Management of Heterogeneous Wireless Sensor Networks*

Markus Anwander, Gerald Wagenknecht, Thomas Staub, and Torsten Braun
Institute of Computer Science and Applied Mathematics
Neubrückstrasse 10
3012 Bern, Switzerland
{anwander|wagen|staub|braun}@iam.unibe.ch

ABSTRACT

Wireless sensor networks (WSNs) are taking a big step forward to productive deployments. Heterogeneous WSNs are gaining importance. Complex problem settings consisting of different environmental conditions require specific sensor nodes for the individual tasks resulting in heterogeneous networks. As the different types of sensor nodes may be incompatible, a more general management architecture for these heterogeneous environments is a necessity. Individual nodes have to be reconfigured and updated during their lifetime. Our WSN management framework supports common management tasks such as monitoring the WSN, configuration of the WSN, code updates, and managing sensor data. Our management architecture consists of the following infra-structural elements: a management station, a number of management nodes and a high number of heterogeneous sensor nodes. All management tasks are controlled by the management station. Management nodes are implemented as wireless mesh nodes.

1. INTRODUCTION

A WSN consists of a number of sensor nodes. They have a large area of applications, e.g., event detection, localization, tracking, monitoring and many more, which require specific nodes to perform the individual tasks. This results in heterogeneous networks.

Currently available sensor nodes are mainly prototypes for research purposes. A number of sensor nodes have been evaluated. We selected four of them to build a heterogeneous sensor network: ESB nodes [1], tmote SKY [2], BTnodes [3] and micaZ [4]. For the management backbone a Wireless Mesh Network (WMN) consisting of Wireless Router Application Platform boards (WRAP) [5] nodes have been selected.

*This work has been supported by the Hasler Foundation under grant number ManCom 2060 and the Swiss National Science Foundation under grant number 200020-113677/1

Contiki [6] is being used as operating system running on the sensor nodes. It is a dynamic operating system with special focus on portability. It is written in C and supports 14 platforms and 5 CPU types. A small TCP/IP [7] stack (μIP) is implemented. Contiki, moreover supports preemptive multithreading, inter-process communication and dynamic runtime linking of standard Executable Linkable Format (ELF) files. Program modules can be updated and loaded at runtime. Contiki and the network simulator COOJA are open source projects and run under BSD license.

To improve current research our concept adds mechanisms to support heterogeneity for management in WSNs. MANNA [8] presents an information architecture and a functional management architecture for WSNs. The management architecture provides functions to establish configurations for sensor network entities. Each of the hierarchical deployed manager nodes is responsible for a cluster of sensor nodes. The information architecture defines the information units and the information exchange among the entities. Currently no implementation of MANNA exists. Guidelines are proposed, but the communication model and other issues are not yet defined. TinyCubus [9] presents a management and configuration framework for WSNs. It also bases on a clustered architecture assigning certain roles to sensor nodes. Another focus is code distribution minimizing the code fragments to be distributed in a WSN. Reliability shall be supported by implicit acknowledgments and retransmissions. The code distribution mechanism has been evaluated in rather friendly environments without high error rates. The Deployment Support Network (DSN) to observe, control, and reprogram a deployed WSN over the air is presented in [10]. Major drawback of this approach is the need of an additional wireless backbone network. The Global Sensor Networks (GSN) [11] provides a middleware for fast and flexible integration and deployment of heterogeneous sensor networks. The key concept in GSN is the usage of virtual sensors, which abstract from implementation details of access to sensor data and correspond either to data streams received directly from sensors. [12] surveys software update techniques in WSNs. Its design space consists of the execution environment at the sensor nodes, the software distribution protocol in the network and optimization of transmitted updates.

The following sections describe the WSN management framework including the management scenario and the management tasks (Section 2), the management architecture

(Section 3), as well as the management protocols (Section 4). Future work is presented in Section 5.

2. MANAGEMENT SCENARIO AND TASKS

A heterogeneous wireless sensor network consists of different types of sensor nodes, which might measure different data and perform different tasks. To operate such a (sub)network the following devices are required: one management station, several mesh nodes and a comparatively high number of heterogeneous sensor nodes. A possible scenario is shown in Figure 1.

The sensor nodes might have different sensors for monitoring the environment. All sensor nodes of one type are able to communicate with each other and build a sensor subnet. Many existing sensor platforms have different radio modules and are thus not able to communicate with each other. The different subnets are interconnected by wireless mesh nodes. They provide interfaces for different sensor subnets and act as gateways. Besides the inter-subnet communication, the mesh nodes perform management tasks. Each mesh node is responsible for one or more subnets. Control of the sensor nodes is done via the management station. The management station is connected to the mesh nodes via Ethernet or via IEEE 802.11.

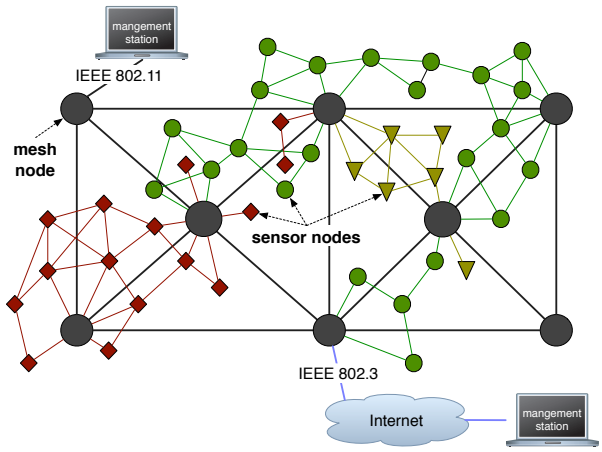

Figure 1: A possible management scenario

From the management point of view there are several tasks required to manage a WSN and its sensor nodes. In general we can divide these into four areas: (1) monitoring the WSN and the sensor nodes, (2) (re)configuring the WSN and the sensor nodes, (3) updating the sensor nodes and (4) managing the sensor data.

The **monitoring task** requires that all sensor nodes in the several subnets are displayed at the management station with all necessary information. This includes sensor node hardware details (e.g. chip, transceiver), sensor node software details (e.g. operating system versions), and dynamic properties (e.g. battery). The node ID and other static information is sent when a sensor node joins the network. Additionally the management station may query sensor nodes. The **(re)configuration task** includes sensor node configuration and network configuration. **Code distribution** mechanisms perform the operating system or the application updates. Mechanisms to handle incomplete, inconsistent and failed updates have to be provided.

3. MANAGEMENT ARCHITECTURE

The management architecture contains the following structural elements: a management station, some mesh nodes as management nodes, sensor node gateways plugged into a mesh node, and the different sensor nodes.

3.1 Management Station

The management station is divided into two parts. It consists of a laptop or remote workstation to access a web interface to control the WSN and a mesh node running the management system for Wireless Mesh Networks (WMNs) [13] including a web server (shown in Figure 2).

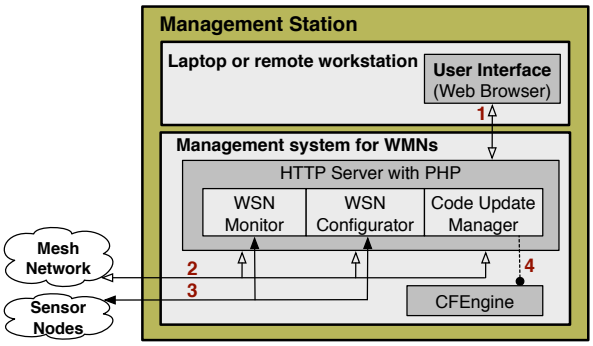

Figure 2: Management station architecture

The communication between the **user interface** and the management system for WMNs is done via HTTPS. The **management system for WMNs** contains a small Linux distribution including all required applications, especially a HTTP server supporting PHP. The HTTP server maintains several modules to handle the requests and transmit them to mesh nodes, sensor nodes or CFEngine [14]. Communication with a mesh node is done via TCP/IP with HTTPS servers running on the mesh nodes (depicted as **2** in Figure 2). The communication between the management station and the sensor nodes is done via TCP/IP (depicted as **3** in Figure 2). For data transmission within the mesh network, CFEngine is used. The **WSN monitor** is responsible for monitoring the whole network. It shows the mesh nodes with their subordinate sensor nodes including all available information of the sensor nodes. The user may request information from a single sensor node. The **WSN configurator** is responsible for the configuration of the WSN. The **code update manager** distributes the uploaded image via CFEngine. It shows the available program versions and performs the updating process.

3.2 WSN Manager

The WSN manager running on the mesh nodes consists of the following components: program version database, WSN information database, sensor database, WSN monitor module, WSN configurator module, and code update manager module (shown in Figure 3). The **sensor database** stores all measured data as tuples (node ID, sensor ID, value,

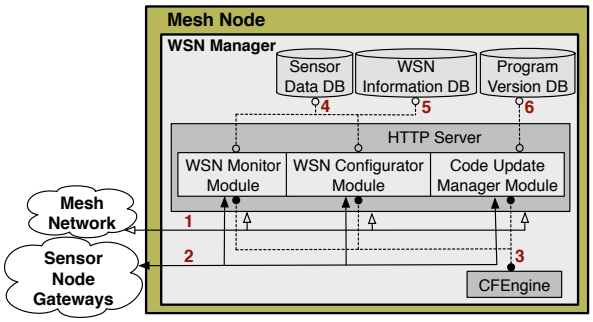

Figure 3: Mesh node architecture

timestamp). The **WSN information database** consists of all infra-structural data. Each entry contains ID, property ID, value, and timestamp. The properties are e.g. chip, transceiver, and battery status. The **program version database** stores the versions of all available programs containing program ID, version, target platform, timestamp, and link to the image. The **CFEngine** is responsible for distributing the databases within the mesh network. The **WSN monitor module** connects to the WSN information DB and to the sensor data DB for responding the requests from the management station. It writes sensor and node data into the databases. The **WSN configurator module** connects to the WSN information database to read and write data. It further queries the sensor node properties and sends commands. The **code update manager module** stores newly uploaded images in the program version DB. It also updates the sensor nodes. It includes methods to reduce the distributed code (differential patch, compression).

3.3 Sensor Node Manager

As shown in Figure 4, the management tasks are handled by a **sensor node manager**. It consists of sensor node monitor, sensor node configurator, sensor data sender, and code updater. The **sensor node monitor** sends the requested values to the mesh node. The **sensor node configurator** executes the configuration requests and notifies the mesh node. The **code updater** receives the image of the application or operating system (differential patch, compressed, or uncompressed) and performs the update. It confirms the success of the update to the mesh node.

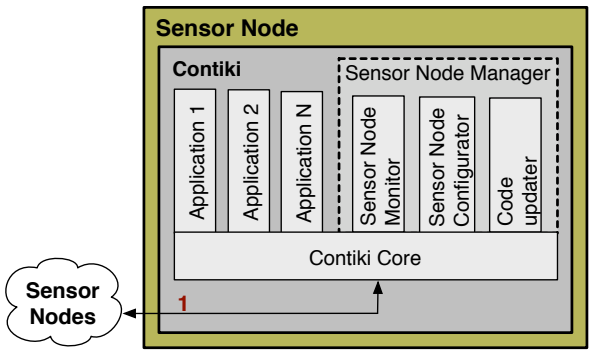

Figure 4: Sensor node architecture

4. WSN MANAGEMENT PROTOCOLS

4.1 WSN Monitoring Protocol

The monitoring protocol enables a management node to get all information about the network topology, all sensor node properties and all measured sensor data. It can be divided into 2 cases: The first case describes how the management station explores the mesh network and the subordinate sensor node networks. The second case describes the situation when the user queries a selected sensor directly.

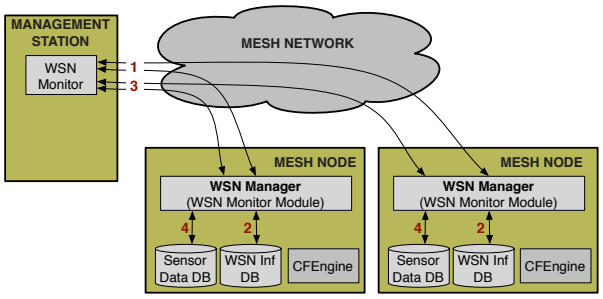

Figure 5: WSN monitor queries the mesh nodes

When the management station joins the mesh network, it connects to the next available mesh node. Because the information is distributed using CFEngine within the mesh network, the information about the WSN topology of distant mesh nodes is several minutes old. In order to receive the actual topology, the management station queries all other mesh nodes (Figure 5). The protocol works as follows: (1) The management station queries all mesh nodes about their subordinated sensor nodes. (2) The WSN monitor modules queries the WSN information DB. (3) The management station requests the current sensor data of every subordinated sensor node. (4) The WSN monitor module queries the sensor data DB.

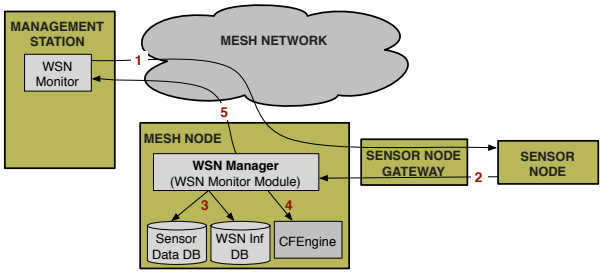

Figure 6: User requests sensor node information directly

The second case is shown in Figure 6 and works as follows: (1) The user requests either sensor node information or sensor data from a single sensor node or from a group of sensor nodes. The request is transmitted via a unicast, multicast or broadcast transport protocol to the queried sensor nodes. (2) The sensor sends the requested information back to the mesh node. (3) The WSN manager module writes the new information into the according database. (4) It copies the information to CFEngine which distributes it within the WMN. (5) The WSN sends a confirmation to the WSN monitor.

4.2 WSN Configuration Protocol

With the WSN configuration protocol the properties of the sensor nodes as well as the network can be configured. Examples are changing sensing intervals or routing tables. It works similar as the WSN monitoring described in 4.1. The request message contains a configuration command.

When a new sensor node joins the following tasks are performed (see Figure 7): (1) First, it broadcasts a 'Hello' message to the WSN configurator module. (2) An initial network configuration is negotiated. Then all available information of the sensor node is requested. (3) The sensor node is registered in the WSN information DB. (4) All available information is propagated to CFEngine for distribution.

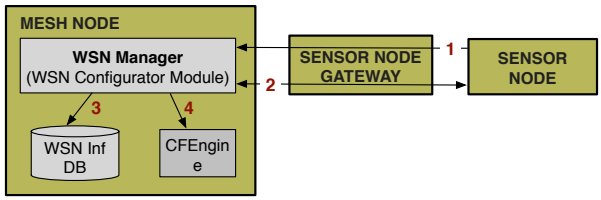

Figure 7: A new sensor node joins the sensor network

4.3 Code Update Protocol

The code update protocol contains mechanisms to upload and distribute the images within the mesh network, notifying the management station about the available programs, and performing the update. The new image of an application or the operating system is uploaded and stored in the program version database and distributed within the WMN. The management station is notified which programs are available in the program version database.

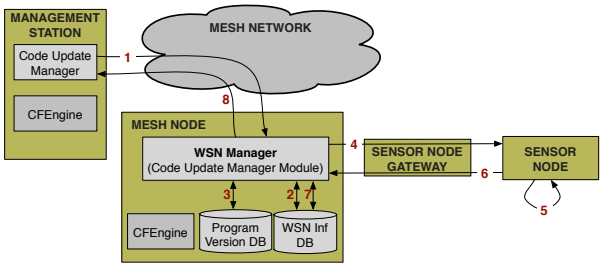

Figure 8: The user initiates the code update for the sensor node

Figure 8 shows the update process of the sensor nodes: (1) The program version and the sensor nodes are selected. The code update manager sends this request to the concerned mesh nodes. (2) The installed program version is checked by querying the WSN information DB. (3) A patch is generated from the old and new image. (4) This is sent to the selected sensor nodes. (5) On the sensor node the update is installed. (6) The update is acknowledged. (7) The WSN information DB is updated. (8) The management station is notified about the success.

5. FUTURE WORK

After defining the management architecture and selecting the appropriate sensor node platforms and operating system, the next tasks concern the implementation of the sensor node management architecture. Other important tasks are the development of reliable communication mechanisms for WSNs. This includes reliable transport protocols for unicast, multicast and broadcast communication. The next step is to develop a reliable point-to-point transport protocol. In detail, the Contiki TCP stack for Distributed TCP Caching (DTC) [15] mechanism and TCP Support for Sensor networks (TSS) [7] will be extended.

6. REFERENCES

[1] J. Schiller, A. Liers, H. Ritter, R. Winter, Th. Voigt. ScatterWeb Low Power Sensor Nodes and Energy Aware Routing. *HICSS-38, Hawaii, USA, Jan 2005.*

[2] Tmote SKY. http://www.moteiv.com. Last visit 04.07.

[3] J. Beutel, M. Dyer, M. Hinz, L. Meier, M. Ringwald. Next-generation prototyping of sensor networks. *SenSys'04, Baltimore, USA, Nov 2004.*

[4] J. Hill, D. Culler. MICA: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro, November 2002.*

[5] WRAP. http://www.pcengines.ch. Last visit 04.07.

[6] A. Dunkels, B. Grönvall, Th. Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *1st IEEE Workshop on Embedded Networked Sensors, Tampa, USA, Nov 2004.*

[7] T. Braun, T. Voigt, A. Dunkels. TCP Support for Sensor Networks. *IEEE/IFIP (WONS 2007), Obergurgl, Austria, Jan 2007.*

[8] L. Ruiz, J. Nogueira, A. Loureiro. Manna: A management architecture for wireless sensor networks *IEEE Communications Magazine, Vol. 41, No. 2, Feb 2003, pp. 116-125.*

[9] P. Marron, A. Lachenmann, D. Minder, M. Gauger, O. Saukh, K. Rothermel. Management and configuration issues for sensor networks *Int. Journal of Network Management, Vol. 15, 2005, pp. 235-253.*

[10] M. Dyer, J. Beutel, L. Thiele, T. Kalt, P. Oehen, K. Martin, P. Blum. Deployment Support Network - A Toolkit for the Development of WSNs *EWSN'07, Delft, Jan 2007.*

[11] K. Aberer, G. Alonso, D. Kossmann. Data Management for a Smart Earth. *SIGMOD Record, Vol. 35, No. 4, Dec 2006.*

[12] C. Han, R. Kumar, R. Shea, M. Srivastava. Sensor network software update management: a survey. *Int. Journal of Network Management, Vol. 15, 2005, pp. 283-294.*

[13] T. Staub, D. Balsiger, M. Lustenberger, T. Braun. Secure Remote Management and Software Distribution for Wireless Mesh Networks *ASWN'07, Santander, Spain, May 2007.*

[14] Cfengine. http://www.cfengine.org. Last visit 04.07.

[15] A. Dunkels, T. Voigt, H. Ritter, and J. Alonso. Distributed TCP Caching for Wireless Sensor Networks. *Annual Mediterranean Ad Hoc Networking Workshop, Bodrum, Turkey, Jun 2004.*