Demo: MARWIS - a Management Architecture for Heterogeneous Wireless Sensor Networks

Gerald Wagenknecht, Markus Anwander, Torsten Braun
Institute for Computer Science and Applied Mathematics
University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
wagen|anwander|braun@iam.unibe.ch

***Abstract*—In this paper we present a demonstration of a novel management architecture for heterogeneous wireless sensor networks (WSNs) called MARWIS. The architecture supports common management tasks such as monitoring, (re)configuration, and updating program code in a WSN. It considers specific characteristics of sensor nodes. To handle large heterogeneous WSNs MARWIS divides it into smaller sensor sub-networks, which contain sensor nodes of the same type. A wireless mesh network (WMN) operates as a backbone and builds the communication gateway between these sensor sub-networks. The mesh nodes perform also the management tasks. The user controls the management tasks using a web-based graphical user interface.**

I. Introduction

A heterogeneous wireless sensor network (WSN) consists of several different types of sensor nodes (SN), such as TelosB [1], MSB [2], MICAz [3], and BTnodes [4]. Various applications supporting different tasks, e.g., event detection, localization, and monitoring, may run on these specialized sensor nodes. In addition, new applications have to be deployed as well as new configurations and software bug fixes have to be performed during the network life time. In a network with thousands of nodes, this is a very complex task and a general management architecture is required.

In [5] we presented MARWIS, a Management Architecture for Heterogeneous Wireless Sensor Networks, which supports common management tasks such as monitoring the WSN, configuration of the WSN, and distributing code updates efficiently and automatically over the network. Furthermore, MARWIS proposes the usage of a wireless mesh network (WMN) as a backbone to build a heterogeneous WSN.

This paper complements [5] with a demonstration of MARWIS and is structured as follows: Section II briefly presents the management architecture MARWIS, including typical management scenarios, description of the structural elements, and the management protocols. Section III describes details of the implementation. Section IV explains practical information about the demonstration.

II. MARWIS

A. Management Scenarios

In a WSN several different applications may run, e.g., event detection, localization, tracking, monitoring. For such applications different types of sensor nodes are required. The sensor nodes of the same type and in the same region can be referred as a sensor sub-network. In general, existing sensor node platforms have different radio modules. This in combination with large distances, results in the inability to communicate directly. Different sub-networks are not able to communicate directly to each other due to different radio modules and to large distance. This results in a heterogeneous WSN built from several sub-networks. To interconnect such a heterogeneous WSN, we propose a WMN as a backbone network connecting the sub-networks. A sensor node plugged into a serial interface (e.g., USB) to a mesh node works as a gateway. The wireless mesh nodes communicate among each other via IEEE 802.11. A possible scenario is shown in Figure 1.

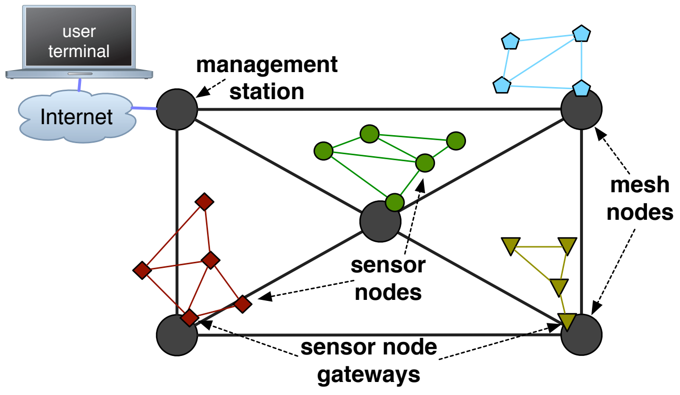


Fig. 1. A possible scenario for heterogeneous WSNs with management devices.

The use of a WMN as backbone has various advantages. The main benefit is the ability to communicate with different types of sensor nodes in several sensor sub-networks. In addition to communication gateway functionality, mesh nodes perform management tasks for the heterogeneous WSN. Moreover, the use of a WMN has advantages by dividing a huge WSN into smaller sensor sub-networks. Usually, a maximum hop count of three to four hops from each sensor node to the gateway can be achieved. This results in a better communication performance with a significantly lower packet delay, jitter and packet loss. Another advantage of using a WMN is that a new sensor node platform can be easily integrated into the heterogeneous WSN without any significant installation effort. A sensor node has just to be plugged into a mesh node.

In a heterogeneous WSN with a large number of different sensor nodes, a comprehensive management architecture is required. In addition to the mesh nodes providing the management functionality, one or more management stations are required. Due to constraints in memory and computational power of the sensor nodes, the management functionalities are shifted to the mesh nodes as well.

From the management point of view there are several tasks required to manage a heterogeneous WSN. In general, the tasks can be divided into three groups: (1) monitoring the WSN and the sensor nodes, (2) (re)configuring the WSN and the sensor nodes, and (3) updating and reprogramming the sensor nodes.

The management tasks include visualization of all sensor nodes in the various sub-networks at the management station. Furthermore, status information about the sensor nodes has to be monitored and displayed. This includes hardware features (micro-controller, memory, transceiver), software details (operating system versions, protocols, applications), dynamic properties (battery, free memory), and position information. The sensor node configuration includes, e.g., sensing intervals of the sensors, running applications on the sensor nodes, or network settings. Updating and reprogramming the sensor nodes are very important tasks, which cannot be performed manually in a large WSN. A mechanism to handle this automatically and dynamically over the network is required, including handling of incomplete, and failed updates.

B. Management Architecture

As shown in Figure 2, the architecture contains the following structural elements: one or more management stations, several mesh nodes as management nodes, sensor node gateways plugged into a mesh node, and the different sensor nodes.

The **management station** is divided into two parts. It consists of a user terminal to access a web-based graphical user interface (GUI) to control the WSN and further ADAM, a management system for WMNs [6], which includes a web server. The user interface displays the WSN topology with the mesh nodes including the subordinate sensor nodes and information about the state of the sensor nodes. ADAM contains a small Linux distribution including all required applications, especially a HTTPS server to handle the requests and transmits them to sensor nodes.

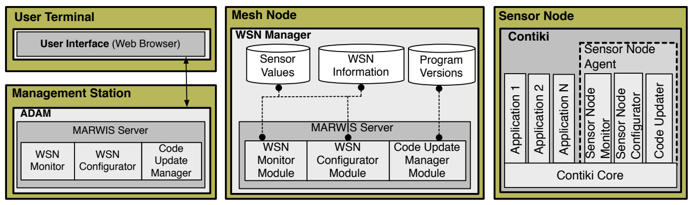


Fig. 2. Architecture of the MARWIS elements.

Modules that provide the management functionalities are located on the mesh nodes. They consist of three databases and the **MARWIS server** with three program modules, such as the WSN Monitor, the WSN Configurator, and the Code Update Manager. The databases store all information about the sensor nodes and the WSN, such as topology (neighbors, address), states of the sensor nodes (battery, memory), versions of programs for the platforms, which can be installed on the sensor nodes, and all data measured by the sensors.

On the sensor node, the management tasks are handled by a **SN agent**. One such module consists of a SN monitor, a SN Configurator, and a Code Updater. These modules can be queried to respond with the current state, to perform configuration tasks, or to update program code.

All communication in the network between the mesh nodes as well as between the sensor nodes is done over TCP/IP.

C. Management Protocols

We consider monitoring, reconfiguration, and code updating the most important tasks of our management architecture.

Monitoring of the WSN can be performed in two ways. First, the management station explores parameters to be monitored by querying the database on the mesh network. Alternatively, the user can query a selected sensor node directly. The first variant is more energy-efficient, as no expensive transmissions between sensor nodes are necessary, but the responded data may not be up-to-date.

With the WSN configuration protocol the properties of the sensor nodes as well as the network can be configured. The procedure is similar to WSN monitoring, but a configuration command is included in the request. As the SN configurator has a universal interface and hides the sensor node type specific characteristics, the configuration is independent of the node type.

The code update protocol consists of three main subtasks. A new image of an application or the operating system is uploaded and stored in the database and distributed within the WMN. The management station is notified about the programs available. Finally, the image is transmitted to the sensor node to be updated. On the sensor node, the update is then performed and acknowledged. The database is updated and, finally, the management station and the user is notified about the success of the update.

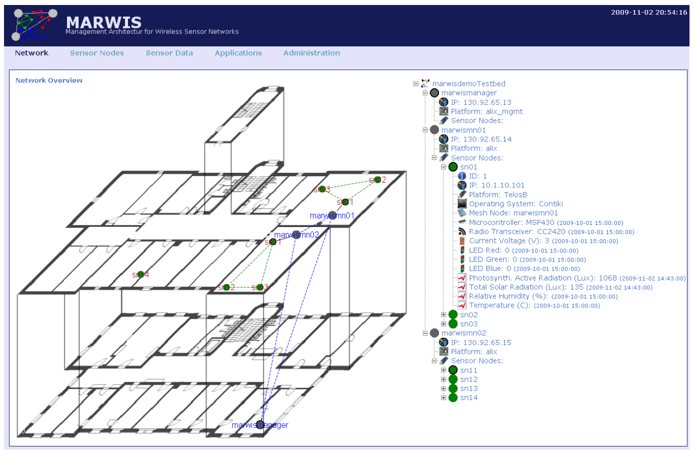
III. Implementation

The management station consists of two devices, a mesh node and a user terminal (e.g., PC). The management system for WMNs is located on a mesh node and contains all necessary programs and configurations to run the management software. The user terminal only has to provide a web browser for the user interface. It connects over HTTPS to the management station.

The user interface visualizes the topology of the networks and the information about the sensor nodes (Figure 3). By clicking on the hosting mesh node (e.g. *marwismn01*), the connected sensor sub-network is shown. The graphics have been created by Graphviz 2.12 [7] using the neighborhood data from the database. By clicking on a sensor node (e.g, *sn01*), the information about the senor node and the operating system is shown. The state of the LEDs are displayed and the installed sensors and their values are represented in a diagram (Figure 4).

The management system for WMNs contains a small Linux distribution (kernel 2.6.28.6) including all required applications, especially a HTTPS server for the connection with the user interface. The modules handling the management tasks and the communication between the mesh nodes are implemented as a server program written in C using UDP/TCP sockets (MARWIS server). The databases are managed with sqlite3 [8].

The mesh nodes run the ADAM software, except for the HTTPS server, as well as the modules handling the management tasks. The Serial Line Interface Protocol (SLIP) is used to communicate with the sensor nodes over a Linux TUN/TAP interface. SLIP connects the IP layer of the mesh node directly to the IP layer of the sensor node gateway.

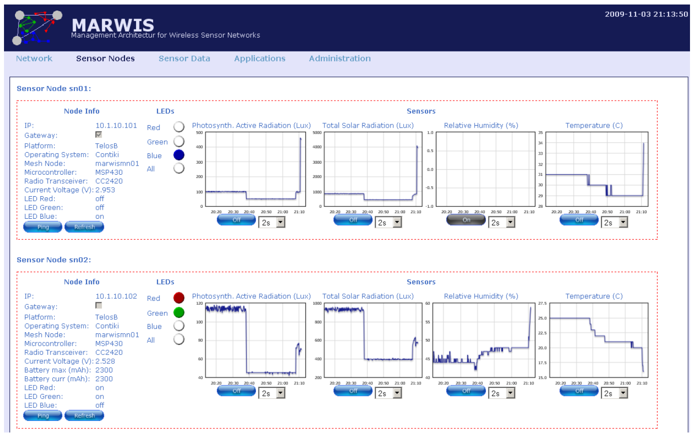


marwisdemoTestbed
- marwismanager
  - IP: 130.92.65.13
  - Platform: alix_mgmt
  - Sensor Nodes:
- marwismn01
  - IP: 130.92.65.14
  - Platform: alix
  - Sensor Nodes:
    - sn01
      - ID: 1
      - IP: 10.1.10.101
      - Platform: TelosB
      - Operating System: Contiki
      - Mesh Node: marwismn01
      - Microcontroller: MSP430 (2009-10-01 15:00:00)
      - Radio Transceiver: CC2420 (2009-10-01 15:00:00)
      - Current Voltage (V): 3 (2009-10-01 15:00:00)
      - LED Red: 0 (2009-10-01 15:00:00)
      - LED Green: 0 (2009-10-01 15:00:00)
      - LED Blue: 0 (2009-10-01 15:00:00)
      - Photosynth. Active Radiation (Lux): 1068 (2009-11-02 14:43:00)
      - Total Solar Radiation (Lux): 135 (2009-11-02 14:43:00)
      - Relative Humidity (%): (2009-10-01 15:00:00)
      - Temperature (C): (2009-10-01 15:00:00)
    - sn02
    - sn03
- marwismn02
  - IP: 130.92.65.15
  - Platform: alix
  - Sensor Nodes:
    - sn11
    - sn12
    - sn13
    - sn14

Fig. 3. User interface: network overview.

We have evaluated several types of sensor nodes and selected four types to build a heterogeneous WSN: TelosB [1], MSB [2], MICAz [3], and BTnodes [4]. For the management backbone, Alix 3D mesh nodes with two IEEE 802.11b/g interfaces have been selected.

The sensor nodes run Contiki [9] as an operating system. The code updater on the sensor node is part of Contiki and responsible for the code replacement. Contiki works with loadable modules to allow replacing applications. To start and finish the application the functions *_init()* and *_fini()* are required and the application has to be compiled as ELF (Executable and Linkable Format) [10] loadable. Standard ELF or CELF (Compact ELF) is used. In contrast to ELF files CELF files are represented with 8 and 16-bit data types, which is adequate for 8-bit micro-controllers. Therefore, CELF files are usually half the size of the corresponding ELF file, but cannot be loaded by a standard ELF file handler. After reception, the referenced variables are checked and functions of the new application are linked and relocated by the Contiki dynamic Link Editor (CLE) and the application is copied to the ROM. Then the code updater starts the application using the *_init()* function.



Sensor Node sn01:

IP:	10.1.10.101
Gateway:	✓
Platform:	TelosB
Operating System:	Contiki
Mesh Node:	marwismn01
Microcontroller:	MSP430
Radio Transceiver:	CC2420
Current Voltage (V):	2.953
LED Red:	off
LED Green:	off
LED Blue:	on

Sensor Node sn02:

IP:	10.1.10.102
Gateway:	
Platform:	TelosB
Operating System:	Contiki
Mesh Node:	marwismn01
Microcontroller:	MSP430
Radio Transceiver:	CC2420
Current Voltage (V):	2.528
Battery max (mAh):	2300
Battery curr (mAh):	2300
LED Red:	on
LED Green:	on
LED Blue:	off

Fig. 4. User interface: sensor node overview.

IV. Demonstration

For the demonstration we will setup a heterogeneous WSN with different types of sensor nodes, similar as shown in Figure 1). We will show the management of such a network using the MARWIS architecture. This includes monitoring and configuration of the WSN and updating applications to the sensor nodes. The required equipment for our demonstration is listed below:

- One laptop and AC/DC power supplies
- 4 Alix 3D mesh nodes
- 3 TelosB sensor nodes
- 3 MSB sensor nodes
- 3 MicaZ sensor nodes
- 3 BTnodes sensor nodes

The requirements for the demonstration are:

- A table (60 cm x 80 cm)
- Wired Internet connection
- One Power outlet
- Poster space/stand

The setup time is about ten minutes.

References

[1] TelosB: http://www.willow.co.uk. Last visit July 2011.
[2] MSB: http://cst.mi.fu-berlin.de/projects/ScatterWeb/ Last visit July 2011.
[3] MICAz: http://www.snm.ethz.ch/Projects/MicaZ. Last visit July 2011.
[4] BTNode: http://www.btnode.ethz.ch. Last visit July 2011.
[5] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, S. Morgenthaler: MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks, *WWIC'08, Tampere, Finland, May'08*.
[6] T. Staub, S. Morgenthaler, D. Balsiger, P. K. Goode, T. Braun: ADAM: Administration and Deployment of Adhoc Mesh Networks *HotMESH'11, Lucca, Italy, June 2011*.
[7] Graphviz: A Graph Visualization Software http://www.graphviz.org. Last visit July 2011.
[8] SQLite: http://www.sqlite.org. Last visit March 2011.
[9] A. Dunkels, B. Grönvall, T. Voigt: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *EmNetS'04, Tampa, FL, USA, November 2004*.
[10] ELF: Executable and Linkable Format. http://www.linux-kernel.de/appendix/ap05.pdf. Last visit July 2011.