On-demand Software Management in Sensor Networks using Profiling Techniques

Zheng Yao[*], Zengyu Lu[*], Holger Marquardt[*], Gerhard Fuchs, Sébastien Truchat, Falko Dressler
Autonomic Networking Group
Dept. of Computer Science 7
University of Erlangen, Germany
{gerhard.fuchs,sebastien.truchat,dressler}@informatik.uni-erlangen.de

ABSTRACT
The heterogeneity and dynamics in terms of hardware and software configurations is steadily increasing in sensor networks. Therefore, software management is becoming one of the most prominent challenges in this domain. We developed a profile-based software management scheme that consists of a dynamic profile-matching algorithm to identify current SW/HW configurations, an on-demand code generation module, and mechanisms for dynamic network-centric reprogramming of sensor nodes. In this demo, we will dynamic node reprogramming based on given objectives (global goals), available sensor nodes, and RPC-style profile matching algorithms. A mobile robot system is employed for decision processes and to store the source code repository. Additionally, techniques for dynamic addressing are included to prevent global pre-configuration of all network nodes.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *wireless communication*; D.2.9 [**Software Engineering**]: Management - *software configuration management*

General Terms
Algorithms, Management, Experimentation.

Keywords
Wireless sensor network, Re-programming, Re-configuration, Network management, Software management, Profiling.

1. INTRODUCTION
With the proliferation of wireless sensor networks (WSN) and sensor/actuator networks (SANET), new application domains appear that make efficient use of such networks [2]. While many challenges have been identified in this area [1], we concentrate on software management techniques for WSN that are dynamic in terms of availability, mobility, and current application demands. Due to limited resources, sensor nodes are usually able to run a single task only. We developed a framework consisting of three methods for SW management in heterogeneous networks: (1) profile matching using light-weight RPCs and dynamic addressing as a helper function, (2) dynamic on-demand code generation, and (3) network-centric reprogramming [3].

These mechanisms can be used in different applications. While, each method can be employed independently, we implemented a common interface for our robot control system "robrain". In this demo, we show a scenario consisting of a WSN assisted by mobile robot systems. These robots are used used because of many reasons: the robots can store source code for many applications and dynamically generate binaries for the sensor nodes, single-hop reprogramming is more reliable, and security issues are limited to one-hop neighborhoods.

2. SCENARIO/DEMO DESCRIPTION
In our lab, we use the Robertino robot platform developed at the Fraunhofer Institute AIS running embedded Linux and Mica2 sensor motes developed at UCB running TinyOS. For direct robot-sensor communication, we installed a single Mica2 mote at the robot. For node reprogramming, we prepared all sensor notes with an initial binary that contains a module for profiling concerns. The robot uses this module to query information about the HW configuration and the currently installed SW, e.g. Mica2 / Mica2dot, temperature measurement / localization. On the robot, we store nesC code and code templates that are described by profiles. This enables the robot to select and adapt the source code while concerning the current context and requirements and, finally, to create a new binary for the sensor node. The robot can install the image over the air. The setup of the demo is depicted in Figure 1. Five Mica2 and Mica2dot sensor motes are used to demonstrate the profile matching and reprogramming approach.

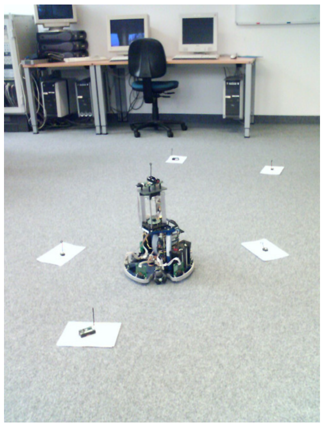


Figure 1. Demo setup

* Zheng Yao, Zengyu Lu, and Holger Marquardt collectively implemented the system

REALMAN'06, May 26, 2006, Florence, Italy.
ACM 1-59593-360-3/06/0005.

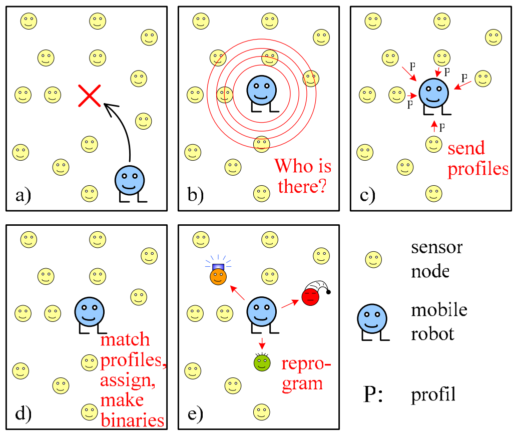


Figure 2. Application scenario for reconfiguration

Figure 2 shows the principal concept of reconfiguration:

a) Depending on the goal, the robot drives to the position in the sensor network where reconfiguration is necessary (we do not assume a particular navigation scheme, various mobility models can be applied).
b) The robot collects information about the environment, builds the context, and explores its neighborhood. In this step, additional actions can be initiated such as preparing the sensor calibration or starting an algorithm for dynamic addressing.
c) All sensor notes respond to the exploration message by sending their current profiles (HW/SW descriptions).
d) The robot uses the information gathered in steps b) and c) for profile matching and to assign the roles of the sensor notes (depending on the current goal). Finally, it creates the new binaries of the sensor notes.
e) The robot reprograms selected sensor notes over the air.

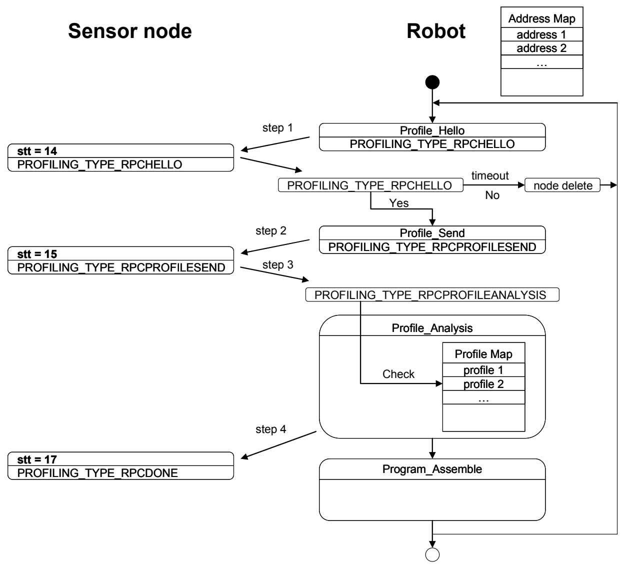


Figure 3. Communication protocol

3. METHODOLOGY

The activity diagram of the reconfiguration process of the mobile sensor network and the communication protocol are shown in Figure 3 and Figure 4. We distinguish between strategic and technical actions. The strategic actions are responsible for the behavior of the entire system. They depend on a global goal and control the reconfiguration process of the sensor network. The technical actions (independent of the goal) provide the functional basics for reconfiguration.

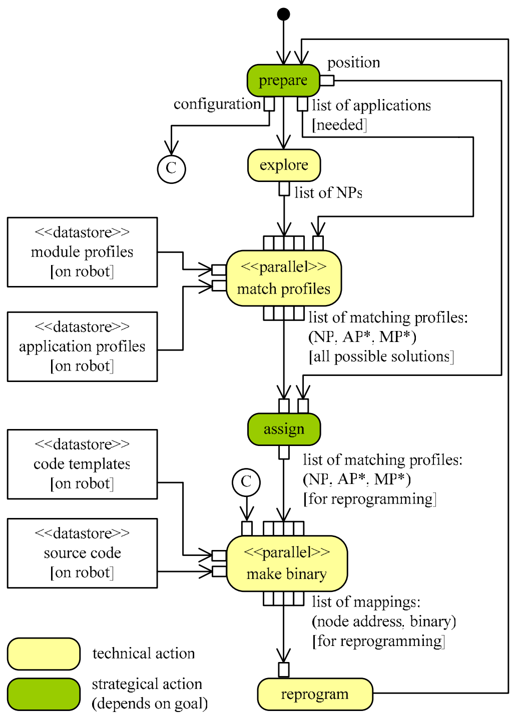
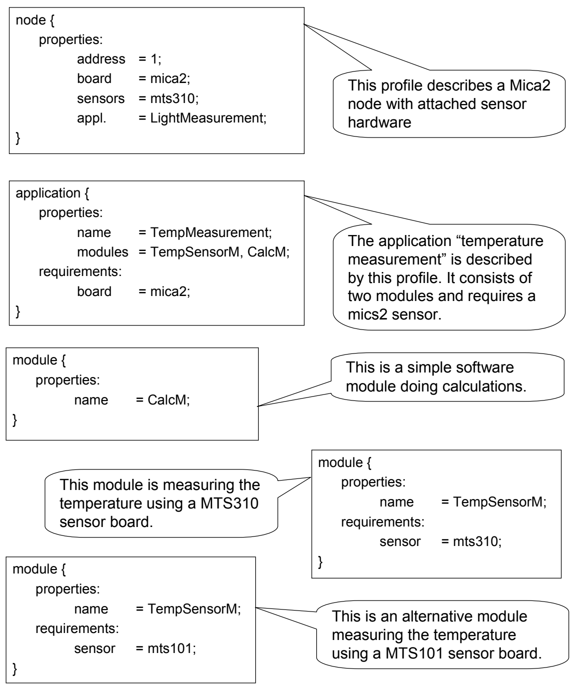

prepare
position
configuration
list of applications [needed]
C
explore
list of NPs
<<datastore>> module profiles [on robot]
<<datastore>> application profiles [on robot]
<<parallel>> match profiles
list of matching profiles: (NP, AP*, MP*) [all possible solutions]
assign
<<datastore>> code templates [on robot]
list of matching profiles: (NP, AP*, MP*) [for reprogramming]
C
<<datastore>> source code [on robot]
<<parallel>> make binary
list of mappings: (node address, binary) [for reprogramming]
technical action
strategical action (depends on goal)
reprogram

Figure 4. Activity diagram of the reconfiguration process (NP = node profile, AP = application profile, MP = module profile; XP* = at least one profile of X)

3.1 Neighborhood Discovery, Profile Matching and Dynamic Addressing

Neighborhood exploration covers two separate steps. The first one, which is optional depending on the configuration of the wireless sensor networks, is the setup of address information. In several scenarios, unique addresses are not necessary to operate a sensor network. Therefore, we propose to initiate a dynamic addressing algorithm first. Based on [5], we implemented a simple addressing scheme for this initial task. The second step is to explore the neighborhood. All nearby sensor nodes must be identified and their profiles must be collected. A simple broadcast to the neighboring nodes is used to send a request to all relevant sensor nodes. Each node sends a reply including its current profile (containing HW/SW info). This profile is taken as input for the following profile-matching algorithm. The primary goal of profile matching is to create all possible combinations of executable source code. Each module or application can be realized using different source files. For example, a module may consist of various sub-modules that can be found in multiple source files.

Some example profiles written in pseudo code are shown in Figure 5. These profiles finally specify the composition of application programs including potential hardware requirements.



```
node {
    properties:
        address   = 1;
        board     = mica2;
        sensors   = mts310;
        appl.     = LightMeasurement;
}
```

This profile describes a Mica2 node with attached sensor hardware

```
application {
    properties:
        name      = TempMeasurement;
        modules   = TempSensorM, CalcM;
    requirements:
        board     = mica2;
}
```

The application "temperature measurement" is described by this profile. It consists of two modules and requires a mics2 sensor.

```
module {
    properties:
        name      = CalcM;
}
```

This is a simple software module doing calculations.

```
module {
    properties:
        name      = TempSensorM;
    requirements:
        sensor    = mts310;
}
```

This module is measuring the temperature using a MTS310 sensor board.

```
module {
    properties:
        name      = TempSensorM;
    requirements:
        sensor    = mts101;
}
```

This is an alternative module measuring the temperature using a MTS101 sensor board.

Figure 5. Examples of profiles used to generate application- and hardware-specific node programs

3.2 Dynamic Code Generation

To be flexible, the robot builds the binaries of the sensor motes just in time. Therefore, it needs a dynamic source code selection and generation system. Code fragments for TinyOS programs are written in nesC. Specific profiles as shown above are connected to these fragments in order to describe functionality and utilization. In order to generate a binary that runs on the nodes described by its node profile, corresponding nesC modules are extracted from the repository and provided to the compiler. The structure of TinyOS programs requires some additional handling in combination with the selection of source files. First, the wiring between the modules must be defined. Based on the available descriptions, templates can be used for an unambiguous wiring. Secondly, some parts of the nesC code have to be adapted to different hardware configurations. We also allow to generate nesC code on demand using code templates. Such templates are filled with variables and algorithms depending on the current context, i.e. the environmental conditions. A template and a configuration defined by a profile will be substituted to a configurable software module that is adapted to a particular hardware configuration. In a final step, the node profile is transformed to a nesC file that can be compiled to a new binary. This binary reflects the application profile and corresponds to the actual hardware capabilities.

3.3 Network-based Reprogramming

The last part of the node reconfiguration using profiling techniques is the re-programming of the nodes for which new binaries have been generated in the last step. We use an extended version of Deluge [4] for this purpose. In the context of code generation, an additional software module will always be installed in any generated binary image that is responsible for network-centric reprogramming.

4. CONCLUSIONS AND FURTHER WORK

We developed a dynamic software management system that can be applied in different application scenarios in WSN. To demonstrate the functionality of the profile-based reconfiguration scheme, we implemented the following modules: dynamic node addressing, RPC-based profile matching, dynamic code generation, and network-centric reprogramming. Even though the developed systems can be flexible used, we created a common interface for our robot control system "robrain". Future work includes the integration of other types of sensor nodes, e.g. BTnodes, the development of control schemes for optimized node selection depending on the node state (resources, energy) and the network state (at least in a given region).

5. REFERENCES

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges," *Elsevier Ad Hoc Network Journal*, vol. 2, pp. 351-367, October 2004.

[2] D. Culler, D. Estrin, and M. B. Srivastava, "Overview of Sensor Networks," *Computer*, vol. 37 (8), pp. 41-49, August 2004.

[3] G. Fuchs, S. Truchat, and F. Dressler, "Distributed Software Management in Sensor Networks using Profiling Techniques," Proceedings of 1st IEEE/ACM International Conference on Communication System Software and Middleware (IEEE COMSWARE 2006): 1st International Workshop on Software for Sensor Networks (SensorWare 2006), New Dehli, India, January 2006.

[4] J. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," Proceedings of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD, USA, November 2004.

[5] Y. Sun and E. M. Belding-Royer, "A study of dynamic addressing techniques in mobile ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 4 (3), pp. 315-329, April 2004.