Lifetime Analysis in Heterogeneous Sensor Networks

Falko Dressler and Isabel Dietrich
Autonomic Networking Group, Dept. of Computer Science 7
University of Erlangen-Nuremberg, Germany
{dressler,isabel.dietrich}@informatik.uni-erlangen.de

Abstract

Wireless sensor networks (WSN) are composed of battery-driven communication entities performing multiple, usually different tasks. In order to complete a given task, all sensor nodes, which are deployed in an ad-hoc fashion, have to collaborate by exchanging and forwarding measurement data. We define the behavior of the overall sensor network based on the parameters lifetime *and* functional density. *The functional density describes the distribution of all necessary tasks in a given geographical area. The lifetime is primarily given by the time each task is successfully performed by at least one node, i.e. the functional density of all necessary tasks. Nodes can become unavailable due to insufficient remaining energy. We assume that sensor nodes can be reconfigured or reprogrammed by a mobile robot system. There are various reasons for considering robots for this reconfiguration, e.g. reliability, security, and deployment issues. In this paper, we evaluate the advantages of exploiting reconfiguration and reprogramming schemes WSN using mobile robots. The primary objective is to increase the lifetime of the overall network. This goal is achieved by optimizing the functional density of heterogeneous tasks. Based on a developed simulation model, we discuss the advantages and performance characteristics.*

1. Introduction

The research on wireless sensor networks (WSN) has become a major research area during the last couple of years [8]. Especially the characteristic capabilities of WSN, such as the number of available resources (energy, processing speed, storage), distinguish sensor networks from other ad hoc networks. Regardless of these resource restrictions, WSN are exposed to requirements such as increased lifetimes and difficult environmental conditions [7]. Many aspects have already been investigated [2], e.g. self-organization issues [11], the interaction of sensor/actuator networks [1], and energy-aware task allocation schemes [10], while others are still work in progress. For example, the deployment of nodes that will build an ad hoc sensor network is an open problem. Basic deployment procedures were investigated in [13]. More sophisticated approaches have been identified: usually, optional mobility is exploited to optimize the initial deployment [15,19,20]. Especially, the use of mobile robot systems leads to optimized deployments [4]. Basically, all these approaches try to optimize the coverage of a given homogeneous WSN under different objectives such as energy constraints or network lifetime.

The consequential next step is to analyze the behavior of a heterogeneous sensor network. In the context of this paper, we understand the term heterogeneous as the possible co-existence of similar sensor nodes with different programmings. This constraint can be easily relaxed by considering a number of variants of sensor nodes that can fulfill similar tasks under different restrictions.

In this paper, we concentrate on the optimized lifetime of a pre-deployed sensor network by considering the functional density and objective-driven reprogramming of individual sensor nodes [12]. Functional density is a measure for the dispersion of the available programs in a given sensor network, i.e. their heterogeneity compared to the location of the particular nodes and, therefore, an enhanced definition of coverage by means of functional diversity. Then, the lifetime of a sensor network can be estimated by monitoring the minimum functional density in all relevant areas. Similar to other approaches that try to increase the coverage in a WSN, we consider the employment of mobile robot systems. Nevertheless, we do not concentrate on re-locating sensors but on reprogramming them. The performance gain can be deduced by evaluating the functional density over the time and examining different deployment strategies and mobility models. We implemented a simulation model in OMNeT++ [18] to perform an in-deep analysis of the proposed methodology. To produce statistically significant results, simulation control techniques were applied to all our simulations.

The contributions of this paper may be summarized as follows. Lifetime and functional density as key characteristics of WSN are described. The possibilities of exploiting mobile robot systems to increase the functional density in a WSN are motivated and discussed. A simulation model is presented that allows multiple performance studies in common ad hoc and sensor networks including mobile robot systems. The functional density as a key parameter to heterogeneous sensor networks is analyzed under different conditions (number of robots, pre-deployment strategy, sensor location). We obtain statistically significant results by evaluating multiple setups with multiple runs.

The remainder of the paper is organized as follows. Section 2 introduces the scenario an application domain. The methodology to estimate lifetime and functional density are discussed in section 3. The developed simulation model is depicted in section 4 followed by some measurement results in section 5. Finally, section 6 concludes the paper.

2. Scenario

Basically, we consider the following scenario: a number of sensor nodes are deployed over a given area. These sensors build a stationary sensor network, i.e. they cannot move or otherwise change their position. The overall application expects a number of different tasks to be simultaneously fulfilled by all nodes. Therefore, all nodes must be differently programmed in order to achieve this demand. Sensors nodes can be reprogrammed at any time by a mobile robot system (if it is within the communication range of the node) but a node can only run one program at a time. This behavior is usual for embedded sensor nodes due to their processing and memory restrictions.

Due to the possible heterogeneity of hardware platforms and the low resources in terms of processing power, available memory, and networking capacities, new approaches for efficient software engineering are needed. An overview to such issues in sensor nodes is provided in [9]. Culler and coworkers describe the necessity for network-oriented software architectures. Issues on the questions of how to configure, reconfigure, program, and reprogram networked embedded systems such as sensor nodes are discussed in [14].

Currently, we are investigating methods for adaptive reconfiguration of sensor nodes using mobile robot systems. Two separate goals should be achieved using these techniques: calibration of sensor hardware and reprogramming based on changes in the environment. In order to address these issues, we apply profiling mechanisms as described in [12].

The use of mobile robots for reconfiguring single sensor nodes and, finally in a global context, larger ad hoc sensor networks has many advantages. For example, the robot systems usually have much more available resources and can store and maintain software modules needed by various sensor nodes. Additionally, applications like sensor calibration can be done only locally. Calibration means to use expensive high quality sensors attached to the few robot systems to verify much cheaper sensors distributed in the field. We discovered that such cheap sensors need a recalibration in regular intervals.

A possible application domain for WSN assisted by mobile robots is agriculture. Imagine an agricultural setting, for example a vineyard or a potato field, where numerous sensor nodes are deployed within the area. The sensor nodes are capable of measuring humidity, nutrient content of the soil, presence of pests, and maybe even the degree of ripeness of the fruits. They can also transmit the collected information to a base station, possibly using other sensor nodes as routers on the way. The base station can be conveniently located off the field (see [3, 5, 6] for information on real-world deployments of such systems). It is intuitive that in each sector (i.e. potato bed or row of vines) there has to be a minimum number of each of the different sensor types for the system to work properly.

This in itself could lead to a reduction of the workload of the farmer, as well as improve the farming decisions by providing a good information base. However, due to the manual actions still required, and due to the necessity of replacing the node batteries from time to time, the system still has room for improvement. This improvement can be gained by introducing mobile robots with the following capabilities. While driving around the field or vineyard, they can monitor the distribution of the different sensors to recognize sensor failures (caused by battery depletion or hardware failures), reprogram nodes in a sector to cover the tasks of failed nodes, inductively recharge depleted batteries, and supply water, pesticides, herbicides or fertilizer to those areas (sectors) that currently have need of them.

In that way, the workload for the farmer could be greatly reduced, while at the same time augmenting the quality of the farm output because the necessary measures are taken immediately without the delay introduced by humans (e.g. robots don't sleep!).

3. Methodology

In the previous section, we explained the necessity and the principle behavior of robot-assisted reprogramming of sensor networks to increase functionality and network lifetime. Here, the basic measures for the optimization are discussed in more detail. Basically, the lifetime of the overall network is the basis for all optimizations. Nevertheless, we cannot simply extract network lifetime from the lifetime of individual nodes. Also, the coverage as defined in most references in terms of covering each point in the network with at least one sensor is not an appropriate measure.

In order to cope with the new demands, we define *func-*

tional density as the primary parameter for measuring the grade of operation, i.e. the lifetime of the network. Let S be the number of available, i.e. running sensor nodes. Then, we can define $N(S)$ being the global number of available sensors and $N_a(S)$ the global number of nodes running program type a. Let Q be the size of the area in which the sensors are deployed. Based on these parameters, we can calculate the overall, i.e. global, functional density $F_a(S)$ of program a as follows:

$$F_a(S) = \frac{N_a(S)}{Q} \tag{1}$$

In most application scenarios, the global characteristics of a WSN are insufficient to describe the functionality. Therefore, we need to arrange the working area Q into smaller areas n (squares) of size $\frac{Q}{n}$ each. Then, we can define the number of sensor nodes running a particular program a in each sector as $N_a^i(S)$ as shown in figure 1. Thus, we can define the functional density $F_a^i(S)$ of program a in sector i as follows:

$$F_a^i(S) = \frac{N_a^i(S)}{Q/n} \tag{2}$$

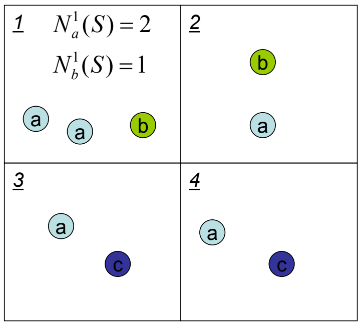

Figure 1. Parameters for calculating the functional density

Now, we are ready to define the lifetime of the network in terms of *operability*. The network is *operable*, if the functional density for each program a and each sector i exceeds a predefined threshold T_a:

$$\forall i : \forall a : F_a^i(S) > T_a \tag{3}$$

Therefore, the network lifetime can be defined as the sum of all time periods where the previous condition holds. In the following section, we provide a simulation model that was used to evaluate the approach.

4. Simulation Model

We used the discrete event simulator OMNeT++ [18] (version 3.2) to build our simulation model. Additional modules from the INET framework (version 20050922) were used for modeling wireless transmissions and mobility.

The basic building blocks of our model are the nodes in the network. We distinguish three types of nodes, namely sensor nodes, robots, and base stations. Each of the node types is composed of multiple modules that represent all elements of the protocol stack, applications, node mobility, and node batteries. Figure 2 displays an exemplary network setup. It contains 50 sensor nodes, the white lamp indicating a particular programming. A base station in the middle of the area and a mobile robot denoted by the penguin on the lower right complete the setup. The channel control module on the left is an INET module responsible for managing wireless transmissions, and the analysisProfiling module is a collector for statistical data.

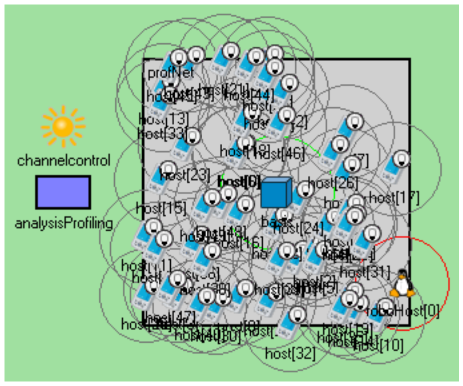

Figure 2. View of the network in the model

4.1. Protocol Stack

Nodes can exchange messages over the air. The physical layer model has to take care of the transmission durations, noise on the channel, and all collisions or other corruptions of messages. The model employs the 802.11 physical layer contained in the INET framework, modified to record a number of packet loss statistics. We also use the corresponding 802.11 mac layer to avoid side effects (like long transmission delays) that would result from the use of a more energy efficient protocol that puts the node to sleep periodically.

The routing protocol AODV was implemented following RFC 3561 [17]. Two timeouts had to be adjusted to take the relatively slow radio operations in a WSN into account. The ”Hello Timeout”, i.e. the period after which each node sends a short message to its neighbors, was incremented from 1 to 10 seconds. The "Active Route Timeout", i.e. the period after which a previously active route is considered as expired and no longer used, was also multiplied by 10, resulting in a 30 second timeout in our model.

4.2. Application

Each of the three node types has its own application model. The base station represents a packet sink for all sensor data in the network. It records statistics on message delays and afterwards simply discards all incoming packets.

The robot application is responsible for reprogramming sensor nodes based on the mechanisms described in the previous section. To simplify the model, the robot does not collect information about its environment, but has global knowledge about the positions, current programs, and battery powers of all the sensor nodes, as well as its own position. As the robot can only reprogram nodes within its communication radius and uses only local information to figure out which nodes to reprogram in the envisioned application scenario, the global knowledge could be replaced by the robot actively inquiring all necessary details from the surrounding nodes. Two reprogramming strategies are available to the robot. First, a node can be picked randomly from all qualifying nodes each time a reprogramming is considered necessary. Secondly, the node with the greatest amount of remaining energy among the qualifying nodes can be chosen. A node is qualified for reprogramming if it is within the robot's communication radius and if there are enough instances of its current program a in the robot's radius so that the removal of one instance would not reduce the number of instances below the threshold T_a.

The modeled sensor applications have two tasks. The first one is to listen for messages from the robot indicating a reconfiguration of the node and to perform the reconfiguration. The second task is to model the behavior of the current program. We model this behavior using the two parameters frequency and permanent energy consumption. The frequency with which sensor data have to be sent to the base station and the additional energy consumption indicating how expensive it is process the data depend on the particular program. We have defined three different programs P_a for the simulations. The first one, P_0, sends one message every 60 seconds and has no additional energy usage. The second program, P_1, has to send a new sensor reading every 10 seconds, but still has no additional energy usage. The third one, P_2, again sends messages every 10 seconds, but also has an additional energy consumption of 5000 mA. All message frequencies are exponentially distributed with the given mean value. The destination of all sensor readings is a base station placed in the middle of the simulation area.

4.3. Mobility

Two different mobility models were employed in our simulations to characterize possible movements of the mobile robots. In the first model, the robot moves randomly according to the Random Waypoint mobility model. In the second model, called Rectangle mobility model, the robot may only move along a fixed rectangular path inside the simulation area. In both cases, the robot speed was fixed at 0.7 m/s, which is an appropriate speed for the chosen area of 200x200m. Additionally, it corresponds to the maximum speed attainable by the "Robertino" robots used in our Labs.

Although the sensor nodes remain stationary throughout the simulation, a special "static" mobility model is applied to them, so that the node positions can easily be obtained and the model remains extensible for future simulations with mobile nodes. There are two variants for node placement: in random placement (RAND), each node independently picks x and y coordinates which are uniformly distributed between 0 and 200. The second method is grid placement (GRID), where the nodes are placed in an equidistant grid inside the simulation area.

4.4. Battery

A battery model has been developed to account for the usage of energy by the sensor nodes. Just as in a real sensor node, there are several consumers that need energy to operate. The main consumer is the radio, but processor, a/d converters, and sensors also consume some amount of energy. The physical layer monitors the current state of the node radio (sleep, idle, receive, or transmit). The energy used by the radio is calculated from the time spent in each of the states. Average consumptions of the typical radio states have been taken from [16]. As the exact state of the node processor at each point in time is hard to determine in a simulation, we make the simplifying assumption that the processor is active whenever the radio is receiving and transmitting, and inactive otherwise. In addition to the detailed accounts for energy drain by the radio and processor, the other consumers are combined to form a third consumer that consumes a fixed but freely configurable amount of energy regardless of the current operations of the node. In our model, battery depletion is the sole cause for node failures.

5. Measurements

All simulation runs were conducted on a simulation area of 200x200 meters. In addition to the sensor nodes and robots, exactly one base node was placed in the middle of the simulation area. The communication radii of all nodes were fixed at 35 m. To reduce the simulation time, all sensor nodes had a battery with a capacity of only 1 mAh. To determine the operability of the network, four sectors were used with a threshold of $T_a = 2$ with $(a = 0, 1, 2)$. The duration of each replication run was fixed at 2100 seconds as most of the sensor batteries are depleted by that time. Table 1 gives an overview over all parameters that were varied in our simulations.

Number of robots	0, 1, 3
Number of sensors	50, 100
Initial deployment	Program 0, random program
Node placement	random, grid
Reprogramming strategy	Random node, node with most energy
Mobility module	Random Waypoint, Rectangle

Table 1. Variable simulation parameters

A variety of statistical data is collected during the simulations. These include communication statistics (like the number of messages sent/received/dropped and end-to-end delays), energy statistics, node positions, mobility traces, operability statistics, and the distributions of programs. These data enabled us to analyze the performance of the network and the robots, especially in terms of operability as defined in this paper, but also in terms of classical performance measures like end-to-end delays and packet loss statistics.

5.1. Random Deployment

We first analyze the network operability with randomly programmed nodes. Figures 3 and 4 depict the mean time when the network becomes operational for the first time and the percentage of the complete simulation time during which the network is operational. The corresponding statistics have been summarized in tables 2 and 3. All setups compared in the figures contain 50 randomly placed sensor nodes. The figures show the differences between setups with none, one, and three robots and between the two reprogramming strategies. It can be seen that the mean time of initial operability decreases with the number of robots employed. The results of replications where the network did not become operational have not been included in the graphics. Instead, a column has been added to table 3 to show the percentage of left-out replications. This explains why the setup with no robots seems to do best in being first operable at zero seconds in every case: either the network is operable from the start or not at all, and the "not at all"-cases (which are 65% of all replications) have been left out as described. Further, figure 4 shows that the increase in total operational time is greatest when one instead of zero robots is used. There is still an increase when using three instead of one robot, but a comparatively small one.

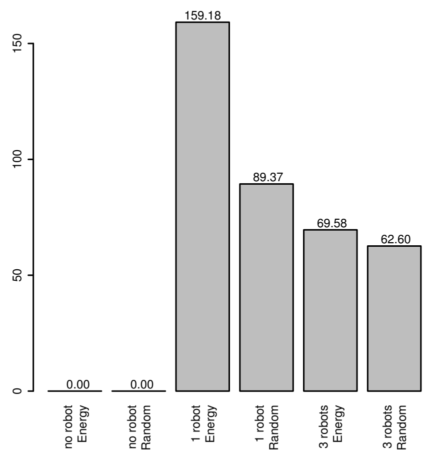

Figure 3. Times of first network operability (seconds), RAND

	Mean	Max
no robot, Energy	0	0
no robot, Random	0	0
1 robot, Energy	159.18	846.03
1 robot, Random	89.37	705.01
3 robots, Energy	69.58	345.02
3 robots, Random	62.6	345.01

Table 2. Times of first network operability (seconds), RAND

It can also be seen that the energy-oriented reprogramming strategy gives only a little improvement (in the case of one robot) and none at all (in the case of three robots) compared to the purely random strategy. The performance difference between the Random Waypoint and Rectangle mobility models is not significant in this case and was therefore not included in in this paper.

Figures 5 and 6 feature the same setup as above, but with nodes placed on a grid inside the simulation area. The results are very close to those with random deployment which means that the deployment strategy does not significantly influence the performance of the robots.

The only disadvantage of the use of mobile robots discernible so far is a slightly higher usage of energy in the sensor nodes resulting in a small decrease of node lifetimes. In the cases with 50 nodes in the simulation area, this is negligible compared to the gain in operable lifetime of the whole network. However, when analyzing the scenarios with 100 nodes in the simulation area, it becomes evident that the robots can even decrease the operable lifetime if the network is very dense. In a setup with 100 nodes randomly placed and randomly programmed, the operable

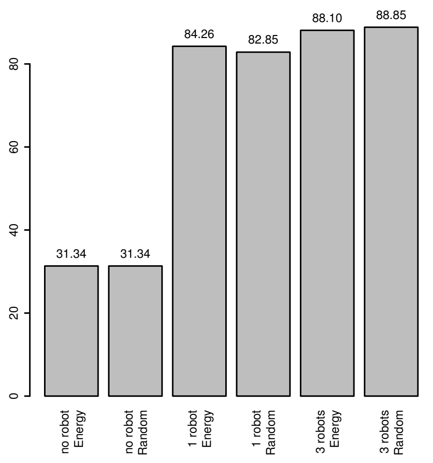

Figure 4. Percentage of network operable time, RAND

	% never operable	Min	Mean	Max
no robot, Energy	65	0	31.34	35.28
no robot, Random	65	0	31.34	35.28
1 robot, Energy	0	49.5	84.26	86.14
1 robot, Random	5	0	82.85	82.85
3 robots, Energy	0	81.83	88.1	89.88
3 robots, Random	0	88.65	88.85	91.98

Table 3. Percentage of network operable time, RAND

lifetime without robots is above 86% and degrades to 84% and 85% with one and three robots, respectively.

5.2. Uniform Deployment

The next step is the analysis of the network behavior with a uniform node deployment, where each node starts with the same program (in the simulations, we used P_0 as defined in the previous section). This measurement corresponds to the envisioned deployment strategy in real scenarios. Obviously, the comparative data for a scenario without robots yield very bad results: the network will never become operable in its entire lifetime. Figures 5.2 and 5.2 show the mean time of first operability and the percentages of network operability for the scenarios with one and three robots, with

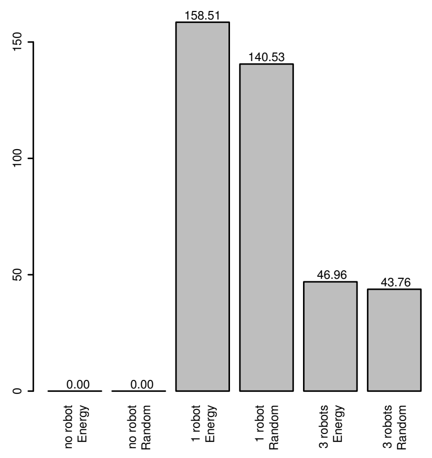

Figure 5. Times of first network operability (seconds), GRID

summary statistics shown in tables 4 and 5. In both cases, 50 nodes have been randomly placed in the simulation area.

	Minimum	Mean	Maximum
1 robot	372.01	993.09	1658.02
3 robots	37.01	281.43	809.5

Table 4. Times of first network operability (seconds)

	% never operable	Minimum	Mean	Maximum
1 robot	5	28.49	42.75	47.02
3 robots	0	74.37	76.48	87.12

Table 5. Percentages of network operable time

The times of first operability increase from never with no robots to a mean of about 1000 seconds with one robot and less than 300 seconds with three robots. The percentage of operable time also increases rapidly, from about 40% with one robot to more than 75% with three robots. This is of course closely related to the times of first operability: the earlier the network becomes operable, the longer can it stay operable during its remaining lifetime. The same qualitative results hold when the nodes are placed in a grid instead of a random deployment. Tables 4 and 5 show the minimum and maximum values as well as the mean for the two measures.

Figure 8 shows the differences in the program distributions over time when employing one or three robots. The robots move according to the Random Waypoint mobil-

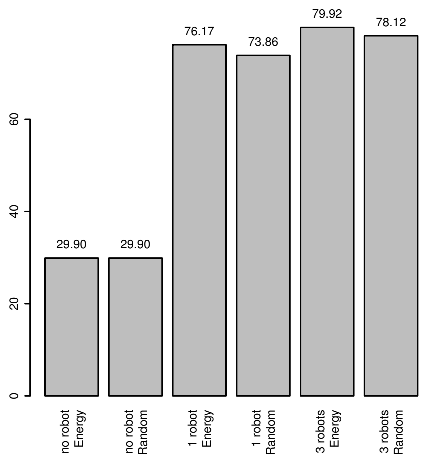

Figure 6. Percentage of network operable time, GRID

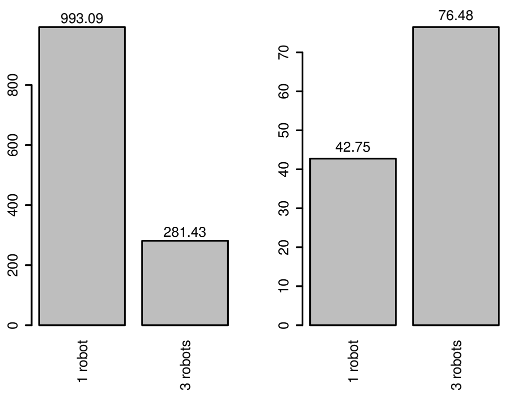

Figure 7. Times of first network operability (left) and percentage of time network was operable (right), RAND

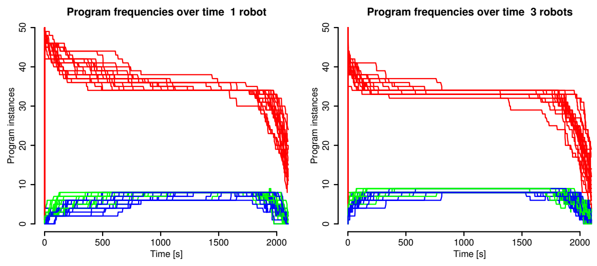

Figure 8. Program distributions over time, Random Waypoint mobility

ity model. Program traces from 20 replications have been plotted in each graph. Both graphs depict a scenario with 50 randomly placed nodes all starting with the same program. Thus, the starting program is initially present on all 50 nodes. In both scenarios, the robot(s) immediately start reprogramming nodes and succeed to establish operability. However, it is obvious that the single robot takes longer for this task than the three robots, and that it has a lot more trouble in keeping the network operable. The operability established by three robots seems to be a lot more robust and stable.

It must also be noted that scenarios with three robots have an advantage towards the end of the network lifetime, as their chances to be at the right spot when a node fails are much higher and thus the time until re-establishing operability by reprogramming another node is reduced.

6. Conclusion

In this paper, we depicted an application scenario for heterogeneously programmed sensor nodes. Mobile robot systems were employed to perform reconfiguration and reprogramming tasks in order to increase the *lifetime* and reliability of the overall system in terms of *functional density*. We developed a simulation model to verify our hypothesis that the WSN can essentially profit from robot-assisted reprogramming. The primary purpose of the simulation model is to evaluate the methodology by analyzing time during which the network is considered operational, i.e. during which the functional density exceeded some given threshold. In order to show the relevance of the methodology, we investigated different mobility models, different deployment strategies, and different node locations. We executed a reasonable number of simulation experiments to obtain statistically significant results.

Finally, we have shown that our strategy of employing mobile robots to reprogram sensor nodes generates significant gains in network operation time if the network distribution is not too dense. We have also shown that the deployment of the sensor nodes can be simplified by initially equipping each node with the same program and leaving the programming of the network to the mobile robots. In summary, it can be said that the proposed method achieves optimized results in terms of operability and increase of the functional density.

References

[1] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Elsevier Ad Hoc Network Journal*, 2:351–367, 2004.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–116, 2002.

[3] A. Baggio. Wireless sensor networks in precision agriculture. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, Stockholm, Sweden, June 2005.

[4] M. A. Batalin and G. S. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. In *International Workshop on Information Processing in Sensor Networks*, pages 376–391, Palo Alto, USA, 2003.

[5] R. Beckwith, D. Teibel, and P. Bowen. Report from the field: results from an agricultural wireless sensor. In *29th Annual IEEE International Conference on Local Computer Networks (LCN)*, pages 471–478, November 2004.

[6] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: sensor networks in agricultural production. *Pervasive Computing*, 3(1):38–45, January-March 2004.

[7] C.-Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[8] D. Culler, D. Estrin, and M. B. Srivastava. Overview of sensor networks. *Computer*, 37(8):41–49, 2004.

[9] D. Culler, J. Hill, P. Buonadonna, R. Szewczyk, and A. Woo. A network-centric approach to embedded software for tiny devices. In *First International Workshop on Embedded Software (EMSOFT 2001)*, Tahoe City, CA, USA, 2001.

[10] F. Dressler and G. Fuchs. Energy-aware operation and task allocation of autonomous robots. In *5th IEEE International Workshop on Robot Motion and Control (IEEE RoMoCo'05)*, pages 163–168, Dymaczewo, Poland, 2005.

[11] F. Dressler, B. Krüger, G. Fuchs, and R. German. Self-organization in sensor networks using bio-inspired mechanisms. In *18th ACM/GI/ITG International Conference on Architecture of Computing Systems - System Aspects in Organic and Pervasive Computing (ARCS'05): Workshop Self-Organization and Emergence*, pages 139–144, Innsbruck, Austria, 2005.

[12] G. Fuchs, S. Truchat, and F. Dressler. Distributed software management in sensor networks using profiling techniques. In *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, 2006.

[13] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[14] V. Handziski, J. Polastrey, J.-H. Hauer, C. Sharpy, A. Wolisz, and D. Culler. Flexible hardware abstraction for wireless sensor networks. In *2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, Istanbul, Turkey, 2005.

[15] A. Howard, M. J. Mataric, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile-sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.

[16] O. Landsiedel, K. Wehrle, and S. Götz. Accurate prediction of power consumption in sensor networks. In *Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, Sydney, Australia, 2005.

[17] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. RFC 3561, July 2003.

[18] A. Varga. The omnet++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, 2001.

[19] G. Wang, G. Cao, and T. L. Porta. Movement-assisted sensor deployment. In *23rd IEEE Conference on Computer Communications (IEEE INFOCOM 2004)*, Hong Kong, 2004.

[20] J. Wu and S. Yan. Smart: A scan-based movement-assisted sensor deployment method in wireless sensor networks. In *24th IEEE Conference on Computer Communications (IEEE INFOCOM 2005)*, Miami, FL, USA, 2005.