RMTool: Component-Based Network Management System for Wireless Sensor Networks

Inuk Jung and Hojung Cha
Department of Computer Science, Yonsei University,
Sinchondong, Seodaemungu, Seoul, Korea
{inukj,hjcha}@mobed.yonsei.ac.kr

Abstract— **This paper introduces a component-based network management system for wireless sensor networks. The research is motivated by a variety of problems that occur due to the unpredictable behavior of wireless sensor networks. Once sensor nodes are deployed, the inner behavior of a sensor field can only be analyzed by monitoring incoming data packets at the sink node or base station, if only such a framework exists. In this paper, we present a component-based management system, RMTool, which allows developers to easily monitor and analyze the network status and interactively configure the network over unexpected problems while running applications over it. The system has been implemented on a multi-threaded sensor network operating system RETOS, which supports run-time loadable kernel modules. The preliminary evaluation shows that RMTool provides management functionalities as designed, and the implementation is efficient due to the component-based module architecture.**

Keywords- Network management; Sensor network monitoring; Sensor network operating system

I. INTRODUCTION

Wireless sensor networks are sophisticated information gathering systems using radio communication and sensing devices to obtain user-interested data from an environment for analysis in large scale. The requirement of sensor networks has complicated in recent years as the applications as well as hardware become heterogeneous. Ideally, sensor networks should be fault-free and have to be operated long enough to provide reliable results. However, this is difficult because of the inherent problems of wireless communication and limited hardware resources.

The most compelling issue that gave the motivation of our research was the lack of information about the inner behavior of a wireless sensor network and the increasing complexity of heterogeneous sensor networks. Currently, there are a number of researches that took sensor network health issues seriously and have proposed solutions, such as network monitoring tools [1][2][3] or even a network debugging mechanism [4]. However, they did not evaluate the performance of their proposed mechanism when operating in the network concurrently with other user applications. Furthermore, sensor network applications are insufficient to provide a minimum level of QoS in such an environment by simply responding to the network using static network parameters over time. Hence, a management system is needed to provide the developer with fundamental network information, to ensure that the basic network infrastructure is functioning correctly, and control over the network parameters, to adapt to constantly changing requirements in the network. The developer should be alerted about any significant problems that need to be dealt with, such as reconfiguring the network parameters or manually relocating nodes that are not able to communicate with the network for some reason. Also, in a large sensor field, a change in requirement of the overall network functionality can be expensive to provide, since such change in the network occurs frequently and may have great impact in the network topology.

As resource is strictly limited in current sensor hardware, many research proposed ways to overcome this while providing maximum functionality. In this paper, we discuss a component-based management system, called RMTool, for sensor networks to provide component wise network management and monitoring, which allows the user to reconfigure the network in run-time for optimal efficiency. The component-based network management system consists of a network analyzer and a network configuration manager. The purpose of the network analyzer is to gather network information and provide a monitoring system to the user through data analysis, whereas the network configuration manager provides the user with some high-level control over the network and applications to optimize their performances or to adapt to any changes in network requirements. The system is implemented using the standard C language on RETOS [5][6][7], which is a wireless sensor network operating system currently being developed to support a multithreading programming environment. The system implements a reconfigurable kernel using dynamic allocation of loadable kernel modules.

II. COMPONENT-BASED NETWORK MANAGEMENT SYSTEM

This section presents the architecture of RMTool, a component-based network management system for wireless sensor networks. The purpose of RMTool is to provide the developer with an environment of both monitoring and real management of a sensor network. RMTool assures a user of the network's functionality and gives the developer some control over the network while running user applications over it. RMTool consists of two major components, the network analyzer (NAZ) and the network configuration manager (NCM).

A. Design Principles

Since network and hardware resources are limited and RMTool is intended to run along with other sensor network applications, it must not interfere with or occupy excessive resources over local applications. In other words, RMTool should not be considered to be a resource competitor by other applications running in the same sensor node. Hence, we strictly constrained our design of RMTool, as shown below:

- minimal resource consumption (e.g., RAM, computation power, battery energy)
- independent from local applications
- simple and robust mechanism

In addition to the constraints from resource limitations, RMTool is also designed to support the following:

- a component-based system for resilient sensor network reprogramming
- a simple interface for accessing its services by developers
- an independent network architecture

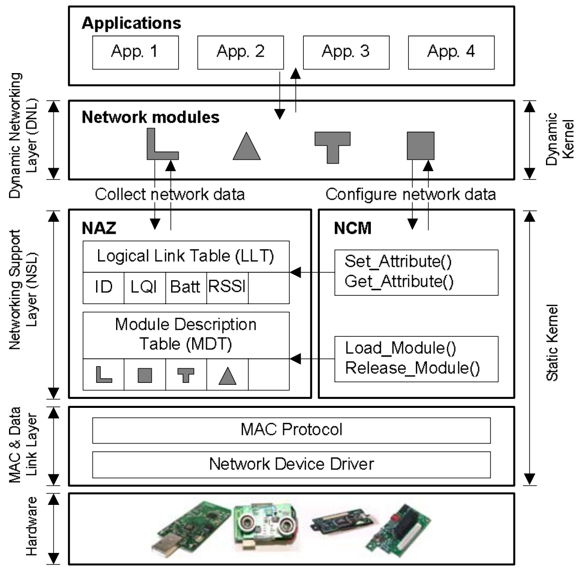


Fig. 1. RETOS Network Architecture

The effort to efficiently manage RAM usage in RMTool can be approached in two ways. One is to actually simplify the implementation of the system. The second approach is to consider reducing the overall RAM used by the entire network, by providing run-time loadable modules for abstracting away unused components in the sensor node hardware. RETOS itself is a modularized system, and maximizes the efficiency of memory usage by loading only the modules required for performing specific tasks. Furthermore, RETOS does not impose any restrictions on module programming for kernel or applications, which suggests that any type of kernel or application components can be loaded in a module-like way. However, any modules that are programmed in kernel level must be crash-safe, since we assume a kernel that is safe from errant applications to protect the sensor node from crashing.

To meet the requirement for providing an easy interface with RMTool, the system is implemented in both the static and the dynamic kernel. Here, we refer the interface to the functions in which developers can selectively choose what network information they want to obtain, since there is much information to obtain within a network. Nevertheless, some information among them can be of no interest. Hence, RMTool has its own policy of network attributes, to free the developer from separately defining them. The collection of fundamental network information and the data structure for storing them is implemented in the static kernel. The APIs of such interfaces and functions of RMTool that tend to need reconfiguration, due to the changing demands of the sensor network, are implemented in the dynamic kernel. Separating the implementation domain within the kernel is to provide safety and reusability of the system.

RMTool provides an independent network architecture. The aforementioned three-layered stack is the architecture provided to serve such needs. In addition, from the developer's point of view, RMTool may be used not only for monitoring the network functionality but also as an overall view of their applications' behavior in the network. The concept of monitoring the network and applications is very different, since topology data and application data are used in different aspects. RMTool is able to sustain the network topology and concurrently acquire application generated data in a reliable way.

B. Network Analyzer (NAZ)

The network analyzer provides a monitoring environment for informing the developer of the current health status of the network. The network analyzer itself does not require much reconfiguration, since its purpose is simple and robust. Within the static kernel, NAZ is implemented in the Network Supporting Layer (NSL), which supports logical connections, neighbor node management, and data transmission. The system overview of the network stack is shown in Fig. 1.

NSL manages a pre-defined static Logical Link Table (LLT) and the functionality for collecting its relevant attributes from the network to deliver them to the user, as shown in Table 1. It also manages a neighbor table of neighbor nodes, which is periodically updated through broadcast queries. Table 2 shows an example of a module description table (MDT), which lists kernel and application modules that are currently available in the sensor node. Thus, a developer can be aware of what type of application tasks or version is operating in specific regions. The tables are updated periodically or with any manual updates.

C. Network Configuration Manager (NCM)

The network configuration manager is designed under the paradigm "Deploy First and Develop Later." It implies that a developer should be able to deploy the network and develop the system to its completeness at a single base station. Since the wireless medium allows abstracting away any one-to-one physical connections exchanging data, sensor network applications themselves can be installed on sensor nodes over the wireless sensor network. Of course, in terms of battery consumption, it is assumed that such expensive use of wireless

TABLE 1. ATTRIBUTES OF THE LOGICAL LINK TABLE (LLT)

Attributes	Description	Attributes	Description
Node ID	Node sequence number	Delivery Rate	Delivery success ratio
Position	Logical position	Delivery Time	Packet delivery time
LQI	Link quality indication	ACK/NACK	Use ACK or NACK
RSSI	Signal strength	Battery Level	Battery residual
CCA	Channel loading status	Neighbor Table	Neighbor table

TABLE 2. EXAMPLE OF KERNEL AND APPLICATION MODULES LISTED IN THE MODULE DESCRIPTION TABLE (MDT)

Kernel	Description	Application	Description
Timer	Timer handler	Temp sensor	Senses temperature
Surge	Routing algorithm	LED blink	led on/off app.
Ultrasound	Ultrasound driver	Packet #	Total sent packets

transmission is performed only with a great advantage that comes with a reasonable trade-off.

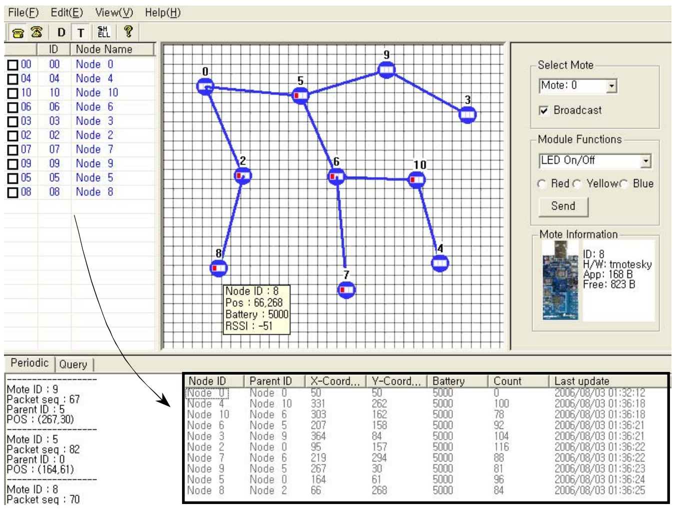
NCM provides a reconfigurable system that has two major purposes: (1) reconfigure the functionality of a kernel or application on sensor nodes online and (2) inject applications into the sensor hardware in run-time over the network. In many sensor network scenarios, applications require diverse functionality, such as sensing different attributes at different regions; furthermore, the role assignment may change over time. However, due to limited hardware resources, the sensor nodes cannot provide all functionalities together. To support such heterogeneity of a sensor network, NCM provides a component-based reconfigurable system by distributing required module code over the network. Modularizing the component and loading only the needed component is the core purpose of RETOS, and NCM applies this feature to our management system. As shown in Fig. 1, NCM has access to NAZ's MDT. It may load and unload user specified modules and also change attributes in the LLT through user queries. However, more overhead might incur for maintaining such a management module system compared to traditional sensor network applications management. Nevertheless, we consider the advantages of the trade-off greater. Through, modularization the user is provided with a high level configuration control, which abstracts away the knowledge of the low level functionalities.

III. EVALUATION

To show the feasibility of our approach, we implemented the component-based network management system using the Tmote Sky hardware running RETOS. We have first evaluated the functionality of the RMTool to see if it meets a certain level of QoS for successfully collecting network information. Then we measured the performance of the network configuration manager while concurrently running other user applications.

A. Functionality

The functional evaluation of NAZ and NCM is first



	ID	Node Name
00	00	Node 0
04	04	Node 4
10	10	Node 10
06	06	Node 6
03	03	Node 3
02	02	Node 2
07	07	Node 7
09	09	Node 9
05	05	Node 5
08	08	Node 8

```
Node ID : 8
Pos : 66,268
Battery : 5000
RSSI : -51
```

```
ID: 8
H/W: tmotesky
App: 168 B
Free: 823 B
```

```
-------------------
Mote ID : 9
Packet seq : 67
Parent ID : 5
POS : (267,30)
-------------------
Mote ID : 5
Packet seq : 82
Parent ID : 0
POS : (164,61)
-------------------
Mote ID : 8
Packet seq : 70
```

Node ID	Parent ID	X-Coord...	Y-Coord...	Battery	Count	Last update
Node 0	Node 0	50	50	5000	0	2006/08/03 01:32:12
Node 4	Node 10	331	262	5000	100	2006/08/03 01:36:18
Node 10	Node 6	303	162	5000	78	2006/08/03 01:36:18
Node 6	Node 5	207	158	5000	92	2006/08/03 01:36:21
Node 3	Node 9	364	84	5000	104	2006/08/03 01:36:21
Node 2	Node 0	95	157	5000	116	2006/08/03 01:36:22
Node 7	Node 6	219	294	5000	88	2006/08/03 01:36:22
Node 9	Node 5	267	30	5000	81	2006/08/03 01:36:23
Node 5	Node 0	164	61	5000	96	2006/08/03 01:36:24
Node 8	Node 2	66	268	5000	84	2006/08/03 01:36:25

Fig. 2. GUI of RMTool displaying the experiment setup

described. The experiment is setup with 10 Tmote Sky hardware; nine intermediate nodes and one sink node. The nodes are randomly deployed in a 20mx20m area. Each intermediate node runs the component-based network manager to collect neighbor information every 5 minutes, and relays it to the sink. A parent node is selected from the neighbor table for relaying data, which is the node with the largest RSSI value.

To validate if NAZ is actually collecting correct data from the network, we have implemented a GUI for displaying network information and issuing queries. The purpose of the GUI application is to efficiently diagnose the current network and have easy control over it, by abstracting away the low level functions.

Fig. 2 shows the display of our experiment setup. Here the user is provided with four panels. The left panel displays a list of nodes that currently take part in the network. The panel in the middle provides a graphic display of the sensor field and the sensor nodes placed at their localized coordinates. The ovals represent nodes currently active in the sensor field. The left topmost node is the sink, and the rest are intermediate nodes. The edges imply the routing path to and from neighbor nodes. If the RSSI drops between a pair of nodes, the color of the edge lightens in the display panel. The lighter the edge, the weaker is the connection between two nodes. The edge disappears when there is no communication between two nodes for some time, which indicates a disconnection. The right panel is an interface for issuing queries to nodes in the network. The type of commands and queries can be selected from the module function box, which lists currently available functions at the particular node. The last panel at the bottom outputs data collected at the sink, which are distinguished between periodic and query respondent messages. Also, RMTool manages a table that summarizes the most general attributes of active nodes, the boxed table in Fig. 2, and is viewed in a separate panel upon request.

NCM has control over the parameters collected by NAZ, and is able to request or even manipulate them by user request. This is to manually configure the network parameters to

TABLE 3. COMMAND INSTRUCTION TABLE

Instruction	Description	Instruction	Description
Shutdown	Shuts down node	StopApp	Stop application
Start	Starts kernel	StartApp	Starts application
Restart	Reboots node	RestartApp	Resets application
InsMod	Inserts module	InsApp	Inserts application
RmMod	Removes module	RmApp	Removes application

optimize the network status. RMTool implements a shell command type interface, which supports two different types of commands. One is a system type command and the other a query type command. The types of instructions for commands are shown in Table 3.

To communicate with the sink, we have implemented a command shell, through which the commands and queries are issued. The system type commands trigger system functions, such as rebooting a node or distributing and installing module code over the network. An example is shown below.

```
> cmd insmod MicroTimer 5
```

Here, 'cmd' implies that the shell command is a system function type command, with an 'insmod' instruction type that wishes to install a module called 'MicroTimer' at the specific node with ID '5'. The query type commands request specific information of a node in the network. Data shown in Table 1 are currently available to acquire. The queries can be issued as a one shot query or a query that collects the requested information over a certain period of time. An example is shown below, where the user wishes to acquire the neighbor table of a node with ID '5' each minute over 10 minutes.

```
> query neighbortable 5, 10, 1
```

B. Performance

Two sets of experiments are conducted to evaluate interactivity and response of the network, taking the number of hop-counts as the control factor. We have experimented up to a maximum of 10 hop-counts for each set.

A HELLO message is flooded every 20 seconds throughout the experiment. Every node that receives a HELLO message replies to the sink. The collected data of the HELLO message is used to update the visual display of the RMTool at the base station. We observed any increase of packet loss when running our component based network management system over this network architecture to see if the overall network performance is declined by the management system.

The first experiment was conducted to validate the interactivity of the network by measuring the time needed to acquire a valid response of the network for a query issued by the user. The query used in this experiment requests the neighbor table, which is provided by the NAZ module. Queries are issued to the user specified source node every 10 seconds. To relieve congestion, intermediate nodes relay queries and the acknowledged data with a 10 second period. As sown in Fig.3, the query reaction time constantly increases with the increase of the hop-count, which suggests that the user is provided with a constant expectation time of receiving a response from the network. Furthermore, it implies that data aggregation of the system does not incur considerable congestion, which may

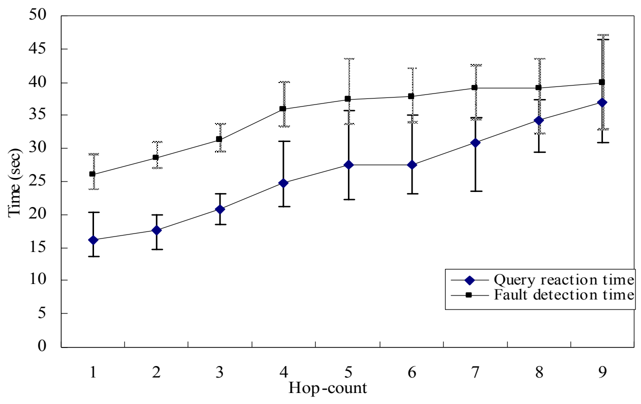


Fig. 3. Response time to a query and a node failure in the network.

cause packet loss or delay of packet reception. The variation of each measurement, the max and min values, show that links over longer hop-counts are weaker and unreliable than shorter hop-counts. The second experiment measured the time needed to detect a node failure in the network. With the same control factor and experiment setup, the battery of the source node, which is again the farthest away node, was taken out and started to measure the time needed by the sink to detect its inactive state. A node failure event is triggered by the neighbor nodes at the instance when the source node is removed from their neighbor tables. As shown in Fig. 3, the fault detection time increases with the sensor field size. However, the ratio of the detection time of a node failure is less than the query reaction time. Since the source node is included by all neighboring nodes, every neighbor node triggers a node failure event, which will be sent to the sink through different routing paths. Hence, data may possibly route through better routing paths, in terms of speed, consequently reducing the detection time of a node failure, compared to a single routing path.

IV. RELATED WORK

Some research has been studied in the literature of wireless sensor network management and maintenance. APTEEN [10] proposed a cluster-based hybrid routing protocol that gives an overall picture of the network at periodic intervals in an energy-efficient way. APTEEN suggested a comprehensive data retrieval method under the argument that a user should always be able to acquire data by using queries, while the data could be alerts about some faultiness in the network that has to be taken care of.

MOTE-VIEW [11] is an application, developed by Crossbow that provides a user interface, which displays the network status to the user. The application implements a simple configuration system that allows a user to set a node's reporting rate or turn its LED's on or off. However, development and significant network configurations are not supported. SNMS (Sensor Network Management System) [1] provides an open architecture, with no pre-set attribute format, for collecting network information, but restricts the mechanism to just monitoring the network. Zhao [2] proposed a network health scan to warn users about early system failures to save time in analyzing the problem at the wrong place. Also, it evaluates the network for weak radio communication regions to improve performance by selectively placing additional sensor nodes. Nevertheless, no advanced mechanism to reconfigure the network online is provided.

TinyCubus [12] introduces a cross-layer framework for heterogeneous sensor networks. There are three components, which are the data management framework, the cross-layer framework, and the configuration engine, respectively. The data management framework and the cross-layer framework support data acquisition from a heterogeneous network. The configuration engine in TinyCubus allows code to be distributed to the network for reconfiguration. TinyCubus is implemented using TinyOS [13], with reconfigured TinyOS code supporting runtime relocation of applications. However, TinyCubus does not invoke a mechanism to separately monitor application and network status. Since, an independent network specific management is not possible, which can cause confusion between application and the network topology.

Sympathy [4] is another network monitoring system proposed by N. Ramanathan. Sympathy is a prototype designed to detect and debug failures in pre- and post-deployment sensor networks using a health metric scheme to analyze prone data at the sink. Sympathy tracks down the root-cause of a detected failure, which is determined by examining and analyzing the data collected at the sink. To evaluate their proposed system the Sympathy debugging mechanism is integrated with the aforementioned network monitoring application, SNMS, for data collection and visual purposes. Here, the network is also not independently managed from the applications executed over the network.

V. Conclusion and Future Work

In this paper, we described a component-based management system for wireless sensor networks. RMTool is designed to provide up-to-date network information and enable the developer to mange the network in a controlled manner. The framework consists of the network analyzer for collecting and analyzing the network status, and the network configuration manager for manipulating dynamic attributes of a sensor network. The component-based implementation is to maximize reusability and optimization of the limited resource of the network. The RMTool implementation is proven to be resource efficient and interactive to user request and network events, which are essential factors for sensor network monitoring applications. Hence, the architecture provides a sophisticated and time-efficient developing environment for monitoring and managing sensor networks.

For future work, we consider defining diverse health metrics to detect complicated network failures more accurately and furthermore we plan to provide an online code debugging tool, which will be an IDE, where a node sends any application code error to the user such as line number of the error occurred within the code.

Acknowledgments

This work was supported in part by the National Research Laboratory (NRL) program of the Korea Science and Engineering Foundation (2005-01352) and the ITRC programs (MMRC) of IITA, Korea.

References

[1] G. Tolle and D. Culler, “Design of an Application-Cooperative Management System for Wireless Sensor Networks,” In Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN), Istanbul, Turkey, January, 2005.

[2] J. Zhao, R. Govindan, and D. Estring, “Sensor Network Tomography: Monitoring Wireless Sensor Networks,” Student Research Poster, ACM SIGCOMM 2001, pp. 64-64, San Diego, CA, August 2001.

[3] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, “An analysis of a large scale habitat monitoring application,” In Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 214-226, Baltimore, MD, 2004.

[4] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, “Sympathy for the Sensor Network Debugger,” In Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys '05), pp. 255-267, San Diego, CA, 2005.

[5] H. Kim and H. Cha, “Towards a Resilient Operating System for Wireless Sensor Networks,” In Proceedings of the 2006 USENIX Annual Technical Conference, pp. 103-108 MA, May 2006.

[6] H. Shin and H. Cha, "Supporting Application-Oriented Kernel Functionality for Resource Constrained Wireless Sensor Nodes," The 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006), Hong Kong, China, December 2006.

[7] S. Choi and H. Cha, “Application-Centric Networking Framework for Wireless Sensor Nodes” In Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS), July, 2006.

[8] Tmote Sky, http://www.moteiv.com

[9] J. Hui and D. Culler, “The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale,” In Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 81-94, Baltimore, MD, 2004.

[10] A. Manjeshwar and D. P. Agrawal, “APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks,” In Proceedings of the 2nd Int. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2002.

[11] Mote-View, http://www.xbow.com.

[12] P. J. Marr´on, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel. “TinyCubus: A Flexible and Adaptive Framework for Sensor Networks,” In Proceedings of the 2nd Europ. Workshop on Wireless Sensor Networks, pages 278–289, 2005.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System Architecture Directions for Network Sensors,” In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX), pp. 93-104, Cambridge, MA, November 2000.