The RETOS Operating System: Kernel, Tools and Applications

Hojung Cha, Sukwon Choi, Inuk Jung, Hyoseung Kim, Hyojeong Shin, Jaehyun Yoo, Chanmin Yoon
Department of Computer Science
Yonsei University
Seoul 120-749, Korea
{hjcha, sukwon, inukj, hskim, hjshin, jhyoo, cmyoon}@cs.yonsei.ac.kr

ABSTRACT
This demonstration shows the programming development suite of the RETOS operating system for sensor networks, which provides a robust and multithreaded programming interface to application programmers. We first demonstrate how to build the RETOS kernel on the TI MSP430, Atmel ATmega 128 and Chipcon CC2430 family of microcontrollers. The application or a kernel module is then compiled and disseminated, via wireless channel, to the target motes. The GUI-based RMon network management tool for RETOS is also demonstrated to monitor the networked sensors, and even to control the system's parameters or applications running on them via a remote shell. The system is demonstrated to run on a mixed set of MSP430, ATmega128 and CC2430-based motes. Overall, our demonstration will convince attendees of the programming convenience of developing sensor network applications using RETOS, which is, indeed, a mature and practical system that can be used to develop real-world applications.

Categories and Subject Descriptors
D.4.7 [**Operating systems**]: Organization and Design

General Terms
Design, Experimentation, Measurement, Performance

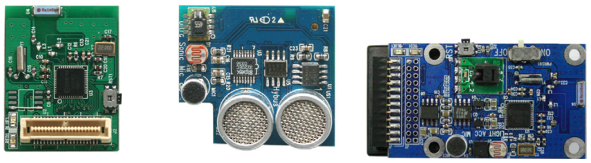
Keywords
Wireless Sensor Network, Operating Systems, Multithreading

1. THE RETOS OPERATING SYSTEM
The RETOS operating system, which has been developed specifically for wireless sensor networks, provides a multithreaded programming interface, system resiliency, kernel extensibility via loadable module, and network abstraction [1,2,3,4]. RETOS is a multithreaded operating system, hence it provides the common thread model of programming interface to application developers. The operating system provides software solutions to separate the kernel from user applications, and supports their robust execution on MMU-less hardware. The RETOS kernel can be dynamically reconfigured, via a loadable kernel framework, so that an application-optimized and resource-efficient kernel is constructed. RETOS' networking architecture is designed with a layering concept to provide WSN-specific network abstraction. RETOS is initially implemented on TI's MSP430-based motes such as Tmote Sky. The operating system currently supports ATmega128-based MicaZ, and the latest CC2430 SoC processor from Chipcon. RETOS is fully functional as a kernel, and a set of real applications has been developed to evaluate the system.

Copyright is held by the author/owner(s).
IPSN'07, April 25–27, 2007, Cambridge, Massachusetts, USA.
ACM 978-1-59593-638-7/07/0004.

2. DEMONSTRATION DESCRIPTION

2.1 Kernel Builds on Hardware Platforms
RETOS is designed to support various types of off-the-shelf microcontrollers and provides the same kernel operations on different hardware. To provide a portable kernel structure across different platforms, the architecture-independent part of the kernel is separated from the hardware-specific part. In our demonstration, we show the kernel build-up procedure for three different microcontrollers – MSP430, ATmega128 and CC2430. In particular, we show the CC2430-based H-mote and related sensor board, shown in Figure 1, in the demonstration.



(a) H-mote Type 1 (b) Sensor board (c) H-mote Type 2

Figure 1. CC2430-based H-mote and sensor board

2.2 Application Code Validation
The microcontroller used for a sensor node typically has a single address space due to its lack of memory management hardware. The kernel and applications usually exist in the same address space, hence the operating system should provide software assistance for error-free applications. The RETOS kernel detects harmful attempts by applications on system safety, and terminates them appropriately. RETOS ensures system resilience with two techniques: dual mode operation and application code checking. Dual mode operation logically separates the kernel and the user execution area. Application code checking evaluates the compiled code's validity via static analysis, and the application code's run-time behavior via a dynamic check. We demonstrate RETOS' code checking process based on the MSP430-based Tmote Sky sensor node. An errant application program is deliberately prepared, and checked at both compile time and runtime.

2.3 Application/Module Loading with ReCode

Once an application is compiled, the binary is loaded into the hardware and runs as a RETOS application in the user mode. Due to the multithreaded programming model, even multiple applications can run concurrently on the same hardware. For kernel reconfigurability, RETOS also provides a mechanism to load the kernel module dynamically in the kernel at run time without rebooting the system. To disseminate the binary code, either application's or the kernel module's, RETOS is equipped with a wireless code dissemination mechanism called ReCode (RETOS Code Dissemination Protocol), for both homogeneous and heterogeneous sensor networks. The code dissemination mechanism is based on epidemic approaches and is designed to compare and exchange application version information in a multi-hop fashion by implicitly connecting nodes based on the application of interest. A unique feature of ReCode is to support the code update mechanism in heterogeneous networks, where multiple and concurrent applications run in a single sensor network. To support code dissemination on this network, ReCode guarantees that the connectivity between sensor nodes, which services the applications, covers the whole network. We define a group of sensor nodes as the code delivery tree, which maintains connections between neighboring groups such that connectivity spans the whole network. In our demonstration, we disseminate an application code and a kernel module using ReCode.

2.4 RMon: The Network Management System

The RETOS development suite provides a network management system, called RMon, to support network management and monitoring of sensor networks deployed in the field. RMon, which runs as a kernel module, assures users of the network's functionality and gives developers control over the network while running user applications over it. From an application developer's point of view, RMon can be used to show an overall view of the applications' behavior on the network, while concurrently monitoring the network's health. A GUI tool is also provided to interactively operate with RMon to visually display data collected by RMon via a shell command interface to allow an administrator to issue commands to network nodes. Figure 2 shows a snapshot of the RMon GUI tool for the deployed sensor nodes. We demonstrate the usage of RMon, which runs on a mixed set of Tmote Sky, MicaZ and CC2430-based H-mote, as shown in Figure 3. The network's health status is displayed in real time, and an individual mote, as well as the entire network, is controlled by specific shell commands.

3. ACKNOWLEDGMENTS

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the National Research Lab. The program was funded by the Ministry of Science and Technology (No. M10500000059-06J0000-05910), and the ITRC (Information Technology Research Center) programs of the IITA (Institute of Information Technology Advancement) (IITA-2006-C1090-0603-0015).

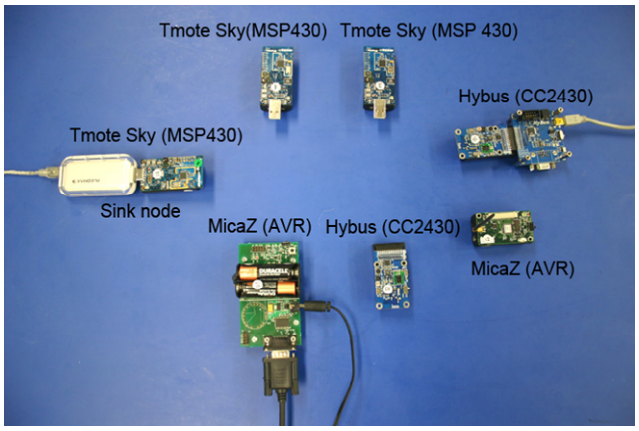


Figure 2. Mixed set of hardware deployment

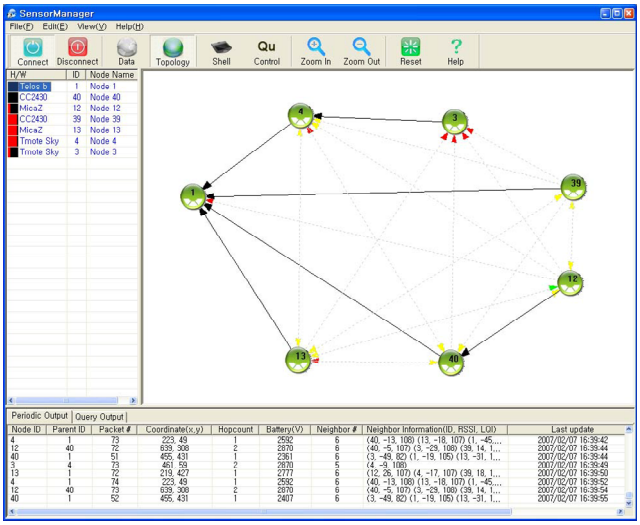


Figure 3. RMon snapshot for Figure 2 deployment

4. REFERENCES

[1] H. Kim, H. Cha, "Towards a Resilient Operating System for Wireless Sensor Networks," *Proc. of the 2006 USENIX Annual Technical Conference*, Boston, Massachusetts, June 2006.

[2] H. Kim, H. Cha, "Multithreading Optimization Techniques for Sensor Network Operating Systems," *Proc. of the 4th European conference on Wireless Sensor Networks (EWSN'07)*, Delft, Netherlands, January 2007.

[3] H. Shin, H. Cha, "Supporting Application-Oriented Kernel Functionality for Resource Constrained Wireless Sensor Nodes," *Proc. of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN'06)*, Hong Kong, China, December 2006.

[4] H. Cha, et al, "RETOS: Resilient, Expandable, and Threaded Operating System for Wireless Sensor Networks," *Proc. of the Information Processing for Sensor Networks (IPSN'07)*, Boston, Massachusetts, April 2007.