MobiCom 2009 Poster: Multi-Mode User-Centric Design of Wireless Sensor Networks for Long-Term Monitoring

Ming Xia[a] **Yabo Dong**[b] **Wenyuan Xu**[c] **Dongming Lu**[b] **Xiangyang Li**[d]
xiaming@zjut.edu.cn *dongyb@zju.edu.cn* *wyxu@cse.sc.edu* *ldm@zju.edu.cn* *xli@cs.iit.edu*
[a]College of Computer Science and Technology, Zhejiang University of Technology, Zhejiang, China
[b]College of Computer Science and Technology, Zhejiang University, Zhejiang, China
[c]Department of Computer Science and Engineering, University of South Carolina, South Carolina, USA
[d]Department of Computer Science, Illinois Institute of Technology, Illinois, USA

I. Introduction

Real-world, long-running wireless sensor networks (WSNs) require intense user intervention during the development, testing, deployment, and maintenance stages. The widely-adopted *network-centric* design principles [1] primarily focus on assuring efficient network resource usage and reliable data delivery, but overlook the underlying heavy burden that users and developers face throughout the entire lifetime of WSNs. To alleviate this burden and to facilitate various tasks real users may encounter, we propose to take a *user-centric* viewpoint to design the entire systems.

To address challenges that arise because of the heavy human intervention, we designed a general Multi-mode user-CentriC (MC^2) framework that can, with simple user input, tailor itself to meet realistic user requirements. We have devised the MC^2 software architecture and designed a sensor platform to support it. Furthermore, we have validated our MC^2 software and sensor platform in a microclimate monitoring system deployed at a wild land world heritage site, Mogao Grottoes [3]. In our current system, 241 data sensors have been deployed, and the network has been running stably for over one year. The MC^2 framework has shown to shorten the time needed for network deployment, and has made the maintenance of the sensor network manageable by non-technical staff.

II. MC^2 Framework

A typical life cycle of a WSN consists of four stages: *testing*, *deployment*, *network operation*, and *maintenance*. Except for the network operation stage, all other stages involve user intervention. To reduce the workload of developers or users in those stages, MC^2 framework contains several utilities.

Testing. This stage involves verifying the correctness of hardware and uploading the customized software to all nodes prior to deployment. MC^2 supports two utilities, *Local Hardware Verification*, which automatically checks the correctness of hardware and returns check results to users, and *Wireless Code Distribution*, which propagates code through wireless communication to avoid arduous mechanical jobs.

Deployment. The main tasks in the deployment stage include sensor calibrations to ensure accurate readings, sensor node placements to assure good network connectivity, and burn-in to detect any early system failures. The supported utilities include: *Online Calibration*, which calibrates sensors without the need of dismounting the sensing board; *Wireless Configuration*, which sets up node parameters (e.g., duty cycle) without the need of wire connection; *Fast Deployment Time Validation*, which tests the communication quality intensively within a short period of time.

Maintenance. To assist fault diagnosis and periodical maintenance, MC^2 supports two utilities, *Remote Fault Detection*, which can identify faulty nodes remotely, and *On-Site Fault Diagnosis*, which enables a user to find out the root causes of a fault in the field.

To integrate all utilities seamlessly without sacrificing network performance, MC^2 framework adopts the concept of modes, and provides both hardware and software support.

II.A. MC^2 Software

The key idea of MC^2 software is to provide an abstracted container, which we call *mode*, to group utilities with similar behaviors, and a general *Mode Management Component (MMC)* to control mode switching, as shown in Figure 1. As such, unrelated utilities are fully decoupled to avoid interference, and we can maximize individual run-time efficiency without compromising the performance of other utilities.

II.A.1. Modes

Factors that can affect behaviors of each utility or task include duty cycle and boot-related characteris-

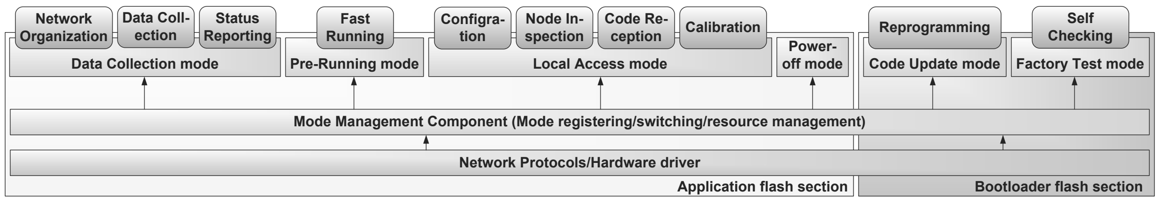


Figure 1: The high-level software architecture of MC^2 framework.

tics. Based on the required duty cycle and boot-related features, we group the utilities into five modes: *Data Collection*, *Pre-Running*, *Local Access*, *Code Update*, and *Factory Test*. Under each mode, we have implemented several modules with each maps to one utility. While modes are exclusive to each other, one or a few modules that belong to the same mode can run simultaneously. We summarize the relationship between utilities, modes and stages in Table 1.

The *Data Collection* mode was designed for the network operation stage, where a sensor node typically runs at a low duty cycle to conserve energy. This mode includes three modules: *Network Organization*, *Data Collection*, and *Status Reporting*. The Network Organization module handles routing, time synchronization, and other network related operations. The Data Collection module samples and transmits data to the base station. The Status Reporting module was designed for remote fault detection and diagnosis, and it periodically reports the node status to the base station.

The *Pre-Running* mode was designed for the task of fast deployment time validation. It contains one module called *Fast Running*. This mode is essentially a high duty cycle version of the Data Collection mode except that it can provide a more detailed status report to assist deployment.

The *Local Access* mode aims to incorporate all utilities that require to fetch/feed information from/to a node directly (e.g., the on-site fault diagnosis). A node in Local Access mode will remain awake in order to respond to users' queries or to react to users' instructions without any delay. The Local Access mode has four modules : *Code Reception*, *Configuration*, *Calibration*, and *Node Inspection*. Each module handles one of the following utilities respectively, Wireless Code Reception, Wireless Configuration, Online Calibration, and On-Site Fault Diagnosis.

The *Code Update* mode is a special mode designed to complete the wireless code distribution process. Once the entire codes are successfully received and stored in external flash memory during Local Access mode, the node will reboot and enter Code Update mode. This mode has one module, *Reprogramming*, which replaces the current program stored in the application flash section with the newly received code.

The *Factory Test* mode is meant for hardware verification during the testing stage, and it includes one module, *Self Checking* module.

Finally, *Power-off* mode is a special mode that does not include any utility, but provides a useful state for transportation, under which a node stays in deep sleep to reduce unnecessary power consumption.

II.A.2. *Mode Management Component*

Mode Management Component (MMC) is responsible to switch the working mode of MC^2 framework according to users' input. A user can instruct the node to enter one of the modes in three ways: resetting, button switching, and wireless switching. Both Code Update mode and Factory Test mode can be entered by resetting a node, as they are stored in the bootloader flash section. Additionally, a user can cycle through the rest of the four modes that are stored in the application flash section either by button switching, whereby a user presses the button located on the node enclosure, or wireless switching, whereby mode switching commands are issued by the SensorMate.

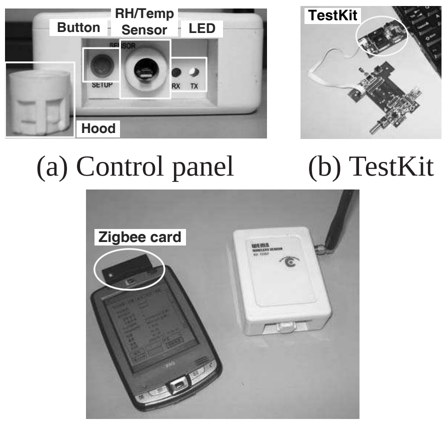
II.B. MC^2 Hardware

MC^2 hardware provides a user-friendly interface to access nodes, and it includes customized sensor nodes, a TestKit, and SensorMate.

The sensor nodes contain a control panel on the weatherproof enclosure and a structure that can fa-

Mode	**Utility**	**Stage**
Data Collection	Monitoring	Network operation
	Remote fault detection	Network operation
Pre-Running	Fast deployment time validation	Deployment
Local Access	Wireless code distribution	Testing
	Wireless configuration	Deployment
	Online calibration	Deployment Maintenance
	On-site fault diagnosis	Maintenance
Code Update	Wireless code distribution	Testing
Factory Test	Local hardware verification	Testing
Power-off	N/A	Prior to deployment

Table 1: The relationship between modes, utilities, and stages.



(a) Control panel (b) TestKit

(c) SensorMate (left) and sensor node (right)

Figure 2: Customized MC^2 hardware

cilitate easy sensor replacements, as shown in Figure 2. The control panel contains a button to switch the working mode locally, and two LEDs for displaying the results of mode switching and data transmission (succeeded/failed). Together, the mode switching button and LEDs facilitate users to maneuver through MC^2 working modes without opening the enclosures.

SensorMate and TestKit were designed for debugging and deployment. SensorMate was implemented on a HP iPAQ PDA with a Zigbee CF card to enable the direct communication to nodes. The TestKit consists of a customized 12 pin wire and a converting board with a USB port for communication between a sensor node and a laptop in Factory Test mode.

III. Case Study: The Microclimate Monitoring at Mogao Grottoes

We have employed the proposed framework in a long-term system deployed at Mogao Grottoes, which contains 492 caves with more than 45,000m^2 of murals and 2,415 painted sculptures. Now the historic relics of Mogao Grottoes are deteriorating due to the inappropriate microclimate in caves [2]. Our project aims to set up a long term monitoring system to collect real-time microclimate measurements, including temperature, humidity and CO_2 density, for scientific studies and to enforce conservation policies.

We have conducted two round deployments at Mogao Grottoes. In our first round exploration, we adopted the network-centric design principle, and in the second round, we used our multi-mode user-centric design. Although the deployment scale of the second round is six times larger than that of the first round, we managed to finish the deployment in almost the same amount of time, showing that MC^2 framework can boost testing and deployment efficiency.

Testing. The largest amount of time saved during this stage is because of wireless software programming. In our first round exploration, we spent 3 hours to program nodes one by one. In our second round, programming all 300 nodes took only 10 minutes.

Deployment. The online calibration feature enables us to calibrate a CO_2 sensor without taking it off the node, which was a mandatory hassle in the first round exploration. Thus, we only spent about 5 hours in calibrating 31 CO_2 sensor nodes. Additionally, the Fast Deployment Time Validation and Wireless Configuration utilities have greatly speeded up the sensor placement and burn-in process, while resulting in higher quality communication links. Such improvement is the direct payoff of Pre-Running mode, since we are able to sample larger number of packets in a shorter period of time. In total, we only spent about 6 days in deploying 241 nodes, while it took us 5 days to deploy only 40 nodes without the help of MC^2 framework.

Maintenance. We have designed a graphical module running at the base station to display node status reported periodically by the Status Reporting module. It can identify a faulty node within 15 minutes after a failure occurs. Upon receiving the node fault notification at the base station, a user can visit the site and to perform on-site diagnosis and fault repair. In particular, the user puts the node into Local Access mode to retrieve node status information via SensorMate. Because of the non-delay communication pattern in Local Access mode, the on-site fault diagnosis can be completed quickly, typically within one minute. Similarly, the user can also rely on SensorMate to quickly re-calibrate the sensor, reprogram and reconfigure the node. In summary, the fault detection and diagnosing tools have made the network maintenance manageable by non-technical staff (in our case historians).

References

[1] D. Estrin, R. Govindan, J. Heidermann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of MobiCom*, 1999.

[2] Y. C. SHI and J. Zhang. The dunhuang caves' main diseases and precautions against them. *Northwestern Seismological Journal*, 19(2):81–87, 1997.

[3] M. Xia, Y. Dong, D. Lu, P. Xue, and G. Liu. A wireless sensor system for long-term microclimate monitoring in wildland cultural heritage sites. In *proceedings of ISPA*, 2008.