Demo Abstract: SenPro: Concurrent System Profiling for Wireless Sensor Networks

Chun-Yu Lin\textsuperscript{1}, Huang-Chen Lee\textsuperscript{1}, Wei-Hsuan Lee\textsuperscript{1}, Chen-Lung Chan\textsuperscript{2}, Chung-Ta King\textsuperscript{1}
\textsuperscript{1}Department of Computer Science, National Tsing Hua University
\textsuperscript{2}Computer and Communication Research Center
101, Section 2, Kuang-Fu Road, Hsinchu, Taiwan
chunyu@cs.nthu.edu.tw, huclee@mx.nthu.edu.tw, mr9562555@cs.nthu.edu.tw, clchan@mx.nthu.edu.tw, king@cs.nthu.edu.tw

ABSTRACT
In this demo, we present SenPro, a profiling tool that profiles the run-time behaviors of a deployed wireless sensor network (WSN). SenPro runs concurrently with the WSN application and provides information of system execution in a minimally intrusive manner, without requiring any modification to the application being observed. We demonstrate, for a broad class of data gathering applications, how SenPro helps the designer to understand the application behavior by collecting and analyzing the metrics at the centralized sink.

Categories and Subject Descriptors
C.4 [Computer Systems Organization]: Performance of System—Design Studies

General Terms
Measurement, Design

Keywords
Wireless Sensor Networks, TinyOS, Profiling

1. INTRODUCTION
Developing wireless sensor network (WSN) applications in dynamic and distributed environments is an error-prone and cumbersome process. Many sensor networks fail to meet application requirements after deployed even though they have been tested in the lab a priori [1]. The reasons can be manifold, including hardware faults (e.g. short circuits and unreliable components), software bugs (e.g. incorrect program logic and race conditions), and network problems (e.g. radio interference and packet collision). For these reasons, ensuring reliable software operations in sensor networks is extremely critical and challenging. A tool to profile the run-time behaviors of the sensor application is very much needed, which can make application testing, analysis, evaluation, and validation easy.

A number of sensor network simulators [2][3][4][5] have been developed to provide testing and evaluation support for individual devices and/or the complete network. Unfortunately, they are only good for testing or evaluation prior to deployment. However unpredictable environmental conditions often manifest at run-time through complex triggering mechanisms, which are difficult to capture at development time. It also makes verification of correct operations of the network very difficult, which in turn increases the chance that buggy code survives the post-deployment testing process and makes its way into production code.

In this demo, we present SenPro, a lightweight profiling tool for wireless sensor networks. SenPro runs above the time-sharing scheduling mechanism that allows SenPro to concurrently execute with the application code. Due to hardware and network limitations, SenPro must collect the information that allows the most precise and meaningful evaluation and observation of the WSN application. Currently, it collects four types of profiling information: power state, lookup function, memory usage, and network condition, which present programmers a high-level view of a deployed sensor network application.

2. TIME-SHARING MECHANISM
SenPro is implemented based on TinyOS [6], the de facto operating system in the research community of WSN. Ideally, SenPro is enclosed in a timer event that periodically collects and delivers the profile information to the sink. However, TinyOS reacts to sensor and timer events in a FIFO fashion. It may delay the execution of SenPro or even block SenPro because of some stalling application tasks.

Therefore, we have to redesign the task scheduler of TinyOS to allow SenPro to preempt the execution of the application code. The basic idea is to switch the executions of the application code and SenPro alternatively so that they run concurrently, a mechanism which will be referred to as concurrent profiling. Nevertheless, it is easy to implement concurrent profiling on modern multithreading operating systems, but when considering the resource-constrained sensor nodes, multithread programming is not feasible. Furthermore, we want the sensor node to respond to the sink when the profiler takes over the control of the CPU, even when some faults in the application code make the processor to stall. Our implementation is introduced next.

3. IMPLEMENTATION
Figure 1 shows an overview of the software architecture of SenPro. The profiling process has four stages and the details are described in the following:
A summary of the information profiled by our system is shown in Table 1. Even though other information could be collected easily, such as the phases of execution in TinyOS, SenPro only profiles the above information on each node. This is primarily a consideration of the computation ability of the sensor nodes and a goal to minimize disturbance to applications.

4. CONCLUSION

Human errors, hardware failures, or problems of the deployed environment may affect the system performance of a deployed WSN or even reduce its lifetime. Ensuring proper operations of a deployed sensor network is a long and laborious process that cannot be answered by using simulation or emulation tools alone. It should be done at run-time. In this paper, we have developed SenPro, a tool for profiling performance data and observing application behavior of a deployed WSN. In the demo, we will use the MicaZ mote [7], developed by Crossbow Inc., to demonstrate the followings:

- Use SenPro to profile the behavior of Oscilloscope, which is a famous demo application in TinyOS, even when a fault happens in a sensor node.
- Show the front-end application that displays the real-time behaviors of the sensor network.

5. REFERENCES


425