Demo Abstract: LiteOS - A Lightweight Operating System for C++ Software Development in Sensor Networks

Qing Cao
Department of Computer Science
University of Illinois at Urbana-Champaign
qcao2@cs.uiuc.edu

Tarek Abdelzaher
Department of Computer Science
University of Illinois at Urbana-Champaign
zaher@cs.uiuc.edu

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Real-time systems and embedded systems*

General Terms

Design

Keywords

LiteOS, LiteC, OpenSC

1 Introduction

Wireless sensor networks offer great potential for a myriad of new applications. Unfortunately, software development for wireless sensor networks is quite demanding compared to the development of other commercial software. Current sensor networks are often programmed by those who have spent considerable time learning about the field. In contrast, to facilitate widespread use of sensor networks, software development of sensor network applications should be accessible to anyone with common programming skills, including those who are not necessarily computer scientists. Application experts in biology, agronomy, and medicine should be able to customize and maintain sensor networks as a form of field instruments. Third party software providers should be able to write code for sensor network applications without steep learning curves. Consumers should be able to buy sensor network products off-the-shelf and download or write software easily to serve their needs.

To achieve this goal, we are building a light-weight object oriented operating system, LiteOS, and programming environment, LiteC, based on C++. Current sensor network code is usually written in NesC [2] and runs on TinyOS [1]. High-level programming solutions to sensor networks, such as TinyDB [3], are designed to facilitate certain application scenarios, but are not aimed for general software development.

The LiteOS operating system and the LiteC compiler (referred to as LiteOS in general) at the University of Illinois provide support for development and execution of C++ programs on generic wireless sensor network platforms. LiteOS contains an API library named the Open Sensor Library (OpenSC), not unlike commercialized APIs such as MFC or OpenGL. It provides the basic functionality modules that the programmer can use for communication, I/O, and common program functions. The LiteOS programming environment provides mechanisms for translating C++ programs built on top of OpenSC into code binaries on different hardware platforms. It implements resource allocation and process scheduling, among other tasks, to run user programs on these platforms.

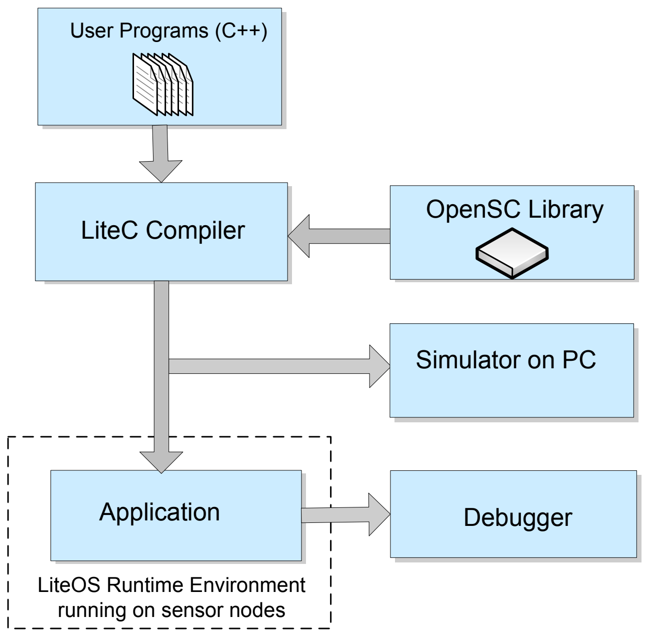

Figure 1. LiteOS Architecture

2 LiteOS Architecture

LiteOS consists of the following three components, shown in Figure 1: (i) The LiteC compiler, which translates user programs written in C++ into binary code, (ii) OpenSC, which provides an API library (implemented as C++ files) to facilitate software development, and (iii) the LiteOS runtime environment, which implements process scheduling and resource allocation. LiteOS also provides two auxiliary tools: an interactive debugger that allows users to see the internal states of running programs using GDB-like commands and an application simulator that simulates sensor network programs on a PC.

2.1 LiteC compiler

The compiler translates user C++ programs into binaries to run on different hardware platforms, such as MicaZ and Tmote. We choose C++ because of its widely known syntax and its support for objects. To run C++ programs on even the most resource-restricted sensor nodes, we only support a subset of C++ features that is deemed necessary. We keep overhead low by disallowing most dynamic features of C++, hence, allowing compile-time optimizations to efficiently reduce the code footprint. We retain the powerful side of C++ by providing support for its primary features, such as the concept of classes, which is necessary for object-based high-level abstractions, and inheritance, which facilitates object declarations and derivations in OpenSC. We refer to this subset of C++ language features as LiteC.

2.2 Open Sensor Classes (OpenSC)

OpenSC is an API library that provides classes to facilitate user program development. OpenSC consists of classes written in C++ that serve common application needs, ranging from routing protocols to time synchronization. OpenSC maintains well-defined interfaces between different classes by allowing only public member functions of one class to be seen by other classes. Classes in OpenSC can be either hardware-dependent or hardware-independent. The LiteC compiler is responsible for the linkage and compilation of classes based for their hardware platform targets.

2.3 LiteOS Runtime Environment

Once programs are translated into binaries, the LiteOS runtime environment allocates resources for these binaries, and executes them on the underlying sensor node hardware. Based on the hardware resources provided, this runtime environment uses different process scheduling and configuration policies: future sensor network platforms will likely provide more hardware resources to make more advanced process scheduling and configurations feasible.

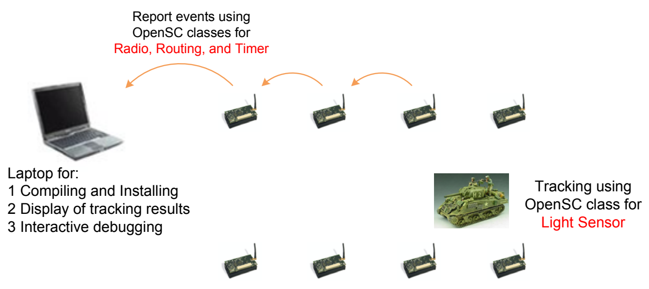

Figure 2. Demo Scenario

3 Demonstration Scenario

For an indoor demonstration, we use the following hardware: a laptop connected with a MicaZ node serving as the base station, twenty-four MicaZ nodes placed in a 4×6 grid for tracking, and a toy tank serving as an object detected by its shadow. The light sensor on each node is used to detect the target. Tracking results are shown on the laptop by a Java GUI program. The demo scenario is shown in Figure 2.

The demo works as follows. First, while the nodes may be physically placed near each other, each node is only allowed to communicate with its four neighbors in the grid. Therefore, nodes form a multi-hop communication network. Second, each node knows its position by its node ID. During the demo, once a node detects a target (by detecting a dramatic light sensor reading change), it reports to the base station through multi-hop communication. Once the base station receives an event report, it shows it on the Java GUI.

On the software side, all programs on motes are developed using C++, compiled using the LiteC compiler, and run under the LiteOS runtime environment. The tracking application uses classes exported by OpenSC to simplify its development. Namely, the tracking module uses the light sensor class that encapsulates interactions with ADC hardware, and the reporting module uses radio, routing and timer classes to deliver packets to the base station.

During the demo, we also show that it is feasible to access the internal program states of the MicaZ motes using the interactive debugging tool mentioned earlier. The tool will display realtime variable values and program status, such as the number of messages that one particular node has delivered and the last light sensor reading on a certain node. The interactive debugging tool features a GDB-like interface, making it easy to use.

We are able to illustrate three points of the LiteOS/LiteC platform in this demo. First, we show that running C++ codes on sensor networks is feasible. Second, by demonstrating the tracking scenario, we demonstrate library support for common sensor network functions. Third, we demonstrate that the interactive debugging tool is effective to help users keep track of program state.

4 References

[1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *Proc. of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–104, 2000.

[2] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*, 2000.

[3] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *SIGMOD*, June 2003.