2006-1323: HANDS-ON PROJECTS IN WIRELESS AND MOBILE COMPUTER NETWORK COURSES

Xiannong Meng, Bucknell University

XIANNONG MENG is an Associate Professor in the Department of Computer Science at Bucknell University in Lewisburg, Pennsylvania, U.S.A. His research interests include distributed computing, data mining, intelligent Web search, operating systems, and computer networks. He received his Ph.D. in Computer Science from Worcester Polytechnic Institute in Worcester, Massachusetts, U.S.A.

Luiz Perrone, Bucknell University

LUIZ FELIPE PERRONE is Assistant Professor of Computer Science, at Bucknell University. He has been developing an elective in Computer Security since the spring of 2003. His research on the application of computer simulation to the study of the security properties of wireless networks is supported by the Office for Domestic Preparedness, U.S. Department of Homeland Security, via the Institute for Security Technology Studies at Dartmouth College

Maurice Aburdene, Bucknell University

MAURICE F. ABURDENE is the T. Jefferson Miers Professor of Electrical Engineering and Professor of Computer Science at Bucknell University. He has taught at Swarthmore College, the State University of New York at Oswego, and the University of Connecticut. His research areas include, parallel algorithms, simulation of dynamic systems, distributed algorithms, computer communication networks, control systems, computer-assisted laboratories, and signal processing.

Hands-on Projects in Wireless and Mobile Computer Network Courses

Abstract

Wireless and mobile computer network courses are becoming increasingly popular in universities and colleges across the nation. This paper collects and analyzes both hardware and software components that are already being used for hands-on exercises in wireless and mobile computer network courses. Most often these hands-on exercises include both programming and laboratory assignments. In traditional wire-based computer network courses, students learn the layered protocols from physical layers such as Ethernet, to network and transport layers such as TCP/IP, and to application layers such as SMTP and HTTP. In learning these concepts and protocols, students have ample opportunities to program at different layers with programming languages such as Java, C++ and C, and can observe clearly how networked computers communicate. In wireless networking environments, for students who wish to learn the basics of programming wireless and mobile networking, it is hard to find standard and well defined programming interfaces and platforms. We have searched and collected information from the Internet about the options available, both in hardware and in software, that have been recently used as laboratory assignments and semester projects in undergraduate and graduate courses. A summary of our findings is presented in this paper.

Introduction

Many universities have used different protocols and products for hands-on experiences in wireless and mobile network courses. The protocols and standards include IEEE 802.11 (a, b, g or simply WiFi), Bluetooth, IEEE 802.15.4 and Zigbee, sensor networks based on the standard Berkeley "mote" platform implemented in products such as WeC, Rene, Dot, MICA, and Telos. Many of the products use TinyOS, a small operating system targeted for minimum hardware.

We present a summary of hands-on laboratory exercises we found on the Internet. In Vassar's CMPU-395, students are asked to implement a simplified version of the IEEE 802.11 MAC protocol emulation on Cybiko devices. Students in Harvard's CS263 were required to implement a simple multi-hop data collection protocol on a kit of four Telos Motes to be integrated and tested in their MoteLab which is a 30-node sensor network. One project in Dartmouth's COSC 78 required students to implement a *location-aware* wireless application using the PlaceLab device positioning framework. The hands-on component of the University of Virginia's CS451/651 uses TinyOS and Berkeley MICA2 Motes to develop applications using light (photo) and sound (audio) sensors.

In the rest of the paper, we focus on specific approaches taken by different universities. Section 2 discusses the project from Vassar College. Section 3 concentrates on the implementation of a simple multi-hop data collection protocol in Harvard University's CS263. Section 4 presents a project used at Dartmouth in COSC 78, where students implement a *location-aware* wireless application using the PlaceLab device positioning framework. Section 5 outlines a sequence of lab exercises in University of Virginia's CS 451/651 course, followed by a summary in Section 6.

The MAC Layer Emulation Project

Vassar University offered a wireless networks course[14] in 2002 in which a hands-on project based on the Cybiko,[2,8] a hand-held game device. It has a software development kit (SDK) that the developers can use to write programs in a C-like programming language on personal computers using Windows or Linux. Once the software is developed and tested, it can be downloaded to the Cybiko device. Other software and games can be downloaded to the device using CyberLoad software. The project implements a simplified version of the 802.11 MAC layer on the Cybiko.[15] The physical layer emulation, equivalent to that of the 802.11 standard, is given to the students, who are then asked to implement a set of well-defined MAC layer interfaces on top of the physical layer interface, so that the application programs may use the set of MAC interface directly. Figure 1 illustrates the concept of the project.

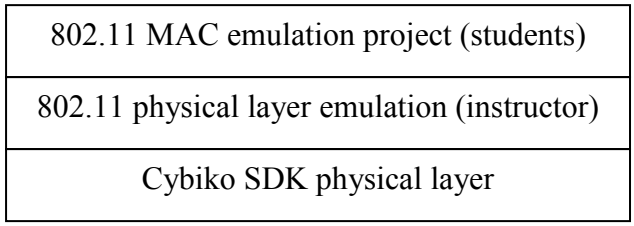

Figure 1. Layers of the Cybiko Project

The MAC layer is simplified so that each device can communicate directly with any other devices in the range using a 4-bit direct destination ID. The other simplified requirements include: no fragmentation or reassembly, no power management, no base station or distribution system, and no WEP (security) support. According to the description of the project,[15] the set of emulated RF-based physical layer interface are as follows:

- void RF_Init (struct module_t * main, char * name)

This routine is called before any other RF physical layer routines can be used. It is passed as a pointer to the application's main module and the name of the application.

- void RF_Service()

This must be called periodically to keep the physical layer and the MAC layer synchronized.

- bool RF_Handler(struct Message * ptr_message)

The layer above must pass events to this function so that the RF layer has a chance to process incoming messages. It returns TRUE if the event was handled by the RF layer.

- int RF_Tx(char *data, int size)

This function takes data from the layer above and broadcasts it to all participating Cybikos.

There is no corresponding function for receiving data. Any incoming data is represented in a Cybiko Message format. MSG_RF_ARRIVAL indicates the fact that a new message has arrived and is stored in a buffer. The sender can be determined via Message_get_sender_id() and the data size can be found with Buffer_get_size().

- bool RF_InUse()

This function returns TRUE if the shared channel is in use.

- clock_t RFIdleTime()

Returns the number of milliseconds since channel was last used.

This set of emulated physical layer interface is written using the Cybiko message system.

Given this set of physical layer interface, students are asked to implement a subset of the IEEE 802.11 MAC layer interface to provide services for the layer above. The exact implementations could vary, but the programming interfaces remain the same, following the specification.

- void MAC_Init(struct module_t * main, char * name)
- void MAC_Service()
- bool MAC_Handler(struct Message * ptr_message)
- int MAC_Send(MACaddr dest, char *data, int size)
- int MAC_Recv(MACaddr *source, MACaddr *dest, char *data, int bufSize)
-

The purpose of these functions is to provide an interface between the two layers: the physical RF layer and the layer above (be it an application or another intermediate layer). The meanings of the parameters are similar to those at the RF physical layer except the MACaddr. The MAC_Send and MAC_Recv send and receive data following the CSMA/CA protocol.

The Cybiko device and its programming interfaces (SDK) are no longer supported by the manufacturer. However, there appears to be interest among the game developers for the device, as demonstrated by searching Cybiko related information on the Web.

A Multi-hop Data Collection Protocol Implementation Project

While Vassar's project was based on IEEE 802.11 wireless network protocol (a.k.a. WiFi), one of the hands-on projects[6] in Harvard's CS263[5] is based on *TmoteSky*[7] that runs TinyOS[10]. TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture that enables rapid innovation and implementation while minimizing code size, as required by the severe memory constraints inherent in sensor networks. There are several manufacturers of Berkeley motes technology. *TmoteSky* is a mote platform for extremely low power, high data-rate, sensor network applications, designed with the dual goal of fault tolerance and development ease. *TmoteSky* supports programming and data collection through the use of USB port. According to its quick start guideline,[11] a collection of Tmote Tools and COM drivers are provided for the Windows platform. These tools include: Cygwin,[1] which is a Linux-like working shell for Windows that provides the basic working environment for TinyOS and Tmotes; Java, which is used for tools that interface between motes and the computer (PC); various drivers for USB and serial ports; TinyOS source code that will be uploaded to the motes; and MSPGCC, which is a compiler that generates code for TinyOS applications. With the tool set installed, programmers can develop applications for Tmotes.

In this project,[6] students are asked to design and implement a multi-hop communication protocol on the mote kit to collect data by a group of motes, each of which then reports back to its parent. Using a spanning tree algorithm, the information from all the sensors on the network can eventually reach a pre-specified base node.

A Location-aware Client/Server Project

Dartmouth's *Computer Network* course[3] offered in spring of 2005 asked its students to implement a location-aware client/server project.[4] The goal of the project is to build a client and server that will enable a location-aware game. Two players (wireless laptop clients) will participate in the game; each player will hunt the other. When the players are located near to each other, they will be able to shoot at each other. The players are represented by two lap-top computers running on a wireless network. The location information of the computers is provided by the PlaceLab[9] software, which provides a low-cost, easy-to-use device positioning for location-enhanced computing applications. Each client detects its current location as determined by PlaceLab. The client periodically transmits its location information to the server, which periodically broadcasts all of clients' locations to all clients. When the clients are within the specified range, they can take actions against each other (e.g. shooting). The key piece of information here in the project is that the PlaceLab running on a wireless-network enabled lap-top can supply the location information of the machine. PlaceLab also provides a Java programming interface through its J9 JVM (Java Virtual Machine).

Wireless Sensor Network Project Using MICA2 Motes and TinyOS

University of Virginia's *CS451/651 – Wireless Sensor Networks*[16] was offered most recently in the fall of 2005. In addition to other class work, a set of five hands-on labs was designed to familiarize students with TinyOS and sensor networks. A sequence of well-designed and coordinated labs (Lab0 to Lab4)[17] walks students through the various stages of constructing and programming a sensor network using Berkeley's MICA2 Motes and its TinyOS interfaces.

Students start by learning the basics of various pieces of hardware, their interconnection, and programming environment (TinyOS). They then proceed in Lab0 to physically connect these devices, including the PC and the sensor devices, and to download an existing program to the MICA2 Motes for execution. In Lab1, students are introduced to the sensing and actuating capabilities of a wireless sensor node. Students are given example code for an application in which a sound is produced when a photo-sensor detects a value above a given threshold. Students are asked to write the code for the second application in which the sensor mote will detect the sound and turn on a green LED. Once students have the basic understanding of the inner working of a sensor network, they are asked in Lab2 to implement a simple sensor application that takes multiple light intensity readings, reports wirelessly to the base, and displays the readings on the PC using TinyOS's MessageCenter Tool. In Lab3 students evaluate the performance of the system built in Lab2 by writing a simple event generator for TOSSIM[12] using nesC, a C-like programming language for TinyOS. The last lab in the sequence is to have students develop a sensor network application which senses the environment and stores the data locally in an external flash memory device. When queried from the PC, the stored data is reported back. The sensor nodes can be dynamically configured.

After these "formal" labs are completed, teams work on a final project in which they implement an application/system using the motes. The projects are defined by the students with help from the instructor.

There are many other projects that are based on TinyOS. For an example, University of Utah's course, CS7962: Embedded Systems, had a number of projects and homework based on TinyOS using Telos Motes.[13]

Summary

The objective of this paper was to collect, summarize, and analyze information on laboratory and project assignments in wireless networking. We presented four readily available and significant hands-on examples that can be easily adapted. We believe that it will help others, as well as ourselves, to design and develop hands-on modules for wireless and mobile networking courses in both electrical engineering and computer science programs. Our goal is to provide examples for either introducing laboratory modules in existing computer network courses or to aid in the creation of new stand-alone wireless and mobile networking courses.

Acknowledgements

The authors would like to thank sincerely Brad Richards, Matt Welsh, and John (Jack) Stankovic for their invaluable information about hands-on experiences in their related course work.

References

[1] Cygwin, http://www.cygwin.com, last accessed December 28, 2005.

[2] Cybiko, http://www.cybiko.com, last accessed December 26, 2005.

[3] Dartmouth College, COSC78: Computer Networks, Spring 2005, Prof. Tristan Henderson, http://www.cs.dartmouth.edu/~cs78/, last accessed December 28, 2005.

[4] Dartmouth College, COSC78: Computer Networks, Project, Spring 2005, Prof. Tristan Henderson, http://www.cs.dartmouth.edu/~cs78/projects/project2.html, last accessed December 28, 2005.

[5] Harvard University, CS263: Wireless Communications and Sensor Networks, Prof. Matt Welsh, Fall 2004, http://www.eecs.harvard.edu/~mdw/course/cs263/fa04/, last accessed December 28, 2005.

[6] Harvard University, CS263: Wireless Communications and Sensor Networks, Prof. Matt Welsh, Fall 2004, Assignment #1, http://www.eecs.harvard.edu/~mdw/course/cs263/fa04/assn1/index.html, last accessed December 28, 2005.

[7] Moteiv, http://www.moteiv.com, last accessed December 28, 2005.

[8] E. Pazera, *Game Developer's Guide to Cybiko*, ISBN 1-55622-854-6, Wordware Publishing Inc. Plano, TX 75074, 2002.

[9] PlaceLab, http://www.placelab.org/, last accessed December 28, 2005.

[10] TinyOS, http://www.tinyos.net, last accessed December 28, 2005.

[11] TmoteSky, http://www.moteiv.com/products/docs/tmote-sky-quickstart.pdf, last accessed December 28, 2005.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", http://www.cs.berkeley.edu/~pal/pubs/tossim-sensys03.pdf, referred to from within http://www.cs.virginia.edu/~cs651/labs/lab3.htm, last accessed December 29, 2005.

[13] University of Utah, CS7962: Embedded Systems, Spring 2005, Prof. John Regehr, Spring 2005, http://www.cs.utah.edu/classes/cs7962/, last accessed December 28, 2005.

[14] Vassar University, CMPU-395: Wireless Networks – Fall 2002, Prof. Brad Richards, http://www.cs.vassar.edu/~cs395/, last accessed December 26, 2005.

[15] Vassar University, CMPU-395: Wireless Networks – Fall 2002, "802.11 Implementation Project", Prof. Brad Richards, http://www.cs.vassar.edu/~cs395/Project/project.html, last accessed December 26, 2005.

[16] University of Virginia, CS451/651: Wireless Sensor Networks – Fall 2005, Prof. John (Jack) Stankovic, http://www.cs.virginia.edu/~cs651/, last accessed December 29, 2005.

[17] University of Virginia, CS451/651: Wireless Sensor Networks – Fall 2005, Prof. John (Jack) Stankovic, Laboratory work, http://www.cs.virginia.edu/~cs651/labs.htm, last accessed December 29, 2005.