# Introducing Networking and Distributed Systems Concepts in an Undergraduate-Accessible Wireless Sensor Networks Course

Sami Rollins
Department of Computer Science
University of San Francisco
San Francisco, CA, USA
srollins@cs.usfca.edu

## ABSTRACT

The field of wireless sensor networks is growing rapidly and there is increasing interest in providing students with a foundation in the area. Curriculum in the field, however, is fairly limited and most courses reach only advanced graduate students. Undergraduate students typically lack the background, for example in distributed systems and networking, to digest the topics and assignments of a standard wireless sensor networks course. In this work, we present our approach to teaching wireless sensor networks to undergraduates and introductory graduate students. We discuss a unique, integrated approach to introducing relevant distributed systems and networking concepts in the context of wireless sensor networking applications. The course provides students who have never previously implemented a networked application with the necessary background to implement sensing applications on the Java SunSPOT sensor. It also provides a structured introduction to prerequisite concepts, such as distributed coordination algorithms, so that students can read and understand research papers. Our experience suggests that there is ample opportunity to expand curricula in sensor networking and reach a broader population of students.

## Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

## General Terms

Human Factors

## Keywords

wireless sensor networks, distributed systems, networking

## 1. INTRODUCTION

The field of wireless sensor networks is growing rapidly. Only a few years ago, work in the field was devoted to hardware and network protocols and only expert researchers attempted to build actual sensor systems. In recent years, the field has become much more broad. Pervasive devices such as mobile phones are touted as the next sensing platform [7]. Moreover, devices such as Sun's SunSPOT platform [10] have made programming sensor systems much more accessible. Whereas a few years ago one might have had to build the sensor device itself before programming it, today platforms such as the SPOT provide a complete networking stack and set of sensors, including light, temperature, and accelerometer, that can be easily programmed using a straightforward Java API. This has made sensor networking much more accessible to a broader population.

Curriculum in the field, however, is still fairly limited. Most courses on sensor networks reach only advanced graduate students. Textbooks and other resources for teaching sensor networks typically focus on low-level details such as radio communication, and are perhaps more appropriate for an electrical engineering curriculum [13, 4]. Further, though many sensor networks courses focus on having students read research papers, undergraduate and introductory graduate students often do not have the relevant background to digest such materials without a bit more preparation. Finally, more junior students have often not been introduced to relevant networking and distributed systems concepts, such as socket programming or concurrency, required to understand and implement sensing applications.

In this work, we present a unique approach to introducing general networking and distributed systems concepts using sensing applications as an overarching theme. We first discuss the challenges of designing such a curriculum for undergraduate and introductory-level graduate students, particularly in a liberal arts environment. We then provide an overview of our course and describe three crosscutting modules: our first module introduces network communication and socket programming by having students implement a simple sensing application; our second module introduces concurrency and multi-threaded programming by having students implement a duty-cycling algorithm for a sensor; our final module uses a layered approach to introduce distributed coordination and link-layer algorithms in order to prepare students to understand research papers on MAC protocols in sensor networks. We conclude with a discussion of our observations of the most and least successful elements of our course.

## 2. STUDENT PREPARATION AND THE LIBERAL ARTS ENVIRONMENT

The computer science department at the University of San Francisco (USF) is similar to that of many liberal arts colleges. We have a very small undergraduate program, with under 40 majors, and a mid-sized Master's degree (MS) program, with roughly 60 majors. This type of environment poses a number of challenges with respect to curriculum design:

- **Large General Education Requirement** - Our undergraduate students have a significant general education requirement, leaving limited room for computer science elective courses. Our major requires only 10 computer science courses total, and students take only one upper-level core course in each of three areas: systems, applications, and theory and languages.
- **Underprepared Students** - Our undergraduate curriculum is fairly horizontal; we have designed the curriculum to avoid long prerequisite chains and encourage more students into the major. Additionally, elective courses are often offered every other year and students end up taking courses when they are available, which is not necessarily the ideal time in their education. The result is that undergraduate students are often somewhat underprepared for upper-level courses. Similarly, our MS students come from extremely diverse academic backgrounds and many are unprepared for typical graduate-level courses in their first year.
- **Limited Course Offerings** - The combination of limited staffing and low enrollment results in the department offering a limited number of courses each semester. We generally have graduate students enrolled in upper-division undergraduate courses and undergraduates will often take our graduate courses as electives. This results in a need to provide an integrated curriculum that can meet the needs of several levels of student.

More specifically, the wireless sensor networks (WSN) course offered at USF enrolls a variety of students at the undergraduate and graduate level. Most students taking the course have not had a networking or distributed systems course. In contrast to advanced graduate students who might take a more typical WSN course, our students are underprepared in the following areas:

- **Basic Knowledge of Networking** - Most of the foundational work in the area of WSN centers around networking algorithms and a central topic in our WSN course is communication protocols. Students who have never taken a networking course are unfamiliar with the Internet protocol stack and the protocols that exist at each layer. Without this background, they may struggle to understand lectures and readings on topics such as MAC-layer protocols in sensor networks.
- **Basic Knowledge of Distributed Algorithms** - Similar to networking, much of the seminal work in WSN applies distributed algorithms that have existed for years. Two notable areas include topology control and time synchronization. Students can gain significant benefit from having some foundational knowledge of these areas before delving into a WSN paper.
- **Sockets** - A significant portion of students enrolling in WSN have never implemented an application that communicates over a network, wired or wireless. For those who have, many have only utilized frameworks such as Rails or Servlets, which hide the low-level communication details from the developer.
- **Threads** - Nearly all students who have taken WSN struggle with concurrency and multithreaded programming. Surprisingly, even for students who have previously implemented a thread scheduler as part of an operating systems course, multithreaded programming in WSN poses a challenge.
- **Programming Proficiency** - A meta-challenge we face is overall programming proficiency. Incoming MS students, in particular, often lack basic programming skills, and many students may lack knowledge of the particular language chosen for a given course.

**Table 1: Overview of topics covered.**

| **Topic** | **Number of Class Periods** |
|---|---|
| Concurrency and Multithreaded Programming | 2 |
| General Networking Concepts | 2 |
| Coordination Algorithms and Media Access Control | 2 |
| Error Control and the Link Layer | 2 |
| Topology Control | 2 |
| Routing | 2 |
| Time Synchronization | 1 |
| Reliability and the Transport Layer | 1 |
| Participatory Sensing | 2 |
| Applications | 2 |

Our WSN course has been designed with these characteristics in mind. The remainder of this paper describes our course and the unique approach we take to introduce broader concepts in the context of WSN.

## 3. COURSE OVERVIEW

The Wireless Sensor Networks (WSN) course at USF is a 15-week, undergraduate-accessible course that introduces several general computing concepts using wireless sensor networks as an overarching example application. The course has been offered twice to date.

The course takes a networking-centric approach to WSN. Table 1 outlines the main topics covered in the course and the number of class periods typically devoted to each topic. Below we overview several choices made in the design of the course:

*Lecture/Discussion Format.*

Class periods typically alternate between interactive lecture and class discussion of research papers. Because there are no good comprehensive resources (e.g., books) that cover all of the necessary topics, lecture periods provide students with a broad overview of each area. Elements discussed in the lectures include the broad research challenges and the algorithms or approaches from other areas (e.g., networking and distributed systems) that may apply. We follow a lecture with a class discussion of a research paper on the given topic. Even undergraduate students with little experience digesting research papers respond favorably to this approach. The lecture periods prepare the students for reading the paper and the discussions allow us to achieve some depth with regard to the topic.

*SunSPOTs.*

The course uses Sun's Small Programmable Object Technology (SPOT) sensor platform [10]. The SPOT is a Java-programmable sensing device that comes equipped with several sensors including a light sensor, accelerometer, and temperature sensor. The SPOTs provide basic networking and use the AODV protocol for communication [9]. Each SPOT also has two switches that can be toggled, and 8 LEDs. This platform is ideal for a teaching environment. Students who have knowledge of Java can easily begin programming without the learning curve associated with similar technology such as the Mote [8].

*Programming-Centric Assignments.*

Students are given a substantial amount of programming to reinforce concepts taught in the class. The most recent offering of the course required 3 laboratory assignments and 4 significant programming projects. Our students typically respond positively to this approach. Not only does it reinforce learning of the material, it prepares our students for development jobs, which is the career path chosen by most of them.

## 4. CROSSCUTTING MODULES

The primary contribution of our approach to teaching WSN is a set of crosscutting modules that use sensor networks as an overarching context in which to introduce general networking and distributed systems concepts. In this section, we describe three modules that are most representative of our approach. We first describe two programming assignments that introduce the concepts of sockets and threads, respectively. We then describe a lecture and discussion module that illustrates our approach to introducing relevant networking and distributed systems concepts to lay the foundation for a more detailed discussion of a research paper.

### 4.1 Network Communication

Network communication is a fundamental tool used in virtually any sensing application. Traditionally, socket programming is introduced in a networking course or, as is the case in our curriculum, a software engineering course. A UDP Pinger, as described by Kurose and Ross [5], is a typical laboratory assignment used to familiarize students with sockets. The assignment asks students to implement a simple client and server; the client sends a simple ICMP message to a server and determines the round trip time based on the response received. This type of assignment gives students a solid understanding of how communication works. They see first hand that, as Kurose and Ross describe, the server program must first be running and have opened the "door" for communication before it can be contacted by the client. They also gain an understanding of the life cycle of a client-server connection.

Unlike a typical graduate course on WSN, which might assume that students are already comfortable with network programming, our WSN course elevates communication primitives to a first-order topic. We spend roughly half of a lecture period (which are 105 minutes) discussing the basics of radio communication and looking at examples using the SunSPOT API. Students are then asked to practice what they have learned by implementing a simple WSN application that requires client-server communication. In the most recent incarnation of the course, students implemented a client application that samples and reports the temperature on a sensor at a regular interval. Students also implemented a server application that collects data from all sensors and records the reading along with the date/time and the address of the sender[1].

Network communication is perhaps the simplest and most straightforward example of a general topic that can be easily integrated into a WSN course. We spend only half a lecture period to introduce the topic and students are able to implement a simple sensing application that illustrates the same concepts as would a traditional client-server application. Students then build upon this application over the course of the semester and, eventually, produce a more complex sensor system.

### 4.2 Multi-threaded Programming

The topic of concurrency and threads can appear in many different computer science courses. Though the topic is often covered in detail in a distributed systems course, students might also encounter this topic in an operating systems course and, more recently, parallelism has started to appear in lower-level courses. At USF, for example, students are required to take an introduction to parallel computing course in the sophomore year and threads are also a topic covered in our sophomore-level software engineering course. Assignments in these courses might include the implementation of a multi-threaded web server, or sorting using MPI.

We have found that, even after seeing threads in one or two lower-level courses, students often struggle to understand the intricacies of their operation. In part, the problem stems from a lack of compelling assignments that are guaranteed to expose potential issues that arise with threads. An assignment such as a multi-threaded web server, for example, may require little coordination between threads, particularly if any shared data is stored in a database such as mysql. Additionally, testing strategies for multi-threaded programs can be tricky. If the assignment works, e.g., serves web pages correctly, students may not take the extra step to verify that threads are created and shut down appropriately, or that they interleave correctly.

In WSN, students must use threads extensively for their work. Students report that the *duty cycling* assignment is the most useful assignment in our WSN course. The key

[1]SunSPOT devices are addressed using a 64-bit IEEE address.

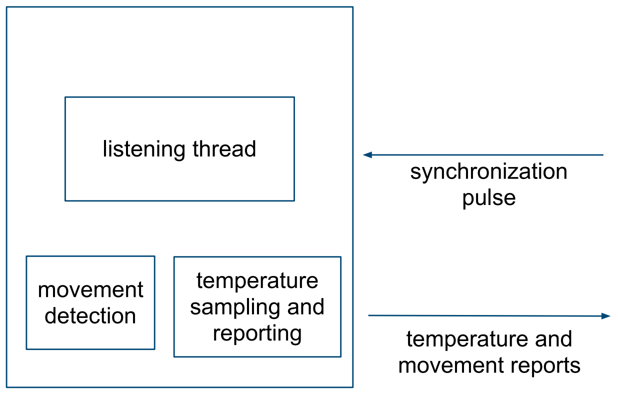
listening thread
synchronization pulse
movement detection
temperature sampling and reporting
temperature and movement reports

**Figure 1: Duty cycle assignment architecture.**

element of the assignment is an algorithm to cycle the sensors such that they are on for $n$ seconds and in a suspended or low-power state for $n$ seconds. Moreover, each sensor must be synchronized, at a very coarse level, with the other sensors in the system.

We prepare students for the assignment by devoting two lecture periods to discussing concurrency and multi-threaded programming. This is necessary as some students, particularly our incoming graduate students, have never previously seen the concepts. For most students, this is enough preparation, though a few students typically require a significant amount of help from the instructor or TA when implementing the assignment.

Figure 1 illustrates the components of the assignment. During the on periods, all sensors sense two elements: (1) sensors periodically sample and report their temperature to a base station and (2) sensors use their accelerometers to detect movement and report any movement to a base station. To ensure that all of the sensors' on periods are roughly synchronized, i.e., to mitigate clock drift, the base station periodically sends a synchronization pulse that indicates when the next sleep period should begin. Though students are free to design their own implementations, students often choose to use at least three threads, one to control each of the components in Figure 1. In order for the sensor to enter a suspended state, all threads, including all system-level threads, must be blocked. This is one of the most challenging aspects of the assignment.

Students find that this assignment gives them a significantly better understanding of multi-threaded programming. Students need not develop complicated test suites as they can immediately see whether the sensors enter and leave the suspended state when required. Below we highlight several key lessons of the assignment:

- **Thread Lifecycle** - Students must have a keen understanding of when threads are blocked and when they are runnable. If a single thread is in a running or runnable state, the sensor will not suspend. Similarly, if a thread unexpectedly becomes runnable during the suspend period then the sensor will exit the suspend period prematurely. Through this assignment, students learn how to tightly control a thread's operation.

- **Inter-thread Communication** - To ensure that all threads enter a blocked state at the same time there must be some communication between them. Students often make the mistake of having each thread use an independent timer to determine when to sleep and wake. They quickly discover that this approach does not work as the threads will not be synchronized and the sleep cycles end up being shorter than required. This problem is typically solved by having a single thread act as a *main controller*. This thread determines when the sensor should sleep and wake, and communicates this information to all other threads appropriately.

- **System-Level Threads** - One of the most challenging aspects of this assignment centers around control of system-level threads that are not started by the user-level program. In particular, the SunSPOT sensors have a routing manager that maintains a routing table and forwards messages within a mesh network. The routing manager must be effectively disabled in order for the sensor to enter a suspended state. Students learn a significant amount from figuring out that system-level threads can prevent suspend of the sensor and from figuring out how to control the system-level threads in order to achieve the desired behavior.

## 4.3 Coordination and the MAC Layer

Because the field of sensor networks is still somewhat immature, most of the literature in the field is restricted to research papers published in computer science conferences. While advanced graduate students may be able to read and digest a paper published at SenSys or Infocom, undergraduates and new graduate students typically lack much of the necessary background to thoroughly understand the ideas proposed. To fully understand and appreciate the benefits of a MAC-layer algorithm such as S-MAC [12], for example, students must have a broad understanding of both distributed coordination algorithms and general MAC-layer protocols.

Our solution to this problem is to take a layered approach to introducing research papers in WSN. Whereas in a typical WSN course students might be expected to read and discuss a new paper every lecture period, we generally devote two lecture periods to each paper. In the first lecture period we introduce the relevant distributed and networking concepts that underlie the solutions proposed by the paper. Once students have this foundation, they are asked to read the paper and discuss it during the subsequent lecture period.

Figure 2 provides an overview of our approach to introducing MAC-layer protocols in sensor networks. In our last incarnation of the course, students read Ye, Heidemann, and Estrin's paper on the S-MAC protocol [12]. We introduced the paper with a lecture covering coordination algorithms in distributed systems, MAC-layer protocols in more traditional networks, and an overview of the challenges of designing MAC-layer protocols in wireless networks. More specifically, our lecture covered mutual exclusion algorithms including time division multiple access (TDMA), frequency division multiple access (FDMA), server-controlled polling algorithms, and ring-based algorithms. Once students understood these broad algorithms, we introduced a conceptual overview of media access control and discussed traditional MAC-layer algorithms such as Ethernet. With this foundation, we then introduced several challenges, including energy efficiency, that arise in the design of MAC protocols for wireless networks.

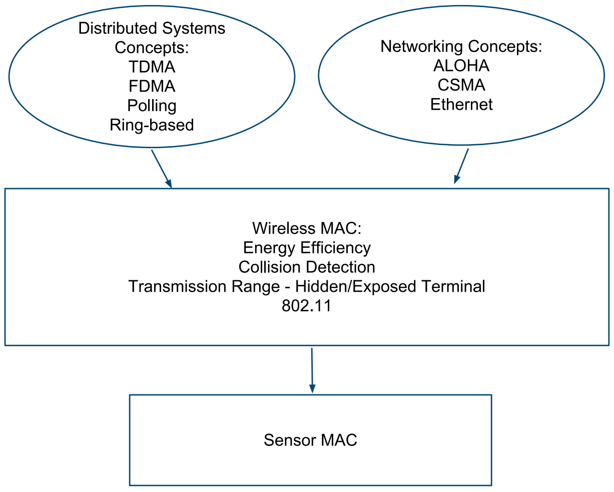
Distributed Systems Concepts: TDMA FDMA Polling Ring-based
Networking Concepts: ALOHA CSMA Ethernet
Wireless MAC: Energy Efficiency Collision Detection Transmission Range - Hidden/Exposed Terminal 802.11
Sensor MAC

**Figure 2: Approach to preparing students to read about MAC-layer protocols for sensor networks.**

Though we are unable to cover each of these topics in as much detail as we would in a distributed systems or networking course, one lecture period has provided sufficient time to overview the important elements of each area. With this foundation, students are able to read and understand the assigned research papers. For some papers, students are even asked to implement versions of the proposed algorithms. While we have not done any formal evaluation, we also believe that this approach will have a positive impact for students who subsequently enroll in a distributed systems or networking course.

## 5. OBSERVATIONS

Our general observation is that the unique approach of integrating distributed systems, networking, and WSN concepts into the same course has been successful. In this section, we highlight several notable elements of the course.

### 5.1 Tangible Systems

One of the most educative aspects of the course is the experience of building actual sensing applications using physical devices. A sensing application is both inherently distributed and requires networked communication in order to function. By giving students lots of hands-on programming exercises with the SunSPOT sensors, they simply cannot avoid gaining experience with both distributed and networking concepts.

More specifically, to provide a tangible illustration of routing algorithms the routing lecture includes a physical demonstration of the path a packet takes through the sensor network. The demonstration works as follows: each student takes a sensor and positions him or herself somewhere in the classroom. The instructor initiates the send of a packet from a base station node located at the front of the classroom. Based on the color of the LEDs on the sensors, students can see which nodes participate in path discovery and which nodes route the packet. This is an extremely useful exercise and gives students a tangible view of how routing works.

### 5.2 Iterative Assignments

An approach that has had mixed success is the use of assignments that are iterative—each assignment builds on the previous. We originally thought this approach would be very successful. It allows students to first practice basic skills, such as socket or thread programming, and then asks them to build on top of what they have done to produce a more sophisticated application. We expected students would appreciate not having to start from scratch for each assignment, and that they could build a final application with a significant amount of functionality and complexity.

The reality, perhaps not surprisingly, is that students are frequently unable to complete assignments satisfactorily, which means they are at a disadvantage starting the next assignment. This has been incredibly frustrating for students, and needs to be resolved in future iterations of the course. One solution under consideration is to provide students with solutions to each assignment after it is due. Student can then either build upon their own solution, or the one provided by the instructor.

### 5.3 Student-Chosen Projects

Each semester we have given students the freedom to propose their own final project for the course. Students can either extend the application they have been building over the course of the semester, or they can choose a completely new project. One of the most exciting successes of the course has been to witness the creative and interesting set of projects proposed by the students. While some students choose the safe bet and implement a small extension to an existing project, some students really try to integrate all they have learned and try something new.

Students have proposed several notable projects in the two semesters the course has been taught. A popular proposal is the use of the sensor as a controller, either for a computer-based game or another device such as a Lego robot. While this assignment does not integrate many of the distributed and network concepts students learn in the course, students often use it as an opportunity to integrate concepts from other courses, for example using Artificial Intelligence techniques to do gesture recognition. Several students have also proposed projects that involve significant experimentation and evaluation. They draw upon what they learn reading and evaluating papers and design and execute experiments to compare different types of sensor network algorithms. Finally, one of the more fun projects was an *automated race monitoring system.* A group of students designed a sensing application and web service to track a runner's movements (speed, etc) and report the results via a web application. The students worked tirelessly on the application, testing running up and down the hallways and getting other students excited about taking the class in the future!

### 5.4 Room for Improvement

There are several notable areas where we feel the course could be improved. First, in both iterations of the course we have taken a bottom-up approach. We first talk about relevant technologies, i.e., the SunSPOT hardware, and then take a networking-centric approach moving from the MAC layer to the application layer. Though applications of sensor networks, such as RACNet [6], tend to be the topics students enjoy most, they often seem "burnt out" by the time we cover the related papers. We would like to explore whether students have enough foundation surrounding the specific challenges and proposed solutions in sensor networks to understand the relevant papers if we were to structure the course in a top-down fashion.

While we do offer students the experience of using actual devices to develop real sensing applications, in the end their applications are only prototypes and they miss the experience of interacting with an actual deployment. We would like to find a feasible way of having students either build and deploy an application over the course of the semester, or otherwise engage with a deployment provided to them. There are several critical skills they would gain from the experience. They would, for example, have to deal with real failures in a long-running system.

## 6. RELATED WORK AND CONCLUSION

There has been increased interest in making material traditionally taught in graduate courses accessible to undergraduates. As Albrecht points out [1], knowledge of distributed systems concepts is becoming critical and undergraduate students are often underprepared for careers in research and development without a foundation in this area. Hnatyshin and Lobo [3] highlight the importance of engaging students in computer networking by providing hands-on exercises to illustrate key concepts. More specifically in the area of sensor networks, Tyman, Bulusu, and Mache underscore the increasing importance of wireless sensor networks and point out that there is a need to make sensor networks accessible to undergraduates [11]. They also point out that with the advent of sensor technology like the Java-programmable SunSPOT, this is imminently possible. Finally, Forster and Jazayeri [2] describe their hands-on approach to introducing undergraduate students to sensor networks to prepare them for jobs in industry.

Our work presents an integrated approach to introducing networking and distributed systems concepts using sensing applications as an overarching motivation. We describe our experience with introducing concepts such as network communication and concurrency by having students implement applications on the SunSPOT sensor platform. We further discuss our success using a layered approach to engage undergraduate students in the process of digesting research papers. Particularly in a small, liberal arts department such as ours, our approach is ideal for introducing students to a broad range of advanced topics.

## 7. REFERENCES

[1] Jeannie R. Albrecht. Bringing big systems to small schools: distributed systems for undergr aduates. In *SIGCSE '09: Proceedings of the 40th ACM technical symposium on Com puter science education*, pages 101–105, New York, NY, USA, 2009. ACM.

[2] Anna Förster and Mehdi Jazayeri. Hands-on approach to teaching wireless sensor networks at the undergraduate level. In *ITiCSE '10: Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 284–288, New York, NY, USA, 2010. ACM.

[3] Vasil Y. Hnatyshin and Andrea F. Lobo. Undergraduate data communications and networking projects using opnet and wireshark software. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 241–245, New York, NY, USA, 2008. ACM.

[4] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, 2005.

[5] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach.* Addison-Wesley Publishing Company, USA, 3rd edition, 2005.

[6] Chieh-Jan Mike Liang, Jie Liu, Liqian Luo, Andreas Terzis, and Feng Zhao. Racnet: a high-fidelity data center sensing network. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 15–28, New York, NY, USA, 2009. ACM.

[7] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Xiao Eisenman, Shane B. an d Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: the design, implementation and e valuation of the cenceme application. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded netw ork sensor systems*, pages 337–350, New York, NY, USA, 2008. ACM.

[8] Crossbow Motes. `http://www.xbow.com`.

[9] Charles E. Perkins and Elizabeth M. Royer. The ad hoc on-demand distance-vector protocol. pages 173–219, 2001.

[10] SunSPOTs. `http://www.sunspotworld.com`.

[11] Damon Tyman, Nirupama Bulusu, and Jens Mache. An activity-based sensor networks course for undergraduates with sun spot devices. *SIGCSE Bull.*, 41(1):34–38, 2009.

[12] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. USC/Information Sciences Institute, IEEE.

[13] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach.* Elsevier/Morgan-Kaufmann, 2004.