Demo Abstract: A Unified Architecture for Flexible Radio Power Management in Wireless Sensor Networks

Kevin Klues, Guoliang Xing, Chenyang Lu
Washington University in St. Louis
{klueska, xing, lu}@cs.wustl.edu

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols—*Protocol Architecture*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Modules and Interfaces*

General Terms

Experimentation, Design

Keywords

Wireless Sensor Networks, Radio Power Management, Architecture, Framework

1 Overview

Radio power management is of paramount concern in wireless sensor networks. While a multitude of power management protocols have been proposed in the literature, their use in real-world systems has been limited. The lack of system support for flexibly integrating different power management policies with a diverse set of applications and network platforms has been the major stopping point. To provide a solution to this problem, we have developed the *Unified Power Management Architecture (UPMA)* for supporting radio power management in wireless sensor networks [1]. In contrast to the monolithic approach adopted by existing power management solutions, UPMA provides (1) a set of standard interfaces that allow different radio sleep scheduling policies to be easily implemented on top of various MAC protocols at the data link layer, and (2) an architectural framework for composing multiple power management policies into a coherent strategy based on application needs. We have implemented UPMA on top of both the Mica2 and Telosb radio stacks in TinyOS-2.0. This demo shows how the UPMA architecture can be used to easily compose different radio power management protocols together in order to achieve better power savings than each of them could achieve individually.

The figure below shows the organization of the UPMA architecture.

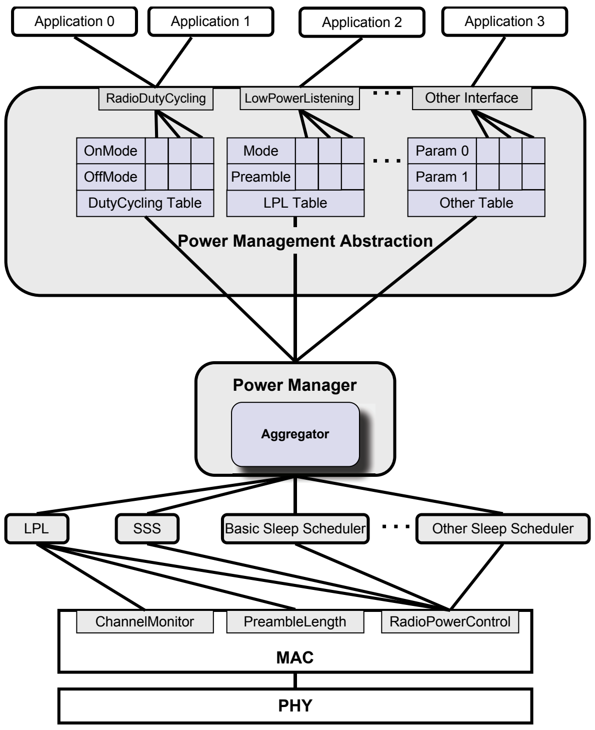

Figure 1. Radio Power Management Architecture

The *Power Management Abstraction* allows applications requiring the use of different power management strategies to coexist on a single node without knowledge of one another [1]. Parameters supplied by each application to their respective power management strategies are passed through the interfaces provided by this abstraction. These parameters are then stored inside a table, with one table existing per interface. Figure 1 shows how this abstraction layer can be used to

[1]The term application is used loosely here. An application in this sense could potentially exist at any level of the network protocol stack as long as it needed to interface with these power management interfaces.

store parameters supplied through the `RadioDutyCycling` and `LowPowerListening` interfaces.

In the same way that the *Power Management Abstraction* allows multiple applications to coexist without knowledge of one another, the *Power Manager* allows multiple power management strategies to coexist. It is used to perform two separate functions. First, it is used to aggregate all of the parameters supplied by different applications to the *Power Management Abstraction*. Second, it coordinates the use of the different power management strategies required by these applications so that they do not interfere with one another.

The combination of these two architectural components provides an intermediate layer between applications and power management strategies that allow each of them to be developed as if they communicate directly with one another. The burden of coordinating their usage is left to the *Power Manager* component. A developer may easily change the coordination policy in use by simply replacing it with an implementation of the appropriate aggregation component. In this demo, we show just how easily these aggregation components can be exchanged, as well as how effective combining different power management strategies on a single node can actually be.

2 Demo Setup

In this demo we show how two different aggregation policies can be used to achieve different levels of power savings in a wireless sensor network setup. Applications running on each node communicate with a simple radio sleep scheduling component using the `RadioDutyCycling` interface exposed by the Power Management Abstraction. Applications pass their desired duty cycle requirements through this interface to the Power Management Abstraction. Depending on the types of applications running, a different aggregation policy may be chosen to achieve the best possible power savings that satisfy each of their duty cycling requirements. A GUI application has been created that allows one to select from among a set of various applications developed within the UPMA framework. These applications are then automatically matched to the power management strategy they require and a choice of the aggregation policy to use is given. The topology of the network is then displayed on the screen, with information pertaining to each node (including its current power consumption) being displayed when that node is selected.

2.1 Combining different Duty Cycles

In our first setup, our network consists of a one-hop cluster of Telosb nodes. The setup consists of a number of slave nodes that periodically send packets to a single master node. Although each slave node only runs a single application, up to 6 different applications can be running in the network at any given time that the master must be prepared to listen for. The on time of the duty cycle for each of the six applications is 200ms, with the off time taking on values of of 200ms, 600ms, 1.4s, 3s, 6s, and 12.6s, respectively. Each application sends a packet at a random time within the 200ms active period of each duty cycle. The master node is able to receive packets from each application by running an aggregate duty cycle according to the OR policy seen in the figure below.

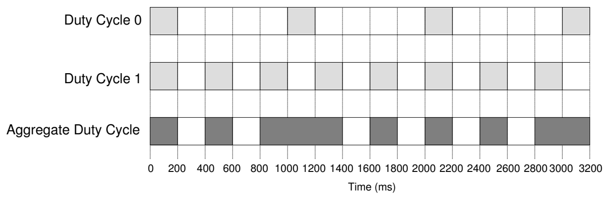

Figure 2. Aggregation of multiple duty cycles

This aggregation policy is implemented as follows. (1) All duty cycles are shifted to begin at the same time instant. (2) They all run periodically according to their own schedule. (3) If *any one* of the duty cycles requires the radio to be on at any particular point in time, the radio will be turned on. (4) Only if *all* duty cycles indicate that the radio should be turned off will the radio ever be turned off.

2.2 Duty Cycles with Density Control

The second setup is similar to the first except that now we are combining two different applications on each of the slave nodes so that they too require the use of an aggregation policy. The second application that each of them runs is a variant of a density control protocol known as PEAS[2]. The network consists of a master Telosb node and 15 slave Telosb nodes placed in a 5×3 grid. Each slave node runs both PEAS as well as an application that is able to specify its duty cycle. Six applications are possible, each with a duty cycle period of 3.2s. The on times of each duty cycle range from 200ms to 1.2s in steps of 200ms. PEAS runs with a duty cycle period of 16s and an on time of 200ms. Inactive nodes send PEAS probe messages at some random time within their 200ms on period, and active nodes send a packet to the master node at some random time during their on period. The probing range of PEAS is limited to 1.5 times the grid width.

The first setup demonstrates the power of this architecture, by allowing the master node to easily select an aggregation policy that will allow it to hear from each of his slaves without having to reimplement the code necessary to do so. The second one demonstrates how additional power savings can be achieved in a network by applying a simple aggregation policy to multiple applications running on a single node that may not originally have been intended to work with one another. Through the use of the GUI interface, visitors to this demonstration will be able to see how easy it is to select the different aggregation polices for each setup, as well as see the actual protocols in action.

3 References

[1] K. Klues, G. Xing, and C. Lu, “A Unified Architecture for Flexible Radio Power Management in Wireless Sensor Networks,” Washington University in St.Louis, Tech. Rep. WUCSE-2006-06, 2006.

[2] F. Ye, G. Zhong, S. Lu, and L. Zhang, “Peas: A Robust Energy Conserving Protocol for Long-Lived Sensor Networks,” in *The 23rd International Conference on Distributed Computing Systems (ICDCS’03)*, May 2003, pp. 169–177.