NETWORK AND SYSTEMS SUPPORT FOR BUILDING ENERGY MONITORING AND CONTROL USING WIRELESS SENSOR NETWORKS

by

Alan Marchiori

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Mathematical and Computer Sciences).

Golden, Colorado

Date _____

Signed: _____
Alan Marchiori

Signed: _____
Dr. Qi Han
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Tracy Camp
Professor and Head
Department of Mathematical and Computer Sciences

ABSTRACT

This thesis explores the application of wireless sensor network technology in residential buildings to monitor and control energy consumption. The technologies developed in this thesis provide a communications backbone so that building systems and appliances can share information. By enabling the free flow of information, we can first collect detailed building performance data. Second, we can rapidly deploy sophisticated whole-house energy optimization strategies.

The centralized building automation and control systems typically found in commercial buildings are typically complex, require expert installation, and costly. In contrast, our peer-to-peer approach utilizing a self-configuring wireless network provides the flexibility and reduced cost required for the residential environment. We first explore the use of non-intrusive load monitoring to automatically extract detailed energy usage information from simple aggregate measurements. This information can, for example, be used to identify wasteful devices. To efficiently distribute sensor data throughout the wireless network, we have developed PIM-WSN, the first general-purpose multicast implementation for IP-based wireless sensor networks. To better understand how occupant behaviors impact energy consumption, we have evaluated 8 energy saving behaviors in homes for 10 weeks. The results were surprising as no single behavior significantly reduced energy consumption. However, we discovered that even motivated homeowners are unwilling to manually implement many energy saving behaviors. To explore automation of these behaviors, we developed a prototype occupancy-based solution and achieved 7%-14% energy savings in office environments. To enable deployments in residential homes, we developed a complete platform based on these demonstrated principles. From several deployments, we observed an 88% reduction in standby losses for home entertainment devices and saved 195 kWh per day by automating basic energy saving behaviors. The completed platform is highly adaptable to new problems and provides sensing and a physical

interface between the building and the control algorithm. This architecture is now available as a resource to implement future control strategies in residential buildings.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation was made possible through the encouragement and support of many people. Fist I acknowledge my adviser, Dr. Qi Han. Her patience and wisdom was essential to keep my research focused. Dr Tracy Camp and the rest of the Toilers research group also provided valuable feedback and assistance throughout this process. Dr. Ren Anderson and the rest of the Residential Buildings group at the National Renewable Energy Laboratory provided significant guidance and funding, ensuring the success of this work.

I also acknowledge all of the professors and teachers I had through college and high school. It is not possible to name them all but, some of the most influential were: Dr. Carla Brodley, my M.S. adviser; Dr. Ismail Jouny and Dr. John Greco from Lafayette College; Daniel Fahringer and John Eliason, two outstanding high school teachers. Without their dedication to teaching, I would not have been prepared to begin this research.

Finally, and most importantly, I acknowledge my wife, two sons, and parents who encouraged and supported me throughout this process.

CHAPTER 1

INTRODUCTION

The U.S. department of energy reports that buildings were responsible for 39% of the total energy consumption in the U.S. in 2009 [1]. In the typical U.S. home, 75% of the energy consumption is used for space heating, cooling, water heating, and lighting. The remaining 25% is consumed by miscellaneous electronic loads (e.g., home entertainment, home office, battery chargers). Due to improvements in building construction, the relative load presented by these electronic devices is predicted to rise to 34% by 2020 [2]. Energy is a precious resource and it is essential that we develop technologies to provide detailed monitoring and control of energy consumption in buildings.

Because a large portion of building energy consumption is closely tied to occupant behavior, numerous monitoring systems have been recently developed [3–5] to provide occupants with detailed feedback on their energy consumption. The theory is that given detailed feedback the occupants can modify their behavior to reduce consumption. The potential impact of such data is significant. However, feedback alone does not always result in savings. A recent study observed an initial 31.9% reduction in energy consumption after installing a monitoring system; however, after a month the reduction fell to only 3.7% [6]. This illustrates that while significant savings are possible, relying on occupants to change their long-term behavior may be difficult. As a result, our ultimate goal is to develop an architecture for intelligent control systems that can be used to automate energy savings in buildings.

Another benefit to developing an architecture for intelligent building automation and control systems is the opportunity to take advantage of a SmartGrid interface and implement demand response. A SmartGrid is an electrical distribution network that incorporates two-way communication between the energy consumer and producer. Demand response allows loads to be varied in response to the state of the

electricity grid. When there is an excess of energy, devices such as hot water heaters, refrigerators, air conditioners, or battery chargers could alter their schedule to immediately use the excess. This capability is useful when paired with wind turbine and solar photovoltaic generation. Gusty winds result in power output that may have significant peaks relative to the mean output. By linking the consumer directly to the generation sources through a two-way communication network we can schedule cyclic devices in phase with the generation, thereby optimizing performance. Conversely, when generation is temporarily decreased or demand begins to peak, operation of these cyclic devices can be temporarily postponed resulting in better synchronization between energy production and consumption. Demand response results in a more efficient grid because less spinning reserve generation (i.e. generators running at low output to respond quickly to changing demand) is required.

1.1 Our Approach

This thesis explores the application of wireless sensor network technology in residential buildings to monitor and ultimately control energy consumption. Wireless sensor networks are built using low-cost computing platforms coupled with industry standard radio systems (e.g., IEEE 802.15.4 [7]). These networks provide a communications backbone so that building systems, appliances, and devices can shared relevant information. This information can then be used by an intelligent building automation and control (BAC) system to minimize energy consumption by implementing demand response, load scheduling, occupancy-based control, intelligent lighting, and other general-purpose automation and control strategies.

Currently, building automation and control systems are deployed in commercial buildings and rely on numerous hardwired sensors using a myriad of communication protocols (BACnet [8], LonTalk [9], ModBus [10], etc.) to communicate data to a central controller. Wireless protocols (ZigBee [11], Wireless HART [12], EnOcean [13], etc.) have been employed to replace some or all of the wired links. However, even

when wireless components are used, a central controller is still responsible for making control decisions. The central controller is typically a programmable logic controller (PLC) that implements basic automation tasks. This thesis presents a completely new approach that is a *paradigm shift* from centralized control to a *distributed approach* for building automation and control. This is motivated by the observation that the microcontrollers typically found on wireless platforms are capable of implementing basic automation and control tasks without relying on a central controller. The end result is by simply utilizing wireless components *we eliminate the need for a central controller*. This approach yields a highly adaptable system that is well suited to implement automation and control in residential buildings.

1.2 Research Contributions and Dissertation Organization

To achieve the goals of detecting energy waste and automating energy savings in buildings, this dissertation makes the following contributions:

Chapter 3: Circuit-Level Load Monitoring for Household Energy Management. There are two basic approaches for monitoring energy consumption in the home. The first is to simply use whole-house energy measurements. This is easy, but does not provide detailed information about the energy consumption of each device. The second approach is to directly monitor the energy consumption of each device in the home. This creates a very detailed picture of household energy consumption, but is costly to implement. This chapter explores an alternative approach to monitor household energy usage, including small devices, by using circuit-level power measurements. We have developed and evaluated two algorithms to disaggregate the circuit-level data into device-level estimates. Our evaluation resulted in an average error less than 5.35% when separating energy usage from the aggregate of three small devices.

Chapter 4: PIM-WSN: Efficient Multicast for IPv6 Wireless Sensor Networks. We envision that wireless building automation and control systems will employ sensors that share data directly with automation and control nodes. One way to implement this information sharing is to use multicast communication. Multicast communication simplifies this task for the sensor by using the network layer to disseminate sensor data in a robust and efficient manner. However, existing solutions for multicast in WSNs are limited because they either support multicast only from a single source node (usually the root node) or they limit the multicast group size to constrain memory usage. To solve both of these problems, we have developed PIM-WSN, a protocol independent multicast (PIM) protocol tailored for IPv6 WSNs. Our design is the first general purpose multicast for WSNs that allows any node to be a mulitcast source and have an unlimited number of subscribers. Using detailed simulations we show that PIM-WSN achieves 1) high packet delivery rate (over 97%), 2) low latency per hop (less than 5 ms), and 3) lower radio utilization than all other comparable protocols (by more than 50%). Using a ten-hop testbed of TelosB motes we have verified our implementation of PIM-WSN for TinyOS 2.x with the Blip IPv6 networking stack which uses only 5,978 bytes of ROM and 235 bytes of RAM.

Chapter 5: Building the Case For Automated Building Energy Management. Two basic questions regarding automated building energy systems are why we need them and how much energy will they save? To address these questions, we have evaluated 8 energy-saving behaviors, as well as the use of an in-home display (IHD), in 10 homes over the course of 10 weeks. In the end we found that no single energy-saving behavior significantly impacted energy consumption. However, for several energy-saving behaviors, fewer than half of the participants reported implementing the behavior. This suggests that automated building

energy management systems may be necessary to realize the full benefits of these behaviors.

Chapter 6: Distributed Wireless Control for Building Energy Management. Chapter 5 suggests that automated building energy management systems may be necessary to reduce the influence of behaviors on energy consumption. Chapter 6 demonstrates how PIM-WSN can be used in a wireless building energy management to share sensor data. Dedicated sensor and energy controller nodes are also described. The sensor nodes transmit data using PIM-WSN. Energy controller nodes automatically detect and utilize the shared information to implement an occupancy-based energy control strategy. This study resulted in an energy savings of 7.1% - 14.6% for the measured loads.

Chapter 7: Achieving Reliable Wireless Automation and Control Using Global Shared Memory. To improve the robustness of wireless building energy management systems, this chapter presents a complete system based on a global shared memory (GSM) networking abstraction. GSM addresses the primary challenge for a wireless building automation and control system, reliability. Reliability is difficult to guarantee using typical networking protocols, like PIM-WSN. Here we trade some efficiency for improved reliability by maintaining a complete copy of the global shared memory on each sensor node. Nodes communicate in their one-hop neighborhood to synchronize all shared values. After network disconnections or other disruptions, the shared values will quickly synchronize to the most recent values. Using this approach, we observed an 88% reduction in standby losses for a home entertainment system. Using published statistics on home entertainment and home office systems, we expect this approach, on average, to reduce energy consumption of these devices by 17%, or equivalently a 2.3% reduction in whole-house energy consumption. This provides a robust

foundation for implementing any general purpose distribute automation and control system.

1.3 Broader Impacts

The broader impacts of this research include disseminating results through publications, software releases, and hardware designs. In addition to these research impacts, we expect that the results from these efforts could be translated with low-risk to the commercial market. In this process there is also the potential to create new jobs supporting these products from the engineering, business, and manufacturing sectors. Finally, if the building automation and control strategies presented in this thesis were adopted nationwide (assuming 50% penetration) for home entertainment, home office devices, and HVAC, national energy consumption would be reduced by 0.42 quadrillion BTU per year.

CHAPTER 2

RELATED WORK

The general topic of building monitoring and control has been studied for decades. This thesis presents a complete system for monitoring and controlling energy usage using a wireless sensor network (WSN). It relies and builds on a large amount of existing work from WSNs. Wireless sensor networks are a relatively new concept that has been growing in popularity for the last decade. Here we provide an overview of the state of the art as well as a historical perspective of research in these fields that is most relevant to this dissertation. We begin with an overview of residential building monitoring and control systems in Chapter 2.1. Chapter 2.2 begins our exploration into WSN technology by examining the most commonly used operating systems used for research. To build up the WSN networking stack, Chapter 2.3, provides an overview of contemporary WSN networking architectures. Finally, Chapter 2.4, completes our exploration by describing high-level abstractions that treat the WSN as a whole and provide abstractions for programming, collecting, and displaying sensor data. Through this exploration we can determine which of these approaches would be useful for building energy monitoring and control.

2.1 Building Monitoring and Control Systems

Perhaps the earliest example of an advanced building monitoring and control system is the Neural Network House [14]. The Neural Network House contains about 75 sensors and actuators wired to a central controller. The goal of the Neural Network House is twofold: appeasing the inhabitants and conserving energy. As the name implies neural networks are used to predict behaviors and then automatically configure the environment by controlling lights and the heating and cooling systems.

One modern solution similar to the Neural Network House is offered by Control4. Control4 is a commercial home automation solution that integrates home theater,

lighting, temperature, and security components with a central controller [15]. There is a significant emphasis on entertainment in the Control4 system. There is no energy monitoring beyond knowing the state of devices (on/off). Control4 is mainly a convenience for the occupant by allowing control and automation of the home's systems through their central server. Both wired and wireless devices are supported.

Recently researchers at the University of California San Diego have developed the Energy Dashboard [3]. This is a large deployment of energy meters on nearly all buildings on their campus. Certain buildings have multiple meters allowing energy usage to be viewed at varying levels of granularity (i.e., whole-building, by floor, HVAC, lighting, etc). This is currently the largest, most detailed, energy metering project that we are aware of. Although the energy data is reported in real time, no supplemental data (occupancy, light, etc) is collected. This is also a passive system that only enables visualization and analysis of the data collected data.

Understanding energy usage in detail is a difficult challenge. One approach is the AC Meter (ACme) [5, 6]. ACme is a wireless energy meter that uses TinyOS and the Blip IPv6 networking stack with an IEEE 802.15.4 radio. By collecting high fidelity energy measurements the authors dissect a building's energy consumption and identify potential savings. ACme was only used for monitoring, however, it provides a good starting point to design a sensor node for both monitoring and control functions.

Tendril is a commercial company with an offering similar to the ACme that enables demand response [16]. To offer a better demand response capability it monitors the energy usage of individual end devices. This energy usage is reported back to a central server using a ZigBee wireless network in the home. When instructed by the central server the device will turn off to reduce demand. Tendril's service partially matches our goals, however, it does not integrate data from different sensors to make control decisions. Also, because energy usage is monitored and transmitted to a central server through an Internet gateway, this node becomes a significant bottleneck.

A WSN-based conference room management system called iSense [17] has also been demonstrated. This system is similar to ours in that it uses several different sensors to detect when a room is being used. If the room is not being used and the lights or HVAC have been left on, it will alert someone to turn them off.

Our vision is a system similar to the Neural Network House or a Control4 home automation system. However, we prefer to allow the occupant to manually control devices and only enforce energy savings measures (such as turning off devices when the home is unoccupied). A significant portion of the functionality of theses systems is beyond our scope, for example, the Neural Network House might detect that you want to listen to music and even which music you want. The most significant challenge we plan to address is to remove the need for a central controller and of course any wires. In this regard, we will blend in ideas demonstrated by the Energy Dashbord, ACme, Tendril, and iSense to produce a fully functioning home energy monitoring and control system.

2.2 Wireless Sensor Operating Systems

The predominant operating system used for research on wireless sensor networks in the United States is currently TinyOS [18, 19]. TinyOS is a component-based operating system with many useful components for common WSN tasks, such as handling a packet [20]. Other components include those for routing, sensing, actuation, and storage. Applications using TinyOS are written in the network embedded systems C (nesC) [21] programming language. In nesC components are defined through interfaces and "wired" together to create a complete application.

One of the reasons TinyOS has been successful is that it is has a good balance of functionality, flexibility, and simplicity. Basic applications can be implemented with only a few kB of memory, so even the most resource constrained nodes can be supported. On the other hand, sophisticated applications can also be built by combining many components. In short, TinyOS is applicable to solve a broad range of

problems and allows the system designer the choice to use many pre-built components and work at a high level, or to implement functionality at a very low level.

TinyOS, however, is not the only option for WSN operating systems. Contiki, developed by Adam Dunkels of the Swedish Institute of Computer Science, has seen significant use, mostly in the European research community. Contiki is an event-driven system like TinyOS; however, rather than using a component structure, protothreads are used to simulate a threaded environment [22]. The growth of Contiki is partially due to robust support for both IPv4 and IPv6 networking standards.

A more traditional approach is to use existing ideas about operating systems and apply them to the sensor node. Nano-RK [23] and MANTIS [24] are UNIX-like operating systems for WSN nodes. Applications are written in standard C. Each system provides a WSN networking stack with its own API. Because these follow the UNIX programming model closely, developers should be able to get started and write simple applications quickly. However, these operating systems do not support as many hardware platforms as TinyOS and Contiki.

LiteOS [25] is another, more recent, operating system for WSNs. It is also a UNIX-like operating system. A key component of LiteOS is a hierarchical file system and wireless shell interface. This allows the user to access the motes in a familiar way, without requiring any application logic to be implemented by the user. The goal of these features is to get the WSN up and running quickly. Application software can then be dynamically loaded onto the mote as needed. This built-in functionality makes the mote look more and more like a general purpose computing platform, however a price is paid for these features in terms of code size and complexity.

Table 2.1 summarizes the key features of several operating systems proposed for WSNs. For comparison we also list the key features of μC/OS-II which is an operating system commonly used for general embedded systems.

This investigation of WSN operating systems yields an interesting result. We prefer a threaded execution model, which suggests MANTIS, Nano-RK, or μC/OS-

Table 2.1: Survey of Operating Systems for Wireless Sensor Networks

	TinyOS[19]	Contiki[26]	MANTIS[24]	Nano-RK[23]	μC/OS-II[27]
Execution Model	Event (Threads via add-on)	Event and Protothreads	Threads	Threads	Threads
Real-time	No	No	Yes	Yes	Yes
Power Mgmt.	Application	Application	Scheduler	Kernel	Application
Networking	Modular	Modular	Built-in	Built-in	Modular
Current Version	2.1.1	2.2.2	1.0 beta	pre 1.0	2.87
Supported Platforms	8	>20	4	2	>100
ROM (kB)	2.5	40	14	18	10

II. However, MANTIS and Nano-RK are immature research grade operating systems and μC/OS-II has no freely available networking support. TinyOS and Contiki have seen wider use and offer more mature development environments. There are also IP implementations available for both of these operating systems (see Chapter 2.3 for more details). Because of these two factors, TinyOS and Contiki are both good choices for our research. The decision between the two is difficult, but as most WSN researchers in the United States use TinyOS, we will also use TinyOS.

2.3 Networking Architectures

Chameleon [28] is a middleware for implementing networking protocols. Each packet is described by a set of attributes and transformers do the work of taking the attributes and creating a packet for the underlying network layer. This allows network-centric applications written using Chameleon to be portable between various network implementations. Additionally, one can utilize Chameleon to bridge two different networks; such as the IP network and the sensor network. This solves the mechanical problem of transporting packets between two different networks, but does not provide an entire sensor networking platform for application developers.

Some may argue that the primary factor limiting progress in sensor networks is the lack of an overall sensor network architecture [29, 30]. The fundamental components of a WSN node are identified at the physical level as: sensing, energy storage, carrier sense, transmit, and receive. Above the physical level is the data-link level containing: media access, time stamping, coding, assembly, and acknowledgments. He proposed the Sensornet Protocol (SP) to provide a common link-layer and below networking abstraction. This is analogous to the IP layer in computer networks. Applications would then be written for SP and therefore be portable from one physical medium to another. There is no consideration or built in support for bridging the SP network with the IP network. This is conceivable, however, the WSN node would then need to implement the upper layers of the networking stack (application and transport) to

be able to communicate with a standard PC. So as with Chameleon, this approach only addresses part of the problem.

Another approach to build a WSN networking architecture is to simply implement Internet style protocols on the sensor node. With this approach there is no need for a complex gateway to convert packets from the IP network to the sensor network. Removing the gateway node alleviates the problem of the gateway server failing and causing the rest of the sensor network to fail as well. The trade off is a somewhat more complex sensor node. In addition, the added processing that is usually necessary for TCP/IP communications will increase the processing load on each node.

A well known solution for using IPv4 on sensor networks is the μIp [31] stack included with the Contiki [26] operating system. Recently Contiki has also added support for IPv6 sensor nodes [32] using 6lowpan. The 6lowpan project [33] from the Internet Engineering Task Force aims to provide a standard way for sensor nodes to transmit IPv6 packets. The Berkeley IP implementation for low power networks (Blip) [34] is the TinyOS implementation of 6lowpan.

For future building energy monitoring and control application it is clear that open standards, such as 6lowpan, will dominate. Building systems are heterogeneous in nature and must interoperate. The pervasiveness of IP communication makes 6lowpan the most logical choice. However, support for 6lowpan is still in its infancy and implementations are evolving quickly. This creates practical challenges, for example, our work in Chapter 4 was developed for Blip 1.x, which is incompatible with Blip 2.x. As a result, the work in Chapter 7 does not use Blip. The system presented in this chapter can easily be adjusted to use a 6lowpan network layer when a mature implementation becomes available.

2.4 High-Level Abstractions

The Tenet architecture [35] views large scale sensor networks as three tiers of nodes. The highest tier consists of Internet connected users and the lowest tier are

the motes. The middle tier manages nodes on the low level tier and translates data between the upper and lower tier protocols. Tenet treats each mote as a general purpose computing device. Before a node can do anything useful, it must first be tasked by a high-level node. Each task is composed of an arbitrary number of tasklets from a fixed tasklet library. Each sensor node executes its tasks in a virtual machine.

Agimone [36] is similar to Tenet in that it also bridges the sensor network with an IP network. However, Agimone uses the idea of mobile agents rather than tasks. A mobile agent is similar to a task that can move on its own from node to node in the network. The agent must be initially inserted into the network by a high-level user. This has the same effect as to treat each node in the network as a general purpose computing platform where the agent implements the application specific logic.

Concierge [37] is a general purpose middleware for wireless sensor networks following the Open Service Gateway Initiative (OSGi) for resource constrained devices. The OSGi framework is a component model enabling the dynamic deployment of interconnected applications (called bundles). Much like the agent-based systems, application bundles can be delivered to nodes over the wireless interface and managed remotely. This could be utilized to implement distributed sensor network applications. However, the price for this flexibility and powerful features is paid for by high minimum requirements. Concierge requires over 100 kB of RAM and a Java Virtual Machine; neither of which is generally available on current sensor nodes.

Imagine linking large numbers of heterogeneous sensor networks together over the Internet. This would create a Global Sensor Network (GSN) [38, 39]. To realize a global sensor network we must standardize how information is presented and extracted from the sensor network. The approach proposed in GSN uses a gateway node to bridge the sensor network with the higher level network (Internet). In GSN the gateway defines *virtual sensors* using XML configuration information. Each virtual sensor can produce one or more data streams. A virtual sensor's data stream can be read directly by a high-level application, or used as input to another virtual sensor.

Data is accessible to the high-level application by executing queries using a SQL-like query language.

Microsoft also has a project to enable querying and visualization of large-scale sensor networks called SenseWeb [40]. The heart of SenseWeb is centralized a database server that is accessible through a Web Service API. Just as in GSN, a special gateway collects sensor data and pushes the data to the SenseWeb database though a Web Service. Once the data is incorporated into the central database it can be queried and displayed using other APIs. One example monitors the SenseWeb database and plots real-time data on an interactive map called SensorMap [41]. SensorMap provides a simple and effective way to visualize the geographical relationship between different sensors and their data. Because data is stored on a central server with it's own APIs an unlimited number of analysis and visualization applications could be constructed.

The Internet-scale Resource-Intensive Sensor Network Service (IrisNet) [42] is a database-centric approach to large scale, Internet connected, sensor networks. IrisNet focuses on high bit rate sensors, such as webcams. To join IrisNet an XML description of the service is created and published in a distributed XML database. A high-level API provides access to the distributed XML database for accessing each device. As with SenseWeb there is no standard way for low level nodes to communicate; it assumes they will all be Internet accessible and able to transmit and parse XML documents.

These high-level abstractions can be separated into two groups where the first three essentially deal with how to deploy application logic on the sensor network and the last three deal with how to expose the sensor network to the Internet. For use in building energy monitoring, any of these high-level abstractions could be applied. However, we leave this integration for future work and focus our efforts on exploring the lower-level support required by building energy monitoring and control.

CHAPTER 3

CIRCUIT-LEVEL LOAD MONITORING FOR HOUSEHOLD ENERGY MANAGEMENT

The first requirement for any intelligent household energy management system is to be able to accurately measure energy usage in the home. Whole-home energy measurement is cheap and easy to set up because only one sensor is placed where the home connects to the power grid. The collected data can provide useful information for large appliances. However, the only way to monitor the energy usage of smaller devices is to install an energy meter on every device of interest. This creates a very detailed picture of household energy consumption, but requires a lot of additional hardware – one meter per device in the home. This chapter explores an alternative, more practical, approach to monitor household energy usage, including small devices, by using circuit-level power measurements. We have developed and evaluated two algorithms to disaggregate the circuit-level data into device-level estimates. Our evaluation resulted in an average error less than 5.35% for each device in the evaluation. We therefore believe that this approach enables the development of highly intelligent automated energy management systems.

3.1 Chapter Overview

A reasonable perquisite to reducing energy consumption in buildings is a practical method to monitor energy consumption in detail. There are two existing approaches to household energy monitoring: **1)** nonintrusive load monitoring (NILM) where aggregate energy usage is measured by a single meter (e.g. a "smart meter") as power enters the home and **2)** complex instrumentation systems where each device's energy consumption is individually metered. NILM is attractive because it is easy; only one energy meter is required to monitor whole-house energy consumption. The whole-house data is analyzed and step changes in power usage are matched to a database of

loads (e.g. a 500 watt step might represent a refrigerator turning on). This approach works remarkably well for *large* (over 150 watt) loads that operate as simple ON/OFF devices or with very simple operating states such as high/medium/low [43]. Low powered loads and those with a large number of device states, such as a dishwasher, or continuously variable energy usage, such as an electric stove, are very difficult to extract from whole-house measurements. The other approach is to measure separately each load in the home [3, 6, 16, 44]. These approaches can provide very accurate data for each device but would require a significant investment in equipment to monitor every device in a home.

There is currently a renewed interest in understanding and reducing energy usage in buildings. NILM was originally developed in the 1980's at MIT by Fred Schweppe and George Hart. In [43] an eight-step process for NILM is described. The steps are: 1) measure power and voltage, 2) normalize, 3) edge detection, 4) cluster analysis, 5) build appliance model, 6) track behavior in terms of models, 7) tabulate statistics, and 8) appliance naming.

Commercial interest in NILM is also growing as evidenced by the Google PowerMeter [4] Project and a system developed by Greenbox [45], in addition to the established NILM products from Enetics, Inc [46]. In each solution a smart meter reports whole-home energy usage to a server for analysis. The information is processed with a NILM algorithm and then presented to the user through a web-based portal. The exact NILM approach taken by these companies is not publicly available. In fact, exact statistics on the accuracy of most NILM system are scarce, however, in [47] a three-home NILM system was studied and resulted in "75 percent to 90 percent of on/off events" being reported. In [43] "preliminary results suggest that NILM usually reports energy consumption within $\pm 10\%$ of the independent sensors." In [48] three limiting assumptions are identified with the original NILM work: loads are distinguishable, loads are steady-state, and data is batch-processed.

In this chapter we describe two approaches for circuit-level non-intrusive load monitoring that enable detailed and practical household energy monitoring. Both approaches are modeled after the traditional NILM process. However, our first approach uses steady-state levels for disaggregation instead of state changes. Our second method is a purely naive Bayesian approach that considers steady-state levels in addition to state changes. Both of these approaches address the limitations of previous NILM algorithms by **1)** considering steady-state power usage in addition to step changes in stead-state power usage and **2)** using circuit-level energy measurements. Using the historical steady-state energy usage pattern for each device allows us to more precisely classify each edge and eliminate some cases of indistinguishable step changes. By measuring energy usage at the circuit level we can overcome the inability of whole-house NILM to monitor small or variable power devices. This is because a) there are fewer devices on each circuit, and b) high-powered devices (stove, hot water heater, air conditioner, clothes dryer, etc) each receive dedicated circuits and will not overshadow lower-powered devices (TV, radios, cell phone charger, etc).

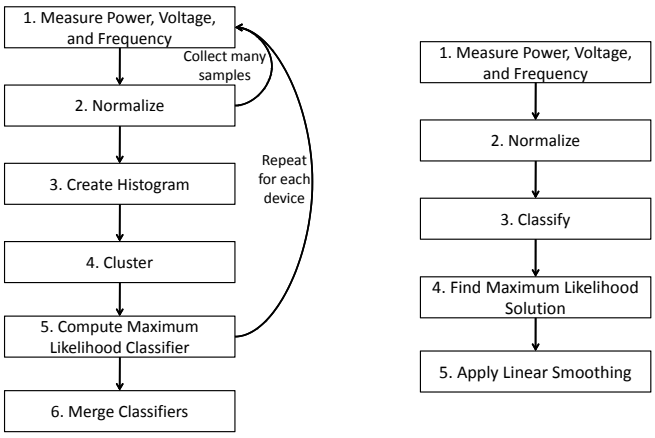
Other researchers are also considering novel ways to extend the basic NILM process. We view these approaches as complementary to ours – each approach extends the capabilities of NILM in a unique way. The ViridiScope [49] couples whole-house power measurements with indirect sensors in the home. The indirect sensors capture additional information (using magnetic, light, and acoustic sensors) that can be use in the disaggregation process. For example, if a light sensor in the kitchen detects an abrupt increase at the same time as the whole-house power consumption increases, we can deduce that the kitchen light has turned on and the observed power increase should be attributed to the kitchen light. Another idea is to detect transient noise caused by devices turning on or off [50]. In this case high frequency sampling is required (100 Hz - 100 MHz) and devices are recognized by their spectral fingerprint. Although both of these approaches are promising, no single approach has yet

been shown to be universally effective (including our own). It is likely that the most effective systems will incorporate ideas from each of these works.

Our long-term goal is the development of a complete household energy management system that integrates both measurement and control functions. For meaningful control, the obvious requirement is to integrate an AC-line switch; however, we believe control of every device is neither practical nor cost-effective. For example, the current EnergyStar criteria for televisions (Version 4.0, effective May 1, 2010) require that they draw no more than 1 watt in *Sleep* mode [51]. Actively disabling an EnergyStar-compliant television would yield very modest energy savings, unlike with older televisions, the most egregious of which can draw 10-20 watts in the *Off* mode. We believe that the energy management system could use circuit-level NILM to identify those devices which should be controlled in order to maximize efficiency while minimizing cost. In addition, if the AC-line switch also included energy monitoring, we could use this data to effectively remove those devices from the unknown aggregate measurements increasing the accuracy on the remaining devices.

3.2 Heuristic Approach to NILM

This approach builds on the basic ideas in the original NILM work, but addresses its weaknesses through several steps to enable more complete device-level energy monitoring. First, we use circuit-level energy measurements. This greatly simplifies the analysis. Because there are only a handful of devices on each circuit, we expect a lower occurrence of indistinguishable devices. Second, our approach uses a probabilistic level-based disaggregation algorithm rather than an edge-based algorithm. Because this algorithm does not require a step change in energy usage to identify devices, we are better able to monitor devices with complex state or continuously variable power usage. While the level-based approach may not perform well with whole-home energy measurements, it is effective here because of the reduced number of devices contained in each circuit-level energy measurement. Training can be done



(a) Training (b) Processing

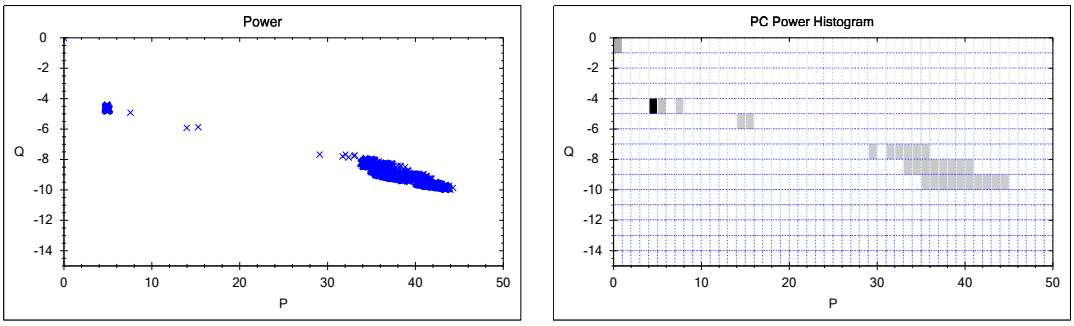
Figure 3.1: Heuristic NILM algorithm.

by monitoring the use of each device as the user turns it on and off or it could be automated by using a control system to switch individual devices and measure the effect on power usage. Both proposed training methods are autonomous and generate samples of each individual device's power usage.

Figure 3.1 shows the steps in our NILM algorithm for circuit-level measurements. *Step (1):* Training (Figure 3.1(a)) and processing (Figure 3.1(b)) both begin by measuring the real (P) and reactive (Q) power, line voltage and frequency once per second. The sampling rate used is the fastest supported by our energy meter [52]. *Step (2):* The power measurements are normalized to a 120V line voltage as in [43]. *Step (3):* Construct a 2-dimensional histogram of the measured power for each device in $P-Q$ space collected during the training period. *Step (4):* Identify clusters that correspond to the different states of the device. Our clustering approach uses thinning of the histogram to compute the location and number of clusters. The thinning process first discards outliers with less than 0.1 percent of the data. The next step is to find local peaks, where all the adjacent cells have lower value. It then increases the value of the peak while decreasing the value of the neighboring cells. The peak-thinning process repeats until every neighboring cell of the local peak has neighbors that are all zero. The number of peaks determines the number of states, K. The (non-thinned) his-

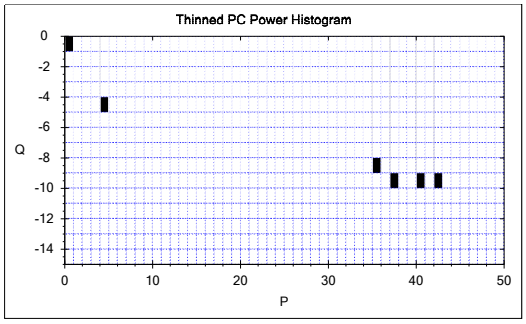
togram value at each peak is the height of the peak. To classify a point we compute the cost to move in a straight line from the unknown point in $P - Q$ space to each peak in the histogram. The cost is the sum of the steps needed to reach the peak. If the height decreases we know there is a different peak that is closer, so we can eliminate the current class as a solution. If we reach a point with zero height we immediately disregard that peak. The lowest cost solution determines which cluster the unknown point belongs to and if there are no peaks reachable from the current location, the point does not belong to any of the device's classes. In this way changes in both $\mathbf{P}$ and $\mathbf{Q}$ are treated equally. *Step (5):* Classify each cell of the histogram and compute the probability mass function (pmf) for each of the K classes. The pmf has the same dimensions as the histogram and represents the probability of being in that cell conditioned on the device being in the state identified by the classifier. The pmf is used in the merge process (next) to compute the maximum likelihood merged classifier. *Step (6):* The merging process creates a classifier by computing all possible combinations of individual device states and their probability. When there is overlap in the merged classifier, the set of classes that maximizes the joint probability is used. One thing to note is that the merged classifier appears blurred when compared to the individual classifiers. This is due to quantization error in the combined measurements. For example, if we have two classifiers with a 1 watt^2 cell size the points in the rectangle with corners at $(0, 0)$ and $(1, 1)$ are placed in the first histogram cell. When we merge the classifier we must account for the fact that this range of values is possible, so the resulting merged range includes the four cells contained by the rectangle with corners at $(0, 0)$, $(2, 2)$.

After the combined classifier is created, we can use it to decompose the circuit-level energy measurement into device-level energy estimates using the process shown in Figure 3.1(b). After measuring and normalizing the power data the next step is to use the combined classifier to classify the most likely state for each device (Step



(a) Measured Data

(b) Histogram



(c) Thinned Histogram

Figure 3.2: Training data for a low-power PC (collected over several hours).

3). Next, we go back to the individual devices' histograms and find the most likely energy usage of each individual device that results in the same total energy as was measured (Step 4). The result is limited to the resolution of the histograms, so we linearly smooth the result so that the sum equals the measured value (Step 5).

Working Through an Example

The measurements and computational steps in creating a classifier for a PC are shown in Figure 3.2. Figure 3.2(a) shows the raw power data plotted in $P-Q$ space. The few samples between the two clusters represent the transient created by switching between standby and active mode. The outlier detection step in the thinning process successfully identified and removed these samples. Figure 3.2(b) shows the histogram of the raw data using a bin size of 1 watt in each dimension. The histogram is then thinned to determine the number of device states. Figure 3.2(c) shows the computed peaks where each peak represents the most likely measurement in each state. This

process is repeated for each device of interest and then each maximum likelihood classifier is computed.

The output of the classifier can now be used to determine which state the device is in. Once the state is known the original histogram is used as an estimate of the pmf *within each state*. We could use the histogram as an estimate of the pmf without constraining the state, however this would bias the estimate to favor the most common state thus make uncommon states highly unlikely. The next step is to generate a classifier for each device of interest and then combine them into a single classifier for the combined energy consumption. This is done by computing every possible combination of each device. The resulting power is summed and the pmf for each device is used to compute the joint probability of the combined solution. The set of classes that maximize the joint probability is saved as the final classification for each cell in the combined classifier.

3.3 Bayesian Approach to NILM

The heuristic method of NILM uses probability estimates based on training data and combines them to disaggregate data. This approach has elements of Bayesian probability. To better understand how this process works we have developed a formalizable purely Bayesian approach. This approach uses a naive Bayes classifier to compute the most likely state of each individual device given a measured aggregate total and detected state change. Our approach is naive because we assume the state of each device is completely independent of the other devices. This is a fair assumption in general; however, for certain devices like a TV and DVD player their operation would be highly correlated. In this approach we use the measured real power, P, as the sole input. This was motivated by the desire to independently train devices and then later combine them. Because the reactive power, Q, does not combine linearly, we cannot train each device independently. This results in less information, there-

fore, to compensate we have added steady-state changes as an additional source of information.

$$S = \{D_1 = s_1, ..., D_n = s_N\} \tag{3.1}$$

To formalize the problem we first define the disaggregated state, S, as in Equation 3.1. Here S is the set of states for each individual device (D_n) where each device is in some known state (s_N). To avoid manually labeling each state, we represent each state as the steady-state power rounded to the nearest 1 watt. Then we let Ω be the state space consisting of all valid states. To disaggregate a power measurement, p, with detected steady state change (edge), e, the problem we must solve is shown in Equation 3.2.

$$\arg\max_{S \in \Omega} Pr\left(S | \sum_{n=1}^{N} D_n = p \bigcap E = e\right) \tag{3.2}$$

Now we apply Bayes' theorem to this probability.

$$Pr\left(S | \sum_{n=1}^{N} D_n = p \bigcap E = e\right) = \frac{Pr(\sum_{n=1}^{N} D_n = p \bigcap E = e)|S)Pr(S)}{Pr(\sum_{n=1}^{N} D_n = p \bigcap E = e)} \tag{3.3}$$

We can then apply the fact that $\sum_{n=1}^{N} D_n = p$ to constrain the search space. Accordingly, this term then is simply 1 and can be eliminated, so the problem to be solved can be re-written as follows.

$$\arg \max_{S \in \Omega: \sum_{n=1}^{N} D_n = p} \frac{Pr(E = e|S)Pr(S)}{Pr(E = e)} \tag{3.4}$$

We compute these probabilities to find the most likely solution. Since the denominator does not depend on S, it can be removed since it is common to all terms; leaving the final classification problem as:

$$\arg \max_{S \in \Omega: \sum_{n=1}^{N} D_n = p} Pr(E = e|S)Pr(S) \tag{3.5}$$

These two terms can be independently computed from training data. We consider the second term, $Pr(S)$, first because this is simply the observed probability of being in a particular state S. Because S actually represents the particular state for each device as $S = \{D_1 = s_1, ..., D_n = s_N\}$ where we have devices $1..N$, and each device is independent, we can compute this as a product of each device separately

as $\prod_{n=1}^{N} Pr(D_n = s_n)$. The first term, $Pr(E = e|S)$, is the probability of seeing a steady-state change (or edge) of size e and ending up in state S.

If we assume that only one device is changing at a time, $Pr(E = e|S)$ is the probability that a single device has changed state by e to end up in S. This probability can be calculated from the training data by looking at the number of occurrences that a change of e ended up in S relative to all changes that ended in S. Let E_s be the set of edges in the training data that resulted in state s. Let I be a function that returns one if a condition is true, and zero otherwise. $Pr(E = e|S)$ under this set of assumptions can then be formalized as:

$$Pr(E = e|S) = \frac{\sum_{i \in E_S} I(e = i)}{\sum_{i \in E_S} 1} \tag{3.6}$$

This analysis shows that in both cases we may train the classifier independently on each device and then use the trained classifiers together to disaggregate a set of devices. This makes it possible to create a database of known devices and then select the particular subset contained in the aggregate measurement of interest.

While not the core topic of this work, we can consider relaxing the assumption that only one device is changing at a time. Let us define a state transition as:

$$T = \{t_1, t_2, ..., t_N\} \tag{3.7}$$

Here T is a transition consisting of the set of power transitions for each of the N devices. In the case of a single device, i changing state only one t_i would be nonzero. The magnitude of an edge defined by a particular transition T would be represented as:

$$e_T = \sum_{i=1}^{N} t_i \tag{3.8}$$

To find the probability $Pr(E = e|S)$, we would find all transitions with an edge of e. For each transition, we would then want to find the probability that that transition resulted in S. Since we trained each device individually, we do not have data on the probability of a transition. Assuming device independence, we can construct the probability of a transition by considering all non-zero device state changes that add

up to e. For example, if T had two nonzero device state changes a and b, with power changes e_a and e_b, we could find the probability of T_S as:

$$Pr(T|S) = Pr(a|S) \cdot Pr(b|S - e_a) + Pr(b|S) \cdot Pr(a|S - e_b) \tag{3.9}$$

This is not commutative as $S - e_a$ is not necessarily $S - e_b$. For the generic case, we must consider every ordering of the individual device transitions of that transition. In other words, if a transition has n nonzero transitions, then there are $n!$ orderings that must be considered.

The exhaustive approach to do this while still being able to train the devices independently requires significant computation. Specifically, finding every combination of transition that has a magnitude of e would need to be found. In the simplest case, the transition for a specific device would have only three possible values: turning on (positive), staying on or off (zero), or turning off (negative). Reducing the problem further to only consider transitions of either turning on, or retaining state, this problem effectively is the subset-sum problem, a known NP Complete problem. As solving this problem would also solve the subset-sum problem, this problem is NP hard. Adding additional transitions per device and adding combinatorics of the devices, this problem will have a worse complexity than just the subset-sum problem. Hence, for anything but a small number of devices that change state, this approach is not tractable. A case might be made for trying both assumptions of two state changes rather than just one, but the generic case is computationally prohibitive.

Working Through an Example

To illustrate the process where we estimate $Pr(S)$ and $Pr(E = e|S)$ we imagine a three-way light bulb. A three-way bulb has two filaments and operates by turning on one filament, the other, and then both. This device has four states: off, low, medium, and high. There may also be a very brief time between two states where the light is actually off. However, because we sample power relatively slowly, this brief off period is not measured. For a 30-50-80 watt light bulb the measured power and edges are

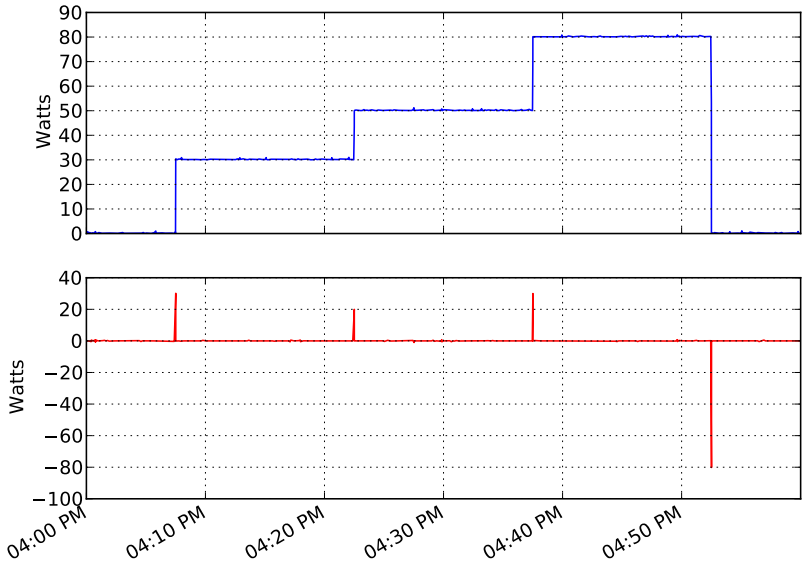


Figure 3.3: Sample 30-50-80 watt light bulb measurements.

shown in Figure 3.3. In this trace the light is in each of the four states for an equal amount of time.

To estimate $Pr(S)$ we construct a histogram of the instantaneous power samples. Because the device spent an equal amount of time in each measured power state: 0, 30, 50, and 80 watts, the estimated probability of being in any one state is $\frac{1}{4}$.

To estimate $Pr(E = e|S)$ we need to compute a probability estimate for all observed state changes (edges) e_M. Because this is conditioned on a particular device state, S, we now compute a histogram for each device state. There were 4 edges detected and 4 distinct steady-state states. In this case the probability is 1 for each edge, because we only observed one distinct edge for each steady-state value. For example, the only edge that resulted in the 50 W state was the +20 W edge. There were two +30 W edges, but they resulted in two distinct states, 30 W and 80 W. The final edge is -80 W which ended in state 0 W.

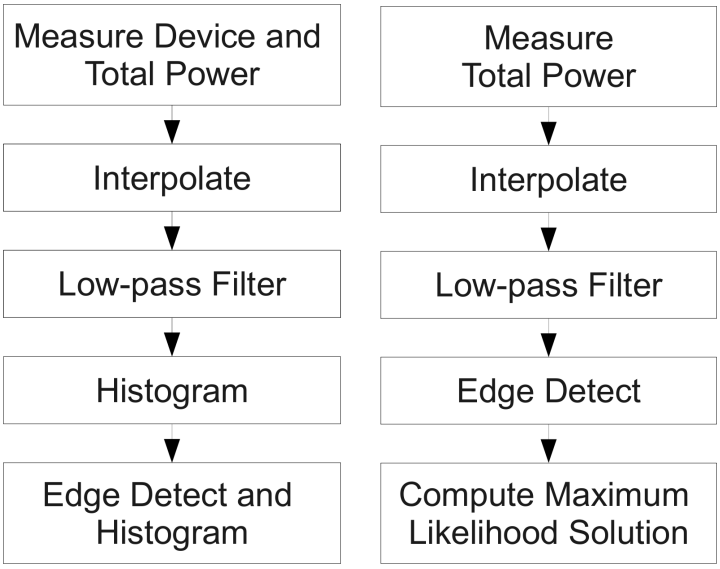
In this example we used a device with obvious states and transitions and used a very precise training procedure. With more complex devices this is not possible and

not necessary for this process to work properly. If we monitor a PC, for example, we see a wide variation over a 10 watt range. This is due to the changing power states of the various peripherals and the changing power requirements of the CPU itself. These complex interactions are difficult to fully understand and control for training purposes. As long as the training period is sufficient to provide a statistically accurate device usage pattern that covers all possible device states, this approach will work.

The final algorithm is shown in Figure 3.4. Both training and processing begin by measuring the aggregate or device-level power. We sample the power at 1 Hz, but it is possible to miss some samples. To correct this problem we use linear interpolation to estimate missing samples. The data is then passed through a low-pass filter to remove high frequency noise. In the training process we then generate a histogram of the power samples using a 1 watt histogram bin. This generates an estimate of the probability density function (pdf) for the device's instantaneous power usage. A histogram is generated for each possible state change in the power measurements. For each detected edge we store the resulting power level for the device. This result allows us to estimate the probability of the device switching into a given power level conditioned on observing a particular edge.

3.4 Evaluation

To explore our approach for circuit-level non-intrusive load monitoring we have conducted several experiments. In each case we use the commercially available power meter called the WattNote from Continental Control Systems, LLC. Each WattNode measures power, voltage, and frequency once per second with $\pm 1.0\%$ accuracy. One WattNode acts as the circuit-level power meter monitoring the aggregate usage of the three devices under test. Each device is then individually monitored by another WattNode. The power used by the WattNodes (< 2 W) is not included in the measured values. These evaluations demonstrate that both of our NILM algorithms are able to disaggregate small (10 W - 30 W) devices using circuit-level measurements.



(a) Training (b) Processing

Figure 3.4: Bayesian NILM algorithm.

This is a significant improvement over existing approaches that only detect devices > 150 W.

Experiment 1 - Long Term Stability

In this experiment we monitor a PC, LCD, and desk lamp for a 24-hour period to evaluate the long-term reliability of each approach under stable conditions. Each device was on for the entire experiment however, we used a random screensaver on the PC. As the name implies the random screensaver selects a screensaver randomly and after a random delay chooses another screensaver. This causes the CPU load to vary from 0 to 50% and the measured power to vary by more than 10 watts. The LCD and lamp show fairly constant power consumption. Because the PC oscillates between many similar states it is more difficult to properly disaggregate than a device with large step changes such as a refrigerator. We use the first two hours of the 24-hour trace for training (and exclude these data from the evaluation). Figure 3.5 shows the measured power for each device. The results from heuristic approach are shown in Figure 3.6 and the Bayesian approach in Figure 3.7.

We use the true device-level power measurements to compute the error of the disaggregation for each method. We see that both methods achieved very good accuracy

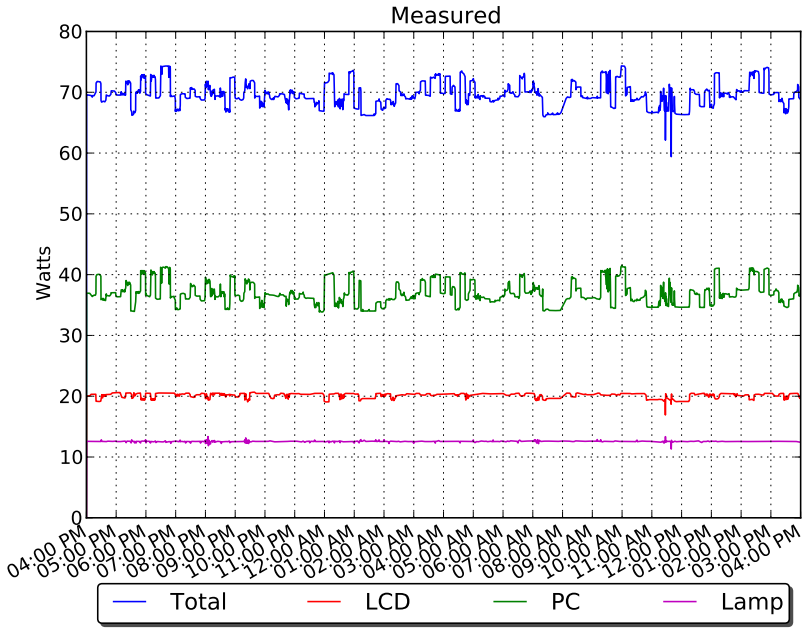


Figure 3.5: Measured power for a 24-hour period.

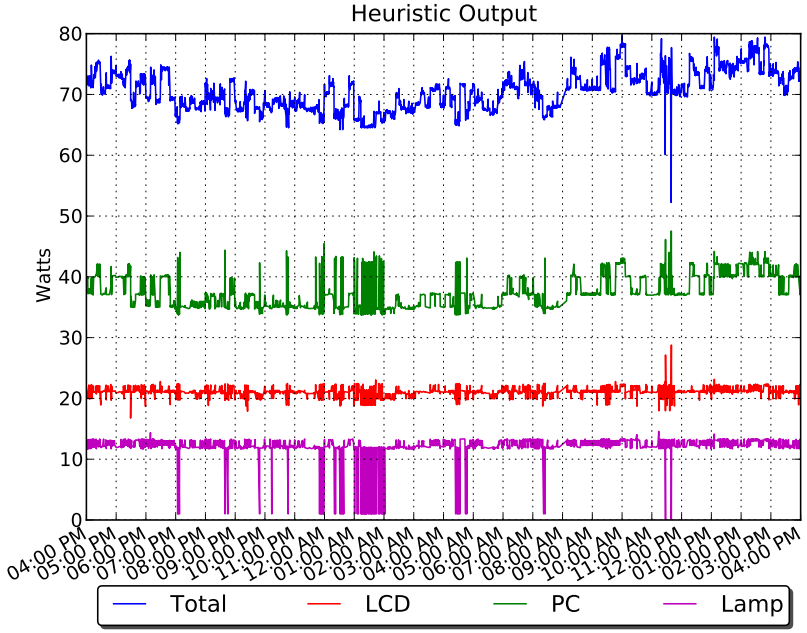


Figure 3.6: Heuristic disaggregation for the same 24-hour period.

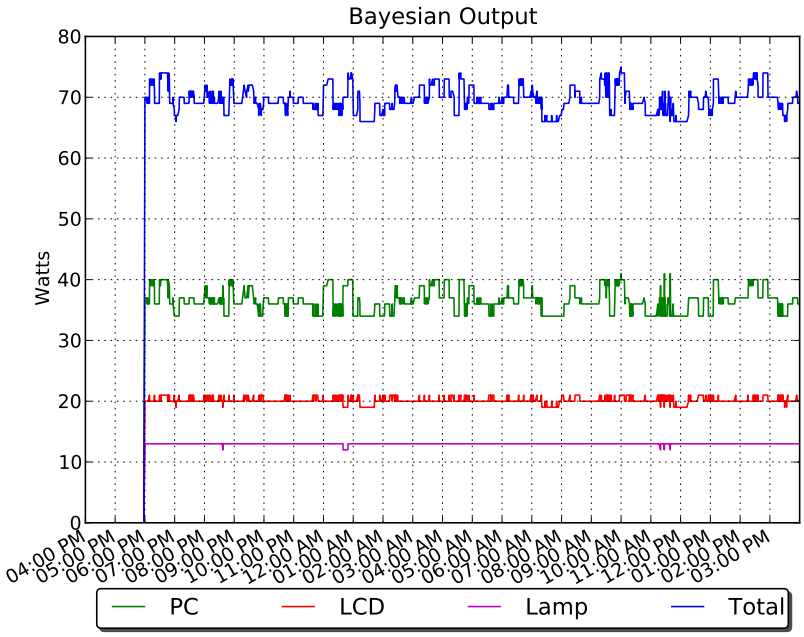


Figure 3.7: Bayesian disaggregation for the same 24-hour period.

over the entire experiment; much better than the ±10% error cited by previous NILM approaches. Overall the Bayesian approach was superior. Because the heuristic approach did not consider edges, it had some difficulty distinguishing between the PC at high power and the lamp off and the PC at low power and the lamp on. In both cases the cumulative power used by these two devices was about 44 W. In this case the state-change information was sufficient to disaggregate these two states. The resulting error rates are shown in Table 3.1, illustrating that circuit-level NILM is able to successfully disaggregate small steady-state loads with high accuracy.

Experiment 2 - Changing States

It is also important to accurately detect and classify devices as they change state. We have previously seen good results with the heuristic approach under changing device states [53] (results summarized in Table 3.1), so we now focus on evaluating the Bayesian approach under changing device states. We replaced the PC from the previous example with a 3-speed fan to increase the number of distinct device states. We collected 6 minutes of training data (Figure 3.8) where we manually cycled each

device through all possible operating modes. Some challenging features in this data are that the LCD in the active state is nearly identical to the fan on low speed. Additionally, when the LCD transitions from active to standby generate transients similar to the lamp turning off or on. As a result in this data neither the steady-state values or state changes alone are sufficient for disaggregation.

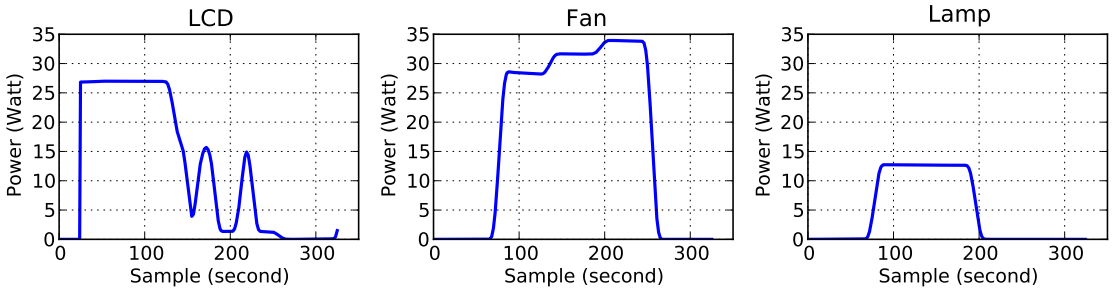


Figure 3.8: Training data for each device.

We then collected data for a one-hour period where we manually changed device states in a different order than was used in collecting the training data. The measured powers and the output of the Bayesian algorithm are shown in Figure 3.9. The experiment started with the LCD in standby and the fan and lamp off. Here we see a small error because the total energy consumption of 1 watt is below our meter's creep limit (the minimum nonzero power). This experiment demonstrates that device state changes are quickly and accurately detected from the circuit-level aggregate measurements. The resulting error rates this experiment is also summarized in Table 3.1.

Table 3.1: Experimental error rates.

	Long Term Stability		Changing States	
	Heuristic	Bayesian	Heuristic	Bayesian
PC/Fan	0.13%	-0.67%	-0.69%	0.74%
LCD	3.10%	0.79%	1.13%	-2.31%
Lamp	-5.35%	2.91%	1.72%	0.73%

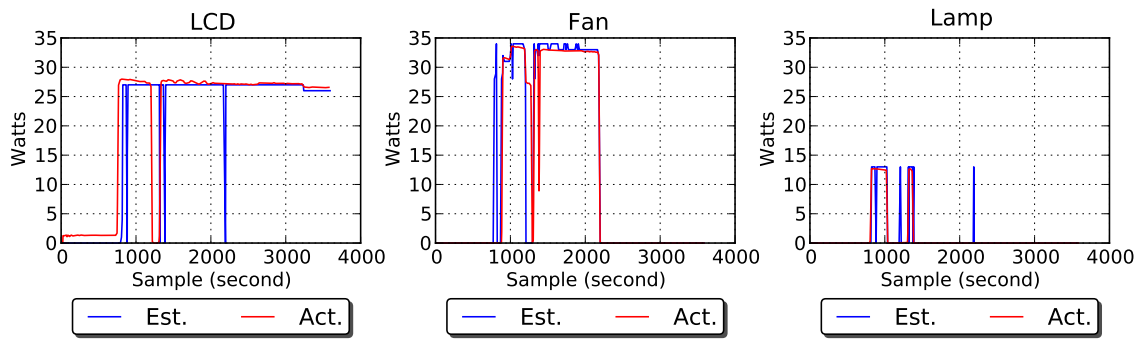


Figure 3.9: Measured and estimated power for each device.

3.5 Chapter Summary

These results are promising; using circuit-level energy measurements we can disaggregate energy usage by low-powered devices. Numerous studies have shown that household energy consumption can be lowered by simply providing real-time aggregate energy usage information [54], we believe that device-level energy information will provide additional useful information allowing occupants to identify wasteful devices and further reduce their consumption. This work represents a small step toward a long-term goal where energy saving behaviors are automated so that the building itself would detect wasteful or unnecessary devices and then disable them based on the sensed occupant behavior.

CHAPTER 4

PIM-WSN: EFFICIENT MULTICAST FOR IPV6 WIRELESS SENSOR NETWORKS

Wireless building energy monitoring and control systems require the robust and efficient communication of sensor data. To provide this, we have developed PIM-WSN, a protocol independent multicast (PIM) protocol tailored for IPv6 wireless sensor networks (WSNs). Multicast IP communication is useful for communicating building sensor data because it eases the task of delivering sensor data to other nodes in the network. Existing solutions for multicast in WSNs are limited because they either support multicast only from a single source node (usually the root node) or they limit the multicast group size to constrain memory usage. Our design allows any node to be a mulitcast source with an unlimited number of subscribers. We constrain the memory usage by approximating multicast group membership using a fixed sized Bloom filter. The efficiency of the protocol degrades as the false positive rate of the Bloom filter increases; however, correct operation is always maintained. Using detailed simulations we show that PIM-WSN achieves 1) high packet delivery rate (over 97%), 2) low latency per hop (less than 5 ms), and 3) lower radio utilization than all other comparable protocols (by more than 50%). Using a ten-hop testbed of TelosB motes we have verified our implementation of PIM-WSN for TinyOS 2.x with the Blip IPv6 networking stack which uses only 5,978 bytes of ROM and 235 bytes of RAM.

4.1 Chapter Overview

Sensor network applications are increasingly being developed that not only passively sense the physical environment but also actively interact with it. In order for this interaction to be more responsive, intuitive, and scalable a distributed approach is essential. Multicast communication is useful for this application because it

allows sensed data to be transmitted directly to multiple receiving nodes. Multicast is heavily studied in IP-based networks, while multicast in wireless sensor networks (WSNs) has focused on specialized protocols with limited functionality (e.g. [55–60]). These limitations constrain the number of nodes in the multicast and which nodes can source multicast traffic to ease implementation on resource constrained sensor nodes. Support is growing for IPv6/6lowpan support in WSNs, however, current implementations do not provide support for multicast. This is because no existing WSN multicast implementations provide the broad support needed for general purpose multicast. Our contribution is a general purpose multicast protocol able to provide multicast for emerging IPv6/6lowpan WSN solutions.

The most common form of multicast is Protocol Independent Multicast (PIM). "Protocol independent" refers to the fact that the it relies on the underlying network routing protocol, rather than implementing its own routing protocol [61]. This is a good match for sensor networks because many routing protocols already exist that have been validated and tuned for a certain application or network topologies. We elect to follow the same, protocol independent, approach for our multicast design and implementation.

In this chapter, we design, implement, and evaluate PIM-WSN, a protocol independent multicast protocol customized for WSNs. The distinguishing features of PIM-WSN are:

- Unlimited number of source and subscriber nodes
- Constant memory usage
- Reliable multicast packet delivery
- No periodic messaging or route maintenance

Through extensive performance studies using carefully designed simulations we demonstrate that PIM-WSN delivers packet delivery comparable to other multicast

approaches (over 95%), low latency (under 5 ms), and significantly outperforms the three other protocols we tested in terms of radio utilization (by more than 50%). We have also implemented and verified PIM-WSN using TinyOS 2.x with the Blip IPv6 networking stack [34]; our implementation requires only 5,978 bytes of ROM and 235 bytes of RAM on the TelosB platform.

4.2 Related Work

Numerous multicast protocols have been proposed for wireless sensor networks. Existing approaches can be classified as: adaptations of mobile ad-hoc network (MANET) protocols, WSN-specific, or geographic. We will not consider geographic protocols (such as [57, 59]) because they rely on extra information (geographic) that we do not. In fact, PIM-WSN used in conjunction with a geographic routing protocol, would effectively create a geographic multicast. We also do not consider the numerous adaptations of link-state protocols (such as OSPF, OLSR, and FRM [62–64]) because this routing paradigm is generally not supported in WSNs.

THe MANET multicast routing protocol Adaptive Demand-driven Multicast Routing (ADMR) has been evaluated for use in wirless sensor networks [55]. The main problem with this (and other) MANET routing protocols is they require storing state for up to every node in the network. In a WSN nodes cannot store this much state information, so the authors explored various policies to reduce memory usage by pruning low quality paths. However, it is not clear if this could result in a disconnection of the multicast. Our solution to this problem may deliver packets to extra nodes but does ensure no node is disconnected (if a path exists). Exploring other MANET multicast protocols does not look like a promising approach. A study of four other common MANET multicast protocols found that they all depend on frequent periodic messaging to maintain multicast routes and do not take any steps to limit memory usage [65]. Both of these issues must be resolved to be useful in WSNs where periodic messaging consumes too much energy and memory is constrained.

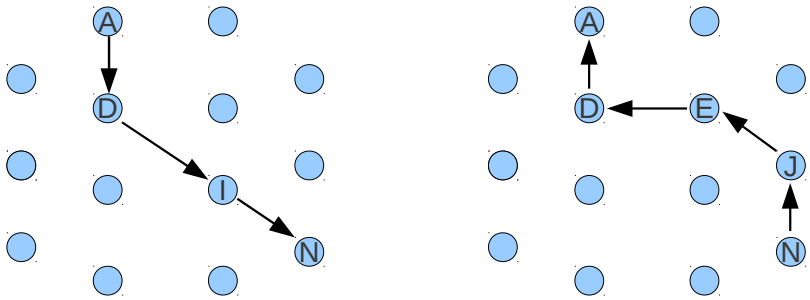
This leaves WSN-specific multicast protocols. A good theoretical analysis of the problem is presented in [56], however, the evaluation is purely theoretical. One of the earliest implementations of multicast in WSNs we found was presented in [60]. This approach bypasses the memory problem by only supporting base to node multicast (not peer to peer) with 8-bit node identifiers. This limits the maximum multicast state, so that even in the worse case, it would fit into a node's constrained memory. In addition, the protocol itself is a standard multicast approach, requiring periodic subscriptions to avoid expiring nodes.

Branch aggregation multicast (BAM) [58] is the only prior work that also uses the protocol independent approach in WSNs. BAM relies on metrics from the routing layer to choose next hop destinations. The significant difference between BAM and PIM-WSN is that BAM appends multicast subscription information to every data packet. This avoids the overhead of sending join messages and storing forwarding information on each node, while increasing the overhead of sending each data packet. This works with short network addressing schemes and a few subscribers, but will not scale well to IPv6 address and more than a handful of subscribers.

4.3 Design of PIM-WSN

Our goal is to keep the protocol as lightweight as possible while balancing memory efficiency, energy efficiency, reliable packet delivery, and low latency. We will first give an overview of PIM-WSN and then describe in more detail how we address the problems that are specific to tailoring PIM-SSM for WSNs.

In PIM-WSN an interested node initiates the multicast data transfer the same way as PIM-SSM, i.e., by sending a unicast *join* message to the source node of the multicast containing the source-group pair (S, G). This message is sent using any available unicast routing protocol. When a source node receives a valid *join* message it responds with a *join acknowledgment* unicast back to the subscriber. The destination node continues sending *join* messages until it receives a *join acknowledgment* or the



(a) A joins N's multicast (b) N acknowledges A's join message

Figure 4.1: PIM-WSN subscription process. Join and join acknowledgement messages may take different paths (as shown). Data flows along the path of the join acknowledgement.

subscription fails after making a maximum number of attempts. After the node receives a *join acknowledgment* it is now a subscriber to the specified multicast and multicast data will be delivered. The subscriber does not need to send any more *join* messages for this multicast (S, G) unless a failure is detected.

The subscription process also serves to inform nodes along the path from the source to subscriber that they must forward packets. This is done by nodes overhearing the *join acknowledgment* message. The multicast source and every node on the path to the unicast destination of the *join acknowledgment* store the address of the next hop towards the destination. These nodes are considered subscribers to the multicast (S, G). This information is later used to forward multicast data packets sent from that (S, G).

Figure 4.1 illustrates node A joining node N's multicast. Arrows indicate the unicast join and join acknowledgement messages. Each node along the path of the join acknowledgement (D, E, and J) updates their local subscription list. For example, node E stores that it must forward multicast packets from N to D. Node D then forwards N's multicast packets to node A. If multiple nodes subscribe to N's multicast, paths to each subscriber are created using the same process. Only one packet is transfered along paths shared by multiple subscribers.

PIM-SSM was designed with wired networks in mind and customizing it for WSN presents several interesting challenges. Three main problems arise with a naive PIM-SSM implementation for WSN. 1) memory usage: storing all the subscription data for each multicast is not possible for a resource constrained devices. 2) reliability: wireless communication is notoriously unreliable; "best effort" delivery works well in wired networks, however, it does not perform well on wireless links. 3) periodic signaling: PIM-SSM keeps forwarding route up to date by each node repeating the subscription process periodically. However, many WSN applications have very low data rates where periodic signaling could dominate energy consumption. Next we illustrate how each of these three issues are addressed in PIM-WSN.

4.3.1 Limiting Memory Usage

We must store subscription data at each forwarding node. Specifically, the triple (D_n, S, G) should be stored at each forwarding node n, where D_n is the next-hop destination of the *join acknowledgment* message along the path from the source to subscriber (S) of mulitcast (group) (G). If we store all the subscription data, the memory usage will quickly exceed the limited storage capacity of resource constrained sensor nodes. On the TelosB storing 213 uncompressed IPv6-style tuples would consume all of the TelosB's memory. This is clearly not an effective approach. Our solution is to use a fixed-sized counting Bloom filter as an improvement to the standard Bloom filters used in [64] to store subscription information.

Because Bloom filters are a probabilistic data structure false-positive results are possible (but not false-negative). As the number of subscribers increases, the false-positive rate also increases. In PIM-WSN, a Bloom filter false positive could result in forwarding multicast data to a node that is not actually subscribed to the multicast. As a result, the efficiency of the protocol is affected. The functionality is still correct because all nodes that are subscribed still receive data. Nodes incorrectly receiving multicast packets can simply drop them. We believe this is a much better

solution than dropping a node from the multicast when memory is exceeded as in other multicast implementations (e.g., [55]).

The counting Bloom filter has three parameters: counter size, number of hash functions, and number of counters. The counter size must be set so that a node is removed after a predetermined number of failures; we use two. If each failure decrements the size by 1, the subscription process must set the initial value to 2. Therefore, we must allocate at least 2 bits per counter. However, to avoid unintentionally removing other nodes that may share common hash values, we should increase this value. Using a counter size on even powers of 2 simplifies the implementation; as a result we use 4-bit counters.

The remaining parameters can be optimized to achieve a certain false-positive rate. Equation (4.1) gives the ideal false-positive rate (P_{fp}) of a Bloom filter as a function of the number of items to be stored in the filter (n), the number of hash functions (k), and the number of counters (m). For a given m and n, the value of k that minimizes P_{fp} is given by Equation (4.2).

$$P_{fp} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \tag{4.1}$$

$$k = \frac{m}{n} \ln 2 \tag{4.2}$$

In PIM-WSN each inserted item is the subscriber triple (D_n, S, G). However, these parameters are application, topology, routing dependent and they are also time varying, so choosing optimized values is nearly impossible. To achieve a constant false-positive rate Bloom filters scale linearly with the number of inserted items. Therefore, memory usage in PIM-WSN also scales linearly with the number of multicast subscribers when the false-positive rate is constrained. However, by allowing the false-positive rate to vary, PIM-WSN achieves the desirable property of constant constant memory usage, even as the number of subscribers increase. Figure 4.2 shows the required number of counters per multicast to achieve a given P_{fp} assuming the

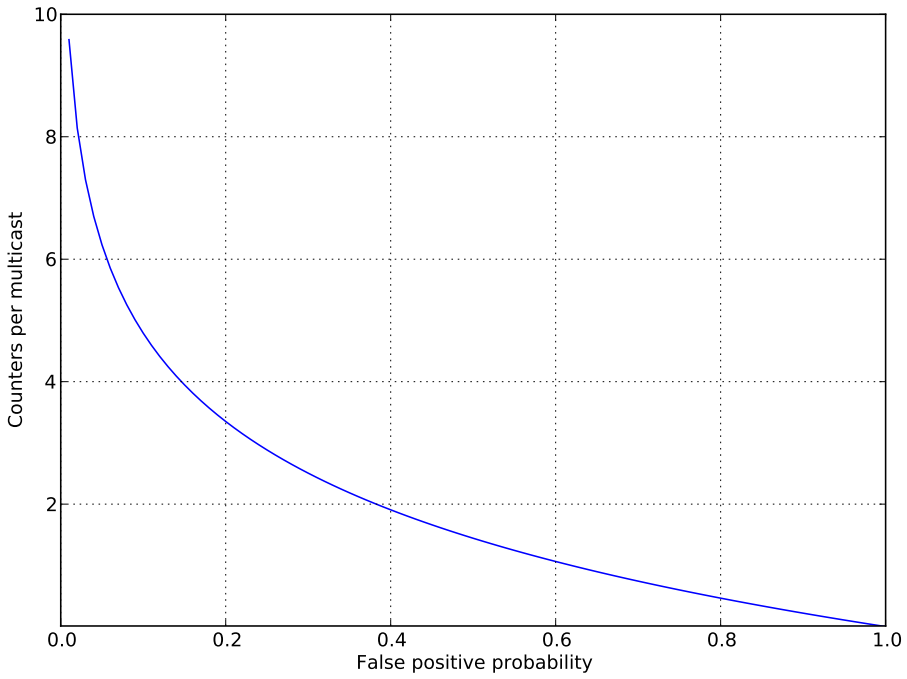


Figure 4.2: The required number of counters per item (multicast) to achieve a given false-positive rate

ideal selection of k. Given that we use 4-bit counters, this translates to a range of 1-5 bytes per multicast to achieve P_{fp}<0.4. The primary advantage of PIM-WSN however, is that we can design the filter for a typical application and the functionality will remain correct even as the false-positive rate varies.

Chapter 4.4.1 explains the values used for our experiments in more detail.

4.3.2 Improving Reliability

PIM-WSN uses one-hop broadcast messages to transfer data to all neighboring nodes at the same time. Common link-layer reliability mechanisms (acknowledgments) cannot be applied to broadcast traffic. Branch aggregation multicast (BAM) [58] also uses one-hop broadcast messages where all neighbor nodes acknowledge the packet. This provides good reliability at the expense of increased energy consumption and congestion. Congestion can be reduced by using delayed acknowledgments, however, this increases latency. Robust broadcast propagation (RBP) [66] proposes

a probabilistic method to achieve reliable broadcasts where a fraction of neighboring nodes is dynamically computed and if at least these many acknowledgments are received the packet is considered to be delivered to all nodes. This improves performance, but still requires every node to transmit an acknowledgement to every packet, resulting in high energy usage and congestion. Our alternative approach is to have the sender designate a single node to acknowledge the one-hop broadcast packet. The address of the designated node is included in the packet header. The result is improved reliability while reducing energy consumption, congestion, and latency.

When a node receives a multicast data packet, it must check if it is designated to acknowledge the transmission. There are two well known ways for a node to acknowledge a data packet: implicitly and explicitly. For improved efficiency, PIM-WSN supports both methods of acknowledgment. The implicit mode is used when the designated node must itself forward the packet; the act of forwarding a multicast data packet acts as an acknowledgment. If a forwarder that is waiting for an acknowledgment receives the same data packet from the node designated to provide the acknowledgment, it knows the packet has been received and may stop retransmitting. There is of course no guarantee the original sender will hear the message being forwarded. So, the original forwarder will retransmit the packet with the same designated node. Whenever a node receives a duplicate data packet and it is the designated acknowledgment node, it will send an explicit acknowledgment to the sender.

4.3.3 Eliminating Periodic Messaging

Proactive route maintenance is used in PIM-SSM to keep forwarding routes up to date. However, if data is sent infrequently, maintaining the forwarding state can dominate energy consumption. Because low data-rate applications are common in WSNs, we instead reactively update multicast routes. If a multicast delivery failure is detected (which is possible because there is a reliability mechanism), we then notify subscribing nodes that a rejoin is necessary. A rejoin is accomplished by the

subscribed nodes on failed forwarding paths rejoining the multicast. This process creates new forwarding paths using current routing information.

A rejoin is triggered after a packet is retransmitted a maximum number of times. To signal this event, the multicast data message is encapsulated in a unicast packed and then sequentially unicast to each subscribed neighbor node. This relies on the unicast routing protocol to route the message (possibly over multiple hops) to the destination node(s). The counting Bloom filter is then decremented on the forwarding node for each (D_n, S, G) triple so that after several failures the node will be removed from the filter. If a node receives a unicast encapsulated multicast packet it assumes there was a delivery failure and repeats the subscription process. If a node receives a unicast data message and it is also a forwarding node it resumes multicast forwarding and follows the usual procedure using one-hop broadcast messages. This localizes the failure to only the effected link, improving efficiency.

In our evaluation we do not send periodic join messages and rely entirely on fault detection. However, it is possible that the unicast data messages may also be lost. For example, if a node became disconnected from the network for an extended period of time. To ensure the correct operation of PIM-WSN it is necessary to detect network disconnect/reconnect events and resubscribe to every multicast.

4.4 Performance Evaluation via Simulation

Our goal is to evaluate PIM-WSN under real world conditions, which are hard to replicate using traditional network simulators like TOSSIM, OmNet++, and ns2. Therefore, we use connectivity traces from real wireless sensor network deployments. Because real network traces are used they inherently contain real-world packet loss and noise. Our simulator, WsnSimPy, is a trace-based simulation tool built on the discrete event simulator SimPy[1]. For validation of the simulation tool, we direct readers to our previous work [67].

[1] `http://simpy.sourceforge.net/`

The connectivity traces are from the "soda" dataset, collected by Ortiz and Culler in a UC Berkeley office space[2]. They were obtained as follows. 46 IEEE802.15.4-compliant TelosB motes are deployed in a 50m x 50m indoor environment, and are constantly listening for packets. One after the other, each mote transmits a burst of 100 packets with a 20 ms inter-packet time and a transmission power of 0 dBm on each of the 16 frequency channels. Timers are used to ensure that all nodes switch channels simultaneously.

4.4.1 PIM-WSN Setup

There are several questions about PIM-WSN that we must now answer. They are: 1) How is the designated acknowledgment node selected? 2) When is a link failure assumed? 3) How is the bloom filter configured? First, we use the weakest link acknowledgment policy. The subscribing node with the worst routing metric (but not worse than a preset threshold) is designated to provide the acknowledgment. Second, we assume a link failure after a multicast packet is transmitted four times without acknowledgment. The rationale to use a relatively low value here is to avoid multicast forwarding paths with poor links. This is useful in our simulation because the network is well connected; i.e. removing links with ETX > 4 still results in a well connected network. In sparse deployments, this might generate excessive subscription traffic if the only available links have ETX > 4. In general we could dynamically assign this threshold based on local network conditions.

Third, we must define the parameters for the Bloom filter used to hold subscription information. Selection of filter size is application specific because it should be large enough to achieve low false-positives with the expected number of multicast subscribers. For our evaluation we set the filter size to achieve good performance with each node subscribing to one multicast. Our simulation has 43 nodes; therefore, we want a low false positive rate after 43 insertions. Solving for the Bloom filter false

[2] `http://wsn.eecs.berkeley.edu/connectivity/`

positive rate, $P_{fp} = 0.1$ with n=43, yields m=206 and k=3 (rounded to the nearest integer). Each counter is 4-bits, so the total memory requirement is 103 octets. This is an awkward size to use in practice so we round down to the nearest word-aligned value and use a fixed-sized Bloom filter of 96 octets.

4.4.2 Simulation Results

For each simulation run we fix the number of source nodes and number of subscribers per source. Each run is assigned a unique value to seed the simulator's random number generator (Mersenne twister [68]). Source nodes are selected randomly without replacement. Subscriber nodes are selected independently for each source without replacement. This does not prohibit a node from being a subscriber to multiple (different) multicasts, or a source of one multicast being a subscriber to one or more other multicasts.

The performance of PIM-WSN is benchmarked against branch aggregation multicast (BAM), simple flooding, and sequential unicast communication. BAM was selected because it is the only other protocol independent multicast approach implemented in WSNs. In addition BAM uses optimized multicast delivery trees and claims to be "very energy efficient." Flooding has the advantage of no setup costs and optimal routing (all routes are used, therefore the optimal route must be used). Finally, comparing to sequential unicast allows us to explore the minimum number of subscribes needed to make multicast a practical solution.

Each protocol was implemented on the same simulated IPv6 stack (similar to the one provided by Blip). BAM had to be modified to work with the routing protocol (RPL) because each node only maintains one route (to the root) in RPL while BAM requires each node to maintain routes to every other node in the network. We solve this problem by initially unicasting the multicast packet from the source to the root. The root maintains a complete routing table and therefore can then implement the BAM algorithm. The computed multicast tree is encoded in the packet header and

sent down the tree to the subscribers following the original BAM protocol. Simple flooding is implemented by modifying PIM-WSN to include a multicast group where every node is implicitly subscribed. This retains the one-hop reliability mechanism of PIM-WSN, where every node will forward each packet once, while removing the subscription overhead. Finally, for our unicast implementation, each source node maintains a list of subscribers that is initialized at the start of the simulation. When transmitting a multicast packet, it is unicast sequentially to each subscriber with a 25 millisecond delay between transmissions. We experimentally found that a 25 millisecond delay was sufficient to avoid collisions due to congestion at the source.

For each simulation we collect packet delivery ratio (PDR), latency, average number of hops, and radio utilization. PDR is computed as ratio of the number of received data packets to how many should have been received (which is 100). In the case of flooding, where every node should receive the packet, we only count receptions by a node if it was selected as a multicast subscriber. Latency is computed by time stamping the multicast packet when it is first transmitted and comparing this to the current time when it is received. In the case of unicast, where packets are transmitted sequentially, the transmission time is the time of the transmission to the first subscribing node. Average hop count is computed by storing the hop count contained in the first reception of each packet if duplicate packets were received. Because latency is strongly dependent on hop count (which may vary from simulation to simulation) we divide the latency by the hop count for comparison. Average radio utilization is computed as the average percentage of the time that the radio was transmitting or receiving. The radio utilization relates the efficiency of each protocol and represents a lower bound on the radio duty cycle using an ideal low-power listening strategy. However, this is not necessarily a good indicator of node energy consumption, because it does not take into account timing or CPU energy consumption. Radio utilization includes every node in the network while PDR, latency, and hop counts are computed

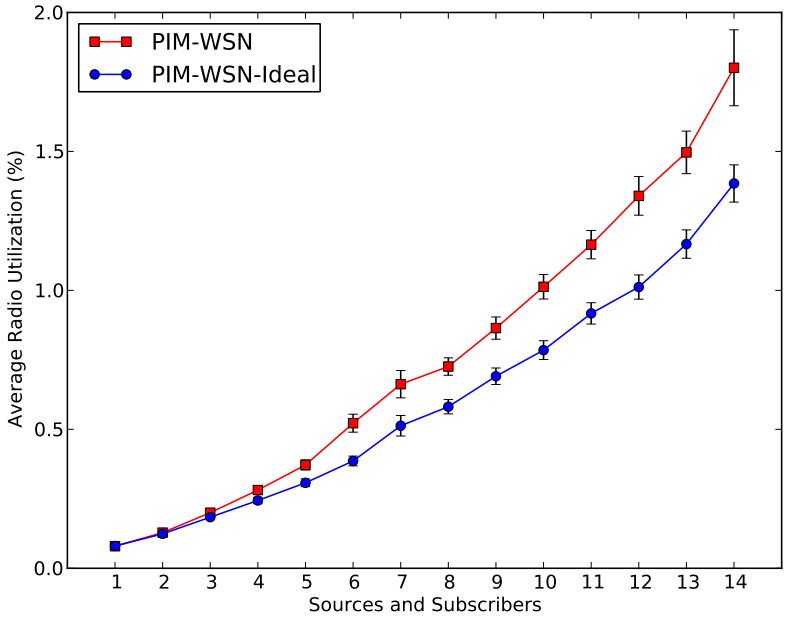


Figure 4.3: The effect of Bloom filter errors.

only on subscribing nodes.

We repeat each trial 20 times with 100 data packet transmissions from each source sent at a rate of 32 packets per minute (PPM) unless otherwise noted. The results are averaged and 95% confidence intervals computed and plotted when visible.

Figure 4.3 shows the effect of bloom filter false positives on PIM-WSN. For this experiment we compare PIM-WSN to PIM-WSN-Ideal. PIM-WSN-Ideal is implemented with an unlimited buffer for storing exact subscription information on each node. As a result there are no false positive results, which result in the unnecessary forwarding of packets. In these experiments we increase both the number of sources and subscribers per source; always keeping the number of sources and subscribers per source equal. For example, if there were 5 source nodes each source also has 5 subscribers. This is necessary to increase the load on the bloom filter and therefore create more false positive results. PDR and latency is not shown because there was no significant effect.

Because false positive Bloom filter results cause PIM-WSN to forward packets unnecessarily, we expect to see an increase in average radio utilization as the number of sources and subscribers increases. Both algorithms had nearly identical radio utilization when there were few sources and subscribers (due to little unnecessary forwarding). When the number of sources and subscribers increases, the number of insertions into the bloom filter also increases, resulting in a higher rate of false positive results. These false positive results cause unnecessary forwarding, which increases the radio duty cycle. In the case of 14 sources and subscribers there are 196 (source, subscriber) pairs that could be inserted into the bloom filter of each node. With this number of insertions into the Bloom filter the false positive rate could be as high as 86.7% (if every multicast went through a single node). The result of these false positives resulted in an increase in radio utilization by 29%. This demonstrates the trade-off PIM-WSN makes to achieve constant memory usage.

Figure 4.4 shows the result of one source node sending data while increasing the number of subscribers from 1 to 14. All methods achieve good PDR; over 95% in all cases. The average PDR over all experiments were 100%, 99.62%, 98.67%, and 97.67% for flooding, BAM, unicast, and PIM-WSN respectively. Flooding is able to do well because our network is rather dense, so there are multiple opportunities for a node to receive each packet. Although PIM-WSN had the worst PDR, we believe 97.67% is good enough for many applications and is very energy efficient. Losses in PIM-WSN arise when a node is forwarding to multiple one-hop neighbors. Because PIM-WSN designates only one node to acknowledge each data packet, it has no way to detect if a non-designated node fails to receive the packet. We could improve the PDR of PIM-WSN by requiring every subscribed neighbor node acknowledge each multicast packet as in BAM. However, BAM achieves only a 2% improvement in PDR compared to PIM-WSN while radio utilization increases by 100%. The best solution might be to dynamically select a variable number of subscribed neighbor nodes to acknowledge

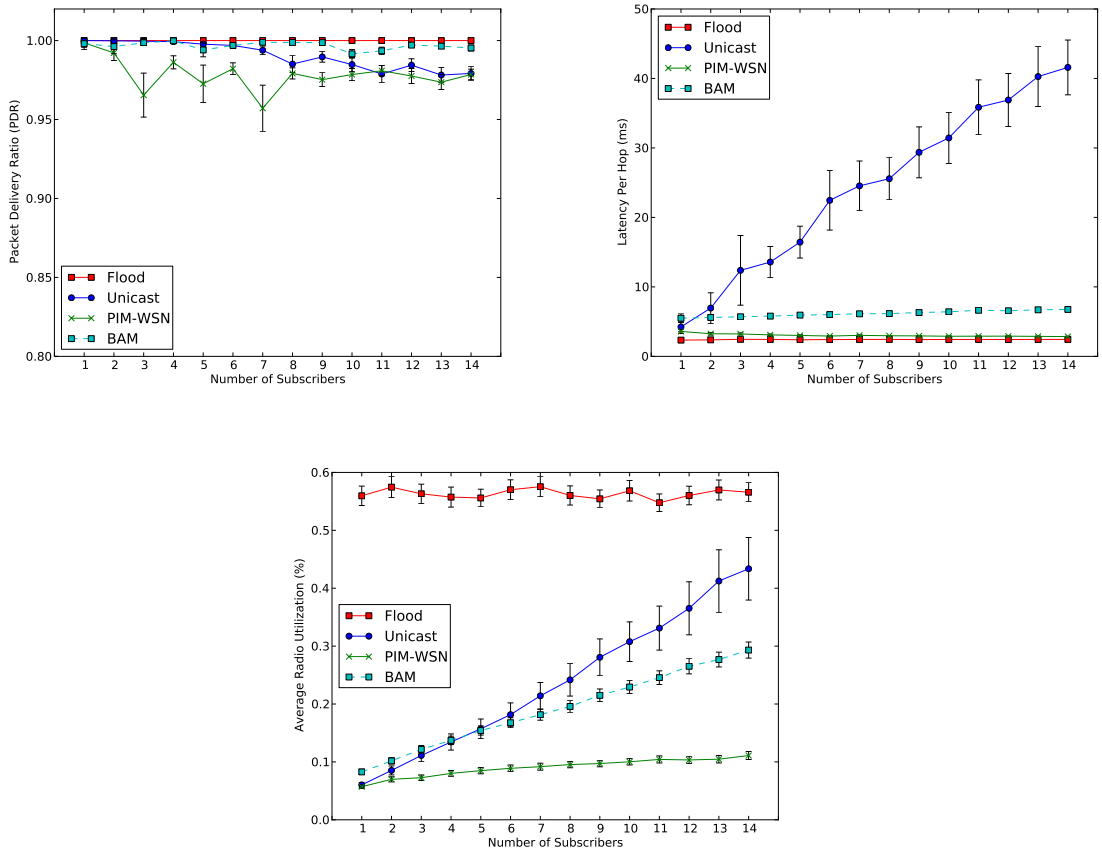


Figure 4.4: The effect of increasing the number of subscribers.

each packet based on the local network conditions; [66] explores this idea for broadcast traffic.

Radio utilization in PIM-WSN was lower than all other protocols, even in the case of just one subscriber. On average it was 52.66% lower than BAM, 61.11% lower than unicast, and 83.99% lower than flooding. Although PIM-WSN has the additional overhead of joining the multicast, it reduces packet transmissions by using a single designated acknowledgement node and by using overhearing for implicit acknowledgments whenever possible. The setup costs happen once while the savings on data transmissions occur every time a packet is sent. The scaling of PIM-WSN is also better than either unicast or BAM. This is interesting because PIM-WSN does not attempt to use optimized routes while BAM does. We see higher radio utilization

in BAM because of two reasons. 1) it does not have a join phase where routes are setup, rather the route is encoded in every multicast packet and as the number of subscribers increases the amount of routing overhead also increases. 2) BAM requires every neighbor node acknowledge each packet. The net effect is that BAM had surprisingly high radio utilization. In fact, unicast was better than BAM until there were 6 subscribers. Flooding had the highest radio utilization and did not depend on the number of subscribing nodes, because every node receives every packet regardless of whether it subscribed or not. If we extrapolate these trends linearly, unicast will surpass flooding at 21 subscribers and BAM at 31; PIM-WSN remains more efficient than flooding.

Flooding and PIM-WSN both achieve very low per hop latency (even better than unicast). Because every node forwards every packet in flooding, it must therefore be forwarded along the optimal path for each packet. PIM-WSN does well because it uses implicit acknowledgments to reduce the overhead of forwarding a packet. The latency of PIM-WSN is the only one to decrease as the number of subscribers increases. This is due to the relative number of implicit acknowledgments increasing; PIM-WSN effectively approaches the case of no explicit acknowledgments, as the number of subscribers increases. BAM has higher latency because of two reasons: 1) in our implementation the packet must first be unicast to the root and then multicast to the subscribers; and 2) BAM uses delayed acknowledgments because every subscribed neighbor node must send an acknowledgement.

Figure 4.5 shows the result of increasing the number of sources while holding the number of subscribers per source constant at 5. This is a direct analog to increasing the number of subscribers. As expected the results follow the same pattern. One notable difference is that the latency of unicast does not depend on the number of sources because each source is independent. The radio utilization of all algorithms increases approximately linearly because no aggregation is performed by any of the

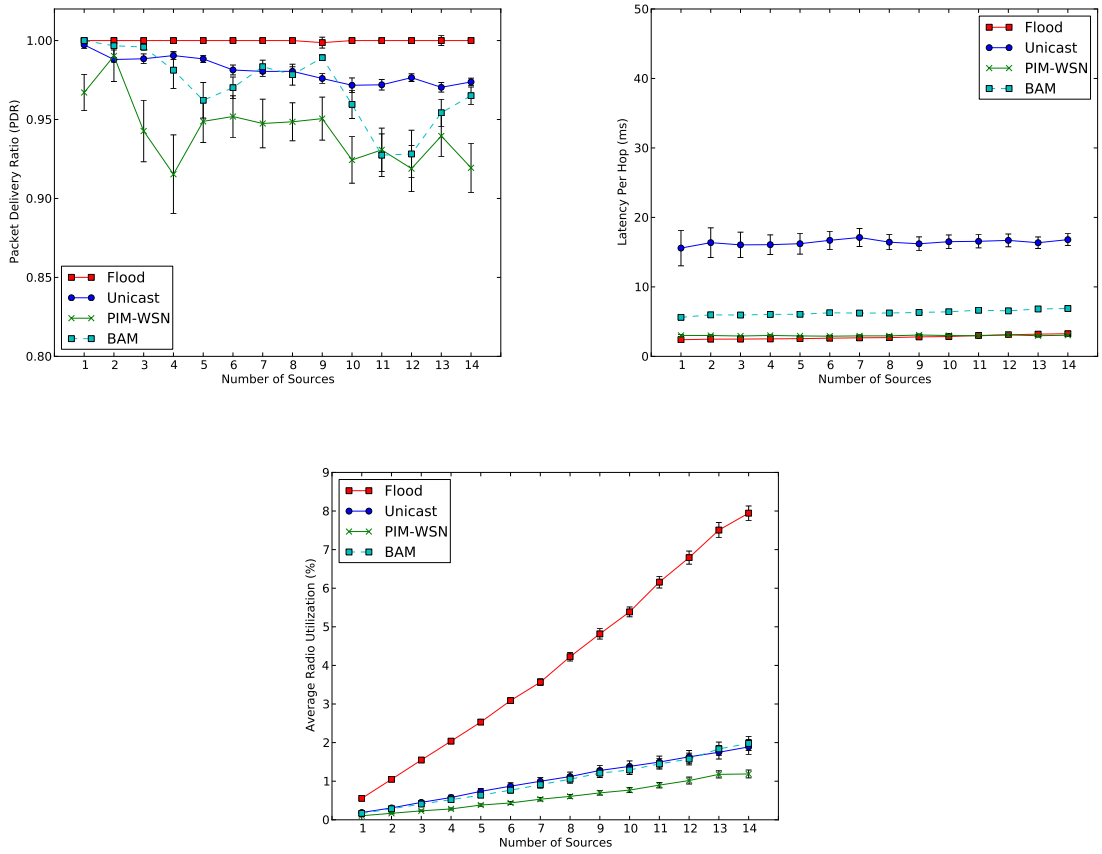


Figure 4.5: The effect of increasing the number of sources.

algorithms between multicast packets from different sources. This result demonstrates that PIM-WSN can efficiently handle multicast data from many sources.

In this experiment we use five source nodes each with five subscribers and then vary the periodic transmission rate to achieve the desired data rate where each data packet contains a 30 byte payload (as per Chapter 4.4). The results are shown in Figure 4.6 and are consistent with the previous cases. Flooding achieves the best PDR. PIM-WSN yields similar PDR to unicast while BAM experiences a steep drop after 120 bytes per second per source. This highlights the weakness of routing every packet through a single node as it quickly becomes a bottleneck.

We have previously seen the lowest latency with flooding, however, here we see a steep increase. Considering that the radio utilization also grows rapidly, the most

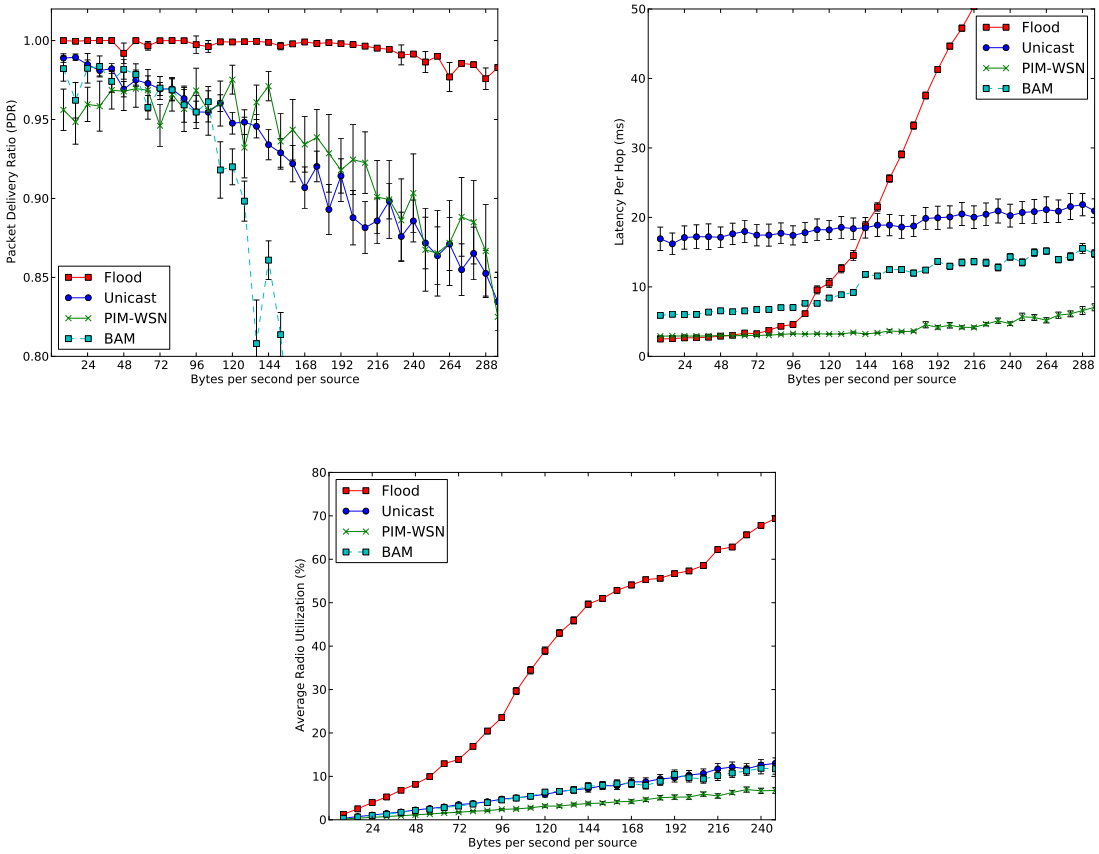


Figure 4.6: The effect of increasing the datarate.

likely cause is congestion. These results demonstrate that PIM-WSN can be used equally well in both low and high data rate applications.

4.5 Implementation

To implement PIM-WSN we use TinyOS 2.x with the Blip IPv6 networking stack as base. Because Blip is still a work in progress, PIM-WSN is implemented above the IP layer. This isolates PIM-WSN from changes in the Blip networking stack. In IPv6 multicast is indicated by the destination address of the packet matching the prefix `ff00::/8`. Instead we use the protocol (next header) field of the IP interface to indicate a multicast packet. PIM-WSN wires to this protocol on the IP interface provided by Blip. PIM-WSN data packets are sent with the IP destination address

`FF02::1` which is translated by Blip into the IEEE 802.15.4 broadcast address `0xffff`. All PIM-WSN packets are sent with a common 20-byte header which could be reduced to 10-bytes by removing diagnostic fields.

Our implementation of PIM-WSN on the TelosB platform requires 5,978 bytes of ROM and 235 bytes of RAM. In a 10-hop linear network with one source and one subscriber PIM-WSN delivered 96.76% of the packets. Repeating the experiment using UDP yielded a comparable delivery rate of 97.63%. The latency (measured via GPIO pins) averaged 326.8 ms from end-to-end for PIM-WSN while UDP averaged 253.6 ms, or a difference of 7.32 ms per hop. The increased latency is because the reliability mechanism in PIM-WSN is implemented on top of IP. The IP interface design forced us to use a longer resend delay because it hides all packet queuing and processing delays at and below the IP layer. Blip implements reliability at layer 2 and is able to use a 15 ms resend delay. Because we do not have access to the internal state of the IP protocol, we were use a 30 ms resend delay to account for the unknown processing and queuing delays. This results in PIM-WSN having slightly higher latency than UDP in practice, however, the primary reason to use PIM-WSN is to enable multicast communication. These experiments are highly favorable to PIM-WSN–they show that even in the worse case of one subscriber there is only a very small performance penalty when compared to native UDP. In our future work, we plan to better integrate PIM-WSN with the IP layer to completely eliminate this problem.

4.6 Chapter Summary

This chapter presented PIM-WSN, a protocol independent multicast routing protocol tailored for IPv6 wireless sensor networks. The primary challenge to implementing multicast in this domain is efficiency in terms of memory and energy. Our novel design addresses the memory problem by approximating multicast group membership

using a fixed sized Bloom filter. Energy efficiency is addressed by modifying the procedure for maintaining the multicast state. Our approach does not remove subscribers due to timeout (thus requiring periodic rejoining) but rather on delivery failure. In the event of delivery failure, PIM-WSN falls back to unicast routing and signals the node to rejoin the multicast. This gives nodes high confidence that they will remain in the multicast even after the failure of multicast forwarding paths. PIM-WSN has been thoroughly evaluated using trace-based simulation and validated on a wireless sensor network test bed of TelosB motes. Testbed results validate the functionality of PIM-WSN and demonstrate that it performs well in practice.

CHAPTER 5

BUILDING THE CASE FOR AUTOMATED BUILDING ENERGY MANAGEMENT

Energy consumption due to buildings comprises a significant portion of our national energy consumption and can be strongly influenced by occupant behaviors. To explore the quantitative effect of occupant behaviors on building energy consumption, we have evaluated eight energy-saving behaviors, as well as the use of an in-home display (IHD), in ten homes over the course of ten weeks. The results showed maximum savings ranging from 0%-20% attributed to the IHD. Additionally, we found evidence that automation is necessary to ease the more tedious tasks such as "unplug when not in use" and "unplug the TV," where less than half of the participants performed the action.

5.1 Chapter Overview

The U.S. Department of Energy (DOE) reports that buildings accounted for 39% of the total energy consumption in the U.S. in 2009 [1]. Because energy consumption is closely tied to occupant behavior, numerous building energy monitoring systems have been recently developed [3–5] that provide homeowners with real-time electrical consumption data to enable more informed energy use choices. However, it is clear that monitoring alone does not always result in savings: For example, a recent study observed an initial 31.9% reduction in energy consumption immediately after installing a monitoring system, then after a month the reduction fell to only 3.7% [6]. This suggests that while significant savings are possible, relying on occupants to change their long-term behavior may be difficult. For significant and *persistent* energy savings, an automated building management system (BMS) combined with a judicious choice of control strategies is required. This chapter presents findings from a ten-week field investigation of the efficacies of various energy-saving control strategies.

The experiment was a simple one: We monitored whole-house energy consumption while occupants manually implemented each energy-saving strategy. To the best of our knowledge this is the first study of its kind. Although our sample size and duration were limited, our results indicate that a much larger study of this type is both feasible and likely to result in useful findings.

5.2 Analysis of Energy Saving Behaviors

There are numerous opportunities to save energy in a building. Some possibilities are to increase insulation, replace old windows, or to buy energy efficient appliances. These approaches are effective, however, they could be costly and do not consider occupant behaviors—which can significantly impact energy consumption. To explore how changing behaviors can reduce energy consumption we first consider to recommendations made by the Natural Resources Defense Council (NRDC) to reduce household energy consumption [69]. However, it is not clear if we were to adopt some or all of these behaviors, what would the quantitative impact on energy consumption be? To answer this question we have slightly modify the NRDC suggestions to separate each action into a distinct components, listed in Table 5.1, and then evaluated the effect of these behaviors in 10 homes over 10 weeks.

Table 5.1: Suggestions for saving energy (adapted from NRDC recommendations)

Short name	Description
Unplug When Not In Use	Unplug seldom-used appliances, like an extra refrigerator in the basement or garage that contains just a few items. You may save around $10 every month on your utility bill. Unplug your chargers when you're not charging. Every house is full of little plastic power supplies to charge cell phones, PDA's, digital cameras, cordless tools and other personal gadgets. Keep them unplugged until you need them.

Table 5.1: (continued)

Short name	Description
Unplug the TV	Use power strips to switch off televisions, home theater equipment, and stereos when you're not using them. Even when you think these products are off, together, their "standby" consumption can be equivalent to that of a 75 or 100 watt light bulb running continuously.
Set Computers to Sleep and Hibernate	Enable the "sleep mode" feature on your computer, allowing it to use less power during periods of inactivity. In Windows, the power management settings are found on your control panel. Mac users, look for energy saving settings under system preferences in the Apple Menu. Configure your computer to "hibernate" automatically after 30 minutes or so of inactivity. The "hibernate mode" turns the computer off in a way that doesn't require you to reload everything when you switch it back on. Allowing your computer to hibernate saves energy and is more time-efficient than shutting down and restarting your computer from scratch. When you're done for the day, shut down.
Take Control of Temperature	Set your thermostat in winter to 68 degrees or less during the daytime, and 55 degrees before going to sleep (or when you're away for the day). During the summer, set your thermostat to 78 degrees or more. Set the thermostat on your water heater between 120 and 130 degrees. Lower temperatures can save more energy, but you might run out of hot water or end up using extra electricity to boost the hot water temperature in your dishwasher.
Use Sunlight	Use sunlight wisely. During the heating season, leave shades and blinds open on sunny days, but close them at night to reduce the amount of heat lost through windows. Close shades and blinds during the summer or when the air conditioner is in use or will be in use later in the day.
Tweak your Refrigerator	Set your refrigerator temperature at 38 to 42 degrees Fahrenheit; your freezer should be set between 0 and 5 degrees Fahrenheit. Use the "power-save" switch if your fridge has one, and make sure the door seals tightly. You can check this by making sure that a dollar bill closed in between the door gaskets is difficult to pull out. If it slides easily between the gaskets, replace them.

Table 5.1: (continued)

Short name	Description
Use Appliances Efficiently	Don't preheat or "peek" inside the oven more than necessary. Check the seal on the oven door, and use a microwave oven for cooking or reheating small items. Wash only full loads in your dishwasher, using short cycles for all but the dirtiest dishes. This saves water and the energy used to pump and heat it. Air-drying, if you have the time, can also reduce energy use. In your clothes washer, set the appropriate water level for the size of the load; wash in cold water when practical, and always rinse in cold. Clean the lint filter in the dryer after each use. Dry heavy and light fabrics separately and don't add wet items to a load that's already partly dry. If available, use the moisture sensor setting. (A clothesline is the most energy-efficient clothes dryer of all!)
Turn Down the Lights	Don't forget to flick the switch when you leave a room. Use fewer lights. Just turn on the lights nearby instead of having all the lights on in a room. Use sunlight whenever possible.

5.2.1 Experimental Setup

Using randomly selected participants from the National Renewable Energy Laboratory (NREL), our goal was to quantify the effect of each of these suggestions on energy consumption. Each household was assigned a different behavior from Table Table 5.1 for each week of the study, including a "control" week where no special behavior was assigned. The behaviors were assigned in a different order for each household, so that for each week only one household was following any particular behavior. The participants were informed that the purpose of the experiment was to evaluate the energy savings potential of each behavior. They were given instructions based on the NRDC recommendations and asked to strictly follow the behavior as described. Because the participants were selected from NREL, we expect them to highly capable and willing to implement the behaviors.

At the start of the each week the participants were given their new behavior and reminded to stop performing the behavior from the previous week. We also asked

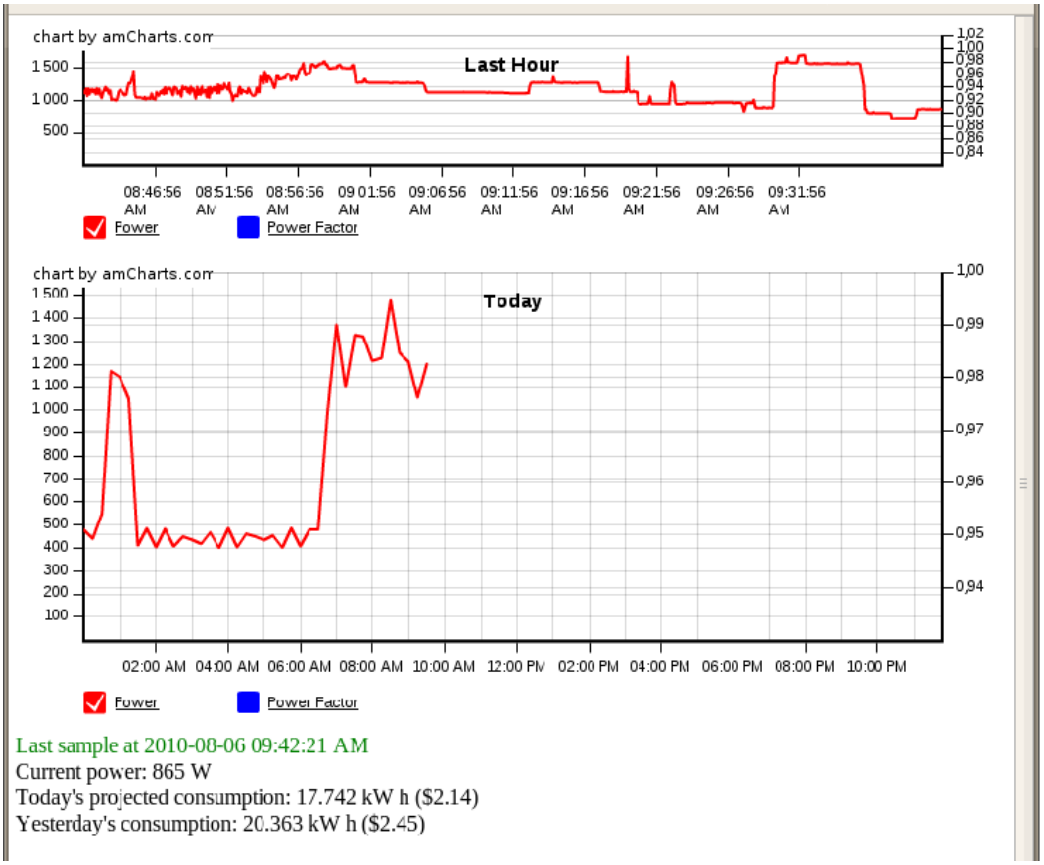


Figure 5.1: Screenshot of the in-home display (IHD). The "Last Hour" graph and textual information (last sample time, current power, and projected consumption) is updated every 5 seconds. The "Today" graph is updated every 15 minutes.

participants to keep diaries of significant events that may impact the study such as vacations, having guests, etc. so that data from these days could be excluded from the analysis. In addition to the NRDC actions, we designated one week per participant to investigate the effect of in-home displays (IHD's) that monitor and display near-real time electricity-use data. Figure 5.1 shows a screenshot of the web-based IHD application that displayed the home's measured energy consumption on a standard laptop computer.

The experiment was conducted over 10 weeks, and during that period we monitored whole-house electrical energy consumption with 5 second resolution. The monitoring equipment is shown in Figure 5.2. Two current transformers (CT's) and a single Continental Control Systems WattNode were placed inside the electrical ser-

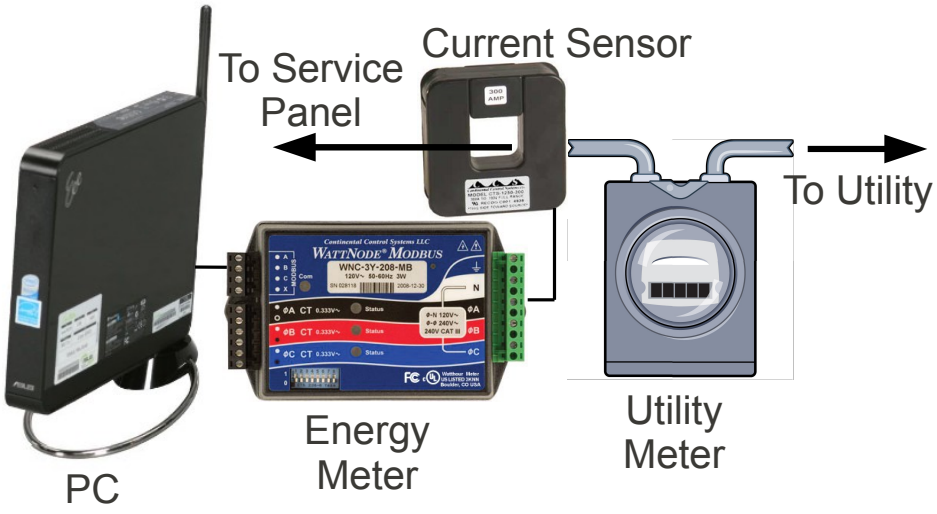


Figure 5.2: Monitoring equipment.

vice panel at each home to monitor electric consumption (all participants had the residential standard split-phase 240VAC service). A small form-factor PC queried the WattNode for power and energy information every 5 seconds. This data was first stored in a local database and then uploaded to a back-end server for analysis and permanent storage.

To compensate for changes in weather on energy consumption using a PRISM-like analysis [70], we also collected hourly average temperatures from the National Weather Service for the length of the study. PRISM is a commonly used statistical procedure designed to separate the energy consumption due to heating and cooling from other uses, and is often used to evaluate the effect of energy-efficiency improvements.

5.2.2 Pre-Experiment Survey

Each participant was asked to complete a short pre-experiment survey designed to collect general information about the participant's home and several factors that may affect energy usage. Table 5.2 and Figure 5.3 summarize the results of this survey. (Some participants did not answer all of the questions.)

Table 5.2: Pre-experiment survey average results.

Number of adults per home	2.20
Number of children per home	0.63
Number of refrigerators/freezers per home	1.78
Approximate hours occupied per day	16.7
CFL lighting use	53%
EnergyStar appliances	38%
Number of households with a central air conditioner	5
Number of households with an evaporative cooler	4
Number of households with a home business or energy-intensive hobby	1
Number of households with hot tubs	1

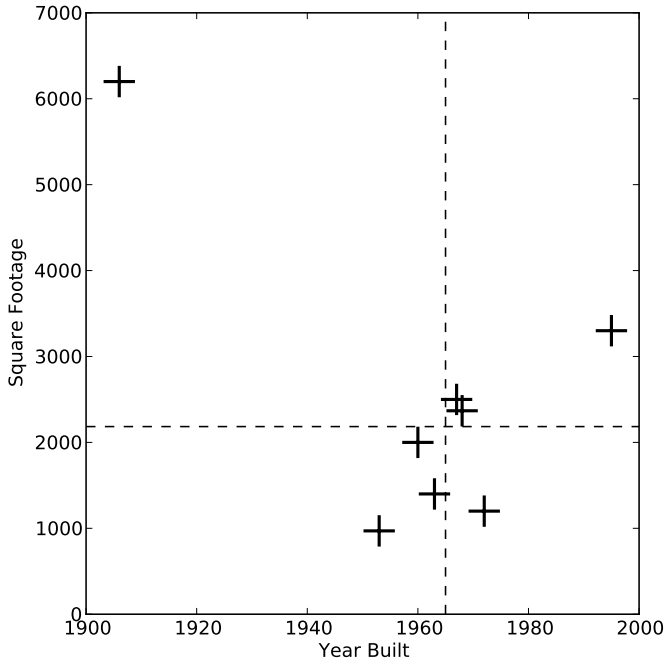


Figure 5.3: Construction year and square footage of participants' homes. Dotted lines indicate median values.

For comparison, the U.S. Census Bureau reports the average size of household is 2.61 (adults plus children, 2007) [71], slightly smaller than our study average household size of 2.83. The median year of construction for our sample is 1965, somewhat older than the national average of 1973 [72], which may explain, in part, the slightly smaller median square footage (2,184 for study sample versus 2,438 U.S. new construction average [73].) The U.S. Energy Information Administration reports that 17% of homes have two or more refrigerators [74]; in stark contrast 50% of the participants in our study had two or more refrigerators.

5.2.3 Results

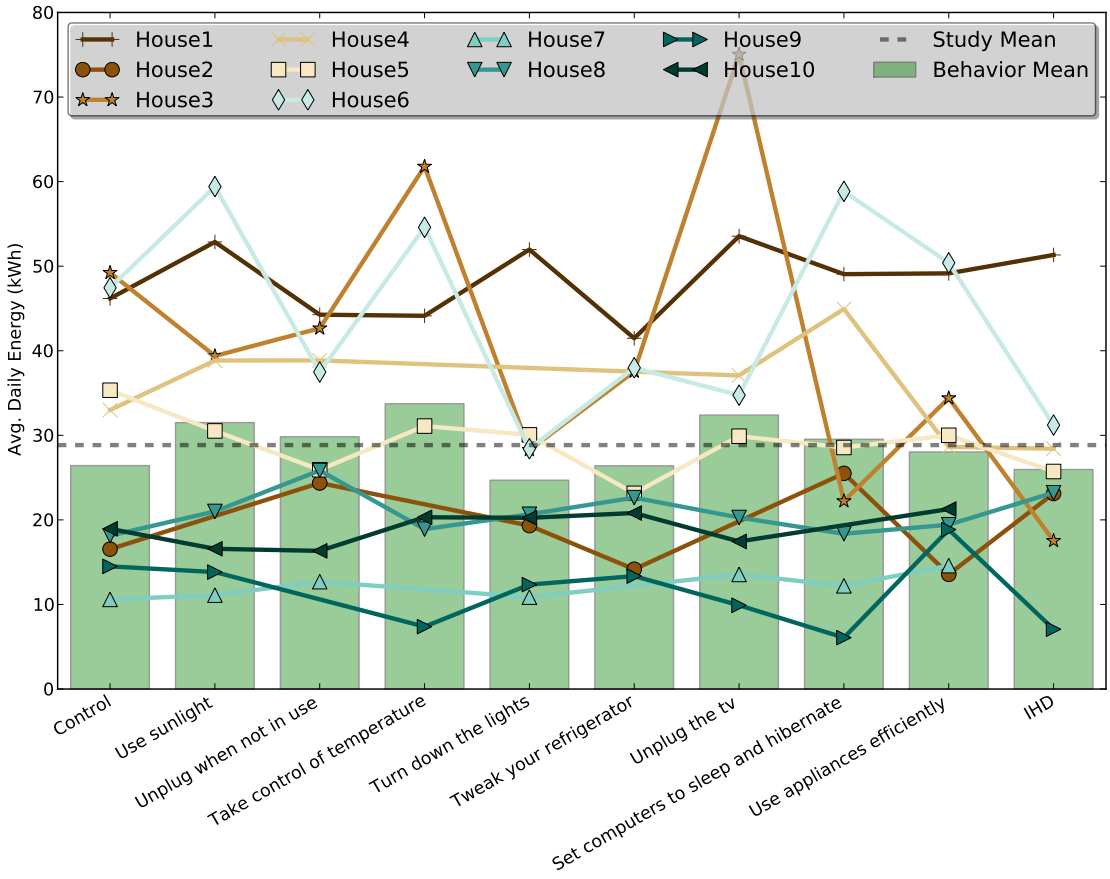


Figure 5.4: Summary of collected data separated by house and behavior.

The raw data are presented in Figure 5.4. Solid lines indicate mean weekday electricity used by each household. Weekend data are excluded for simplicity, as occupancy patterns are more likely differ dramatically between weekdays and weekends than between different weekdays. The vertical bars correspond to the mean daily energy usage for each behavior across the study sample. The dotted line at 28.84 kWh is the study-wide mean daily energy consumption. As expected, there is wide diversity and variation in energy use patterns. For example, House7 shows very little change in consumption, while House3 shows large variability from week to week.

To determine the effect of each behavior on energy consumption, we construct a linear model for mean daily energy consumption. This is shown in Equation 5.1,

where Y_{ijk} is the energy consumption in house j with behavior i on day of the week k, α is the intercept, β_i is the effect of behavior i, γ_j is the effect of house j, δ_k is the effect of day of the week k, $\rho_j * t_{ijk}$ is the effect of ambient temperature relevant to house j, behavior i, and day of the week k, and η_{ijk} are random errors assumed to be independent and normally distributed with constant variance.

$$Y_{ijk} = \alpha + \beta_i + \gamma_j + \delta_k + \rho_j * t_{ijk} + \eta_{ijk} \tag{5.1}$$

We found that replacing the mean daily outdoor temperature with the cooling degree-hours computed assuming a fixed set point (as is customary in PRISM analysis [70]) had no significant effect on the results, so we use the simpler mean daily temperatures.

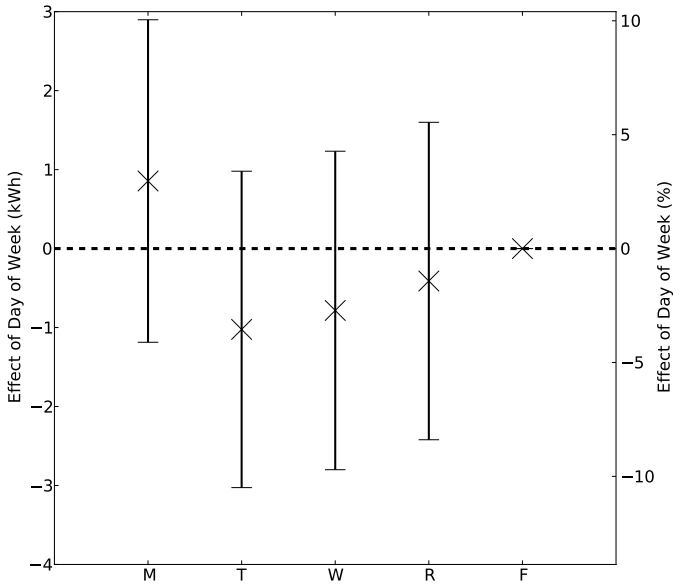


Figure 5.5: Change in energy usage by day of week relative to Friday (90% confidence intervals shown).

Figure 5.5 shows the computed coefficients for effect of the day of week relative to Friday. The confidence intervals are at 90%. Clearly the day of week is not significant in this study, even with 90% confidence. However, given a larger sample size, it may be possible to identify variations due to the day of week.

Figure 5.6 shows the computed effect of each behavior (the β_i terms from Equation 5.1). Due to the small sample size, no results were significant with 95% confidence,

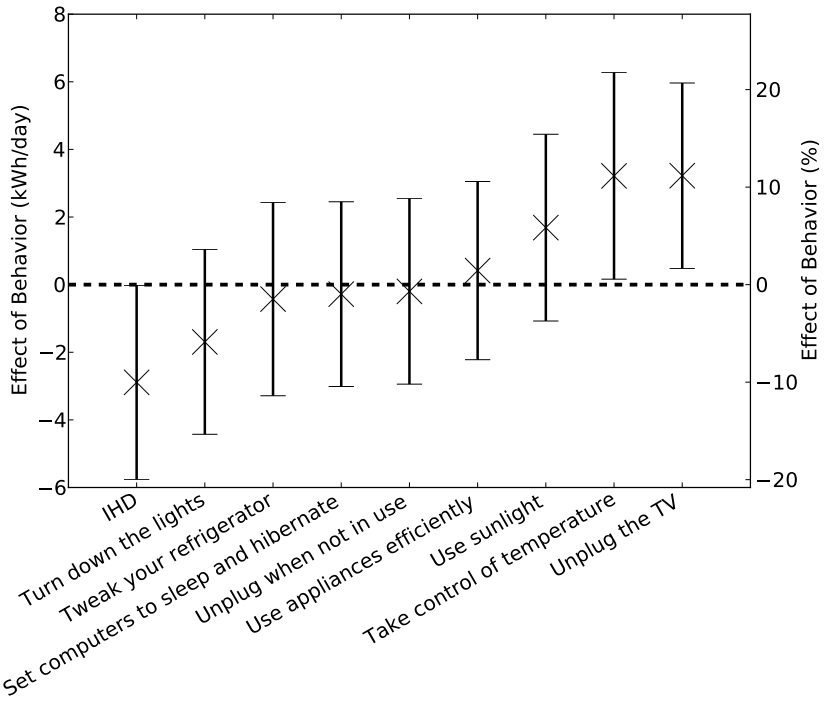


Figure 5.6: Change in energy usage by behavior relative to baseline (control) behavior (90% confidence intervals shown).

so we relaxed the confidence to 90% for discussion. The in-home display (IHD) was significantly better at reducing energy consumption than any single behavior for our study group. This is not surprising, because during the IHD week participants were able to use the provided feedback to reduce their energy consumption by targeting any number of end-uses. Over one week, this resulted in a $0 \sim 20\%$ energy savings. This correlates very well with the $0 \sim 18\%$ reported by the Electric Power Research Institute (EPRI) [75] based on several IHD studies.

Due to the small sample size and short duration of our study, no other results showed significant energy savings. However, there is some indication that "turn down the lights" may yield energy savings. This avenue for savings could diminish over time because our pre-study survey indicated only 53% of lighting was using CFL or other energy efficient technologies. If this percentage is increased, the potential savings will decrease.

The behaviors "Tweak your refrigerator," "Unplug when not in use," "Use appliances efficiently," and "Set computers to sleep and hibernate" were not found to be effective. Several possible explanations exist: First, the effect of these behaviors may have been too small to see over our short study period. Second, because these represent the most common-sense energy saving behaviors it is possible that the participants were already performing these behaviors. Finally, the occupants may have only partially performed the behavior (if at all). To explore this possibility we conducted a post-experiment survey, described in Chapter 5.2.4.

The behaviors "Use sunlight" and "Take control of temperature" showed possible increases in energy consumption, with the latter being significant. These results are most likely a result of miscommunication or performing the behavior incorrectly. For example, "Take control of temperature" suggested summer/winter set points for HVAC. If the participant did not normally use their air conditioner, but turned it on to comply with the behavior, we would expect increased consumption, although this was not the intent. (This does, however, illustrate the dangers of generalizing when specifying energy saving behaviors, as "one-size-fits-all" approaches may lead to undesirable consequences.)

The significant increase for "Unplug the TV" was most likely caused by some other factor not accounted for in our model. Examining the raw data in Figure 5.4, we see that House3's energy consumption is nearly double while performing this behavior. For comparison, we excluded House3's results for this behavior and recomputed the results (See Figure 5.7). This results in a significant change for "Unplug the TV" as well as slight changes for the other behaviors as well. The new results show "Unplug the TV" to be similar in effect to "Tweak your refrigerator," "Unplug when not in use," "Use appliances efficiently," and "Set computers to sleep and hibernate."

The overall conclusions that we can draw from these results are mixed. Our IHD results are consistent with previous, more extensive studies, and demonstrate that significant energy savings are possible. However, no single behavior that we tested

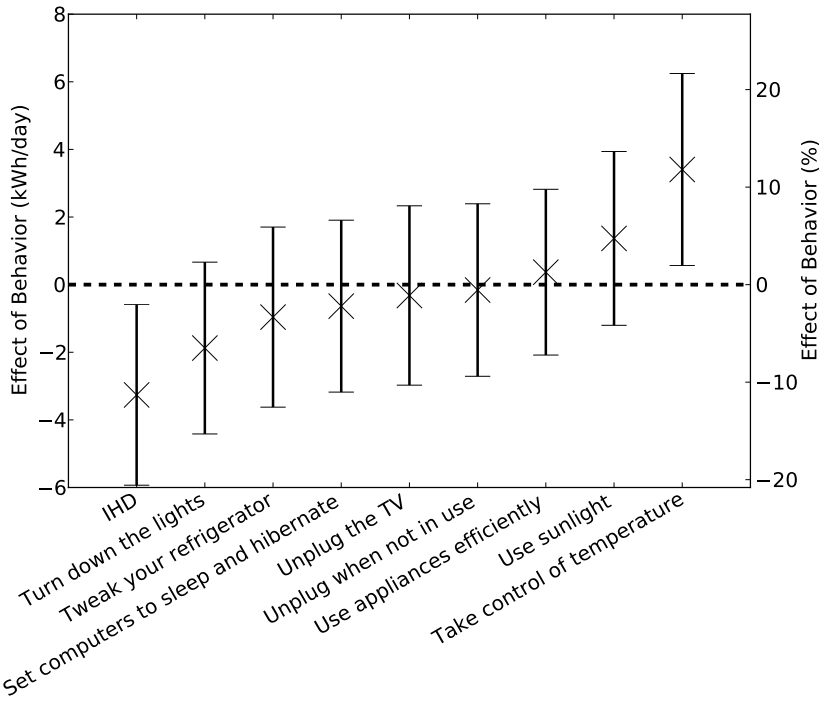


Figure 5.7: Change in energy usage by behavior relative to baseline (control) behavior with House3 excluded from behavior "Unplug the TV" (90% confidence intervals shown).

was responsible for statistically significant whole-house energy savings. The results for "Unplug when not in use" contradict our previous study where we saw significant savings using automated occupancy-based control. To reconcile this we plan to reevaluate this behavior using our automated BMS and a much larger statistical sample. Finally, the increased energy consumption for "Take control of temperature" suggests that temperature control is an important component of energy consumption and that more advanced control techniques (such as [76]) are essential.

5.2.4 Post-Experiment Survey

At the conclusion of the experiment, we asked participants to complete a second short survey. The survey was sent to all participants before they received any results from the study. Nine of the ten participants completed the survey.

Table 5.3: Survey results to: "What do you expect the effect of this behavior to be on your daily energy consumption?"

Behavior	Used Much Less	Used Less	No change	Used More	Used Much More
IHD	1	3	2	1	0
Unplug when not in use	0	2	2	5	0
Unplug the TV	0	3	4	2	0
Set computers to sleep and hibernate	0	2	4	2	1
Take control of temperature	2	0	2	1	3
Use sunlight	0	2	4	1	1
Tweak your refrigerator	0	1	4	3	0
Use appliances efficiently	2	0	3	2	1
Turn down the lights	0	3	2	3	1

Question one asks "'What do you expect the effect of this behavior to be on your daily energy consumption?" Table 5.3 shows the results. These results were surprising because for each behavior, at least one participant expected it to increase daily energy consumption. The largest expected increase was for "Take control of temperature," which matches the observed result. This supports our speculation that the participants turned on their air conditioners specifically to comply with the behavior.

At the other extreme, more people expected the IHD to provide energy savings than any other behavior. Perhaps the most surprising survey result was that some people expected the behaviors: "Unplug when not in use," "Use appliances efficiently," and "Set computers to sleep and hibernate" to increase energy consumption.

Table 5.4: Survey results to: "How well did you (and your family) follow each behavior?"

Behavior	Not at all		About half of the time		All of the time
IHD	2	0	4	1	0
Unplug when not in use	1	0	4	4	0
Unplug the TV	4	1	0	0	4
Set computers to sleep and hibernate	0	0	3	2	4
Take control of temperature	0	0	1	3	4
Use sunlight	0	0	2	3	3
Tweak your refrigerator	4	2	0	0	2
Use appliances efficiently	0	0	2	3	3
Turn down the lights	0	1	2	3	3

The final question was "How well did you (and your family) follow each behavior?" The results are shown in Table 5.4. These results can be used to estimate what energy saving behaviors people are willing to implement manually. Because our participants were all employees of NREL, we expect this group to have higher compliance than if we selected from the entire population. "Take control of temperature" had the highest reported compliance. This makes sense because it involves setting the thermostat one time. The behaviors with lowest reported compliance were "Unplug the TV" and "Tweak your refrigerator." Unplugging the TV requires an extra step each time the TV is turned on/off, so it is not surprising that people did not perform this behavior consistently. We suspect that the low compliance for "Tweak your refrigerator" was due to the need to find and use a refrigerator thermometer and wait a long time between making adjustments. The low levels of compliance suggest that these behaviors could benefit from automation.

5.3 Chapter Summary

In our quest to reduce residential energy consumption, we have conducted an experimental study to quantify potential savings from eight different behaviors as well as real-time monitoring via an IHD. The results of this study indicates that none of suggested behaviors were as effective at reducing energy consumption as the IHD. This is not surprising; given appropriate information savvy homeowners can significantly reduce their energy consumption on their own. However, not all homeowners are equally interested in saving energy, and our field study of limited duration does not address the issue of persistence.

These results suggest that automated BMSs are necessary to reduce energy consumption and they should focus on interaction with the homeowner (as with the IHD), lighting, and possibly HVAC systems. We observed little to no effect from instructing people to manually unplug miscellaneous electrical loads, computers, or

televisions and the post-experiment survey confirmed that less than half of the participants actually unplugged devices as instructed.

CHAPTER 6

DISTRIBUTED BUILDING ENERGY MANAGEMENT USING PROTOCOL INDEPENDENT MULTICAST

Automated building energy management systems are essential to enabling the development of mass-market, low-energy buildings. In existing and future buildings, the impacts of occupant behaviors contribute significantly to the total energy efficiency. Chapter 5 suggests that there are many ways to save energy by altering behaviors, however, most people are not willing to significantly change their behaviors. One solution is to automate these energy saving behaviors. In this chapter we demonstrate a prototype framework where building systems can share information in order to optimize performance. Using protocol independent multicast, sensors and controllers efficiently share information in a distributed peer-to-peer fashion. In this study, the system achieved an energy savings of 7.1% - 14.6% by implementing an occupancy-based control policy. Based on the results of this work we have identified several key areas for future work.

6.1 Chapter Overview

The U.S. department of energy reports that buildings were responsible for 39% of the total energy consumption in the U.S. in 2009 [1]. Because energy consumption is closely tied to occupant behavior, numerous building monitoring systems have been recently developed [3–5] to provide occupants with detailed energy consumption information. The impact of this data is significant; however, monitoring alone does not always result in savings. A recent study observed an initial 31.9% reduction in energy consumption immediately after installing a monitoring system; however, after a month the reduction fell to only 3.7% [6]. This illustrates that while significant savings are possible, relying on occupants to change their long-term behavior may

be difficult. One alternative solution is to build systems that automate the energy saving behaviors.

Automated building management systems (BMS) are expected to save an average of 5% to 10% in residential households [75, 77, 78]. Although these savings are significant, it equates to only $5-$10 per household per month [79]. To be practical, a BMS must pay for itself within a few years, which means it should cost no more than a few hundred dollars. As a result, deploying a WSN with sensors and actuators dedicated exclusively for building management is too costly. However, if we leverage existing sensors already in the home, we can significantly reduce the cost of the BMS. For example, many homes have security systems which sense the states of doors and windows and detect motion. Existing HVAC systems sense temperature. Everyday household appliances have numerous on-board sensors ranging from the simple refrigerator door switch to the complex sensing techniques possible with an idle PC's microphone and camera. It is even possible to collect device-level energy usage in many appliances for free [80]. Of course none of these systems currently share this very useful information. One reason is that there are no standards defining how to share and consume this information.

Our proposed solution uses a wireless sensor network (WSN) to share this information. Wireless sensor networks utilize low-powered low-cost wireless nodes communicating over an ad-hoc network. Standardization is emerging in the form of IEEE 802.15.4 [7] and 6lowpan [33]. The dominant communication paradigm in WSNs is from the sensor nodes to a base station for processing and storage. Using this approach we could easily construct a centralized building management system that would process all sensed data and make optimized control decisions. However, for a BMS to be more responsive, intuitive, robust, and scalable, a distributed approach is essential. We define a distributed BMS as one where control decisions are made locally at the point of control using information received directly from any other nodes in the network. For example, a light might receive sensor data from a wall switch,

motion sensor, and light sensor. The control policy could be to turn on the light if the switch is on and there has been motion in the last 15 minutes and the ambient light level is below 200 lux; otherwise the light will be off to save energy.

Implementing a WSN-based distributed control system requires an efficient means of sharing information between devices. The two general approaches to information sharing are to pull or push the data. In a pull-based system, controllers would periodically poll the sensors, pulling the relevant information into the controller. This places the burden on the control point to collect necessary information in a timely manner while the sensor only needs to respond to requests. In a push-based system, the sensors disseminate information to the controller when it is available. This makes the controller's job much easier by transferring more responsibilities to the sensors. The ZigBee Smart Energy Profile 2.0 allows both forms of information sharing [81]. For both cases it is implemented in the application layer with sequential unicast communications, which creates redundant messaging when two nearby nodes are consuming the same information. For example, two (or more) lights in the same room might rely on the same set of sensors. Using unicast communication requires unique messages for each sensor used by each controller. However, the broadcast nature of wireless communication, makes it possible to improve efficiency by allowing any interested node within a shared communication area to receive the same information. We have implemented this approach using standard IP multicast that we have adapted for WSNs. The result is that sensors can push information that is then efficiently delivered to all interested control points. The use of IP multicast distributes the responsibility for information sharing to the *network* rather than either the sensor or controller. Because this is implemented at the network layer, redundant packet transmissions can also be eliminated, which improves energy efficiency, information timeliness, and network utilization.

6.2 Prototype System

Our prototype BMS using PIM-WSN was deployed in two graduate student offices on our campus. The deployment is depicted in Figure 6.1. The portion of the building shown is approximately 120 feet by 40 feet. We configured the transmit power on each mote to -10 dBm to simulate a physically larger and more interesting network topology. This results in a maximum hop count of 5 from the base station in CH131 to the motes in CH123. The control algorithm is distributed and implemented directly on each energy controller. The base station is required by Blip to provide multihop routing for unicast packets. Multicast data is then forwarded along the routes selected by unicast routing protocol.

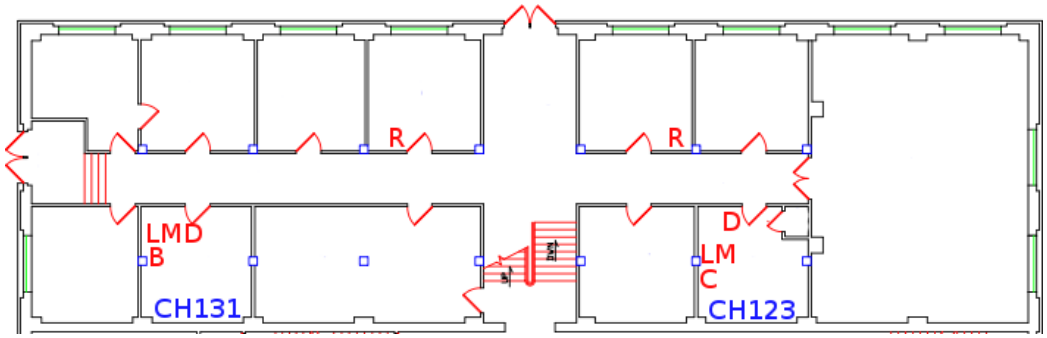


Figure 6.1: A building management system prototype deployment area. The offices with energy controllers are CH131 and CH123. Letters indicate the approximate location of nodes. Node types are abbreviated as: Light, Motion, Door, Energy Controller, Relay, and Base-station.

6.2.1 Sensors and Controllers

Each office is outfitted with three sensors: 1) passive infrared (PIR) motion sensor, 2) door sensor (magnetic reed switch), and 3) ambient light sensor. To consume this data, each office has an energy controller node. The energy controller plugs into a standard electrical outlet and provides two outlets: one switched and one non-switched. The energy controller also contains a power meter that measures real-time power usage and total energy usage independently for each outlet. A power

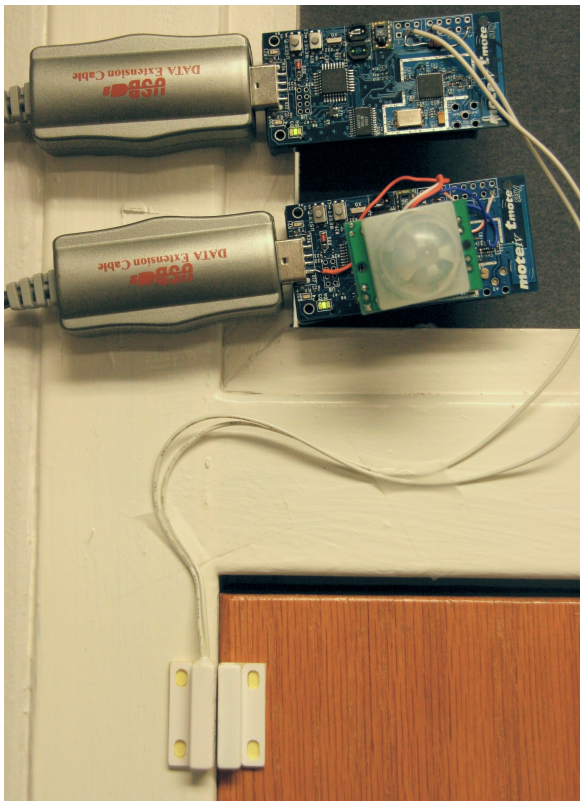


Figure 6.2: Door and PIR motion sensors.

strip is plugged into each outlet and all essential devices (PCs, refrigerator, network equipment, etc.) are plugged into the non-switched power strip. Non-essential devices (LCD, printer, coffee pot, microwave, etc.) are plugged into the switched power strip. The sensors and energy controller each use a modified TelosB mote programmed with TinyOS 2.x and our implementation of PIM-WSN. The premise is that the energy controller will detect and process the available sensor data and intelligently control the switched outlet to save energy by switching off non-essential devices when the office is unoccupied.

1) Motion sensor

The motion sensor is a PIR sensor (Parallax #555-28027) with a motion detection range of approximately 20 feet (lower mote in Figure 6.2). It is configured to send repeated pulses when there is continuous motion. The sensor output is attached to the TelosB's expansion connector on an interrupt-enabled GPIO pin. This allows the sensor to wake up the TelosB when motion is detected. Every transition of the GPIO

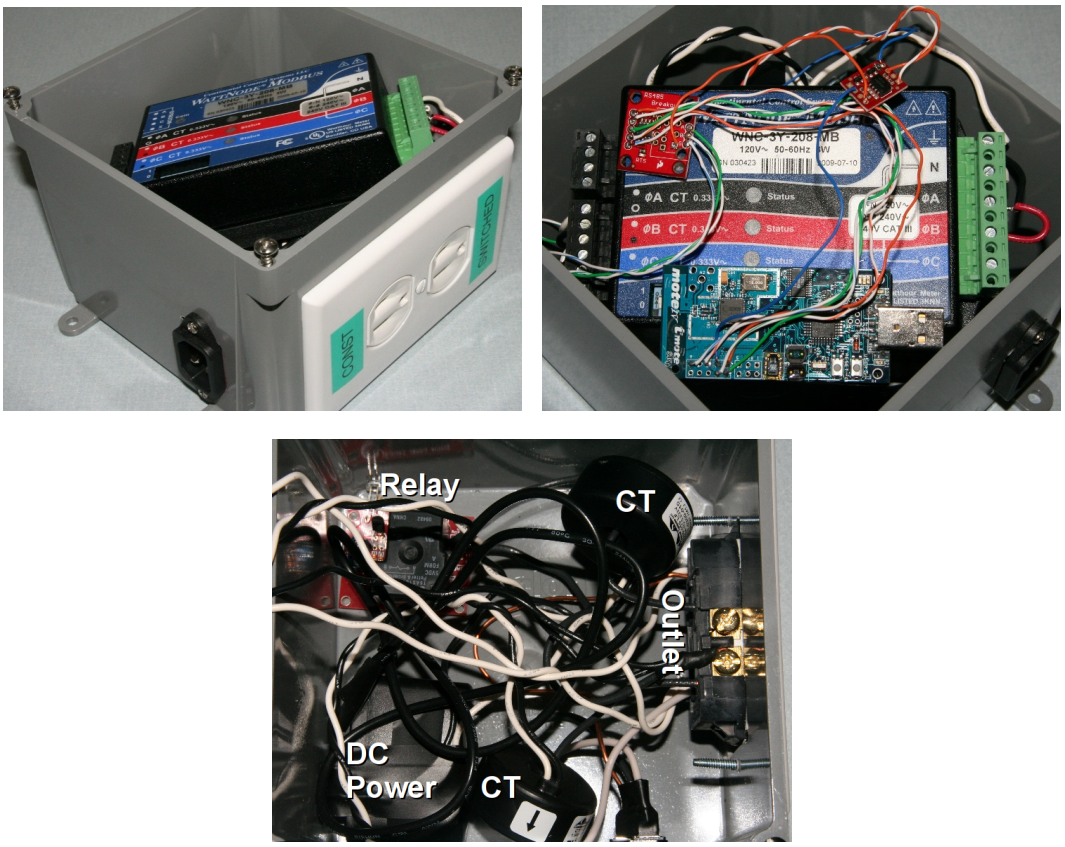


Figure 6.3: Energy controller node.

pin causes a single multicast packet to be transmitted indicating the motion sensor output (motion or no motion). Because the sensor requires 3.3V-5.0V to operate reliably, it is powered directly from the USB interface.

2) Door sensor

The door sensor is a magnetic proximity switch (C&K Components #MPS45WGW) attached to each office door (upper mote in Figure 6.2). The switch is interfaced to the TelosB in the same configuration as the motion sensor. An interrupt-enabled GPIO pin allows the sensor to wake the mote when the state of the door changes. When the switch changes value a single multicast packet is sent indicating the current state of the door (open or closed). Although multicast packet delivery is generally very reliable, it is not guaranteed. Because it is likely to only send one or two packets (unlike the motion sensor that generally sends several), we also use a periodic timer

to send the door state once per minute. This allows the energy controller to miss packets and still maintain acceptable functionality.

3) Ambient light sensor

The ambient light sensor is the TelosB's on board Hamamatsu S1087 photodiode read by the MSP430's internal ADC. The photodiode is polled randomly once every 0.75 ± 0.25 seconds. Typical ADC readings range from 0 (dark) to the several hundred for normal office lighting. If the sampled light value is more than 25 higher or lower than the last reported value it is multicast immediately. Otherwise, one sample is sent every 100 readings. This results in a data rate of at least one packet every 75 ± 25 seconds while still being responsive to rapid changes, such as the lights being turned on or off.

4) Energy Controller

The energy controller is a TelosB with a Tyco relay (# T9AS1D12-5) that is rated for 220V@30A and a WattNode[3] energy meter to allow power monitoring. We use a standard 6"x6"x4" electrical box to house this equipment. Power comes in through an IEC C14 connector, passes through a 15A current transformer (CT), and travels in parallel to the non-switched side of a standard NEMA 5-15 outlet and the AC relay. The output of the relay passes through another 15A CT and then on to the switched side of the outlet. The WattNode, relay, and TelosB are powered from the non-switched AC supply so the energy controller's power usage is included in the non-switched power measurement. Because of this configuration, we are measuring total power and switched power; the non-switched power usage is simply the difference between the two measurements. The power meter is configured to average instantaneous power readings over a 20 second sliding window. The TelosB then samples the real (W) and reactive (VAR) power, line voltage (V), frequency (Hz), and total energy used (kW h) once every five seconds for each phase (total and switched). The sensor data is then augmented with the current occupancy value

[3] `http://www.ccontrolsys.com/products/wattnode_modbus.html`

(true/false) and encoded in a single packet and transmitted sequentially as a multicast packet and as a serial packet (for logging).

Figure 6.3 shows the energy controller. The top figure shows the power input and outlets. The middle figure is TelosB with interface circuitry. The WattNode uses the Modbus [10] serial communications protocol over an EIA-485 physical link, so an EIA-485 adapter was added on `uart0` of the TelosB. The relay requires 200 mA to activate, so a supplemental AC/DC power supply was also added. Because the mote was powered via USB (to collect diagnostic information) an opto-isolator was used to interface to the relay (the WattNode's EIA-485 interface is already isolated). The internal wiring is shown in the bottom figure.

6.2.2 Control Algorithm

Each energy controller is preprogrammed with the room number it is deployed to so that it can search for sensors (see Chapter 6.2.3) in the same room (each sensor is also programmed with its room number). When a door, motion, or light sensor is detected in the same room, the energy controller subscribes to that node's multicast and begins receiving sensor data. Our control algorithm (Figure 6.4) relies on detecting when the occupancy changes and then switching the relay on or off accordingly. Occupancy detection is a difficult problem and not our focus, so we use a simple but effective algorithm tailored to our office environment. Each office has a single door. We assume that the office is occupied if the door is open and that the door is shut when unoccupied (this is nearly always true). As a result, occupancy can only change after a door-close event. Therefore, when the door is open, the energy controller switches to the occupied mode. After detecting a door-close event it starts a 60-second timer. While this timer is running it counts the number of motion events received. When there is constant motion the motion sensor will send one motion event per second, but even when there is no motion one or two (false) motion events per hour. To reduce the impact of false motion events, we use a threshold of 5 or

more motion counts in the 60-second interval to indicate that the room is occupied. In practice this algorithm works very well in our offices and could be easily applied in residential homes by interfacing to a security system with door and motion sensors.

```
if door open then
  // room is occupied
  setRelay(close)
else
  // assess occupancy
  count = 0
  for 60 seconds do
    wait for motion event or timeout
    count += 1
  end for
  if count ≥ 5 then
    // room is occupied
    setRelay(close)
  else
    // room is not occupied
    setRelay(open)
  end if
end if
```

Figure 6.4: An occupancy detection algorithm executed each time the door state changes

6.2.3 Service Discovery

One remaining challenge is to decide how the energy controller initially subscribes to the multicast from each sensor. One approach is to hard-code the source address of each sensor, by definition this is not a very flexible solution. Instead, we have implemented a service discovery protocol that is similar to the Simple Service Discovery Protocol (SSDP) [82]. SSDP is used by Universal Plug and Play (UPnP) to detect other UPnP devices. It uses HTTP formatted messages over a predefined multicast group. In our implementation we use two fixed format messages, rather than the variable format HTTP messages, to simplify processing. We have also assigned a special multicast group in PIM-WSN where all nodes are assumed to subscribe. This effectively allows PIM-WSN to broadcast the service discovery messages to every node

in the network. The two messages are: *advertisement* and *query*. Common to both of the messages are two 8-character fields defining the sensor type and domain. The sensor types we used are: motion, light, door, and energy. The domain is used to indicate the room number of the sensor: CH123 or CH131.

There are two ways to detect a sensor. The first is at startup, because up it will initially transmit several *advertisements* to the service discovery multicast group. If a node is interested in the advertised sensor it can then join the multicast immediately. The second way to detect a sensor is to have the interested node send a *query* message to the service discovery multicast group. The query message allows wildcard searches on the type and domain values. All nodes that receive the query will check to see if their service description matches, and if so, the node replies with a unicast advertisement.

6.3 Experimental Results

Figure 6.5 shows the power and cumulative energy usage logged by the energy controller in CH131 over two typical days. On day one (Figure 6.5(a)) the relay was disabled to detect and compute wasted energy. On day two (Figure 6.5(b)) the relay was enabled to control the wasted energy. The non-switched devices are: two PCs, one laptop, two small refrigerators, and an Ethernet switch. The switched devices are: two LCD displays, a laser printer, two powered speakers, a desk lamp, a microwave, and a coffee pot. The large spikes in the switched data are due to the coffee pot, microwave, and laser printer. The oscillations in the non-switch data are due to the refrigerators.

Figure 6.5(a) shows the first day where the room was occupied for 7h 52m 46s. The switched devices consumed a total of 1.6362 kW h and the non-switched devices consumed 5.0972 kW h. Of the switched total, 0.7008 kW h was consumed (wasted) while the room was unoccupied. The minimum power usage was 33 watts. This reveals a potential savings of 42.8% of the total energy used by switched devices

or equivalently 10.4% of the total (switched plus non-switched) measured energy consumption. Performing the same analysis on the other office yields a potential savings of 6.89%.

On the next day the relay was enabled and the results are shown in Figure 6.5(b). On this day the room was occupied for 9h 56m 21s. The switched devices consumed a total of 0.9120 kW h wile the non-switched devices consumed 5.3426 kW h. The non-switched devices consumed 4.8% more on this day, most likely due to the increased occupancy. Despite this fact, the switched devices now consumed 0.7242 kW h less than the previous day. The total energy consumption (switched plus non-switched) was 6.2546 kW h or 7.1% less than the previous experiment. The same analysis for the other room shows that the total energy consumption was reduced by 14.6%.

On the first day we measured that 0.7008 kW h of electricity was wasted. On the second day we controlled the devices to reduce waste and the total measured energy usage was 0.7242 kW h less than the total on the previous day. These results are very consistent between the two days. If we then assume an average daily savings of 0.7 kW h and then multiply by 365 for a conservative estimate of the yearly savings (because unoccupied time, and therefore savings, is expected to be greater on weekends and holidays) the result is 255 kW h. Then, if we assume this savings is typical over all 41 offices in our building, the estimated building-wide savings becomes approximately 10 MW h per year. This equates to an annual reduction of approximately 7.8 tons of CO_2 [4] and a savings of approximately $1,000.

Over the last year our building's total energy consumption was approximately 300 MW h; however, this includes HVAC and lighting. We could expand our system to include these systems or for calculation exclude them from the total energy usage. According to [1] in an average building HVAC and lighting represent 53% of the building's total energy consumption. This can be used to compute the total energy consumption excluding HVAC and lighting as 141 MW h. Fully deployed, our BMS

[4] 1.5 lbs CO_2 per kW h

is expected to reduce this by 10 MW h, or over 7%. Because this value is close to our achieved savings, it gives confidence that the offices used in this study are representative of the average energy consumption in the building.

If we assume the building already has sensors able to detect room occupancy (and they share this data), the only additional hardware required to implement this system is a simplified energy controller node for each office (the energy monitor function is not needed). To achieve a one year payback period (assuming 0.7kW h per day savings), the resulting budget is $25 per node. This is more than the cost of our TelosB motes, but, commercial IEEE 802.15.4 devices like the XBee are currently priced around $20 each. The relay that we used is currently priced at $1.40 each. This gives us confidence that this type of distributed control system could achieve a one year payback period.

6.4 Chapter Summary

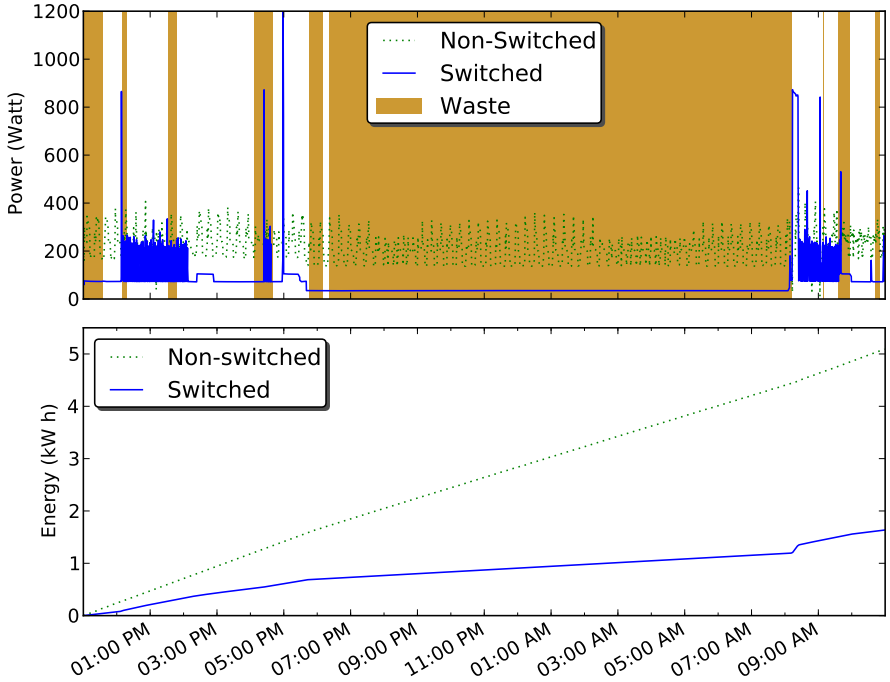
Advanced building management systems will eventually become common in residential and commercial buildings because occupant behaviors have a significant impact on the total energy consumption. To be successful these systems must be responsive, intuitive, robust, and scalable. Our approach is a fully distributed architecture using WSN-class nodes coupled with an efficient multicast communication protocol. This allows each controller to autonomously locate and receive relevant sensor information from other nodes in the network. Because control decisions are made at each control point, if a sensor or communication link fails the controller can still make reasonable control decisions. Our prototype system achieved an energy savings of 7.1% - 14.6% by implementing an occupancy-based control policy.

Based on the results of this work we have identified the following key areas for future work:

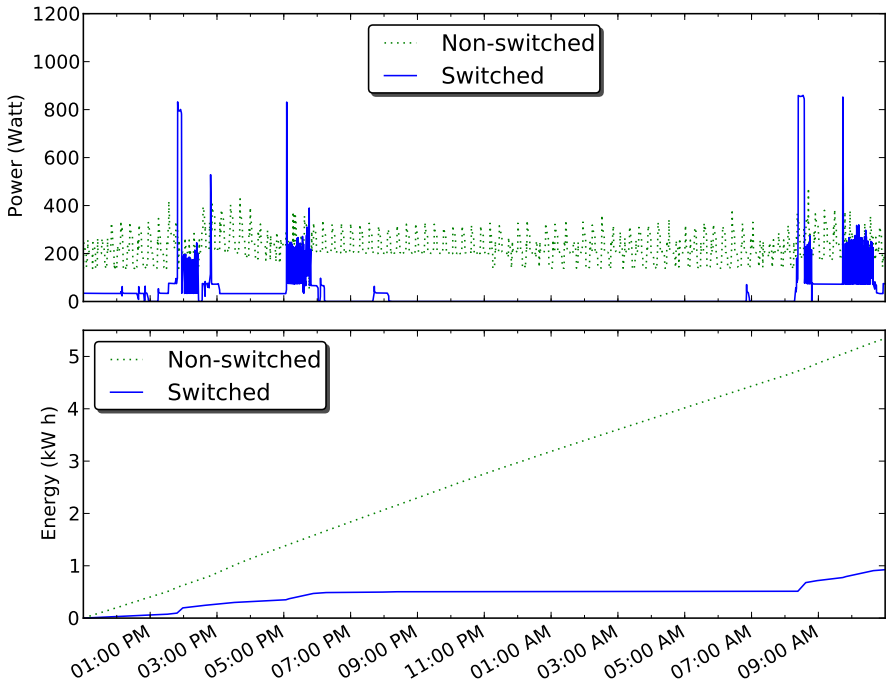
Reliability - Although PIM-WSN achieves good packet delivery (>97% under normal conditions [83]), missing just one packet can cause a control algorithm to

fail. In our case the "door open" packet was crucial to receive or the occupant could walk into a room with all their devices powered off. We consider any packet delivery failure as a network disconnect, even if it is a transient event. To solve this, first we need a robust way to detect these disconnections. Second, once the node regains communication the missed packets should then be delivered to the node. This is reminiscent of the Trickle algorithm [84]. To be applied in this domain the algorithm must support rapidly changing data from many sensors in the network.

Minimizing sensor power consumption - To minimize the number of packets sent, PIM-WSN uses one-hop broadcast messages to deliver packets to multiple nodes simultaneously. Most modern low-power protocols focus on unicast rather than multicast or broadcast and as a result their performance in these cases is greatly reduced. In order to support battery powered or energy harvesting sensor nodes, efficient low power communication is essential. Synchronized low-power protocols represent one approach to alleviating this problem.



(a) Monitoring only.



(b) Monitoring and control.

Figure 6.5: Experimental results over a typical day.

CHAPTER 7

ACHIEVING RELIABLE WIRELESS AUTOMATION AND CONTROL USING GLOBAL SHARED MEMORY

Improving energy efficiency in residential homes is a challenge that must be solved. As improvements in building construction improve the mechanical efficiency of buildings, the relative impact of other building loads is heightened. To mitigate the increasing significance of electrical loads, we have proposed a distributed wireless building automation and control architecture. This architecture allows a paradigm shift from the typically centralized building automation and control systems to a distributed approach which we believe is more appropriate for residential homes. Our previous implementation of this architecture relied on protocol independent multicast (PIM) to share sensor data. Our evaluation identified two key areas for improvement: reliability and supporting low-powered sensor nodes. In this chapter, we present a complete hardware and software solution for building general purpose wireless automation and control systems with guaranteed reliability and support for ultra-low duty cycled sensor nodes. Sensor data is shared through a global shared memory abstraction that ensures the eventual delivery of sensor data. Essentially this abstraction provides infinite retransmissions. In our evaluation, we observed an 88% reduction in standby losses for a home entertainment system. Using published statistics on home entertainment and home office systems, we expect this approach, on average, to reduce energy consumption of home entertainment device by 17%, or equivalently a 2.3% reduction in whole-house energy consumption. This work also provides a general purpose distributed automation and control framework that can be extended to implement other energy saving measures such as intelligent lighting and adaptive HVAC systems.

7.1 Chapter Overview

Current generation building automation and control (BAC) systems (e.g., SCADA) rely on highly planned heterogeneous deployments of sensors and actuators communicating to a central controller. When considering BAC for residential homes, the large number of existing homes precludes any approach that requires significant planning or professional installation. Even for new construction, the cost of current BAC systems is likely too high. Instead, what is needed is a low-cost modular system that can be bought by any homeowner, taken out of the box, and just works. This chapter presents our alternative approach for a wireless distributed BAC system that meets these requirements.

Our model for a wireless BAC is composed of three classes of nodes: 1) sensing 2) controlling, and 3) a mesh-networking core. Sensing nodes provide the inputs to the control system and are expected to be physically separate from the actual control nodes themselves. This is to allow sensors to be placed near a specific building feature, such as a door, and then several controllers may consume this input while being located near the device, system, or appliance (e.g., a lamp). The sensing nodes monitor temperature, light, humidity, motion, door state, etc. Controller nodes make control decisions by locally processing data from the sensing nodes. The networking core ensures that sensed data are delivered rapidly and reliably to the controller nodes. This architecture, first proposed by Schmid, et al. [85], takes advantage of infrastructure power, when available, to greatly simplify and reduce energy consumption of the sensing nodes.

The significance of this model is that there is no central device responsible for any part of the system; the BAC system is fully distributed. Sensor data is received by any control node within range of the sensor and then the controller will ensure the data is efficiently transmitted to all other control nodes in the network. Each controller makes independent control decisions using the available sensor data. This approach

was selected to improve reliability and expandability of the system. Reliability is critical, and our distributed approach eliminates the possibility of a single failure disabling the entire system. Expandability is a key feature that allows a small control system to be initially deployed and then additional sensors or controllers to be added at a later time. Because the control strategy is fully implemented by the control node, no configuration changes or software updates would be needed by existing nodes in the network to support the additional nodes.

The primary challenge in using WSN techniques for distributed control is that the dominant communication paradigm in WSNs is base-station-centric while distributed control requires a peer-to-peer model. We have previously demonstrated how PIM-WSN (Chapter 6) can be used to share sensor data. In this work PIM-WSN achieved >97% packet delivery, but this may not be sufficient for a BAC system. The challenge of guaranteeing reliability has been solved in traditional WSNs for applications such as code update [84]; however, for distributed automation and control minimizing latency is an additional consideration that has less support from existing approaches.

To implement a highly reliable distributed BAC, we have deployed a global shared memory (GSM) abstraction for WSN-class nodes to efficiently share building information in a distributed peer-to-peer fashion. Specialized sensing and control nodes have been developed and deployed in a residential home implementing two basic control strategies where an average energy savings of 195 watt hours per day (71 kWh per year) was achieved. Further analysis shows that the expected energy savings is 17.5% for home office and entertainment devices, on average, yielding a 2.3% reduction in whole-house energy consumption.

7.2 Related Work

One of the earliest example of an advanced building automation and control system is the Neural Network House [14]. The Neural Network House contains about 75 sensors and actuators wired to a central controller. The goal of the Neural Network

House is twofold: appeasing the inhabitants and conserving energy. As the name implies neural networks are used to predict behaviors and then automatically configure the environment by controlling lights and the heating and cooling systems. This is similar to our high-level approach, however, our system utilizes wirelessly connected distributed controllers.

Wireless sensor networks have been previously applied to monitor appliance energy consumption [6], intelligently control lighting [86], and to improve heating and cooling efficiency [76]. However, these have been special-purpose systems requiring manual configuration and dedicated servers. Our approach is to develop a general purpose distributed WSN control architecture that can be applied in residential buildings.

7.3 System Hardware Design

Building automation and control systems currently rely on numerous sensors, typically hardwired, throughout the building using a myriad of communication protocols (BACnet, LonTalk, ModBus, etc.) to communicate data to a central controller. Wireless protocols (ZigBee, Wireless HART, EnOcean, etc.) have been employed to replace some or all of the wired links. However, even when wireless components are used, the central controller is still responsible for making control decisions. The central controller is typically a programmable logic controller (PLC) that implements basic automation tasks. Our proposed approach is a paradigm shift from centralized control to a distributed approach. This is motivated because the microcontrollers typically found on wireless platforms are capable of implementing basic automation and control tasks without requiring any additional resources. The end result is, by simply utilizing wireless components, we eliminate the need for a central controller entirely.

Because there is no currently available WSN platform that includes a complete set of building sensors, we have developed a dedicated building sensor node based on the Epic core module [87]. This node provides sensing of several key building parameters

(temperature, humidity, light, motion, and door state), described in Chapter 7.3.1. To make use of this information for control and to provide a robust mesh-networking core, we have also developed an automation controller based on the Berkeley Wireless AC Meter [6]. A mechanical relay was added, in addition to the energy meter, to enable switching of an attached device and is further described in Chapter 7.3.2.

7.3.1 Sensor Node

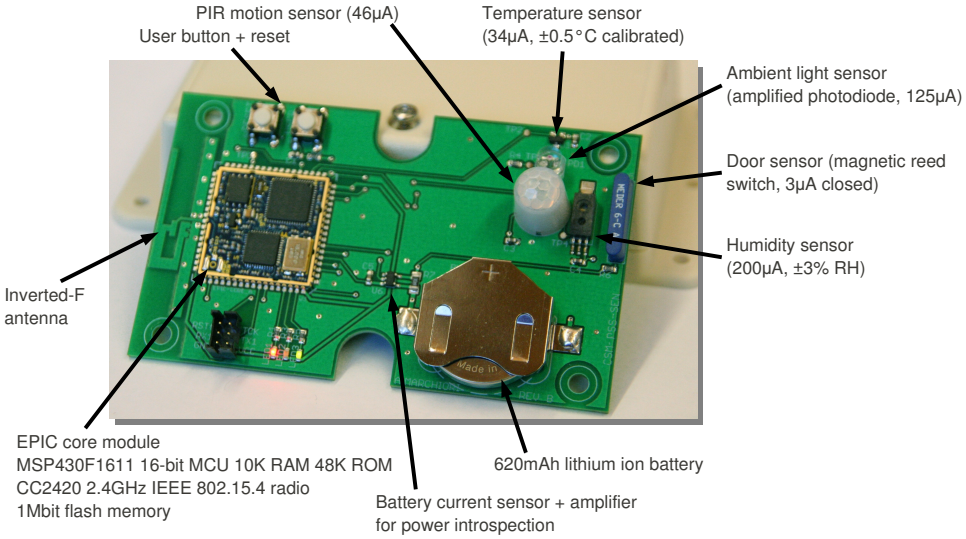


Figure 7.1: Wireless building sensor.

The sensor node is shown in Figure 7.1. The main components are the Epic core module (left), coin cell battery (lower right), and sensors (upper right). The battery is a 3V CR2450 manganese dioxide lithium cell with a nominal capacity of 620 mAh. Because node lifetime is the most significant design consideration, we have selected an array of low-power sensors, summarized in Table 7.1.

Table 7.1: Sensors used in our building sensor node.

Manufacturer	Model	Description	Active Current
Honeywell	HIH-5030	Humidity	200 μA
Panasonic	AMS302	Photodiode+Amp	125 μA
Panasonic	AMN41122	PIR Motion	46 μA
Microchip	TC1046	Temperature	35 μA
MEDER	MK06	Door Switch	3 μA

To estimate the lifetime of the sensor node we consider the primary components of power consumption. All of the sensors, except the motion sensor, can be duty-cycled to reduce energy consumption. The motion sensor must remain on to properly detect motion because the sensor has a 30-second startup time. Duty-cycling this sensor would result in large amounts of time waiting for the sensor to turn on and potentially missed motion events. We experimentally measured the current consumption with all sensors and the microcontroller on to be 2.2 mA. Sampling the sensors, with appropriate startup delays, takes approximately 100 ms. Therefore, the energy required to read all sensors (excluding the PIR), is approximately 660 μJ. The motion sensor continuously uses 46 μA. We assume the power consumption of the microcontroller and radio is negligible in sleep mode; the resulting idle energy consumption is 11.9 J per day.

The remaining energy consuming task is transmitting data. The low-power communication protocol uses link-layer acknowledgments and will retry a transmission up to three times with a randomized delay up to 200 ms (see Chapter 7.4.1). We experimentally measured a failed transmission to use 22.5 mA for 534 ms, consuming 36 mJ. A successful transmission used 22.5 mA for 78 ms, consuming 5 mJ. Because sampling the duty-cycled sensors consumes much less energy than transmitting a data packet, we will always sample the sensors just prior to every transmission. The final estimate for energy consumption is shown in Equation (7.1) where α is the number of failed transmissions, β is the number of successful transmissions, $\alpha + \beta$ is the total number of sensor readings, and the motion sensor was powered for δ days.

$$\begin{aligned} E(\alpha, \beta, \delta) &= \alpha * 36\text{mJ} + \beta * 5\text{mJ} + \\ &\quad (\alpha + \beta) * 660\mu\text{J} + \delta * 11.9\text{J} \end{aligned} \tag{7.1}$$

Referencing the battery datasheet, we estimate that under a constant 420 μA discharge, the battery releases approximately 5,800 J of energy. We combine this with Equation (7.1) and compute the trade-off between sampling rate and node lifetime in

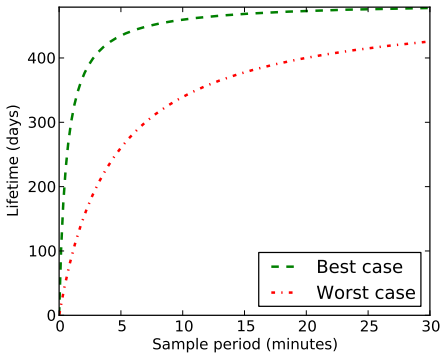


Figure 7.2: Theoretical sensor lifetime with 620mAh battery.

Figure 7.2. The two curves shown bound the node's lifetime. The best case is when every transmission is successful ($\alpha = 0$). The worst case is when every transmission fails ($\beta = 0$). In practice we expect something in between. The lifetime resulting from only powering the motion sensor constantly is 487 days. We see from the figure that in the best case this maximum lifetime quickly approaches the limit as the sample period is increased to 5 minutes. After 5 minutes there is little benefit in terms of lifetime. As a result we have selected 5 minutes as the sensor sampling period to balance data collection and node lifetime.

Table 7.2: Sensor node energy consumption by function.

Function	Best-Case (435 days)	Worst-Case (261 days)
Motion detection	89%	54%
Other sensors	1%	$< 1\%$
Data transmission	10%	46%

Using a 5 minute sampling period, we now examine the energy consumption by each function in Table 7.2. In the best case the motion sensor dominates energy consumption. The other sensors (humidity, light, temperature) consume very little energy because they can be effectively duty-cycled. For example, although the humidity sensor consumes the most current (200 μA), it has a short 70 ms startup time. Sampling this sensor once every 5 minutes yields a 0.023% duty-cycle, resulting in low energy consumption.

A successfully acknowledged data transmission required approximately 78 ms; every 5 minutes this yields a very low 0.026% radio duty-cycle. In the worst-case, the energy consumed by data transmission significantly increases to a relatively low 0.178%. This illustrates that even low-duty cycle radio communications significantly impact energy consumption.

7.3.2 Automation and Control Node

To provide automation for the BAC system, we have adapted the ACme developed at U.C. Berkeley [6] to include a Panasonic DK series mechanical latching relay. This relay is rated for loads up to 10 A and 250 VAC. Because the relay latches in both the on and off states, it only uses energy to change state (2 mJ). Additionally, no heat sink is required, unlike a solid state relay. This allows most small household devices to be efficiently controlled while minimizing energy consumption by the controller. While active (not switching), the node consumes 0.25 watts of power. We retain the Analog Devices ADE7753 single phase multifunction energy metering IC, central to the ACme, so that the attached device can be monitored as well as controlled. The enclosure is larger than the second generation ACme (4.11 x 2.23 x 2.50 in) to accommodate the relay. The internal circuit board and assembled device are shown in Figure 7.3.

7.4 System Software Design

Because BAC systems interface with line-powered systems, devices, and appliances, it is also useful to utilize the available power for the mesh-network and automation and control nodes. This allows the self-powered sensing nodes to be optimized for low-power operation. As a result of this separation, there are two distinct network protocols in the system. The first is a one-way, sensor to router, asynchronous communications protocol described in Chapter 7.4.1. This design allows very low-duty

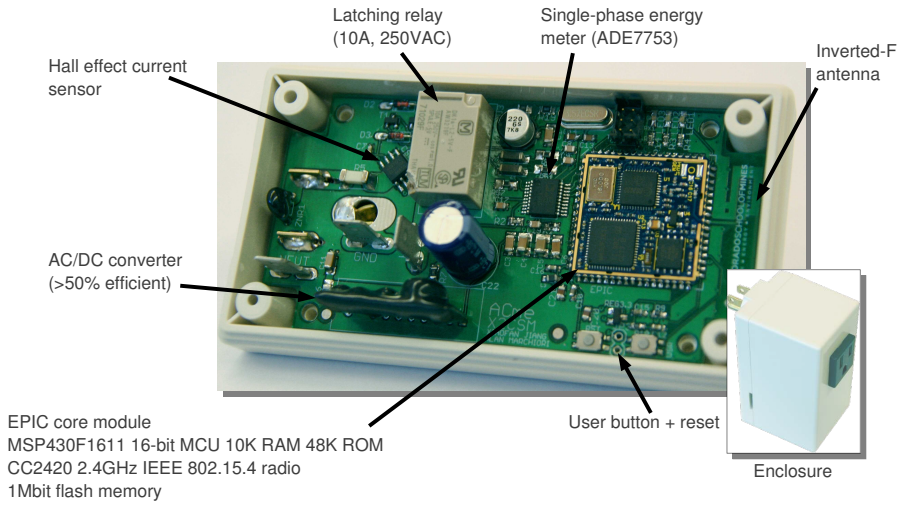


Figure 7.3: Device automation controller.

cycle communication from the sensor nodes. The powered automation and control nodes implement the receiving side of this protocol while also sharing received data using our global shared memory (GSM) abstraction, described in Chapter 7.4.2.

7.4.1 Sensor Communication Protocol lp*Send*

Sensor communication is a challenging task considering we would like the sensor node to last for years, if possible. The most common method to reduce the energy costs of radio communication is to duty-cycle the radio. Low power listening is the usual strategy where nodes periodically turn on their radio for short amounts of time to listen for incoming packets. If a data transmission is detected in one of these listen periods, the radio is kept on to receive the packet. Transmitters prefix their data transmissions with a series of messages indicating a data transmission will occur in the near future. This puts a significant burden on the transmitting node. In our application, the transmitting nodes are energy constrained, while the receiving nodes are powered by the household electric supply. This distinction justifies our approach of optimizing the protocol to minimize energy for the transmitting node.

Our sensor communication protocol is refereed to as *lp*Send for low-power send. This protocol is a one-way, sensor to router, asynchronous communications protocol. Two-way communication is possible in this architecture and is explored in [85], but is not essential to this application. The *lp*Send protocol is intended to support very-low duty-cycled operation of sensor nodes and enable long battery lifetimes. As a result, it maintains no routing state. Instead, a data transmission is initiated by a *probe* message containing a response window. Routing nodes that receive the *probe* reply with a random delay within the requested response window. Data is then sent to the source of the first response received using software retransmissions and acknowledgments provided by TinyOS. For our deployment we limit the number of *probe* messages sent to three. The response window is 100 ms. Upon receiving a probe response, the software link layer is allowed up to 10 retransmissions with 15 ms delay. The transmission fails if no probe responses are received or the link layer fails to deliver the message.

At this point the protocol is complete but there is a simple improvement. The *lp*Send protocol can also implement dynamic power control. Jeong, Culler, and Oh analyzed several power control algorithms for various traffic patterns and found zero to 37% reduction in power consumption with various traffic patterns [88]. Although reducing power consumption is one possible benefit, our primary concern is reducing contention in the network. Congestion could be a problem because in response to the *probe* message, the *lp*Send protocol requires all routing nodes to send a *probe response* message. Power control effectively limits the number of these messages by limiting the number of routing nodes receiving each *probe*.

The basic algorithm for dynamic power control is to transmit a broadcast message and count the number of responses. If the count is greater than a threshold, power is reduced. If the count is less than another threshold, power is increased. The CC2420 radio has 32 distinct power levels (0 - 31), so we apply a simple algorithm to continuously adjust the power level. The transmit power arbitrarily starts at 16 and for every

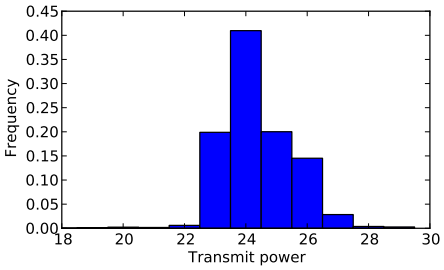


Figure 7.4: Typical sensor transmission power distribution

probe retransmission it is increased by 2. After a successful transmission, we add a small delay (10 ms) to count additional responses. If the number of responses is more than 3, the power is decreased by 1 for the next transmission. This adapts the transmit power to maintain communication with three routing nodes in the network. Our approach does not decrease the size of the transmit power adjustment as suggested by Jeong, Culler, and Oh. Because we expect long deployment times (months or years), our goal is not to converge to a single optimal value. Instead, our goal is to enable the nodes to respond quickly to the current network conditions. Figure 7.4 shows a histogram of the transmit power of a particular node over one week. We found no correlation between transmit power and time of day. In our deployment (Chapter 7.7), this protocol delivered 99.96% of the data packets to the routing nodes.

7.4.2 Automation and Control Protocol (GSM)

The communication challenge for the routing nodes is to 1) receive sensor data using the *lp*Send protocol and 2) share both sensor and energy data with other nodes in the network. Implementing the receiving side of the *lp*Send protocol is straightforward. The second challenge is more difficult because few existing solutions are available for peer-to-peer information sharing.

One exception is the TeenyLIME middleware [89]. TeenyLIME maintains a shared tuple-space between one-hop neighbor nodes. Our design is similar, with two significant differences. First, information is shared between all nodes in the network, across

multiple hops. Second, no modifications were made to the TinyOS 2.x distribution to accommodate our protocol, improving the interoperability with the TinyOS core and simplifying implementation for those already familiar with TinyOS programming.

We have previously utilized multicast communication to allow control controllers to dynamically subscribe and receive sensor data in BAC systems [83, 90]. However, one challenge facing any multicast implementation is to store the necessary state information. Our solution to this problem was to approximate the multicast state information. As a result the multicast protocol would revert to broadcast whenever the multicast forwarding rules were too complex. Furthermore, occupancy detection, a critical component of any BAC, is a good candidate for a broadcast protocol because it requires data from all motion sensors to make a proper occupancy assessment.

To explore the difference in network traffic generation from multicast and broadcast, we consider a hypothetical 64-node network. This network was constructed geometrically by iteratively adding nodes starting from an arbitrary initial location and sequentially placing a node randomly within the communication range of the last placed node. We then randomly select a source node and set of multicast subscribers. To estimate the minimum number of transmissions, we construct a graph of this network including all nodes on the shortest path between the source and each subscriber. The order of the multipoint relay (MPR) set of this graph approximates the minimum number of transmissions required for the source to reach each subscriber node. In the case of broadcast, the order of the MPR set for the entire network, generated from the randomly select source, is used. In both cases the classic MPR selection algorithm was applied [91]. The median number of transmissions required are shown in Figure 7.5 for both multicast and broadcast. For each number of multicast subscribers, 64 different (source, subscriber) sets were randomly selected. Because these results significantly depend on the network topology, density, and (source, subscriber) set, we only consider the general trend that as the number of multicast subscribers increase the savings afforded by multicast is quickly diminished. When considering

this fact, we expect that in many cases mutlicast offers little benefit while consuming significant resources. For a BAC where reliability is arguably more important than efficiency, these resources may be better utilized to buffer and retransmit sensor data than to maintain the multicast state. Therefore, we focus our work on using broadcast communication to implement the reliable information sharing needed by a WSN-based BAC system.

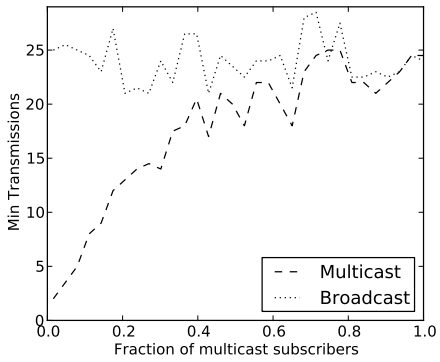


Figure 7.5: Required transmissions for increasing multicast subscribers in a random network.

Our criteria for a reliable broadcast algorithm is that in a connected network, for a given packet with infinite retransmissions and time, the packet must be delivered to every node in the network at least once. Simple flooding where every node retransmits the packet the first time it is received is often cited as having good reliability because at each node, every neighbor node will transmit the packet resulting in multiple opportunities to receive the same packet. For example, a node with n neighbors would have up to n opportunities to receive the packet. This, however, does not meet our definition of a reliable broadcast algorithm because it is *possible* for a node to not receive any of these n transmissions.

More intelligent broadcast protocols can be split into three classes: probabilistic, area based, and neighbor knowledge [92]. We considered each class of intelligent broadcast protocol for use in our application and found that while these algorithms improve efficiency, none can achieve guaranteed reliability. It is clear that probabilis-

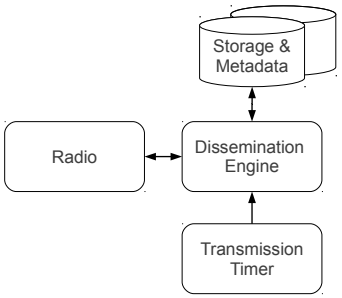


Figure 7.6: GSM protocol architecture.

tic methods will not be appropriate and that area based methods require location information, which we do not have. The neighbor knowledge methods, however, have a less obvious failing. This is their reliance on a neighbor table covering one or more hops. In our experiments, we found that errors in the neighbor table were not uncommon and that these algorithms perform poorly with erroneous neighbor information. Our final conclusion is that in order to achieve information sharing with guaranteed reliability, our algorithm cannot rely on a neighbor table.

We refer to our information sharing architecture as GSM because it implements global shared memory. It is based on the dissemination concept presented in TEP118 [93]. Dissemination, as defined in TEP118, has been implemented by Drip and DIP [94]. These protocols can synchronize data items smaller than a single radio packet between all nodes in the network. GSM provides this capability as well. However, dissemination relies on each data *item* having a predefined unique key. Due to this requirement, the senders and receivers of data must agree upon unique keys at compile time. This approach would prohibit adding a new sensor to a deployed system without reprogramming nodes in the network. To solve this problem we assign a unique key to each *data type* in the network and extend the dissemination interface to allow *any node* to disseminate instances of each data type.

The high-level architecture of GSM is shown in Figure 7.6. The primary components are the dissemination engine and the storage subsystem. Received packets are processed asynchronously to the transmission timer. All radio transmissions are

initiated from the transmission timer. This approach is used to limit contention in the network.

The storage subsystem is a generic component that is instantiated with a type key and maximum size. The component allocates memory for the maximum number of data items, headers, and metadata. The header contains the 48-bit unique identifier of the source for the data item, type key, and tag. The unique source identifier plus tag create a globally unique identifier for each GSM data item. Metadata includes a version number, last transmission time, and various other flags. The total memory overhead for each data item is 17 bytes. Interfaces are provided to create, update, and search for data items. Events are generated when a data item is changed locally or remotely.

The dissemination engine receives events from both the storage subsystem and the radio. When a data item is changed by the local node, the version number is incremented and the data item is flagged for transmission. Each time the transmission timer is fired a packet is transmitted by the node. If data has changed, or another node requested a particular item, a data packet is transmitted otherwise a summary packet is transmitted.

Data packets contain exactly one data item, including the header and version number. Receiving nodes process data packets and update their storage subsystem, if needed. Dissemination begins as a flood because each node unconditionally transmits every updated data items one time. Reliability is ensured by the summary messages.

Summary packets contain the globally unique identifier and version number for up to 12 data items. Nodes sequentially iterate through all stored data items when there are more than 12 items available. Receiving nodes process summary packets by comparing the transmitted version number to the local version number for each data item. To account for version number wrap-around, the version numbers are compared using modular arithmetic (i.e., the difference of the version numbers is compared to half the maximum value of the version type). If a new data item is available, a request

packet is sent to the source of the summary message at the next transmission event. If an old data item is detected, the node will retransmit that particular data item at the next transmission event.

This is a simple protocol that maintains a consistent global shared memory space across a multihop network. Because nodes transmit a packet every time the transmission timer fires, specifying the transmission timer becomes critical. If the transmission timer is fast, latency will be reduced but contention will be likely. To mitigate contention, we apply the desync algorithm [95] to desynchronize the transmission timer of neighboring nodes. Desync dynamically adjusts the phase of each transmission timer creating a loose TDMA schedule with no additional overhead.

7.5 Scalability Analysis

There are two primary factors affecting the scalability of the GSM protocol, these are memory constraints and bandwidth constraints. First, because each node stores a complete copy of the shared memory space, the available storage space will limit the size of the shared memory space. In our deployment (Chapter 7.7), sensor readings are 20 bytes each and the metadata overhead for each data item is 9 bytes. Resulting, a deployment of 100 sensors would require 2.9KB of memory. Our implementation stores the shared data in RAM, however, it would be possible to utilize external flash memory, especially if data was modified infrequently. Because this protocol is intended primarily for residential BAC systems where there are approximately a few sensors per room, we do not expect memory to be a limiting factor.

The second factor is the bandwidth available for data transmission. In GSM each node transmits at a fixed rate (governed by the transmission timer). In the best case with no packet loss and an ideal TDMA schedule each data item is transmitted once by each node. The effective data rate for GSM is therefore limited by the time it takes for every node to transmit each data item once. If we assume each node has a 20-byte piece of data to report, the transfer rate is only dependent on the report

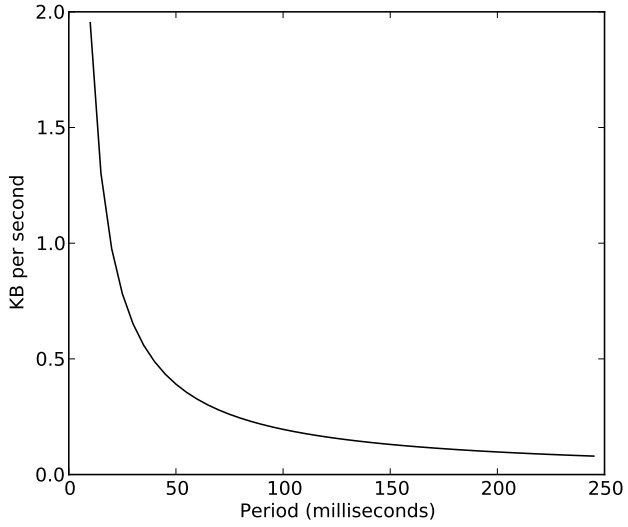


Figure 7.7: Maximum data rate for GSM where every node has data to report for varying transmission timer periods.

interval. Figure 7.7 shows the maximum data rate for GSM under this condition. For explanation, consider the a case with 10 nodes and a 100 ms transmission timer. In total there are 200 bytes (10 nodes * 20 bytes / report) of data transferred. Each node will use 10 transmissions (one for each data item) at the 100 ms timer period, consuming a total of 1 second. The data rate is therefore, 200 bytes / 1 second. This is also shown in Equation (7.2), where it is clear that the number of nodes cancel and the maximum data rate is bound only by the timer period and report size.

$$\text{Maximum data rate} = \frac{20\frac{\text{bytes}}{\text{report}} \times 10 \text{ nodes}}{10 \text{ nodes} \times 100 \text{ ms}} \tag{7.2}$$

In our deployment the data rate of each sensor is limited to one 20 byte sensor reading every minute, or effectively 0.33 bytes per second per sensor. The sensor data rate is shown in Equation (7.3).

$$\text{Sensor data rate} = \frac{20 \text{ bytes}}{60 \text{ seconds}} \tag{7.3}$$

The minimum timer period that can support a given number of sensors can be computed by setting the maximum data rate to the sensor data rate and solving for the timer period. Equation (7.4) shows this relationship where α is the number of

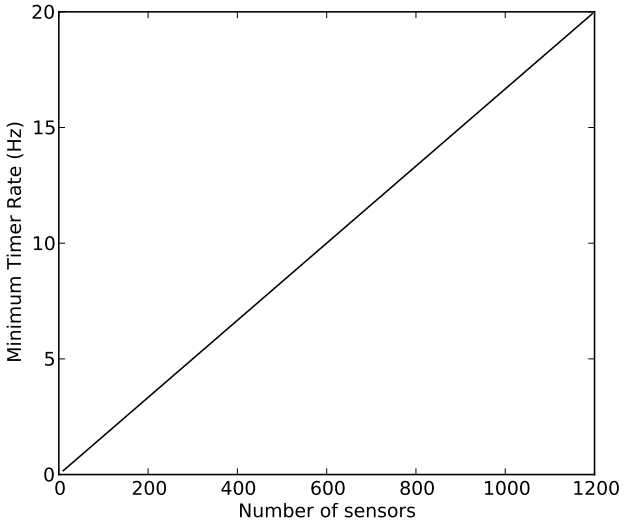


Figure 7.8: Minimum required transmission timer rate for GSM to support a given number of sensors.

sensor nodes and β is the timer period (in seconds). The timer rate is the reciprocal of the period which yields a linear relationship between the timer rate and the number of nodes, shown in Figure 7.8.

$$\frac{20\frac{\text{bytes}}{\text{report}}}{60 \text{ seconds}} \times \alpha \text{ nodes} = \frac{20\frac{\text{bytes}}{\text{report}} \times \alpha \text{ nodes}}{\alpha \text{ nodes} \times \beta \text{ seconds}} \tag{7.4}$$

It is clear from this result that GSM can support hundreds of building sensor nodes. Even with a relatively slow 1 Hz transmission timer approximately 60 sensor nodes could be supported. A faster transmission timer will decrease latency and increase data rate. However, the transmission timer period must always be long enough so that every node in the one-hop neighborhood can transmit once in the timer period. For the TelosB platform this requirement is approximately 5 milliseconds per one-hop neighbor. If we expect no more than 10 one-hop neighbors we could therefore use up to a 20 Hz transmission timer and support a total of 1,200 sensor nodes in a single network. One solution to enable even larger networks is to deploy multiple GSM networks on different radio channels and then replicate relevant data across the networks.

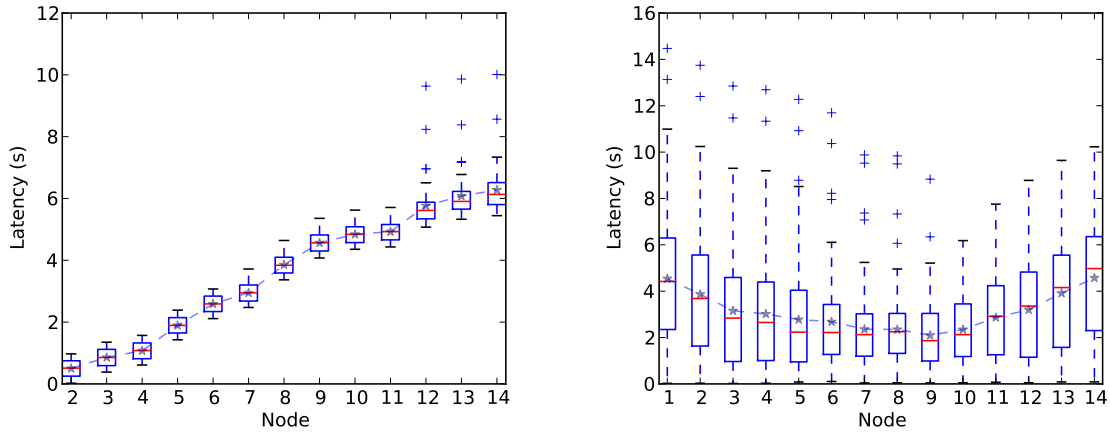
7.6 GSM Evaluation

The design of the GSM protocol ensures that every node will eventually synchronize all data items with the most recent values available in the network (i.e., packet reception rate is guaranteed to be 1, given sufficient time). This is true even when there are disconnections in the network or a node is reset or added to an active network. This property is necessary for a BAC where reliability is essential. Because the protocol guarantees the eventual delivery of every data item, the critical metric is the latency of delivery which we evaluate below.

For all latency measurements, received packets were locally timestamped and every node received every data item. Clock synchronization is achieved with a simple fast-flooding protocol initiated by node 1. Synchronization error is on the order of tens of milliseconds, which is sufficient for monitoring latency on the order of hundreds of milliseconds per hop.

The transmission timer used has a period of 1 second. Desync was used on each node to dynamically adjust the phase of the transmission timer. This period was chosen because the testbeds used have rather dense topologies (average neighbor count > 25) and the long period reduces the effect of time synchronization errors. In practice, this period can be significantly shortened to improve performance.

Figure 7.9 shows testbed results where we enforce a linear topology in software (i.e., nodes only accept packets from adjacent nodes). In (a) node 1 is the only node sourcing GSM data at a rate of one packet per minute. Every node received every update from the source node. The outliers for nodes 12, 13, and 14 are due to packet loss between nodes 11 and 12, however, this loss was detected and these nodes did synchronize with some additional delay. In general, we expect the latency to increase approximately linearly. Assuming independent clocks (they are not) and uniform random arrival times, the expected delay waiting for the transmission time to fire would be 0.5 seconds. This closely matches the computed average latency of 0.46 seconds per hop in our 13-hop linear network.



(a) Node 1 sources data.

(b) All nodes source data.

Figure 7.9: Latency in a linear network.

In (b) we configure all 14 nodes to source data at a rate of 1 packet per minute. Nodes near the ends of the network show higher latency than the nodes in the middle because, on average, they are further from the source node. The overall latency has also increased due to queuing delays because only one data item can be transmitted at a time. If multiple data items need to be transmitted, the dissemination engine arbitrarily chooses one to transmit, while the others must wait for the next transmission timer event. In these results, the median latency at node 8 was just over 2 seconds. The average hop count at this node was 2.07, resulting in an average latency of approximately one second per hop.

To explore latency in a more realistic environment, we used the moteLab [96] testbed. MoteLab is a experimental wireless sensor network deployed in Maxwell Dworkin Laboratory, the Electrical Engineering and Computer Science building, at Harvard University. This testbed is particularly relevant to evaluate networking for a BAC because 2-4 nodes are located in most offices, as we would expect for a wireless BAC system. MoteLab contains 184 nodes, but only 73 were available for our experiments. This is still far more than we would expect in a residential BAC, so it demonstrates performance for a large BAC deployment. For this experiment, we

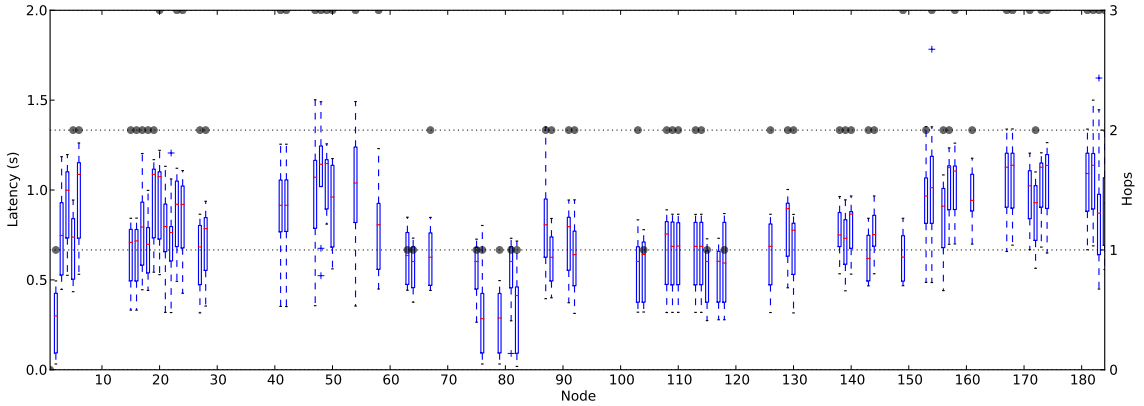
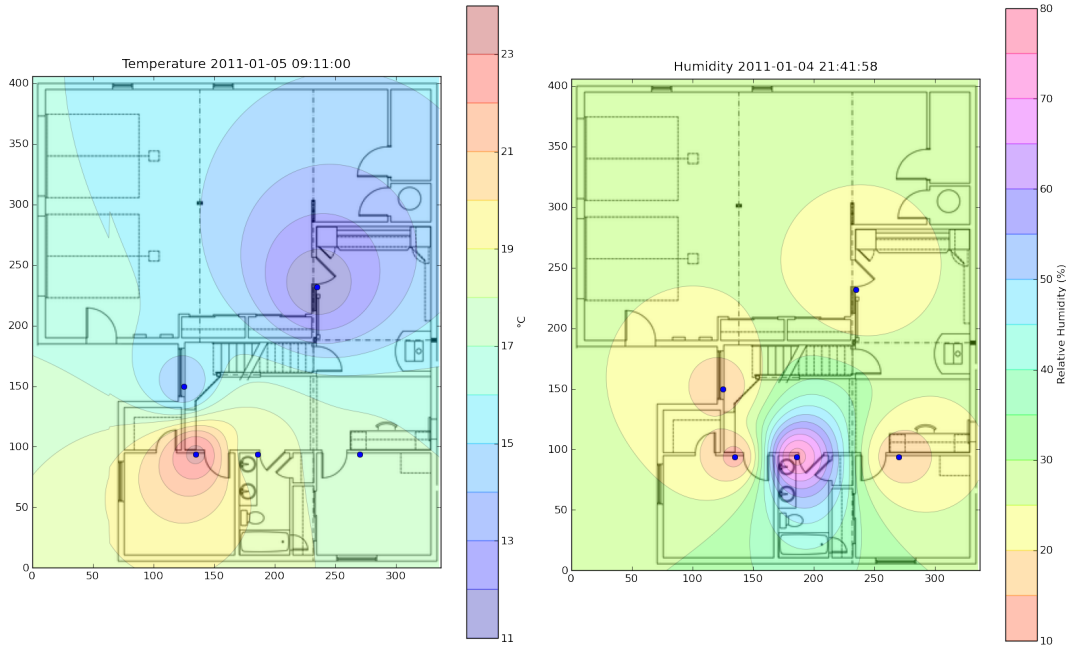


Figure 7.10: Latency (boxplot) and minimum hop count (circles) in moteLab with node 1 as the source.

again designate node 1 to transmit data at one packet per minute and we verify that each data item was successfully received by every node in the network. Figure 7.10 shows the resulting latency. Because latency is closely tied to path length, we also plot the minimum path length from node 1 to the other nodes in the network (paths with ETX > 4 are discarded). These results show the path length has less effect than expected. This is likely due to packet delivery along high-loss paths. For example, nodes 100-120 are all two or three hops from node 1, but the latency of the 3-hop nodes was only increased by approximately 0.1 seconds when compared to the 2-hop nodes. Because our previous results show the latency increase approximately 0.5 seconds per hop, we conclude that these nodes are often reachable in two hops, even though these paths are of poor link quality.

7.7 Deployment

To demonstrate and evaluate the functionality of this architecture, we have implemented three distributed control strategies: occupancy-based control, linked control, and demand response. For each building sensor node, there is a GSM data item that stores the most recent readings from the sensor. The sensor nodes use the *lp*send protocol to transmit data to the infrastructure powered automation and control nodes. These nodes then update the corresponding GSM data item for the sensor.



(a) Temperature (b) Humidity

Figure 7.11: Temperature and humidity plots.

Figure 7.11 illustrates some of the sensed environmental data interpolated using Shepard's method [97]. This interpolation does not take walls into account, but it enables quick visual inspections of the data. This data could be used by the HVAC system or simply displayed to the user.

Occupancy-based control detects whole-house occupancy and disables non-essential devices while the home is unoccupied. Whole-house occupancy detection is deceivingly simple and reliable using a combination of door and motion sensors. At a minimum, a single door and motion sensor is required for every exterior door. The occupancy detection algorithm is implemented locally by each controller in the network. We assume that all exterior doors are closed when the building is unoccupied. Therefore, the occupancy can only change immediately following an exterior door close event. After detecting this event event, a 3-minute timer is started. During this interval the number of motion events received are counted. To minimize false detections, we use a threshold of 3 or more motion counts in the interval to indicate

that the building is occupied. In the unlikely event of a misclassification, subsequent motion events are used to detect the error and operation changes to the occupied mode [98].

Table 7.3: Power consumed by controlled devices.

Device	Standby (W)	Active (W)
TV	0	105
Speaker	0	10
Subwoofer	10	15
TV	1	78
DVD Player	0	15

In our deployment we implemented occupancy-based control for two televisions. Using a P3 Kill-a-watt, we measured the standby and active power of each device, shown in Table 7.3. Surprisingly, the only device with significant standby power was the subwoofer. Although this limits the savings due to mitigating standby losses, savings can still be achieved by avoiding active loads due to devices being left on. Informally, we have found this to happen with some regularity.

Linked control is useful when several devices are used together. Common examples are a TV and DVD player or a desktop computer and printer. When one device is turned on the corresponding linked devices are turned on automatically. This strategy is even more effective than occupancy-based control at eliminating standby losses because the duration for which any of these devices is used must be less than the time the home is occupied. In our deployment, the powered speaker and subwoofer were linked to one TV while the DVD player was linked to the other TV. Because the televisions implement occupancy-based control, these devices inherit this control strategy as well. Including the additional load created by each controller (0.25 watts), automating these devices with occupancy and linked control decreases the standby losses by 88% from 11 watts to 1.25 watts. When devices are unintentionally left on, this savings is significantly increased. Based on typical occupancy and usage schedules, we expect a savings of 195 watt hours per day, or 71 kWh per year.

Our implementation of demand response assigns a priority to each device. A demand response event is initiated by a node updating a GSM demand data item with a priority and duration of the event. All nodes in the network receive this data item and devices of lower priority are disabled for the designated duration. Demand response is useful to help utilities balance generation and demand. Because we did not have access to utility demand and generation data, we were not able to evaluate this feature beyond verifying the functionality.

To quantify the expected savings using these control strategies for the average home, we use data from the Building America Research Benchmark for miscellaneous electrical loads [99] . This data includes information regarding standby power consumption, operating times, and house occupancy information. If the occupancy-based control strategy was applied to home office and home entertainment devices in the benchmark, the energy consumption of these devices would be reduced by 10%. Employing both occupancy and linked control to these devices results in a 17.5% savings. The total reduction on whole-house energy consumption using both control strategies is expected to be 2.3%, or 182 kWh, per year per household. Nationwide, assuming 50% penetration, this would reduce energy consumption by 0.04 quads per year (the total energy consumption for buildings in the US is approximately 20 quads per year) and yield a savings of 1.5×10^{10} pounds of CO_2.

7.8 Chapter Summary

This chapter presented the complete implementation of a reliable distributed BAC system using a custom wireless platform and protocols. By separating the tasks of sensing and controlling, we are able to leverage the availability of infrastructure power to create a reliable mesh-network core with ultra-low powered sensing nodes. Building state information is synchronized between all nodes in the network creating a global shared memory space. Reliability is guaranteed by the network protocol, even when nodes are added or removed. Because control decisions are made locally by each

node, the system has no single point of failure and can be expanded at any time. Testbed results show typical latency is less than 0.5 seconds per hop and depending on network topology, higher performance is possible. The average expected energy savings by applying this control approach to home entertainment and home office equipment is 17.5%, resulting in a whole-house energy reduction of 2.3%. Because this architecture is fully open and distributed, evolving deployed systems to include new automation and control strategies enables existing sensors to be leveraged for increased energy savings.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this thesis we have explored the use of wireless sensor networks to develop a new class of distributed building automation and control systems. The central idea is to wirelessly share sensed building information. Shared building information could include simple measurements (e.g., temperature, humidity, light), computed device-level state information (e.g., TV on, dishwasher running). Individually these separate bits of information have limited uses but when viewed together they enable very detailed analysis of building state and occupant behaviors. The wireless sensor network is the critical component to enable this detailed analysis. Our approach is vastly different than systems that rely on wired or wireless sensors to communicate data to a central controller. Because control decisions are made at the point of control, it is possible to develop systems that automatically adapt to a wide range of building parameters.

8.1 Summary of Dissertation Research

Chapter 3 presents two approaches for circuit-level energy disaggregation. Whole-home energy measurement is cheap and easy to set up because only one sensor is placed where the home connects to the power grid. The collected data can provide useful information for large appliances. However, the only way to monitor the energy usage of smaller devices is to install an energy meter on every device of interest. This creates a very detailed picture of household energy consumption, but requires a lot of additional hardware – one meter per device in the home. We have developed and evaluated two algorithms to disaggregate the circuit-level data into device-level estimates.

Chapter 4 describes PIM-WSN, the first general purpose multicast protocol for IPv6 wireless sensor networks. Multicast is one approach to share sensor information

between building systems. However, existing solutions for multicast in WSNs are limited because they either support multicast only from a single source node (usually the root node) or they limit the multicast group size to constrain memory usage. Our design allows any node to be a mulitcast source with an unlimited number of subscribers. We constrain the memory usage by approximating multicast group membership using a fixed sized Bloom filter. The efficiency of the protocol degrades as the false positive rate of the Bloom filter increases; however, correct operation is always maintained. Using detailed simulations we show that PIM-WSN achieves 1) high packet delivery rate (over 97%), 2) low latency per hop (less than 5 ms), and 3) lower radio utilization than all other comparable protocols (by more than 50%). Using a ten-hop testbed of TelosB motes we have verified our implementation of PIM-WSN for TinyOS 2.x with the Blip IPv6 networking stack which uses only 5,978 bytes of ROM and 235 bytes of RAM.

Chapter 5 explores the quantitative effect of occupant behaviors on building energy consumption. To do this we have evaluated eight energy-saving behaviors, as well as the use of an in-home display (IHD), in ten homes over the course of ten weeks. The results showed maximum savings ranging from 0%-20% attributed to the IHD. Additionally, we found evidence that automation is necessary to ease the more tedious tasks such as "unplug when not in use" and "unplug the TV," where less than half of the participants performed the action.

Chapter 6 illustrates how a distributed BAC system can be built using PIM-WSN. We propose a general framework where building systems can share information in order to optimize performance. To be successful, such a system must be responsive, intuitive, robust, and scalable. Using protocol independent multicast, sensors and controllers are allowed to efficiently share information in a distributed peer-to-peer fashion. Our prototype system achieved an energy savings of 7.1% - 14.6% by implementing an occupancy-based control policy.

Chapter 7 concludes by describing the specialized WSN sensor nodes developed specifically to implement distributed BAC and a global shared memory abstraction for reliably sharing sensed building information. The global shared memory approach guarantees the eventual synchronization of data in the network addressing potential reliability problems with PIM-WSN. This approach will enable the development of off-the-shelf automation systems that can be reliably deployed and used by the typical homeowner. In our study, we observed an 88% reduction in standby losses for a home entertainment system. Using published statistics on home entertainment and home office systems, we expect this approach, on average, to reduce energy consumption of these devices by 17%, or equivalently a 2.3% reduction in whole-house energy consumption. This work provides a general purpose distributed automation and control framework that can be extended to implement other energy saving measures such as intelligent lighting and adaptive HVAC systems.

8.2 Future Research Directions

There are several remaining challenges described below to fully realize the potential presented by this architecture.

Protocol efficiency for GSM needs to be studied further. The current implementation was designed for low data rate applications, such as building monitoring, where the sensors transmit one packet per minute. Other applications may require higher data rates. To achieve higher data rates, modification to the GSM protocol will be required. One possible improvement is to develop a priority transmission scheme to estimate the impact of sending each data item. For example, if a node overhears a data item transmitted by each of its known neighbors, it is unlikely to reach a new node if transmitted, so it would be assigned a low priority. Conversely, if a node has a new data item that has been transmitted by a minority of its neighbors, that item would be assigned a high

priority. An extension of this idea is to modify the TDMA approach to include a contention window where nodes could transmit high priority data before their scheduled transmission time. This could be especially useful at decreasing the end-to-end latency in large networks while still maintaining the high reliability of GSM.

Energy harvesting sensor nodes are essential to developing a user friendly system. Many indoor energy harvesting technologies have been studied including: solar, piezoelectric, vibration, thermoelectric, and acoustic. Of these, solar provides the highest power density and is the most cost effective solution. Monocrystalline silicon cells can provide 25% efficiency under ideal conditions, however, indoor lighting causes their efficiency to drop to 10% at 50 lux [100]. Typical residential indoor light levels range between 1 and 100 lux and we have found that the sensor node consumes approximately 12 Joules per day. At 50 lux and 10% cell efficiency, a 6 square inch cell would generate this amount of energy in less than one hour. The significant challenge is to develop power control strategies that adapt the operation of the sensor node to the current and predicted lighting conditions.

Energy storage also needs to be considered for the energy harvesting sensor node. One possible solution is to use an ultracapacitor. Long-life ultra-capacitors are readily available that reliably support millions of charge-discharge cycles, where most batteries would need replacement after fewer than 1,000 cycles. When fully charged, a 10 Farad ultra-capacitor could power the sensor node for up to 24 hours with no additional charging. Coupled with indoor solar charging, and adaptive power control strategies, we expect this solution to provide continuous power under typical indoor lighting conditions.

Additional building automation strategies have also been explored by other researchers. Those expected to yield the highest energy savings are HVAC control (28% reduction [76]) and lighting control. Our architecture provides a unifying framework capable of integrating many different automation and control strategies with a common platform. New sensors and actuators will be developed to implement these capabilities.

Traditional distributed control problems, such as industrial process control, would also benefit from wireless sensor networks. Because distributed control systems require reliable communication for stable operation, GSM is an appropriate solution. WirelessHART [12] is a currently available wireless solution for industrial process control that uses centralized coordination for wireless communication. In comparison, GSM is fully distributed with no central controller. A thorough comparison of these two approaches for industrial process control would provide valuable insights.

LIST OF ABBREVIATIONS

AC . . . . . Alternating Current

ADC . . . . . Analog to Digital Converter

ADMR . . . . . Adaptive Demand-driven Multicast Routing

BAC . . . . . Building Automation and Control

BAM . . . . . Branch Aggregation Multicast

BAS . . . . . Building Automation System

BMS . . . . . Building Management System

CFL . . . . . Compact Fluorescent Light

CH . . . . . Chauvenet Hall

CPM . . . . . Closest-Fit Pattern Matching

CPU . . . . . Central Processing Unit

CRC . . . . . Cyclic Redundancy Check

CT . . . . . Current Transformer

CTP . . . . . Collection Tree Protocol

DAG . . . . . Directed Acyclic Graph

DC . . . . . Direct Current

DCOCLK . . . . . Digitally Controller Oscillator Clock

DCO . . . . . Digitally Controller Oscillator

DOE . . . . . Department Of Energy

DOW . . . . . Day Of week

DVD . . . . . Digital Versatile Disc

EIA . . . . . Electronic Industries Association

EPRI . . . . . Electric Power Research Institute

ETX . . . . . Expected Transmission Count

FRM . . . . . Free Riding Multicast

FTSP . . . . . Flooding Time Synchronization Protocol

GPIO . . . . . General Purpose Input/Output

GSM . . . . . Global Shared Memory

HTTP . . . . . HyperText Transfer Protocol

HVAC . . . . . Heating, Ventilating, and Air Conditioning

IC . . . . . Integrated Circuit

ID . . . . . Identification

IEC . . . . . International Electrotechnical Commission

IEEE . . . . . Institute of Electrical and Electronics Engineers

IHD . . . . . In-Home Display

IP . . . . . Internet Protocol

KB . . . . . Kilobyte

LCD . . . . . Liquid Crystal Display

LQI . . . . . Link Quality Indicator

MAC . . . . . Media Access Control

MANET . . . . . Mobile Ad-Hoc Network

MCU . . . . . Microcontroller

MIPS . . . . . Million Instructions Per Second

MOLSR . . . . . Multicast Optimized Link State Routing

MOSPF . . . . . Multicast Open Shortest Path First

MPR . . . . . Multipoint Relay

NEMA . . . . . National Electrical Manufacturers Association

NILM . . . . . Non Intrusive Load Monitoring

NP . . . . . Nondeterministic Polynomial

NRDC . . . . . Natural Resources Defense Council

NREL . . . . . National Renewable Energy Laboratory

OLSR . . . . . Optimized Link State Routing

OS . . . . . Operating System

OSPF . . . . . Open Shortest Path First

PC . . . . . Personal Computer

PDA . . . . . Personal Digital Assistant

PDR . . . . . Packet Delivery Ratio

PIM . . . . . Protocol Independent Multicast

PIR . . . . . Passive Infrared

PLC . . . . . Programmable Logic Controller

PPM . . . . . Packets Per Minute

PRISM . . . . . Princeton Scorekeeping Method

RAM . . . . . Random Access Memory

RBP . . . . . Robust Broadcast Propagation

RFC . . . . . Request For Comment

ROM . . . . . Read Only Memory

RPL . . . . . IPv6 Routing Protocol for Low power and Lossy Networks

RRQ . . . . . Read Request

RSSI . . . . . Received Signal Strength Indicator

RTOS . . . . . Real Time Operating System

RX . . . . . Receive

SCADA . . . . . Supervisory Control And Data Acquisition

SCLK . . . . . System Clock

SNR . . . . . Signal-to-Noise Ratio

SPI . . . . . Serial Peripheral Interface

SSDP . . . . . Simple Service Discovery Protocol

SSM . . . . . Source Specific Multicast

TDMA . . . . . Time Division Multiple Access

TEP . . . . . TinyOS Extension Proposal

TFTP . . . . . Trivial File Transfer Protocol

TOSSIM . . . . . TinyOS SIMulator

TX . . . . . Transmission

UDP . . . . . User Datagram Protocol

USART . . . . . Universal Synchronous/Asynchronous Receiver/Transmitter

USB . . . . . Universal Serial Bus

US . . . . . United States

VAC . . . . . Volts Alternating Current

VAR . . . . . Volt Ampere Reactive

WPAN . . . . . Wireless Personal Area Network

WRQ . . . . . Write Request

WSN . . . . . Wireless Sensor Network

REFERENCES CITED

[1] U.S. Department of Energy. Buildings energy data book. http://buildingsdatabook.eren.doe.gov/, 2009.

[2] C. Barley, C. Haley, R. Anderson, and L. Pratsch. Building America System Research Plan for Reduction of Miscellaneous Electrical Loads in Zero Energy Homes. Technical Report NREL/TP-550-43718, National Renewable Energy Laboratory, November 2008.

[3] Y. Agarwal, T. Weng, and R. K. Gupta. The energy dashboard: Improving the visibility of energy consumption at a campus-wide scale. In *Proceedings of The First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, 2009.

[4] Google Inc. Google PowerMeter. http://www.google.org/powermeter/, accessed 2009.

[5] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and implementation of a high-fidelity ac metering network. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2009.

[6] X. Jiang, M. V. Ly, J. Taneja, P. Dutta, and D. Culler. Experiences with a high-fidelity wireless building energy auditing network. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[7] IEEE Computer Society. Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf, 2006.

[8] S. Bushby and H. Newman. The BACnet communication protocol for building automation systems. *American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) Journal*, 33(4), 1991.

[9] Echelon Corporation. Protocol specification: EIA/CEA 709.1-B-2002. http://www.echelon.com/communities/energycontrol/developers/lonworks/protocol/, accessed 2011.

[10] Modbus Organization, Inc. MODBUS protocol specification. http://www.modbus.org/, accessed 2010.

[11] ZigBee Standards Organization. Zigbee specification, 2008.

[12] HART Communication Foundation. HART communication protocol specification. `http://www.hartcomm.org/`, accessed 2010.

[13] EnOcean GmbH. Enocean. `http://www.enocean.com/`, accessed 2011.

[14] M. C. Mozer. The Neural Network House: An Environment that Adapts to its Inhabitants. In *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, 1998.

[15] Control4. Control4 website. `http://www.control4.com/`, accessed 2009.

[16] Tendril Networks, Inc. Tendril consumer website. `http://www.tendrilinc.com/`, accessed 2009.

[17] K. Padmanabh, A. Malikarjuna, S. Sen, S. P. Katru, A. Kumar, S. Pawankumar, S. K. Vuppala, and S. Paul. isense: A wireless sensor network based conference room management system. In *Proceedings of The First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, 2009.

[18] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[19] P. Levis, D. Gay, V. H, J. Hauer, B. Greenstein, M. Turon, J. Hui, K. Klues, C. Sharp, R. Szewczyk, J. Polastre, P. Buonadonna, L. Nachman, G. Tolle, D. Culler, and A. Wolisz. T2: A second generation OS for embedded sensor networks. Technical report, Technische Universitat Berlin, 2005.

[20] J. Hill, P. Bounadonna, and D. Culler. Active message communication for tiny networked sensors, `http://www.tinyos.net/papers/ammote.pdf`, access date: January 2008.

[21] D. Gay, M. Welsh, P. Levis, E. Brewer, R. V. Behren, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI)*, 2003.

[22] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[23] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-rk: An energy-aware resource-centric rtos for sensor networks. *Proceedings of the IEEE International Symposium on Real-Time Systems (RTSS)*, 0, 2005.

[24] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: System support for multimodal networks of in-situ sensors. In *Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*. ACM Press, 2003.

[25] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He. The liteos operating system: Towards unix-like abstractions for wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[26] A. Dunkels, B. Grunvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets)*, 2004.

[27] J. Labrosse. *MicroC/OS-II: The Real-Time Kernel*. CMP Books, 2 edition, 1998.

[28] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys)*, 2007.

[29] J. Polastre, J. Hui, P. Levis, J. Zhao, D. E. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[30] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *Proceedings of the 10th Workshop on Hot Topics in Operating Systems (HotOS)*, 2005.

[31] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[32] M. Durvy, J. Abeill, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Poster abstract: Making sensor networks ipv6 ready. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008.

[33] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC4944, 2007.

[34] S. Dawson-Haggerty and A. Tavakoli. Berkeley ip implementation for low-power networks. `http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip`, accessed 2009.

[35] O. Gnawali, K. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler. The tenet architecture for tiered sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[36] G. Hackmann, C. Fok, G. Roman, and C. Lu. Agimone: Middleware support for seamless integration of sensor and ip networks. In *Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2006.

[37] J. S. Rellermeyer and G. Alonso. Concierge: a service platform for resource-constrained devices. In *Proceedings of the ACM EuroSys 2007 Conference*, 2007.

[38] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. In *Proceedings of the 8th International Conference on Mobile Data Management (MDM)*, 2007.

[39] K. Aberer, M. Hauswirth, and A. Salehi. The global sensor networks, middleware for efficient and flexible deployment and interconnection of sensor networks. Technical report, Ecole Polytechnique Federale de Lausanne (EPFL), 2006.

[40] A. Kansal, S. Nath, J. Liu, and F. Zhao. Senseweb: An infrastructure for shared sensing. *IEEE MultiMedia*, 14(4), 2007.

[41] L. Luo, A. Kansal, S. Nath, and F. Zhao. Sharing and exploring sensor streams over geocentric interfaces. In *Proceedings of the 16th International Conference on Advances in Geographic Information Systems (ACM GIS)*, 2008.

[42] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: an architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 2(4), Oct.-Dec. 2003.

[43] G.W. Hart. Nonintrusive appliance load monitoring. In *Proceedings of the IEEE*, 1992.

[44] M. Kazandjieva, B. Heller, D. Gal, P. Levis, C. Kozyrakis, and N. McKeown. PowerNet: A magnifying glass for computing system energy. In *Proceedings of the Stanford Energy & Feedback Workshop: End-Use Energy Reductions through Monitoring, Feedback, and Behavior Modification*, 2008.

[45] Greenbox Technology Inc. Greenbox website. `http://getgreenbox.com/`, accessed 2009.

[46] Enetics Inc. Enetics website. `http://www.enetics.com/`, accessed 2010.

[47] G. W. Hart. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society Magazine*, 8(2), Jun 1989.

[48] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong. Power signature analysis. *IEEE Power and Energy Magazine*, 1(2), Mar-Apr 2003.

[49] Y. Kim, T. Schmid, Z. M. Charbiwala, and M. B. Srivastava. ViridiScope: Design and implementation of a fine grained power monitoring system for homes. In *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp)*, 2009.

[50] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd. At the flick of a switch: detecting and classifying unique electrical events on the residential power line. In *Proceedings of the 9th International Conference on Ubiquitous Computing (UbiComp)*, 2007.

[51] U.S. Environmental Protection Agency. TVs key ENERGY STAR product criteria. `http://www.energystar.gov/index.cfm?c=tv_vcr.pr_crit_tv_vcr`, Accessed 2010.

[52] Continental Control Systems, LLC. WattNode modbus. `http://www.ccontrolsys.com/products/wattnode_modbus.html`, accessed 2010.

[53] A. Marchiori and Q. Han. Using circuit-level power measurements in household energy management systems. In *Proceedings of The First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, 2009.

[54] A. Faruqui, S. Sergici, and A. Sharif. The impact of informational feedback on energy consumption-a survey of the experimental evidence. The Brattle Group, Inc., May 2009.

[55] B. Chen, K. Muniswamy-Reddy, and M. Welsh. Ad-hoc multicast routing on resource-limited sensor nodes. In *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN)*, 2006.

[56] R. Flury and R. Wattenhofer. Routing, anycast, and multicast for mesh and sensor networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2007.

[57] Q. Huang, C. Lu, and G. Roman. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, 2002.

[58] A. Okura, T. Ihara, and A. Miura. Bam: branch aggregation multicast for wireless sensor networks. In *Proceedings of The Second IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2005.

[59] J. A. Sanchez, P. M. Ruiz, and I. Stojmenovic. Energy-efficient geographic multicast routing for sensor and actuator networks. *Computer Communications*, 30(13), 2007.

[60] A. Sheth, B. Shucker, and R. Han. VLM2: A very lightweight mobile multicast system for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.

[61] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4:153–162, 1996.

[62] J. Moy. Multicast extensions to OSPF. RFC1584, 1994.

[63] A. Laouti, P. Jacquet, P. Minet, L. Viennot, T. Clausen, and C. Adjih. Multicast Optimized Link State Routing. Technical report, Institut national de recherche en informatique et en automatique (INRIA), 2003.

[64] S. Ratnasamy, A. Ermolinskiy, and S. Shenker. Revisiting IP multicast. *ACM SIGCOMM Computer Communication Review*, 36(4):15–26, 2006. ISSN 0146-4833.

[65] S. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[66] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza. RBP: robust broadcast propagation in wireless networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[67] A. Marchiori, L. Guo, J. Thomas, and Q. Han. Realistic performance analysis of WSN protocols through trace based simulation. In *Proceedings of The Seventh ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, 2010.

[68] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1), 1998.

[69] Natural Resources Defense Council (NRDC). How to reduce your energy consumption. `http://www.nrdc.org/air/energy/genergy.asp`, accessed 2010.

[70] Margaret F. Fels. PRISM: An introduction. *Energy and Buildings*, 9, 1986.

[71] U.S. Census Beureau. America's families and living arrangements: 2007. `http://www.census.gov/population/www/socdemo/hh-fam/p20-561.pdf`, 2009.

[72] U.S. Census Beureau. American housing survey for the united states: 2007. `http://www.census.gov/prod/2008pubs/h150-07.pdf`, Issued 2008.

[73] U.S. Census Beureau. Characteristics of new housing: 2009. `http://www.census.gov/const/www/charindex.html`, 2009.

[74] U.S. Energy Information Administration. U.S. household electricity report. `http://www.eia.doe.gov/emeu/reps/enduse/er01_us.html`, 2005.

[75] B. Neenan. Residential electricity use feedback: A research synthesis and economic framework. Technical Report 1016844, Electric Power Research Institute, 2009.

[76] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.

[77] R. Anderson and D. Roberts. Maximizing residential energy savings: Net zero energy home technology pathways. Technical Report NREL/TP-550-44547, National Renewable Energy Laboratory, 2008.

[78] C. Barley, C. Haley, R. Anderson, and L. Pratsch. Building america system research plan for reduction of miscellaneous electrical loads in zero energy homes. Technical Report NREL/TP-550-43718, National Renewable Energy Laboratory, 2008.

[79] U.S. Department of Energy. Electric power monthly. `http://www.eia.doe.gov/cneaf/electricity/epm/epm_sum.html`, accessed 2010.

[80] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler. Energy metering for free: Augmenting switching regulators for real-time monitoring. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[81] ZigBee Alliance and HomePlug Powerline Alliance. Smart Energy Profile 2.0 Technical Requirements Document, April, 2010.

[82] Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright. Simple service discovery protocol/1.0. draft-cai-ssdp-v1-03, 1999.

[83] A. Marchiori and Q. Han. PIM-WSN: Efficient multicast for IPv6 wireless sensor networks. In *Proceedings of the 12th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, 2011.

[84] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[85] T. Schmid, R. Shea, M. B. Srivastava, and P. Dutta. Disentangling wireless sensing from mesh networking. In *Proceedings of the 7th Workshop on Embedded Networked Sensors (HotEmNets)*, 2010.

[86] M. Ceriotti, M. Corr, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. L. Cigno, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele. Is there light at the ends of the tunnel? wireless sensor networks for adaptive lighting in road tunnels. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN/SPOTS)*, 2011.

[87] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler. A building block approach to sensornet systems. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008.

[88] J. Jeong, D. Culler, and J. Oh. Empirical analysis of transmission power control algorithms for wireless sensor networks. In *Proceedings of the 4th international conference on Networked Sensing Systems (INSS)*, 2007.

[89] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco. Programming wireless sensor networks with the teenylime middleware. In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware (Middleware)*, 2007.

[90] A. Marchiori and Q. Han. Distributed wireless control for building energy management. In *Proceedings of The Second ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, 2010.

[91] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical report, Institut national de recherche en informatique et en automatique (INRIA), 2000.

[92] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*, 2002.

[93] P. Levis and G. Tolle. Dissemination of small values. `http://www.tinyos.net/tinyos-2.x/doc/html/tep118.html`, accessed 2010.

[94] K. Lin and P. Levis. Data discovery and dissemination with DIP. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[95] J. Degesys, I. Rose, A. Patel, and R. Nagpal. DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[96] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: a wireless sensor network testbed. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[97] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 23rd ACM National Conference*, 1968.

[98] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the Second ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys)*, 2010.

[99] R. Hendron and C. Engebrecht. Building america research benchmark definition: Updated december 2009. Technical Report NREL/TP-550-47246, National Renewable Energy Laboratory, 2010.

[100] J. F. Randall and J. Jacot. The performance and modelling of 8 Photovoltaic materials under variable light intensity and spectra. In *World Renewable Energy Congress VII & Expo*, 2002.

APPENDIX - PUBLICATIONS

This dissertation has yielded the following publications and submissions:

A.1 Appeared

1. Alan Marchiori and Qi Han, A Foundation for Interoperable Sensor Networks with Internet Bridging, The Fifth Workshop on Embedded Networked Sensors (HotEmNets), 2008.

2. Alan Marchiori and Qi Han, A Two-stage Bootloader to Support Multi-application Deployment and Switching in Wireless Sensor Networks, The 7th International Conference on Embedded and Ubiquitous Computing (EUC), 2009.

3. Alan Marchiori and Qi Han, Using Circuit-Level Power Measurements in Household Energy Management Systems, The First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency in Buildings (BuildSys) held in conjunction with ACM SenSys, 2009.

4. Alan Marchiori, Lin Guo, Josh Thomas, and Qi Han, Realistic Performance Analysis of WSN Protocols Through Trace Based Simulation, The Seventh ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN), 2010.

5. Alan Marchiori and Qi Han, Distributed Wireless Control for Building Energy Management, The Second ACM Workshop On Embedded Sensing Systems For

Energy-Efficiency in Buildings (BuildSys), 2010.

6. Alan Marchiori, Douglas Hakkarinen, Qi Han, and Lieko Earle, Circuit-Level Load Monitoring for Household Energy Management, IEEE Pervasive Computing's Special Issue on Smart Energy Systems, Jan.Mar. 2011.

7. Alan Marchiori and Qi Han, PIM-WSN: Efficient Multicast for IPv6 Wireless Sensor Networks, The 12th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), 2011.

A.2 Under Review

1. Alan Marchiori, Qi Han, William C. Navidi, and Lieko Earle, Building the Case For Automated Building Energy Management, submitted to Pervasive and Mobile Computing, 2011.

2. Alan Marchiori and Qi Han, GSM: Global Shared Memory Approach for Wireless Building Automation and Control, submitted to The 31st Annual IEEE International Conference on Computer Communications (INFOCOM), 2012.

3. Cory Jensen, Nicholas Gerstle, and Alan Marchiori, Evaluating Energy Efficiency Retrofit Potential in Small Commercial Buildings Using Wireless Sensor Networks, submitted to the 3rd ACM Workshop on Embedded Sensing Systems for Energy Efficiency in Buildings (BuildSys), 2011.