Stony Brook University

Data-Adaptive Hierarchical Protocols for Wireless Sensor Networks and Health Monitoring Systems

A Dissertation Presented

by

Guofeng Hou

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

May 2012

Stony Brook University

The Graduate School

Guofeng Hou

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

K. Wendy Tang - Dissertation Advisor
Associate Professor, Department of Electrical & Computer Engineering

Thomas G. Robertazzi
Professor, Department of Electrical & Computer Engineering

Dantong Yu
Adjunct Professor, Department of Electrical & Computer Engineering
Brookhaven National Laboratory

Yuefan Deng
Professor, Department of Applied Mathematics & Statistics

Eric Noel - Dissertation Co-Advisor
AT&T Labs

This dissertation is accepted by the Graduate School

Charles Taber
Interim Dean of the Graduate School

Abstract of the Dissertation

Data-Adaptive Hierarchical Protocols for Wireless Sensor Networks and Health Monitoring Systems

by

Guofeng Hou

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2012

Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power wireless sensor nodes that are small in size and communicate untethered over short distances. These wireless sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on the collaborative effort of a large number of nodes. Networking together hundreds or thousands of cheap wireless sensor nodes allows users to accurately monitor a remote environment by intelligently combining data from the individual nodes. Such networks require robust wireless communication protocols that are energy efficient, data transmission efficient and support long network lifetime.

In this Ph.D. dissertation, we analyzed and implemented the low-energy adaptive clustering hierarchy (LEACH) protocol, a leading protocol architecture for wireless sensor networks. LEACH combines the ideas of energy-efficient cluster-based routing and media access together with application-specific data aggregation to achieve a desired performance. In an effort to improve the power consumption and network lifetime, we proposed Dynamic LEACH, or D-LEACH, a data-adaptive hierarchical protocol based on LEACH. The idea of D-LEACH is to dynamically change the likelihood for each node to send data to the base station, based on the similarity of the data within each node cluster. We implemented D-LEACH on the TinyOS platform, ran experiments to analyze its performance, and compared against other major protocols. The analysis shows that in many cases D-LEACH achieves a superior performance when compared to LEACH and XMesh protocols' power consumption, total data received, and network lifetime.

In the second part of this dissertation, we proposed and implemented a vital sign monitoring system based on wireless sensor network hardware. Our goal was to design a wireless sensor system, the Health Tracker 2000, which can monitor users' vital signs, notify relatives and medical personnel of users' status during life threatening situations. The Health Tracker 2000 combines wireless sensor networks, existing vital sign monitoring technology to inform medical personnel of users' health status. The use of wireless technology makes it possible to deploy our system in all types of homes and facilities. Since radio frequency waves can travel through walls and fabric, our system can send vital signs information to a central monitoring computer via a miniature transmitter network. Such information can then be easily accessed from any location over the Internet.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am extremely grateful to Professor Wendy Tang, my Ph.D. dissertation advisor and mentor, for the numerous comments and suggestions that she made during my Ph.D. studies at Stony Brook. Professor Tang taught me what a researcher need to know about research.

I would also like to thank Dr. Eric Noel, my Ph.D. dissertation co-advisor, for his many very helpful comments and suggestions.

In addition, I appreciate the support from the Electrical and Computer Engineering Department. I had the privilege of interacting with wonderful, bright, and talented people here. It has been a wonderful experience studying at Stony Brook.

Finally I would like to thank my wife Lili and my parents, for their unlimited love and support. They have stood by me in everything I have done, providing constant support, encouragement and love.

Chapter 1

Overview

Wireless sensor networks consisting of a large number of small nodes have become very popular over the past few years. The nodes sense environmental changes and report them to other nodes over a flexible network architecture. Sensor nodes are best used for deployment in hostile environments or over large geographical areas. Some basic concepts and architecture of sensor networks are introduced below.

1.1 Usage of Sensor Networks

Sensor networks are useful in a variety of domains, such as:

Environmental Observation

Sensor networks can be used to monitor environmental changes. An example could be water pollution detection in a lake located near a factory that uses chemical substances [1]. Sensor nodes could be randomly deployed in unknown and hostile areas to relay the exact origin of a pollutant to a centralized authority so as to take appropriate measures and limit pollution

spreading. Other examples include forest fire detection, air pollution and rainfall observation for agriculture [2,3].

Military Monitoring

The military uses sensor networks for battlefield surveillance. Sensors could monitor vehicular traffic and track enemy positions [4].

Building Monitoring

Sensors can also be used in large buildings or factories to monitor environmental changes. In this situation, thermostats and temperature sensor nodes are deployed throughout the building's area [5]. In addition, sensors could be used to monitor vibrations that could damage the structure of a building [5].

Healthcare

Sensors can be used in biomedical applications to improve the quality of the provided care. Sensors can be implanted in the human body to monitor medical problems such as cancer and help patients maintain their health [6].

1.2 Wireless Sensor Networks Model

To monitor the environment, a wireless sensor network consists of hundreds or thousands of low cost nodes which could either be placed at fixed locations or randomly deployed. Sensors usually communicate with each other using a multi-hop approach, in which data from sensor nodes traverse one or multiple nodes to reach a destination. That way, data can reach destinations

beyond sensors' radio range. The flow of data converges to special nodes called base stations (sometimes also referred to as sinks). A base station is typically located on the periphery of the sensor network. It collects data from the sensor nodes and transmits it to a remote control station. In other words, a base station links the sensor network to another network (like a gateway) to disseminate the data sensed for further processing. Base stations have enhanced capabilities over simple sensor nodes since they must do complex data processing; this justifies the fact that base stations have workstation/laptop class processors, sufficient memory, energy, storage and computational power to perform their tasks well. Usually, the communication between base stations is initiated over high bandwidth links [7].

As it may be inconvenient or impossible to recharge sensor nodes batteries, one of the biggest challenges of sensor networks is power consumption, which is greatly affected by the communication distance between nodes. Therefore, all aspects of the node, from the hardware to the protocols, must be designed in an energy efficient manner. To solve this issue, several solutions for reducing energy consumption have been introduced, including aggregation points, clustering, sleep mode, and randomized assignment of high energy-consumption tasks to nodes.

Aggregation points are introduced in the network to reduce the total number of messages exchanged between nodes and reduce energy consumption. Usually, aggregation points are regular nodes that receive data from neighboring nodes, perform some processing, and forward the filtered data to the next hop [8].

Similar to aggregation points is clustering. Sensor nodes are organized into clusters, each cluster having a "cluster head" as the leader. The communication within a cluster must travel

through the cluster head, which then is forwarded to a neighboring cluster head until it reaches its destination, the base station [9].

Another method for saving energy is having nodes place themselves in sleep state when idle and wake up as required by new tasks.

When tasks are not uniformly distributed across nodes (i.e. aggregation), randomized rotation is used to balance energy-consumption across nodes [8].

1.3 LEACH Protocol

LEACH (Low Energy Adaptive Clustering Hierarchy) protocol [8,9] was designed for wireless sensor networks. There, the data from individual nodes are sent to a central base station, sometimes located far from the sensor network, through which the end-user can access the data. There are several desirable properties for protocols on such networks: Use 100's - 1000's of nodes; Maximize network lifetime; Maximize network coverage; Use identical battery-operated nodes.

Conventional network protocols, such as direct transmission, minimum transmission energy, multi-hop routing, and clustering all have drawbacks that prevent them from achieving these desirable properties. LEACH includes distributed cluster formation, local processing to reduce global communication, and randomized rotation of cluster-heads. Together, these features allow LEACH to achieve the desired properties. Initial simulations show that LEACH is an energy-efficient protocol that extends system lifetime beyond a general-purpose multihop approaches [8,9]. A detail discussion of the LEACH protocol is in Section 3.1.

1.4 Research Challenges

Due to sensors' limited communication bandwidth and energy, several design issues must be addressed so as to achieve an effective and efficient operation of wireless sensor networks.

Energy Saving Algorithms

Since sensor nodes use batteries for power that are difficult to replace once consumed (remember that often sensor nodes are deployed in remote and hostile environments), it is critical to design algorithms and protocols that utilize minimal energy. To do so, researchers must reduce communication between sensor nodes, simplify computations and apply lightweight security solutions.

Location Discovery

Many tracking applications require knowledge of the physical location of a sensor node in order to link sensed data with the object under investigation. Furthermore, many routing protocols need the location of sensor nodes to forward data across the network. Location discovery protocols must be designed in such a way that minimum information is exchanged between nodes to discover their location. Cost is another factor that influences design; manufacturers try to keep the cost at minimum levels. If the cost is high, the adoption and deployment of sensor technology will be prohibited.

Security

Security solutions are constrained when applied to sensor networks. For example, cryptography requires complex processing to encrypt transmitted data. Secure routing, secure discovery and verification of location, key establishment and trust setup, attacks against sensor nodes, secure group management and secure data aggregation are some of the many challenges that need to be addressed within a security context.

1.5 Simulation and Implementation of LEACH Protocol

In this dissertation, we implemented the LEACH protocol on both TinyOS and ns2 platforms.

TinyOS

TinyOS is a free and open source component-based operating system and platform targeting wireless sensor networks (WSNs). TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes [10]. It is intended to be incorporated into Smartdust. Smartdust is a hypothetical system of many tiny microelectromechanical systems (MEMS) such as sensors, robots, or other devices, that can detect, for example, light, temperature, vibration, magnetism or chemicals. Smartdust MEMS are usually networked wirelessly and are distributed over some area to perform tasks, usually sensing [11]. TinyOS started as collaborative effort between the University of California Berkeley, Intel Research and Crossbow Technology. It has since grown as an international consortium, the TinyOS Alliance.

It is very challenging to implement LEACH protocol on TinyOS platform, as there are many challenging technical issues we had to solve.

Our TinyOS implementation of LEACH can be run on both hardware motes and TOSSIM, an emulation environment for TinyOS applications. Execution of TinyOS applications on TOSSIM is identical as on hardware motes [67].

ns2

ns2 stands for network simulator (ver 2). It is a discrete event simulator targeted at networking research. ns2 is an object-oriented simulator developed as part of the VINT project at the University of California in Berkeley. ns2 is extensively used by the networking research community right now. It provides substantial support for simulation of TCP, routing, multicast protocols over wired and wireless (local and satellite) networks, etc. The simulator is event-driven and runs in a non-realtime fashion. It consists of C++ core methods and uses Tcl and Object Tcl shell as interface, allowing the simulation script to describe the model to simulate.

We ported the MIT's LEACH simulation which is implemented on ns2 version 2.1b5 (released in year 2000) to the latest ns2 version available at the time of this research, ns 2.33. There are many technical challenges we had to solve, due to the significant software architecture and component changes in ns2 over the last 10 years [13].

1.6 D-LEACH Protocol

In an effort to improve the power consumption and network lifetime of LEACH, we propose Dynamic LEACH, or D-LEACH. The idea behind D-LEACH is to dynamically change

the likelihood for each node to send data to the cluster head, based on the similarity of data within each node cluster. We implemented D-LEACH on the TinyOS platform, ran experiments, analyzed the performance of D-LEACH protocol, and compared its performance with other major protocols. The results show that D-LEACH achieved a much superior performance than LEACH protocol and XMesh protocol, in term of power consumption, total data received and network lifetime.

1.7 A Wireless Health Monitoring System

We proposed and implemented a vital sign monitoring system based on wireless sensor network hardware. In this research, our goal was to design a wireless sensor system, the Health Tracker 2000, that can monitors users' vital signs and notifies relatives and medical personnel of their status during life threatening situations.

The Health Tracker 2000 combines wireless sensor networks, existing vital sign monitoring technology to simultaneously monitor vital signs of the users. The use of wireless technology makes it possible to install the system in all types of homes and facilities. Radio frequency waves can travel through walls and fabric, sending the vital signs information to a central monitoring computer via a miniature transmitter network. Such information can easily be accessed from any location over the Internet.

Chapter 2

Sensor Networks Background

2.1 Sensor Networks Components

The main components of sensor nodes include a sensing unit, a processing unit, a transceiver, and a power unit as shown in Fig. 2.1. Each component is described in the next sections.

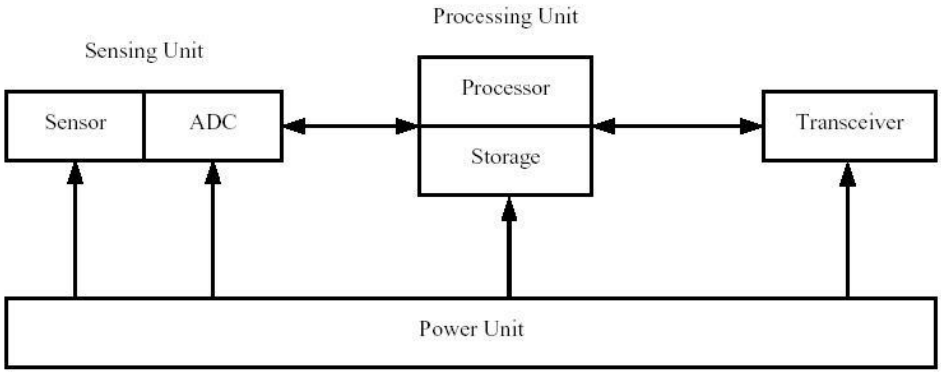


Figure 2.1: Sensor Network Components

Sensing Unit

The main function of the sensing unit is to sense or measure physical data from the target area. The sensor generates an analog voltage or signal that corresponds to the observed

phenomenon. The measured signal is digitized by an analog-to-digital converter (ADC) and delivered to the processing unit for further analysis [14,15]. The sensing unit is the current technology bottleneck because the sensing units have much lower transmission speed than the semi-conductors [16]. Sensing technology has not progressed as fast as semi-conductors. Also, sensors are being applied to the real physical world, while the processing unites and transceivers operate in a somewhat controlled environment. Sensing units are front-end components within sensor nodes used to transform one form of energy into another.

Processing Unit

The processing unit plays a major role in managing collaboration with other sensors to achieve predefined tasks. There are currently several families of processing unit, including microcontrollers, microprocessors, and field-programmable gate arrays (FPGAs) [17]. FPGAs consume more energy and are not compatible with traditional programmable methodologies. However, they can be reprogrammable and reconfigurable to eliminate deployment costs [18]. Non-volatile memory and interfaces such as ADCs can be integrated onto a single integrated circuit [14,18]. The processing unit needs storage for tasking and to minimize the size of transmitted messages by local processing and data aggregation [16]. Flash memory is widely used due to its cost and storage capacity.

Transceiver

There are three deploying communication schemes in sensors including optical communication (laser), infrared, and radiofrequency (RF). Lasers consume less energy than radio and provide high security, but require a line of sight and are sensitive to atmospheric conditions. Infrared, like lasers, needs no antenna but is limited in its broadcasting capacity. RF is the most

easy to use but requires an antenna. Various energy consumption reduction strategies have been developed such as modulation, filtering, and demodulation. Amplitude and frequency modulation are standard mechanisms. Amplitude modulation is simple but susceptible to noise [14]. The RF Monolithics TR1000 and Chipcon 1000 are commercial radios widely used in sensor applications [14,18]. Chipcon 1000 is easily programmed for operation at frequencies between 300 MHz and 1000 MHz [18].

Power Unit

Power consumption is the main bottleneck of sensor networks. Any energy preservation schemes can help to extend sensor networks' lifetime. Batteries used in sensors can be categorized into two groups; rechargeable and non-rechargeable. Often in harsh environments, it is impossible to recharge or change a battery. New sensors are developed to be able to renew their energy from solar or vibration energy [14,16]. Alkaline batteries have a wide voltage range and large physical size whilst lithium provides a constant voltage supply but with very low nominal discharge currents. Nickel Metal Hydride can be recharged but with a significant decrease in energy density [14]. Two major power saving policies can be found in [18]: DPM and DVS. Unused devices can be shut down and activated when required. This is called "Dynamic Power Management (DPM)" which requires support from the operating system and stochastic analysis to predict future events. In another approach, Dynamic Voltage Scheduling (DVS), power can be varied to allow for a non-deterministic workload. DVS is used in the TinyOS operating system.

Discussion

In order to develop an efficient application, high performance hardware components are required. The current research aim is to build the smallest sensor node with the least energy consumption. The Smartdust project [19] was established to develop a very small sensor node, a few cubic millimeters in size, which can remain suspended in the air. To conclude, a sensor node consists of various components, all of which must combine to achieve the predefined goal.

2.2 Technology Survey

Sensor nodes are small. Fig. 2.2 presents the Mica2 sensor node, which is the most popular research platform at the moment. Founded in 1995, Crossbow Technology, Inc. is the leading end-to-end solutions supplier in wireless sensor networks and the largest manufacturer of wireless sensor networks [66].

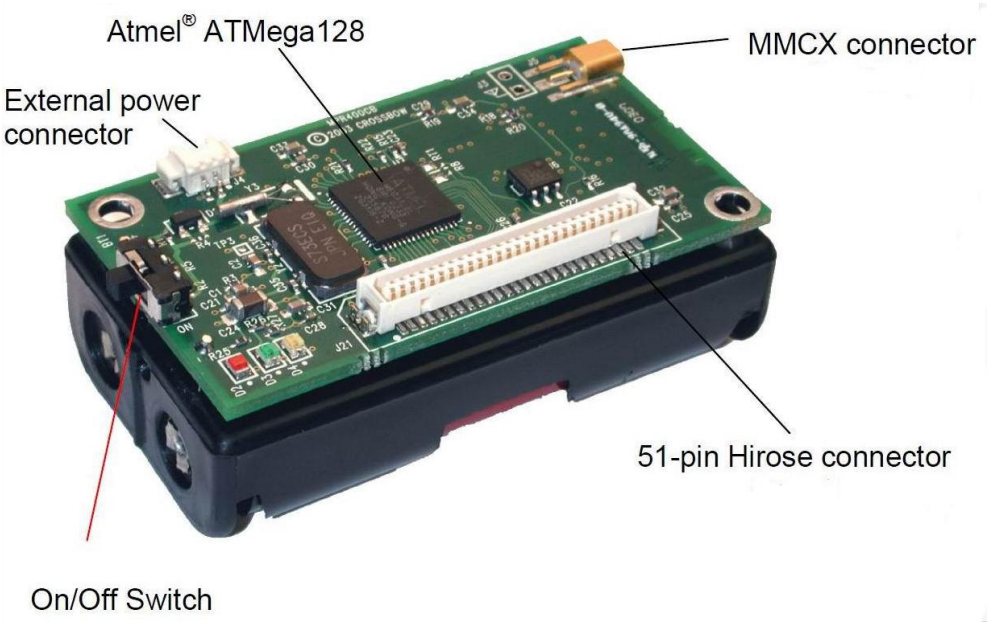


Figure 2.2: Mica2 Sensor Node

The main components of a typical MICA2 sensor node include an antenna and a radio frequency (RF) transceiver to allow communication with other nodes, a memory unit, a CPU, the sensor unit (i.e. thermostat) and the power source which is usually provided by batteries. The operating system running on Crossbow sensor nodes is called TinyOS and was initially developed at the University of California, Berkeley. TinyOS is designed to run on platforms with limited computational power and memory space. The programming language of TinyOS is stylized C and uses a custom compiler called NesC. Though it may work on other platforms, the supported platforms are Linux RedHat 9.0, Windows 2000, and Windows XP. We installed TinyOS on one of the supported platforms, Windows XP, to develop TinyOS applications. Further information can be obtained from the official TinyOS website http://webs.cs.berkeley.edu/tos/.

Some of the capabilities of the recent sensor network platforms organized by device class are listed in [51].

The recent research and development of first-generation wireless sensor network platforms is now feeding back on itself to help system engineers define a new generation of hardware better able to meet network demands.

Hardware Progression

The progression of sensor-network hardware is influenced by Moore's Law on the design. For all platform classes except special-purpose sensor nodes, Moore's Law promises an increase in performance for a given power budget. A Mica2 node has roughly eight times the memory and communication bandwidth of its predecessor, the Rene node, despites involving the same power and cost. Gateway and high-bandwidth devices have achieved similar performance

growths without significantly changing their power or cost requirements. In contrast, the special-purpose sensor nodes (such as Spec) use advances derived from Moore's Law to reduce their power consumption and cost requirements while maintaining the same performance level.

Part of the performance increase in the generic-sensor-node class is due to new CMOS radios specifically designed for low data rates and low power consumption. In addition, to improve raw radio performance metrics, the communication interfaces provided by low-power radio now include specialized hardware support to help reduce the peak load placed on the CPU. Low-power controllers can burst data out over the RF channel at rates several times faster than previous generation of radios. Moreover, early hardware designs used the microcontroller to duty cycle the radio and check for channel activity [14]. Next-generation radios will have built-in state machines that perform this operation automatically [51].

Software and Interface Standards

Engineers and researchers in the field of low-power wireless technology are pursuing a protocol-standardization effort aimed at allowing future devices to interoperate with one another. The 802.15.4 standard provides a specification of the RF channel and signaling protocol to be used [56]. Built atop 802.15.4 is the Zigbee protocol, a specification of the application-level communication protocol between devices. To put Zigbee and 802.15.4 in perspective relative to the platforms we've discussed here, 802.15.4 determines which radio hardware to use and Zigbee determines the content of messages transmitted by each networked node [56]. Following the availability of the first 802.15.4 radios, researchers have sought to develop TinyOS drivers. When these drivers are completed and released, existing sensor-network applications will be able to take advantage of the new capabilities of the 802.15.4 chips [51].

Even as the standardization process advances, it is not clear whether a comprehensive set of standard protocols will ever be available to meet all application requirements. Unlike traditional Internet applications, sensor network applications demand protocols that are optimized for their unique communication patterns. Additionally, the for-members-only nature of Zigbee standards and other proprietary solutions impose additional hurdles on any widespread sensor-network standard-setting process and adoption. In this environment, TinyOS's ability to allow application developers to assemble custom protocols from individual networking building blocks will continue to be the preferred sensor-network development strategy. Developers will likely start with generic TinyOS protocol implementations, then customize as needed to satisfy application-specific requirements [51].

2.3 Routing Protocols

Routing in sensor networks is very challenging due to several characteristics that distinguish them from contemporary communication and wireless ad-hoc networks. First of all, it is not possible to build a global addressing scheme for the deployment of the sheer number of sensor nodes because of the very large number of nodes and the associated overhead. In sensor networks, globally unique addresses would need to be very large--at least as large as Ethernet's 48 bits address--compared to the typical few bits of data attached to them. Therefore, classical IP-based protocols cannot be applied to sensor networks. Local addressing is needed. Second, in contrary to typical communication networks, all applications of sensor networks require the flow of sensed data from multiple regions (sources) to a particular sink. Third, generated data traffic has significant redundancy in it since multiple sensors within the vicinity of a phenomenon may

generate same data. Such redundancy needs to be exploited by the routing protocols to improve energy and bandwidth utilization. Fourth, sensor nodes are tightly constrained in terms of transmission power, on-board energy, processing capacity and storage; and thus require careful resource management.

To leverage these differences, many new algorithms have been proposed to solve the problem of routing data in sensor networks. These routing mechanisms have considered the characteristics of sensor nodes along with the application and architecture requirements. Almost all of the routing protocols can be classified as data-centric, hierarchical or location-based. There are few routing protocols based on network flow or quality of service (QoS) awareness, such as Maximum lifetime energy routing, Maximum lifetime data gathering, Minimum cost forwarding, Energy-Aware QoS Routing Protocol, and SPEED which are discussed later. Data-centric protocols are query-based and depend on the naming of the desired data, which helps eliminate many redundant transmissions. Hierarchical protocols aim at clustering the nodes so that cluster heads can do some aggregation and reduction of data in order to save energy. Location-based protocols utilize the position information to relay data to desired regions rather than the whole network. The last category includes routing approaches that are based on general network-flow modeling and protocols that strive to meet some QoS requirements along with the routing function. In this section, we explore the routing mechanisms for sensor networks developed in recent years. Each routing protocol is discussed under the proper category.

2.3.1 Data-centric Protocols

In many applications of sensor networks, it is not feasible to assign global identifiers to each node due to the sheer number of nodes deployed. Such lack of global identification along

with random deployment of sensor nodes makes it hard to select a specific set of sensor nodes to be queried. Therefore, data is usually transmitted from every sensor node within the deployment region with significant redundancy. Since this is very inefficient in terms of energy consumption, routing protocols that will be able to select a set of sensor nodes and utilize data aggregation during the relaying of data have been considered [57]. When data are measured or arrive from a neighbor, the sensor needs to decide whether or not they are important enough to forward them. The data may also be combined with other received data, in order to minimize the number of bits to forward. Such data aggregation (also referred to as data fusion) from multiple sensors is important, because of severe energy and bandwidth limitations as well as for numerous other reasons, including reliability [57]. This consideration has led to data-centric routing, which is different from traditional address-based routing where routes are created between addressable nodes managed in the network layer of the communication stack.

Data-centric routing protocols are quite energy efficient since the query is performed only when it is needed and global topology needn't be maintained. While there are also disadvantages, such as the naming depends on different applications and must be done firstly, and the process of data querying and matching introduces extra communication load and may result in broadcast storm [59].

In data-centric routing, the sink sends queries to certain regions and waits for data from the sensors located in the selected regions. Since data is being requested through queries, attribute based naming is necessary to specify the properties of data. SPIN [20] is the first data-centric protocol, which considers data negotiation between nodes in order to eliminate redundant data and save energy. Later, Directed Diffusion [21] has been developed and has become a

breakthrough in data-centric routing. Then, many other protocols have been proposed either based on Directed Diffusion [22][23][24] or following a similar concept [25][26][27][28].

Flooding and gossiping [29] are two classical mechanisms to relay data in sensor networks without the need for any routing algorithms and topology maintenance. In flooding, each sensor receiving a data packet broadcasts it to all of its neighbors. This process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. On the other hand, gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbor, which picks another random neighbor to forward the packet to and so on.

Sensor Protocols for Information via Negotiation (SPIN) [20] is among the early work to pursue a data-centric routing mechanism. The idea behind SPIN is to name the data using high level descriptors or meta-data. Before transmission, meta-data are exchanged among sensors via a data advertisement mechanism, which is the key feature of SPIN. Each node, upon receiving new data, advertises it to its neighbors and interested neighbors (i.e. those who do not have the data) and retrieves the data by sending a request message. SPIN's meta-data negotiation solves the classic problems of flooding by using techniques such as redundant information passing, overlapping of sensing areas and resource blindness thus, achieving a lot of energy efficiency.

Directed Diffusion [21,30] is an important milestone in the data-centric routing research of sensor networks. The idea aims at diffusing data through sensor nodes by using a naming scheme for the data. The main reason behind using such a scheme is to get rid of unnecessary operations of network layer routing in order to save energy. Direct Diffusion suggests the use of

attribute-value pairs for the data and queries the sensors in an on demand basis by using those pairs.

Energy-Aware Routing

Shah et al. [27] proposed to use a set of sub-optimal paths alternatively to increase the lifetime of the network. These paths are chosen by means of a probability function, which depends on the energy consumption of each path. Network survivability is the main metric that the approach is concerned with. The approach argues that using the minimum energy path all the time will deplete the energy of nodes on that path. Instead, one of the multiple paths is used with a certain probability so that the whole network lifetime increases.

Rumor routing [22] is another variation of Directed Diffusion and is mainly intended for contexts in which geographic routing criteria are not applicable. There, generally Directed Diffusion floods the queries to the entire network when there are no geographic criteria to diffuse tasks. However, in some cases there is only a little amount of data requested from the nodes and thus the use of flooding is unnecessary. An alternative approach is to flood the network if the number of events is small and the number of the queries is large. Rumor routing is between event flooding and query flooding. The idea is to route the queries to the nodes that have observed a particular event rather than flooding the entire network to retrieve information about the occurring events.

Gradient-Based Routing

Schurgers et al. [23] have proposed a slightly changed version of Directed Diffusion, called Gradient-Based Routing (GBR). The idea is to keep the number of hops when the interest is diffused through the network. Hence, each node can discover the minimum number of hops to

the sink, which is called the height of the node. The difference between a node's height and that of its neighbor is the gradient for that link. A packet is forwarded on a link with the largest gradient.

2.3.2 Hierarchical Protocols

Similar to other communication networks, scalability is one of the major design attributes of sensor networks. A single-tier network can cause the gateway to overload with increase in sensor density. Such overload might cause latency in communication, inadequate tracking of events, queue overflow and message loss. In addition, the single-gateway architecture is not scalable for a large set of sensors covering a wide area of interest since the sensors are typically not capable of long-haul communication. To allow the system to cope with additional load and to be able to cover a large area of interest without degrading the service, network clustering has been pursued in some routing approaches [59].

Dynamic clustering in Hierarchical routing protocols can strengthen the connectivity and prolong lifetime of the network, but its main disadvantage is using single hop communication, not suitable in large area applications [59].

The main aim of hierarchical routing is to efficiently maintain the energy consumption of sensor nodes by involving them in multi-hop communication within a particular cluster and by performing data aggregation and fusion in order to decrease the number of transmitted messages to the sink. Cluster formation is typically based on the energy reserve of sensors and sensor's proximity to the cluster head [31][32]. LEACH [8,9] is one of the first hierarchical routing approaches for sensors networks. The idea proposed in LEACH has been an inspiration for many

hierarchical routing protocols [25][34][35][36], although some protocols have been independently developed [37][38]. These protocols are discussed in this section.

LEACH

Low-Energy Adaptive Clustering Hierarchy (LEACH) [8,9] is one of the most popular hierarchical routing algorithms for sensor networks. The idea is to form clusters of sensor nodes based on the received signal strength and use local cluster heads as routers to the sink. This will save energy since the transmissions will only be done by the cluster heads rather than all sensor nodes. Optimal number of cluster heads is estimated to be 5% of the total number of nodes [8,9].

PEGASIS & Hierarchical-PEGASIS

Rather than forming multiple clusters, Power-Efficient GAthering in Sensor Information Systems (PEGASIS) [34] forms chains from sensor nodes so that each node transmits and receives from a neighbor and only one node is selected from that chain to transmit to the base station (sink). Gathered data moves from node to node, is aggregated and eventually is sent to the base station. The chain construction is performed in a greedy way. PEGASIS eliminates the overhead caused by dynamic cluster formation in LEACH and decreases the number of transmissions and reception by using data aggregation. However, PEGASIS introduces excessive delay for distant nodes on the chain. In addition, the only one node that is selected from the chain to transmit data to the base station can become a bottleneck.

TEEN and APTEEN

Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [25] is a hierarchical protocol designed to be responsive to sudden changes in the sensed attributes such as

temperature. Responsiveness is important for time-critical applications, in which the network operates in a reactive mode. TEEN pursues a hierarchical approach along with the use of a data-centric mechanism. The sensor network architecture is based on a hierarchical grouping where closer nodes form clusters. This cluster forming process goes on the second level until the base station (sink) is reached. The Adaptive Threshold sensitive Energy Efficient sensor Network protocol (APTEEN) [36] is an extension to TEEN and aims at both capturing periodic data collections and reacting to time-critical events.

Energy-aware Routing for Cluster-based Sensor Networks

Younis et al. [38] have proposed a different hierarchical routing algorithm based on a three-tier architecture. Sensors are grouped into clusters prior to network operation. The algorithm employs cluster heads, namely gateways, which are less energy constrained than sensors. It is assumed the location of sensor nodes is known. Gateways maintain the state of the sensors and sets up multi-hop routes for collecting sensors' data. A TDMA based MAC is used for nodes to send data to the gateway. The gateway informs each node about slots in which it should listen to other nodes. It also informs each node when to transmit and which slot to use for transmission. The command node (sink) communicates only with the gateways.

Self-organizing Protocol

Subramanian et al. [37] not only describe a self-organizing protocol but develop a taxonomy of sensor applications as well. The taxonomy is based on the network configuration of the sensor nodes. Network configuration, in this scenario, refers to the physical placement of the various sensors and the connectivity of these nodes to nodes in the infrastructure. The network

configuration determines the amount of routing intelligence that needs to be put into sensor nodes.

Based on such taxonomy, they have proposed architectural and infrastructural components necessary for building sensor applications. The architecture supports heterogeneous sensors that can be mobile or stationary. Some sensors probe the environment and forward the data to a designated set of nodes that act as routers. Router nodes are stationary and form the backbone for communication. Collected data are forwarded through the routers to more powerful sink nodes. Each sensing node should be reachable to a router node in order to be part of the network.

2.3.3 Location-based Protocols

Most of the routing protocols for sensor networks require location information of sensor nodes. In most cases location information is needed in order to calculate the distance between two particular nodes so that energy consumption can be estimated. Since, there is no addressing scheme for sensor networks like IP-addresses and they are spatially deployed on a region, location information is used to route data in an energy efficient way. For instance, if the region to be sensed is known, using the location of sensors, the query can be diffused only to that particular region which will eliminate a significant number of transmissions. Some of the protocols discussed here are designed primarily for mobile ad hoc networks and consider the mobility of nodes [39][40][41]. However, they are also applicable to sensor networks where there is less or no mobility.

It is worth noting that there are other location-based protocols designed for wireless ad hoc networks, such as Cartesian and trajectory-based routing [42][43]. However, many of these

protocols are not applicable to sensor networks since they are not energy aware. In order to stay within the theme of our survey, we limit the scope of coverage to energy-aware location based protocols.

Location based routing protocols are energy efficient when sensor nodes are deployed densely. That is, when the density of sensor nodes is small, these protocols may not keep the connectivity efficiently [59].

MECN and SMECN

Minimum Energy Communication Network (MECN) [40] sets up and maintains a minimum energy network for wireless networks by utilizing low power GPS. The small minimum energy communication network (SMECN) [41] is an extension to MECN. In MECN, it is assumed that every node can transmit to every other node, which is not possible all the time. In SMECN, possible obstacles between pair of nodes are considered. However, the network is still assumed to be fully connected as in the case of MECN. The subnetwork constructed by SMECN for minimum energy relaying is provably smaller than the one constructed in MECN if broadcasts are able to reach all nodes in a circular region around the broadcaster. As a result, the number of hops for transmission will decrease. Simulation results show that SMECN uses less energy than MECN and maintenance cost of the links is reduced. However, constructing a sub-network with smaller number of edges introduces more overhead in the algorithm.

GAF

Geographic Adaptive Fidelity (GAF) [39] is an energy-aware location-based routing algorithm designed primarily for mobile ad hoc networks, but may be applicable to sensor networks as well. GAF conserves energy by turning off unnecessary nodes in the network

without affecting the level of routing fidelity. It forms a virtual grid over the covered area. Each node uses its GPS-indicated location to associate itself with a point in the virtual grid. Nodes associated with the same point on the grid are considered equivalent in terms of the cost of packet routing. Such equivalence is exploited in keeping some nodes located in a particular grid area in sleeping state in order to save energy. (At most one node in a particular grid area is on). A node in the sleeping state wakes up after an application-dependent sleep time. Thus, GAF can substantially increase the network lifetime as the number of nodes increases.

GEAR

Yu et al. [44] have suggested the use of geographic information while disseminating queries to appropriate regions, since data queries often include geographic attributes. The protocol, namely Geographic and Energy Aware Routing (GEAR), uses an energy aware and geographically informed neighbor selection heuristics to route packets towards the target region. The idea behind Direct Diffusion is to restrict the number of interests, i.e. the specific datasets that the applications are interested in by only considering a certain region, rather than sending the interests to the whole network. GEAR compliments Directed Diffusion in this way and thus conserves more energy.

In GEAR, each node keeps an estimated cost and a learning cost for reaching the destination through its neighbors. The estimated cost is a combination of residual energy in the node and distance to destination. The learned cost is a refinement of the estimated cost that accounts for routing around holes in the network. A hole occurs when a node does not have any closer neighbor to the target region than itself. If there are no holes, the estimated cost is equal to the learned cost. The learned cost is propagated one hop back every time a packet reaches the

destination so that route setup for the next packet will be adjusted. There are two phases in the algorithm:

1. Forwarding packets towards the target region: Upon receiving a packet, a node checks its neighbors to see if there is one neighbor closer to the target region than itself. If there is more than one, the nearest neighbor to the target region is selected as the next hop. There is a hole if the neighbors are all further than the node itself. In this case, one of the neighbors is picked to forward the packet based on the learning cost function. This choice can then be updated based on the convergence of the learned cost during the delivery of packets.

2. Forwarding the packets within the region: If the packet has reached the region, it can be diffused in that region by either recursive geographic forwarding or restricted flooding. Restricted flooding is good when the sensors are not densely deployed. In high-density networks, recursive geographic flooding is more energy efficient than restricted flooding [44]. In that case, the region is divided into several sub regions and copies of the packet are created. Each copy is sent to one sub regions. This splitting and forwarding process continues until the regions with only one node are left.

2.3.4 Network Flow and QoS-aware Protocols

Some routing protocols pursue other approaches such as network flow and QoS. In some approaches, route setup is modeled and solved as a network flow problem. QoS-aware protocols consider end-to-end delay requirements while setting up the paths in the sensor network. Sample of these protocols are discussed in this section.

Maximum Lifetime Energy Routing

Chang et al. [45] present an interesting solution to the problem of routing in sensor networks based on a network flow approach. The main objective of the approach is to maximize the network lifetime by carefully defining link cost as a function of the remaining node energy and the required transmission energy using that link. The solution to this problem maximizes the feasible time the network lasts. In order to find out the best link metric for the stated maximization problem, two maximum residual energy path algorithms are presented and simulated. The two algorithms differ in their definition of link costs and the incorporation of nodes' residual energy. Simulation results show that the proposed maximum residual energy path approach has better average lifetime than the minimum transmitted energy approach for both link cost models [45].

Minimum Cost Forwarding

The minimum cost forwarding protocol [24] aims at finding the minimum cost path in a large sensor network, which will also be simple and scalable. The cost function for the protocol captures the effect of delay, throughput and energy consumption from any node to the sink. The protocol is not really flow-based. However, since data flows over the minimum cost path and the resources on the nodes are updated after each flow, we have included it in this section. After each data flow is done, a setup phase will be executed to set the cost value in all nodes.

SPEED

A QoS routing protocol for sensor networks that provides soft real-time end-to-end guarantees is described in [48]. The protocol requires each node to maintain information about its neighbors and uses geographic forwarding to find the paths. In addition, SPEED strive to ensure a certain delivery speed for each packet in the network so that each application can

estimate the end-to-end delay for the packets by dividing the distance to the sink by the speed of the packet before making the admission decision. Moreover, SPEED can provide congestion avoidance when the network is congested.

2.3.5 Discussion

Protocols, which name the data and query the nodes based on some attributes of the data are categorized as data-centric. Many of the researchers follow this paradigm in order to avoid the overhead of forming clusters, the use of specialized nodes etc. However, the naming schemes such as attribute-value pairs might not be sufficient for complex queries and they are usually dependent on the application. Efficient standard naming schemes are one of the most interesting future research direction related to this category.

On the other hand, cluster-based routing protocols group sensor nodes to efficiently relay the sensed data to the sink. The cluster heads are sometimes chosen as specialized nodes that are less energy-constrained. A cluster-head performs aggregation of data and sends it to the sink on behalf of the nodes within its cluster. The most interesting research issue with cluster based routing protocols is how to form the clusters so that the energy consumption and contemporary communication metrics are optimized. The factors affecting cluster formation and cluster-head communication are open issues for future research. Moreover, the process of data aggregation and fusion among clusters is also an interesting problem to explore.

Protocols that utilize the location information and topological deployment of sensor nodes are classified as location-based. The number of energy-aware location-based approaches found in the literature is rather small. The problem of intelligent utilization of the location information in order to aid energy efficient routing is the main research issue. Spatial queries and

databases using distributed sensor nodes and interacting with the location-based routing protocol are open issues for further research.

Although the performance of these protocols is promising in terms of energy efficiency, further research would be needed to address issues such as Quality of Service (QoS) posed by video and imaging sensors and real-time applications. Currently, there is little research that looks at handling QoS requirements in an energy constrained environment such as sensor networks.

Chapter 3

Implementation of LEACH Protocol on the TinyOS Platform

3.1 LEACH Protocol Architecture

The LEACH (Low Energy Adaptive Clustering Hierarchy) protocol [8,9] developed by Wendi Heinzelman is designed for sensor networks to support end-users remotely monitor the environment. In such a situation, the data from the individual nodes must be sent to a central base station, often located far from the sensor network, through which end-users can access the data. There are several desirable properties for protocols on such networks: Use 100's - 1000's of nodes; Maximize system lifetime; Maximize network coverage; Use identical, battery-operated nodes.

Conventional network protocols, such as direct transmission, minimum transmission energy, multi-hop routing, and clustering all have drawbacks that prevent them from achieving the desirable properties as discussed in Section 2.3. LEACH includes distributed cluster formation, local processing to reduce global communication, and randomized rotation of the cluster-heads. Together, these features allow LEACH to achieve the desired properties. Initial

software simulations conducted by Wendi Heinzelman show that LEACH is an energy-efficient protocol that extends system lifetime [8,9]. Much of the material in this section was taken from Dr. Wendi Heinzelman's papers [8,9].

LEACH protocol employs a clustering stage before transmitting data wherein a sensor becomes a cluster head and will transmit data from any sensor belonging to the cluster head to the base station. This differs from the standard method where each sensor transmits to the base station [7], which is always further away than the cluster head and thus requires more transmission power. Thus the LEACH protocol helps to maximize the lifetime of the system by minimizing the energy used to transmit data to the base station.

LEACH is a cluster based approach with random periodic cluster head selection so as to distribute load across all nodes. The nodes within the cluster communicate with the cluster head via a TDMA MAC (a fixed schedule for communicating with non-cluster nodes). The cluster membership is adaptive as the nodes select its cluster head in terms of the received signal strength from all cluster nodes. After receiving data from the nodes within its cluster, the cluster head aggregates the data. The cluster head sends the aggregated data directly to the sink or user.

Based on the network communication model described above, the following can be said about LEACH protocol: The sources and users are stationary and events monitored are continuous; the data dissemination mechanism is broadcasting.

LEACH protocol has several energy efficient features such as: Optimization of the energy used by shutting down nodes' radios, load balancing, and only two hops from any node to the sink or the user. LEACH's distributed hierarchical approach makes it scalable.

However, LEACH has also some shortcomings:

- Failure of the cluster head is a problem,
- Cluster head selection is a difficult problem to optimize,
- All nodes must be capable of long range communication to the base station,
- Time synchronization between nodes.

The operation of LEACH is divided into rounds. Each round begins with a set-up phase, where the clusters are organized, followed by a steady-state phase, where data are transferred from the nodes to the cluster head and on to the base station.

LEACH forms clusters by using a distributed algorithm where nodes make autonomous decisions without any centralized control. Sensor nodes elect themselves to be a cluster head with a certain probability (based on the amount of battery capacity they have left). So the cluster-head position is randomly rotated among the sensors. Each sensor node chooses a random number between 0 and 1. If this random number is less than the threshold T(n), the sensor node is selected as a cluster-head [8,9]. T(n) is given by

$$T(n) = \begin{cases} \dfrac{P}{1 - P[r \bmod (1/P)]} & \text{if } n \in G, \\ 0 & \text{otherwise,} \end{cases} \qquad (3.1)$$

where P is the desired likelihood for a node to become a cluster head (P = k/N, k is the expected number of cluster head nodes for this round, N is the total number of nodes alive in the network); r is the current round; G is the set of nodes that have not been selected as a cluster-head in the last 1/P rounds.

Therefore, only nodes that have not already been cluster heads recently, and which presumably have more energy available than nodes that have recently performed this energy-intensive function, may become cluster heads in the next round.

Each non-cluster-head node determines which cluster to join. It determines its cluster for the current round by choosing the cluster head that requires the minimum transmit power, based on the received signal strength of the advertisement from each cluster head.

After each node has decided to which cluster it belongs to, it informs the cluster head node that it will be a member of the cluster. Each node transmits a join-request message (Join-REQ) back to the selected cluster head using a non-persistent CSMA MAC protocol (when a node has data to send it listens to the channel to try to determine if any other node is currently transmitting). After sensing a busy channel, the node enters a back-off state by setting a randomized timer. When the timer expires, the node again senses the channel. If it is busy, the node resets the timer and repeats the back-off procedure. If the channel is free, the node transmits the packet [8].

The cluster heads in LEACH act as local control centers to coordinate the data transmissions within their cluster. The cluster head node sets up a TDMA schedule and transmits this schedule to the nodes within its cluster. The TDMA schedule transmitted consists of a list of member node ids and their allocated transmission time slot information. Every node in the cluster is assigned a transmission slot of equal length. The whole operation is broken into frames, where nodes send their data to the cluster head during their allocated transmission slot at most once per frame, and go to sleep until it is time to transmit data. This ensures that there are no collisions among data messages and also allows the radio components of each non-cluster

head node to be turned off at all times except during their transmit time, thus reducing the energy consumed by the individual sensor nodes. Once the TDMA schedule is known by all nodes in the cluster, the set-up phase is complete and the steady-state operation (data transmission) can begin. A flowchart of this distributed cluster formation algorithm is shown in Fig. 3.1.

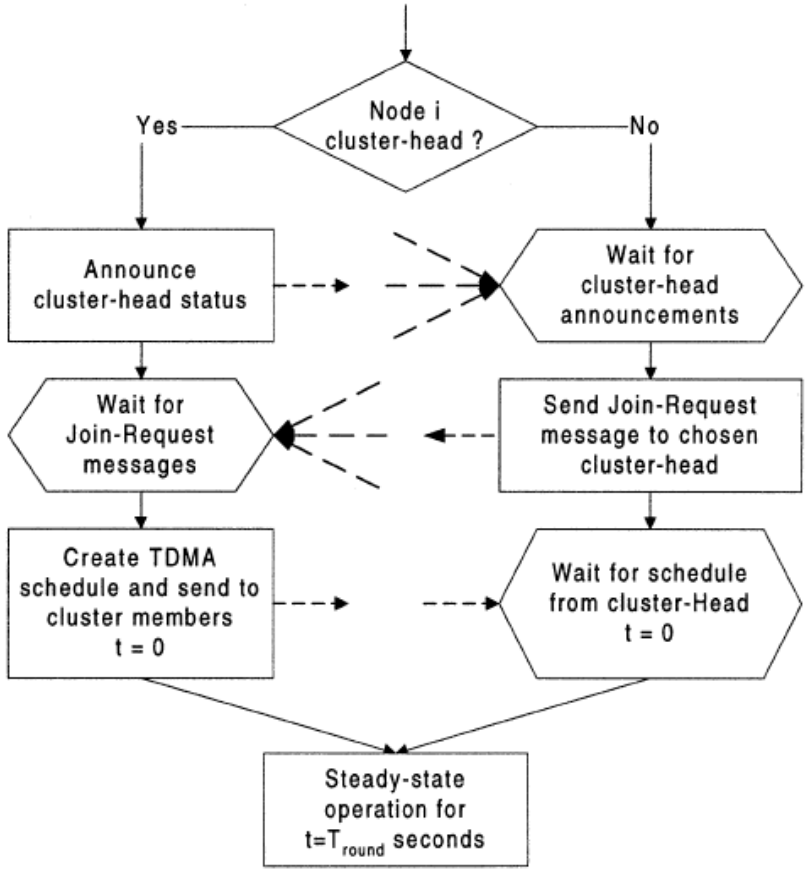


Figure 3.1: Flowchart of the Distributed Cluster Formation Algorithm for LEACH

The time line for one round of LEACH is illustrated in [8]. Data transmissions are explicitly scheduled to avoid collisions and increase the amount of time each non-cluster head node can remain in the sleep state.

To reduce energy dissipation, each non-cluster head node uses power control to set the amount of transmission power based on the received strength of the cluster head advertisement. Furthermore, the radio of each non-cluster head node is turned off until its allocated transmission time. The cluster head must be awake to receive all the data from the nodes in the cluster. Once the cluster head receives all the data, it performs data aggregation to enhance the common signal and reduce the uncorrelated noise among the signals. Individual signals are combined into a single representative signal. The resultant data are sent from the cluster head to the BS.

LEACH-C, MTE, Static-Clusters Protocol

LEACH-centralized (LEACH-C) is a protocol that uses a centralized clustering algorithm and the steady-state portion of LEACH.

While there are advantages to using LEACH's distributed cluster formation algorithm, the LEACH protocol offers no guarantee about the placement and/or number of cluster head nodes. Since the clusters are adaptive, obtaining a poor clustering set-up during a given round will not greatly affect overall performance. However, using a central control algorithm to form the clusters may produce better clusters by dispersing the cluster head nodes throughout the network. This is the basis for LEACH-C [8,9].

During the set-up phase of LEACH-C, each node sends information about its current location (possibly determined using a GPS receiver) and energy level to the base station (BS). In addition to determining good clusters, the BS needs to ensure that the energy load is evenly distributed across all the nodes. To do this, the BS computes the average node energy, and any node with energy level below this average is excluded from the cluster head selection for the current round. Using the remaining nodes as possible cluster heads, the BS identifies clusters

using the simulated annealing algorithm [49] to solve the NP-hard problem of finding optimal clusters [52]. This algorithm attempts to minimize the amount of energy for the non-cluster head nodes to transmit their data to the cluster head, by minimizing the total sum of squared distances between all the non-cluster head nodes and the closest cluster head. The steady-state phase of LEACH-C is identical to that of LEACH.

For MTE (minimum transmission energy) routing, each node runs a start-up routine to determine its next-hop neighbor, defined to be the closest node that is in the direction of the BS. Data packets are passed along via next-hop neighbors until they reach the BS. As there is no central control in MTE routing, it is difficult to set up fixed MAC protocols (e.g., TDMA), so each node uses CSMA to listen to the channel before transmitting data. If the channel is busy, the node backs off; otherwise, the node transmits its data to the next-hop node. When a node runs out of energy, the routes that contain that node are recomputed to ensure connectivity to the BS.

For static clustering, nodes are organized into clusters initially by the BS using the same method as in LEACH-C to ensure that good clusters are formed. These clusters and cluster heads remain fixed throughout the lifetime of the network. As in LEACH and LEACH-C, nodes transmit their data to the cluster head node during each frame of data transfer, and the cluster head aggregates the data and sends the resultant data to the BS. When the cluster head node's energy is depleted, the nodes in the cluster lose communication ability with the BS and are essentially "dead."

3.2 Analysis and Simulation Model

For even moderately-sized networks with tens of nodes, it is extremely difficult to analytically model the interactions between all the nodes. Therefore, researchers used simulation to evaluate LEACH and compare it to other protocols. MIT's researchers compared LEACH to LEACH-C, MTE routing, and static clustering in terms of system lifetime, energy dissipation, and amount of data transfer. They used the network simulator ns2 [13] and developed the models described below:

Mobile Node

The wireless model in the ns2 release was originally ported as CMU's Monarch group's mobility extension to ns2. It includes the internals of a mobile node, routing mechanisms and network components that are used to construct the network stack for a mobile node. The components include Channel, Network interface, Radio propagation model, MAC protocols, Interface Queue, Link layer and Address resolution protocol model (ARP). Figure 3.2 shows the implementation of a mobile node under CMU Monarch's Wireless Extensions [54].

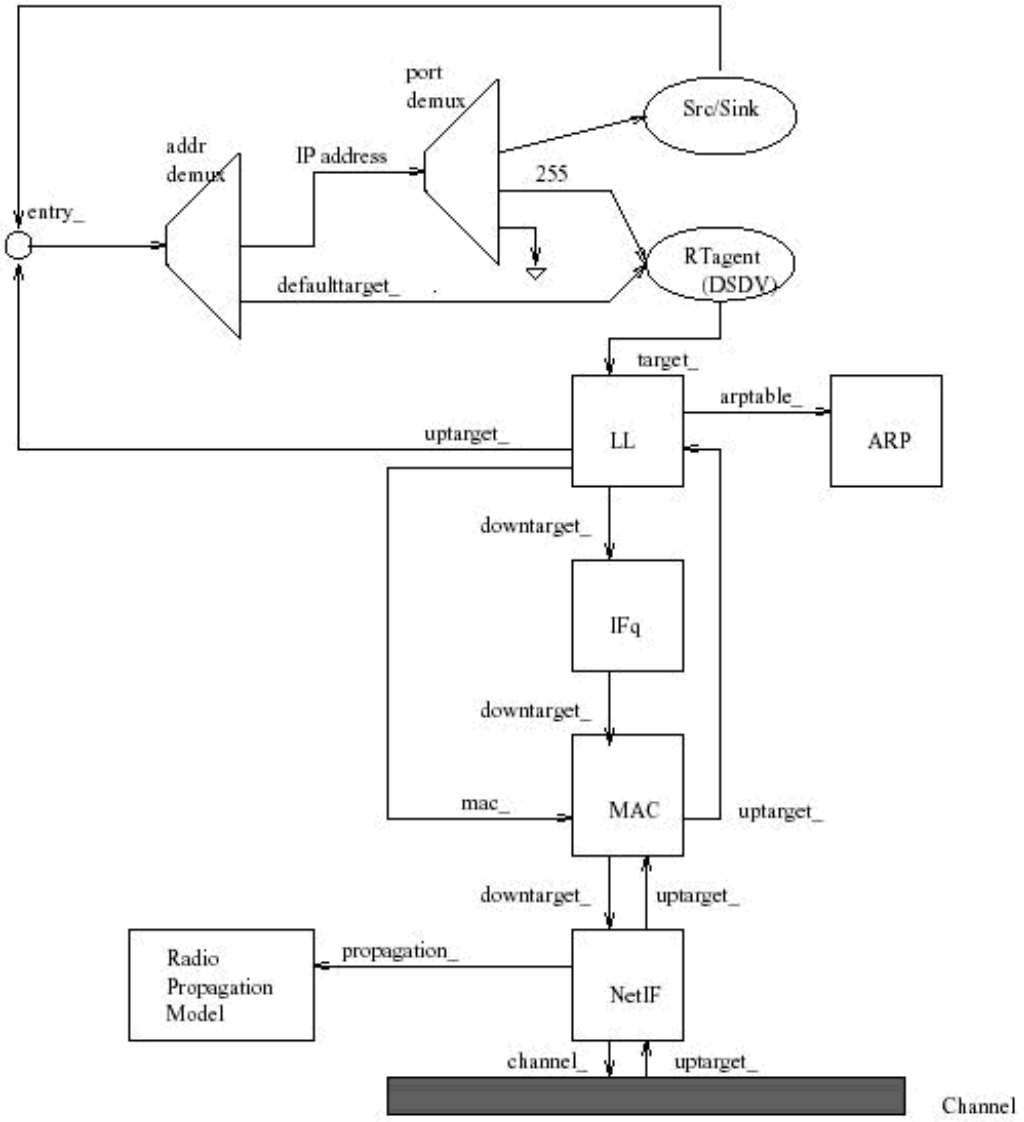


Figure 3.2: Schematic of Mobile Node under CMU Monarch's Wireless Extensions to ns

Resource-Adaptive Node

A new type of node, the Resource-Adaptive node, was added to ns2 in MIT's implementation. The new features of the Resource-Adaptive Node include the Resources and the Resource Manager. The resource manager provides a common interface between the

application and individual resources. The resources can be anything that needs to be monitored, such as energy and node neighbors. The application updates the status of the node's resources through the resource manager.

Radio Energy Dissipation Model

The LEACH radio energy dissipation model is shown in Figure 3.3. In this model, the transmitter dissipates the energy to run the radio electronics and the power amplifier while the receiver dissipates the energy to run the radio electronics.

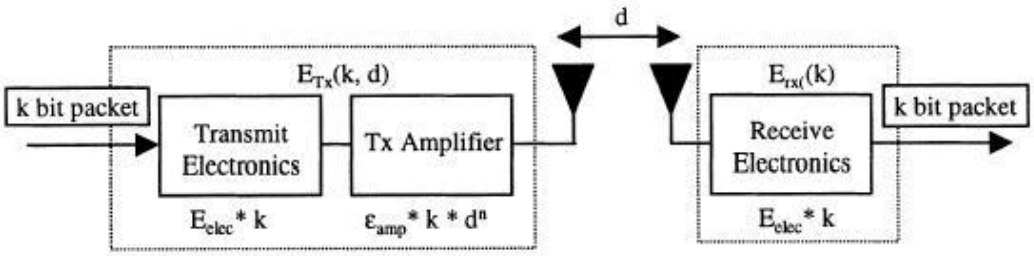


Figure 3.3: LEACH Radio Energy Dissipation Model [8]

Terms in Fig. 3.3	Stands for
E_{elec}	the electronics energy, the energy dissipated for the radio to transmit one bit of data
ε_{amp}	the transmit amplifier energy factor
d	the distance between the transmitter and receiver
E_{Tx}	the energy expensed for transmitting data
E_{rx}	the energy expensed for receiving data

MAC protocol

The protocol in the implementation is a combination of carrier-sense multiple access (CSMA), and time-division multiple access (TDMA). TDMA is implemented with the

application by only having the application send data to the agent during the specified TDMA time-slot. CSMA is implemented in the MAC class.

Since LEACH is an application-specific protocol architecture, it is implemented as a subclass of ns2's application class. The LEACH protocol is implemented as described in the paper which first presented LEACH protocol [8]. In addition, the base station is a special node that has no energy constraints and is the node to which all data are eventually sent. A base station application is implemented to perform the base station's functions.

3.3 ns2 Simulation of LEACH

We implemented the LEACH protocol in ns 2.33 based on MIT's ns2 extension. When MIT's researchers presented the LEACH protocol, they implemented LEACH in the ns 2.1b5 release dated year 2000. Since then, ns2 has evolved and several new versions were released. During our simulation work, we referred to the ideas of the MIT's sensor network extensions, modified and merged the MIT's LEACH protocol simulation codes in the ns 2.33 release, the latest release available at the time of this research. After coding and debugging, we validated our simulation model. It produced simulation results which agree with the original MIT's simulation results.

We ported all the MIT's LEACH modules in the ns 2.33 version. All features of LEACH are implemented in the ns 2.33 simulation. It was a challenging task as some of the ns 2.33 modules and objects were quite different from those in the ns2 version of MIT's extension. More than 30% of the code has been modified and added. Additional code was added to support

the new features of the Resource-Adaptive node, which are introduced by the LEACH protocol and implemented in the MIT's extension.

The simulation was executed on a Linux platform. The simulation results matched well the MIT's simulation results.

For equivalent inputs, we compared our simulation results to that of MIT's simulation result figures. The comparisons are shown in Figure 3.4-3.7.

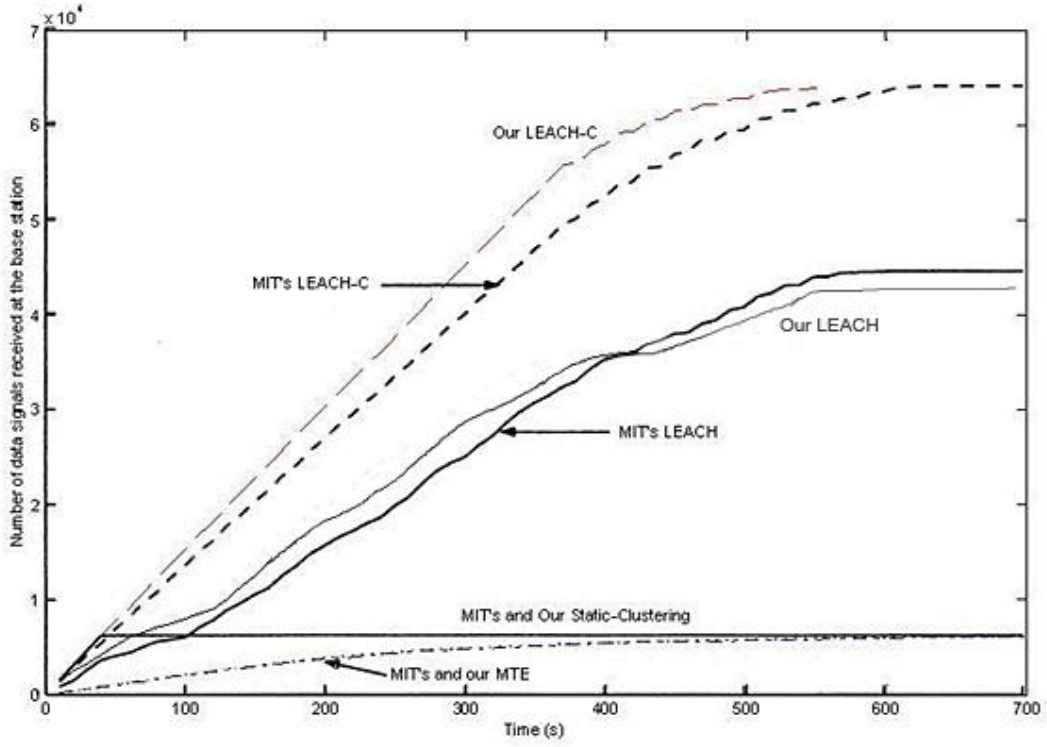


Figure 3.4: Comparison of Total Amount of Data Received at the BS over Time

Fig. 3.4 shows the total number of data signals received at the BS over time for a given amount of energy. It shows that LEACH and LEACH-C send much more data to the base station (BS) in the simulation time than MTE routing and Static-Clustering. The reason MTE requires so

much time to send data from the nodes to the BS is that each message traverses several hops. In the other protocols, each message is transmitted over a single hop, to the cluster head, where data aggregation occurs. The aggregate signals are sent to the BS, greatly reducing the amount of data transmitted [8]. Our simulation results of LEACH and LEACH-C agree well with MIT's results. Our results of MTE and Static-Clustering are identical to those of MIT.

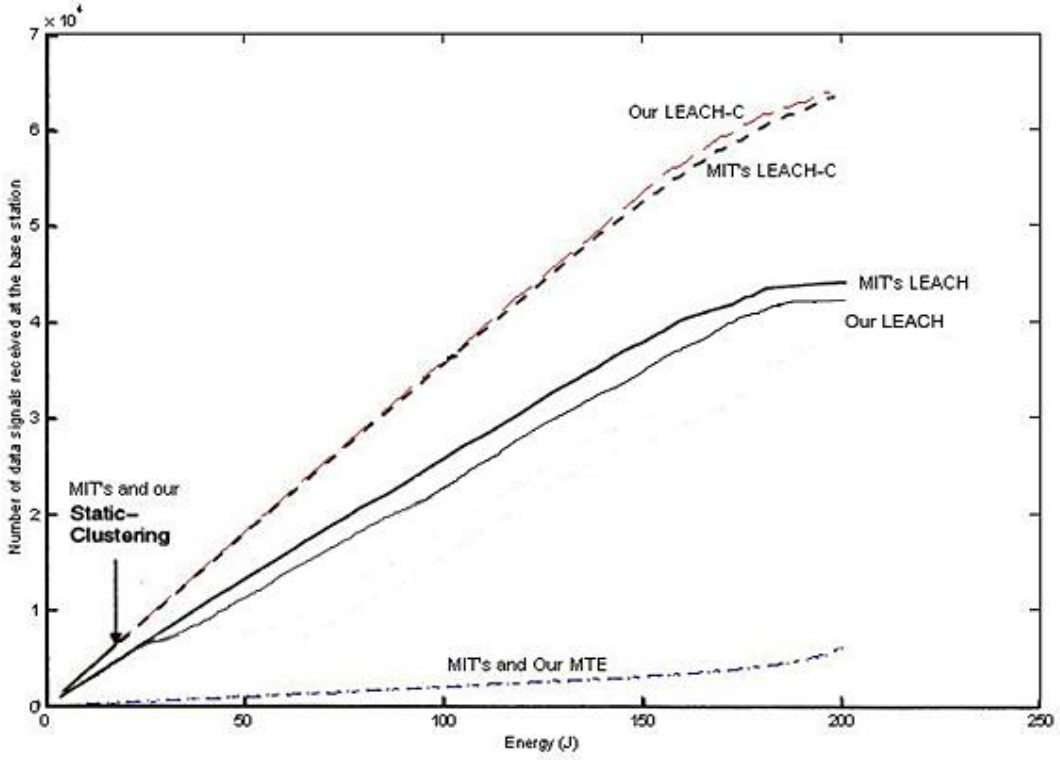


Figure 3.5: Comparison of Total Amount of Data Received at the BS per Amount of Energy Consumed by the Network

Fig. 3.5 shows the total data received at the BS for a given amount of energy. This graph shows that LEACH and LEACH-C deliver the most data per unit energy, achieving both energy and latency efficiency. A routing protocol such as MTE does not enable local computation to reduce the amount of data that needs to be transmitted to the BS [8]. Our simulation results of

LEACH protocol agrees with that of MIT. Our results of LEACH-C, MTE and Static-Clustering are almost identical to those of MIT.

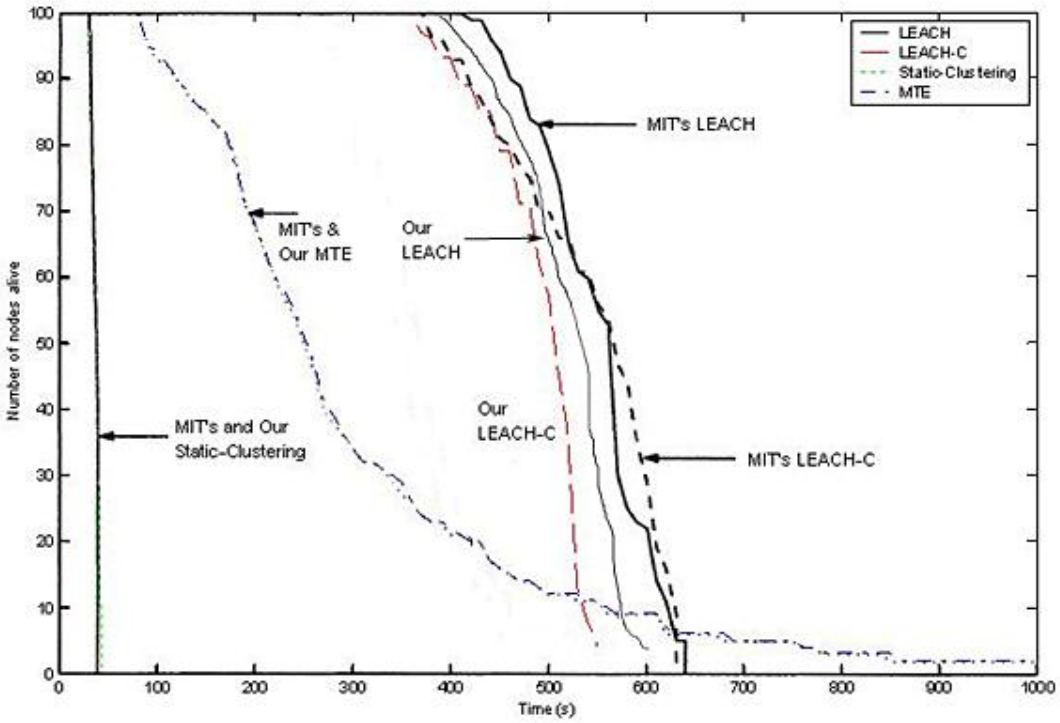


Figure 3.6: Comparison of Number of Nodes Alive over Time

Fig. 3.6 shows the total number of nodes that remain alive over the simulation time. Nodes remain alive for a long time in MTE because a much smaller amount of data has been transmitted to the BS. If we plot the total number of nodes that remain alive per amount of data received at the BS (Fig. 3.7), we see that nodes in LEACH can deliver about ten times more data than MTE for the same number of nodes that ran out of battery. There are two reasons why MTE requires more energy to send data to the BS (hence, causing more node deaths for the same amount of data delivery): collisions and lack of data aggregation. Figures 3.6 and 3.7 show why static clustering performs poorly: the cluster head nodes die quickly, ending the lifetime of all

nodes belonging to those clusters. Therefore, rotating the cluster head position enables LEACH to achieve a longer lifetime than static clustering [8]. In Figures 3.6 and 3.7, our results of LEACH and LEACH-C agree well with MIT's. Moreover, our results for MTE and Static-Clustering are almost identical to those of MIT.

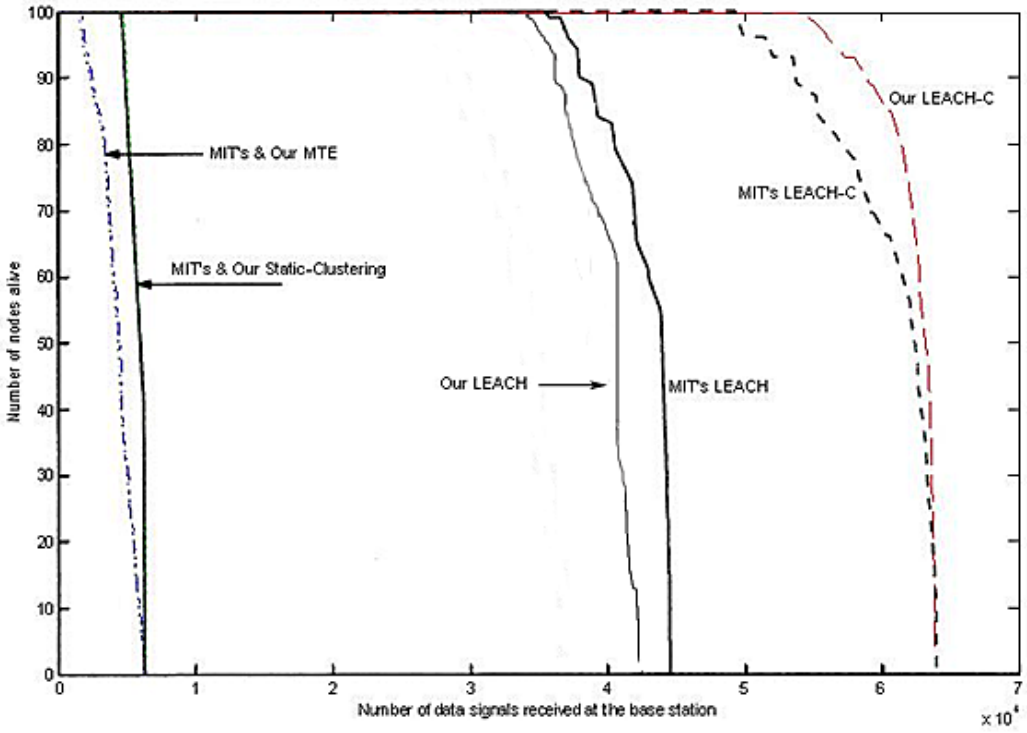


Figure 3.7: Comparison of Number of Nodes Alive per Amount of Data Received at BS

As shown by Figures 3.4-3.7 our simulation results agree well with MIT's results. This validates our simulator and simulation results. As the time of these writings, MIT's simulation results were the only available LEACH simulation results available for comparison.

Note that some of our results of LEACH and LEACH-C in Figures 3.4-3.7 don't match MIT's results exactly. This is because the LEACH protocol has inherent randomization, for

example, the cluster head selecting algorithm. And LEACH-C protocol uses the simulated annealing algorithm, which also randomizes the cluster-head selection process. Therefore, we cannot expect a perfect match with MIT's simulation results. For MTE and Static-Clustering, neither protocol involves randomization, so the simulation results agree perfectly with MIT's.

ns 2 Simulation of LEACH Incorporating Crossbow Hardware

Since Crossbow is one of the leading manufacturers of sensor network hardware, we modify our ns2 simulation model to account for Crossbow motes. To do so, we derived the Crossbow motes radio energy model and simulated LEACH with Crossbow sensor network hardware.

The data we used in the analysis were extracted from the Chipcon hardware manual, the manufacture of motes' transceiver. The data are listed in Table 3.1. The "Distance" column represents the distance between the motes, and the "Current" column represents the minimal transmitting current required for the motes to communicate with each other successfully.

Distance (ft)	Current (ma)	Distance (ft)	Current (ma)
2	8.6	260.96	10.8
21.92	8.8	280.88	11.1
41.84	9	300.8	13.8
61.76	9	320.72	14.5
81.68	9.1	340.64	14.5
101.6	9.3	360.56	15.1
121.52	9.3	380.48	15.8
141.44	9.5	400.4	16.8
161.36	9.7	420.32	17.2
181.28	9.9	440.24	18.5
201.20	10.1	460.16	19.2
221.12	10.4	480.08	21.3
241.04	10.6	500	25.4

Table 3.1: Distance Between Motes and the Minimal Transmitting Current Required

According to the electronic communication theory, in a wireless channel, the electromagnetic wave propagation can be modeled as falling off with distance by a power law function [8].

Based on the hardware data in Table 3.1 and the expected power law behavior, we derived the relationship between the distance and the transmitter energy consumption, and constructed a function that maps any distances into Crossbow motes data. In this way, we can get the current consumption required for any given distance. To do so, we used Eviews, a statistics software package, to analyze the data. Details about this data analysis and regression can be found in Appendix A.

As a summary of the analysis in Appendix A, the Crossbow motes radio energy model could be summarized as below:

When transmitting data,

when distance < $d_{crossover}$, Current = 8.874057 + 2.94E-05 * $Distance^2$ mA (3.2)

when distance > $d_{crossover}$, Current = 11.61814 + 1.94E-10 * $Distance^4$ mA (3.3)

where $d_{crossover}$ = 295 ft.

When receiving data, Current = 9 mA.

The results shown above in Equation 3.2 and Equation 3.3 agree well with the electronic communication theories [8]. The first part relates to $Distance^2$; that corresponds to the Friss free

space model. The second part relates to $Distance^4$; that corresponds to the two-ray ground propagation model. The communication theory indicates that if the distance between the transmitter and receiver is less than a certain cross-over distance ($d_{crossover}$), the Friss free space model should work, and if the distance is greater than $d_{crossover}$, the two-ray ground propagation model should work. As we can see from the results above, our analysis result agrees with the theory.

In our following simulation, we used a 100-node network where nodes are randomly distributed in a 100*100 meters grid of 100*100 points. The base station was located at grid point (50, 175). The bandwidth of the channel was set to 38,400 bps. Thus, to transmit an *L*-bit message for a distance *d*, the amount of energy that the radio consumes is:

When transmitting data, *Energy*

$= L* 8.874057 * 2.7 * 0.001 /38400 + L * 2.94E-05 * 2.7 * 0.001 /38400 * Distance^2*0.305^2$, *when distance < 90 meters.*

$= L * 11.61814 * 2.7 * 0.001 /38400 + L * 1.94E-10 * 2.7 * 0.001 /38400 * Distance^4*0.305^4$, *when distance > 90 meters.*

When receiving data, $Energy = L * 9 * 2.7 * 0.001 /38400$

We incorporate this Crossbow motes radio energy model as well as other Crossbow sensor hardware data parameters into the ns2 simulation. The simulation details and results are presented in Section 3.5.

3.4 Implementation of LEACH on the TinyOS Platform

TinyOS is an open source component-based operating system and platform targeting wireless sensor networks (WSNs). TinyOS is an embedded operating system written in the nesC programming language (nesC is also used in Crossbow motes) as a set of cooperating tasks and processes [10]. It is intended to be incorporated into Smartdust. TinyOS started as collaboration between the University of California, Berkeley in co-operation with Intel Research and Crossbow Technology, and has since grown to be an international consortium, the TinyOS Alliance.

We successfully implemented the LEACH protocol in nesC on the TinyOS platform. Our implementation exactly implemented the LEACH protocol architecture. The implementation works properly on both Crossbow MICA2 hardware and TOSSIM, the standard network emulator on the TinyOS platform. It was very challenging to implement LEACH protocol on TinyOS platform, as there were many difficult technical problems we had to solve, in both software and hardware aspects. To the best of our knowledge, no one has ever implemented LEACH on a real sensor network.

Figure 3.8 shows the System Flowchart of our LEACH Implementation on the TinyOS platform. The Data Transportation Module implements data traffic generation, data sending and receiving, and data aggregation which ran when the node is a cluster head. The Time Synchronization Module takes care of the synchronization of the nodes. All the nodes are synchronized. The cluster head sends a TDMA schedule to all its member nodes. The member nodes can only send data in the time slot allocated for this node.

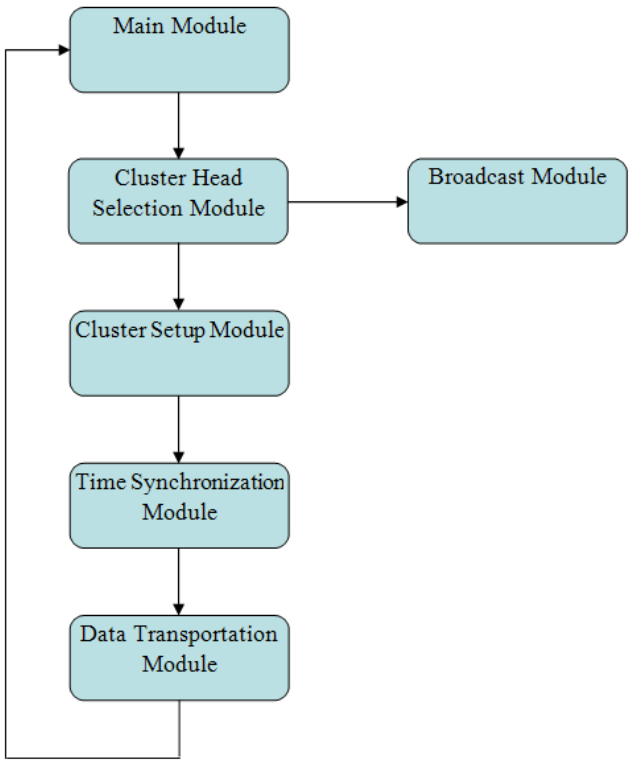


Figure 3.8: TinyOS LEACH Implementation System Flowchart

We ran multiple series of our LEACH TinyOS implementation to do experiments and analyze the LEACH protocol performance. The experiments below were executed on PowerTOSSIM with a network size of 100 sensor nodes, which is a typical network size for sensor networks research.

TOSSIM and PowerTOSSIM

TOSSIM is a scalable event-driven emulation environment for TinyOS applications. TOSSIM emulates entire TinyOS applications. It works by replacing components with emulation implementations. TOSSIM is a discrete event emulator. When it runs, it pulls events of the event

queue (sorted by time) and executes them. Depending on the level of emulation, emulation events can represent hardware interrupts or high-level system events (such as packet reception). Additionally, tasks are emulation events [67].

TOSSIM is a Mote emulator that implements the lowest layer of components in the TinyOS API, and can run many emulated Motes in parallel. TOSSIM emulates Mote applications, and requires that an identical code base run on every node.

PowerTOSSIM is an extension of TOSSIM. PowerTOSSIM provides low-level and accurate per-node power consumption information. It captures the detailed, low-level energy requirements of the CPU, radio, sensors, LEDs and other peripherals. In PowerTOSSIM, TinyOS components corresponding to specific hardware peripherals are instrumented to obtain a trace of each device's activity during the emulation run. PowerTOSSIM employs a novel code-transformation technique to estimate the number of CPU cycles executed by each node, eliminating the need for expensive instruction-level emulation of sensor nodes. PowerTOSSIM's approach to estimate the number of CPU cycles is to: (1) instrument the PowerTOSSIM binary to obtain an execution count for each basic block (run of instructions with no branches) executed by the emulated CPU; (2) map each basic block to its corresponding assembly instructions; (3) determine the number of CPU cycles for each basic block using simple instruction analysis; and (4) combine the emulation basic block execution counts with their corresponding cycle counts to obtain the total CPU cycle count for each emulated mote [12].

PowerTOSSIM provides accurate estimation of power consumption for a range of applications (including Beacon, CntToLeds, CntToLedsAndRfm, CntToRfm, Oscilloscope, OscilloscopeRF, Sense, SenseLightToLog, SenseTask, SenseToLeds, SenseToRfm, TinyDB and

Surge, etc.), and scales to support very large emulations [12]. PowerTOSSIM achieves an accuracy of within 0.45-13% of the true power consumption of nodes running an identical program [12].

PowerTOSSIM tracks the power state of each hardware component of the emulated motes by generating specific power state transition messages that are logged during the emulation run. This is accomplished by instrumenting the TOSSIM emulated hardware components with calls to a new component which tracks hardware power states for each mote and logs them to a file during the run [12].

PowerTOSSIM provides run-time configurable debugging output, allowing a user to examine the execution of an application from different perspectives without needing to recompile. TinyViz is a Java-based GUI that allows you to visualize and control the emulation as it runs, inspecting debug messages, packets, and so forth [67]. The emulation provides several mechanisms for interacting with the network; packet traffic can be monitored. In the TinyViz GUI, there is “Power Profiling” tab which displays the power consumption data of the nodes in the network in real time.

We executed our LEACH TinyOS implementation multiple times to analyze the performance of LEACH. Figure 3.9 and Figure 3.10 show a snapshot of the TinyViz GUI when running our TinyOS LEACH implementation on PowerTOSSIM.

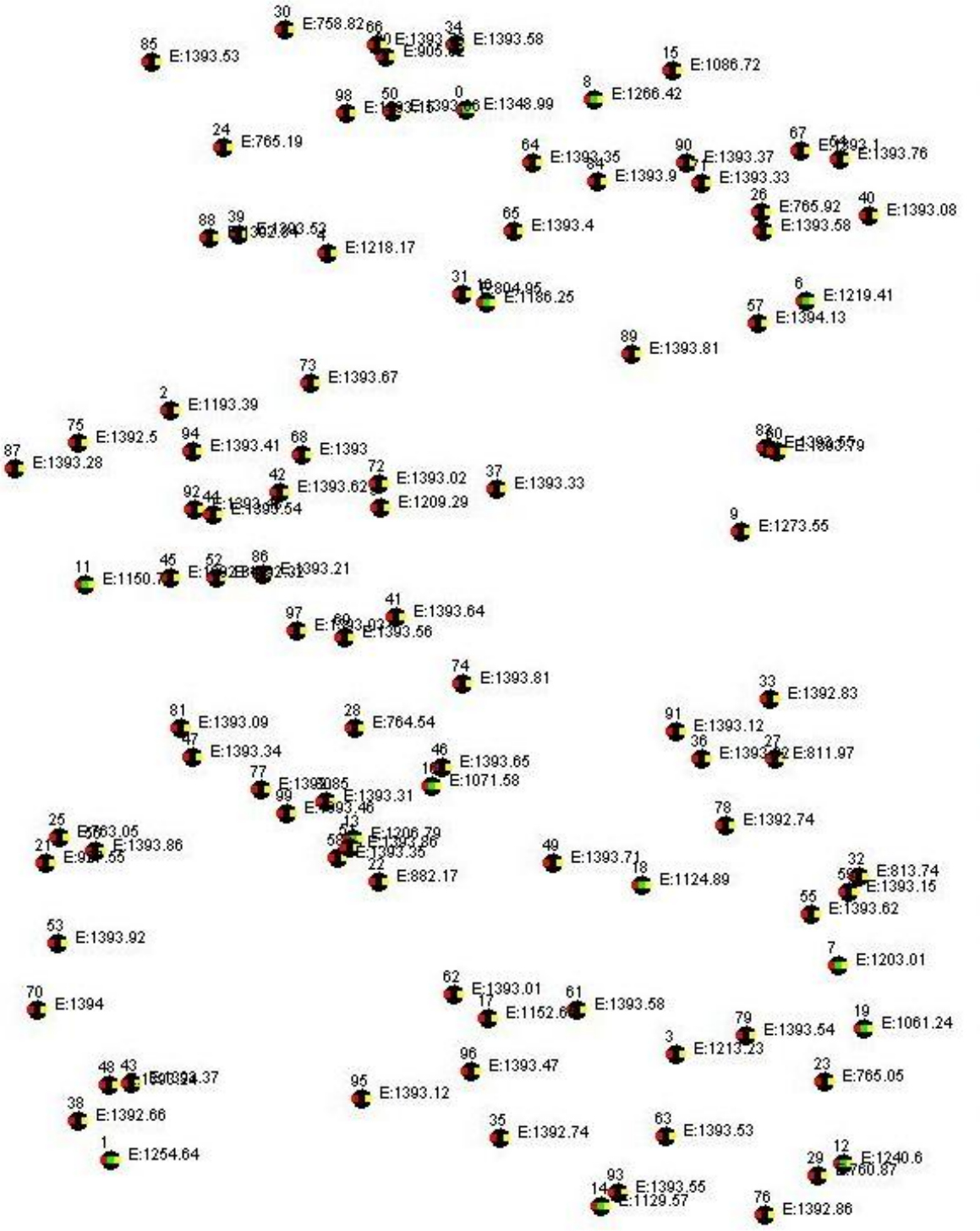


Figure 3.9: Snapshot of the TinyOS LEACH Implementation Running on PowerTOSSIM GUI Displaying the Motes Layout

Figure 3.10: Snapshot of the TinyOS LEACH Implementation
Running on PowerTOSSIM GUI Displaying the Output Debug Messages

In the following table we provide a definition for Figure 3.10 output debug messages:

Debug Window Message	Explanation
[yy] USR2:In total: Consume xx packets in this turn.	Means node yy sent xx packets in this round.
[yy] USR2:Till now: Consume xx packets.	Means node yy has sent a total of xx packets from time 0 till now to the base station.
[yy] USR2:Got xx data packets in this frame.	Means node yy received xx packets in this frame, where xx equals to the number of member nodes in the cluster. Node yy is a cluster head in this frame, with xx member nodes in its cluster.

Table 3.2: Definition for Figure 3.10 Output Debug Messages

Each 10 emulated seconds, we pause the PowerTOSSIM emulation, record and calculate the metrics, then resume the emulation.

By summing up the numbers of the data packets that were send to the base station by each node at every sample point, we get the number of data items received at the base station by that time. Time is the emulated time shown on the PowerTOSSIM GUI. When switching to the "Power Profiling" window while an emulation is running, PowerTOSSIM displays power consumption results on its GUI, showing the total amount of energy consumed so far by each component of each node.

By summing up the energy consumed by the 100 emulated nodes, we get the total energy consumed by the whole network so far. During the process of the emulations, we monitor the total consumed energy by every node. When a node consumed a total energy that is equal to or greater than the maximum energy limit for each node (which is set to 8J for this emulation), we

turned off that node. We calculate the number of nodes that are still alive in the network. Once all the nodes reach its energy limit, the network life ends.

After the emulation is done and metrics are collected, we plot the following metrics: Data Received, Time, Energy Consumed, and Nodes Alive; and analyzed the data. The emulation results are summarized in Section 3.5.

3.5 Results of the LEACH Implementation

In this section, we compared our ns2 LEACH simulation results to our TinyOS LEACH implementation experimental results.

All the parameters that are used in our LEACH ns2 simulations and TinyOS implementations are listed in Table 3.3 below.

Description	Parameter	Value
Cross-over distance for Friss and two-ray ground attenuation models	$d_{crossover}$	90 m
Minimum receiver power needed for successful reception	$P_{r\text{-}thresh}$	0.473 nW
Radio electronic energy	E_{elec}	817 - 1316 nJ/bit
Compute energy for beamforming	E_{BF}	10 nJ/bit
Bitrate	R_b	38400 bps
Signal wavelength	λ	0.328 m
Radio amplifier energy	$E_{friss\text{-}amp}$ $E_{two\text{-}ray\_amp}$	22.2 pJ/bit/m^2 0.0016 pJ/bit/m^4

Number of nodes		100
Network size		100m * 100m
Base station location		(50, 175)
Radio propagation speed		$3*10^8$ m/s
Processing delay		300 μs
Total energy of each single node		8 J

Table 3.3: LEACH Simulation and Implementation Parameters

The comparison of our ns2 LEACH simulation results to our TinyOS LEACH implementation experiment results are shown in Figure 3.11 to 3.14. In these figures, "LEACH" represents the result of our ns2 simulation of LEACH protocol. "TinyOS" represents the result of our TinyOS LEACH implementation experiments.

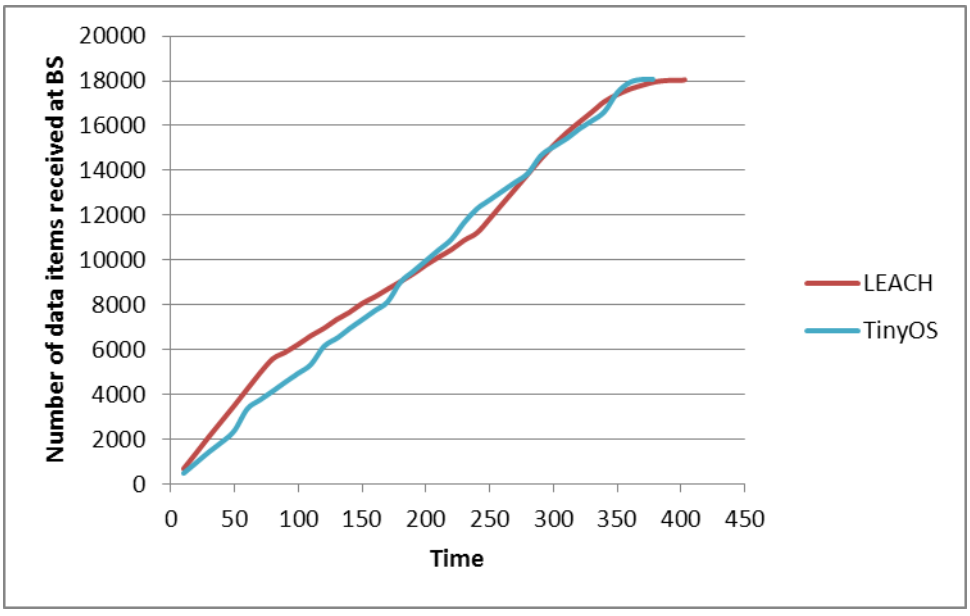


Figure 3.11: Total Amount of Data Received at the Base Station over Time

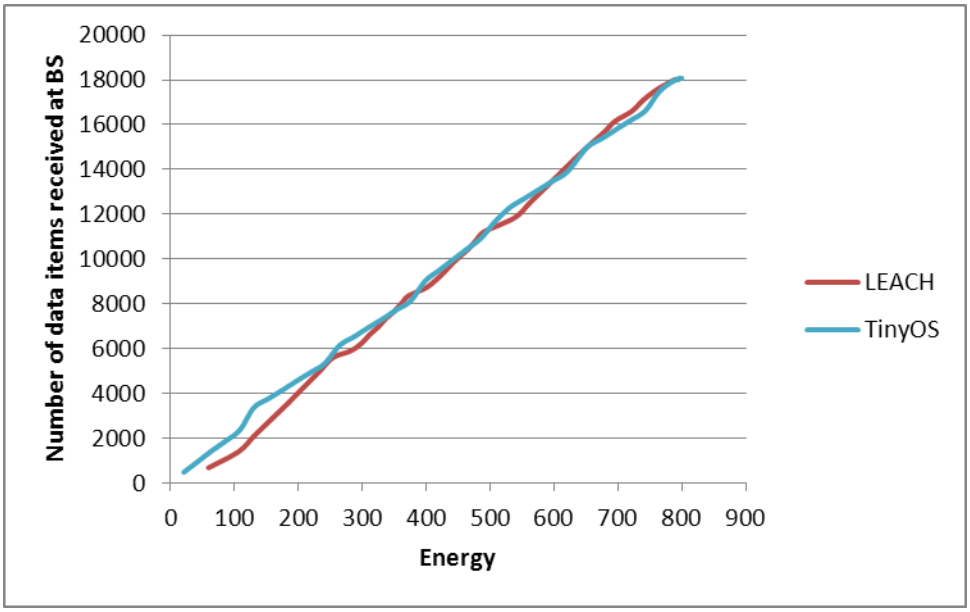


Figure 3.12: Total Amount of Data Received at the BS per Total Energy Consumed

Figure 3.11 shows the total number of data signals received at the base station over time for a given amount of energy. Figure 3.12 shows the total data received at the base station over the total amount of energy consumed by all nodes in the network.

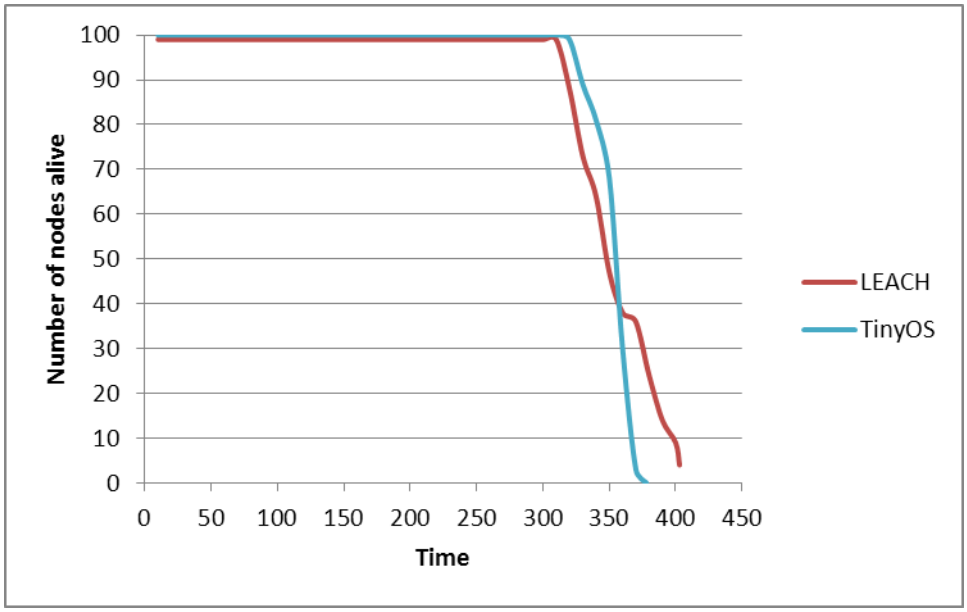


Figure 3.13: Number of Nodes Alive Function of Time

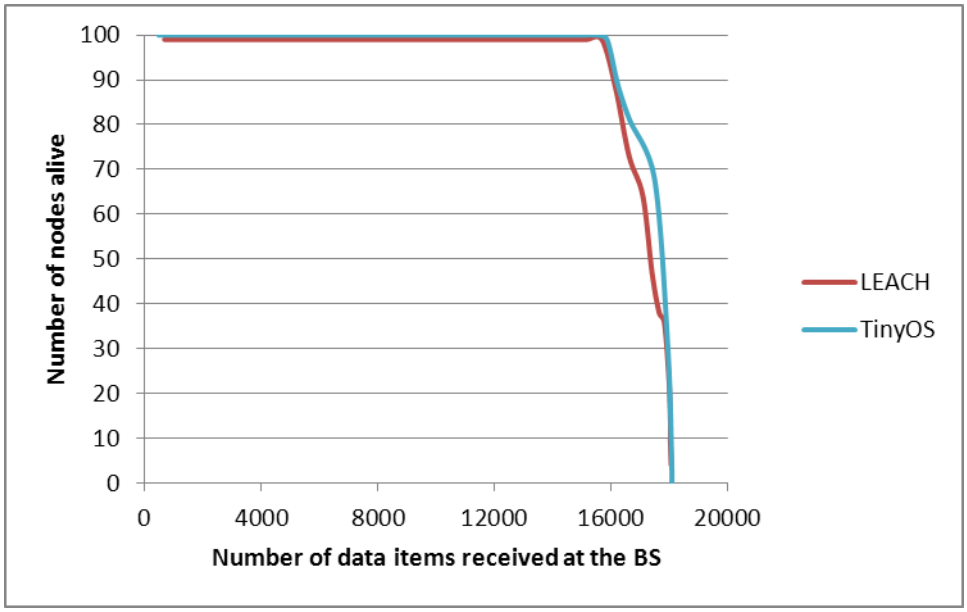


Figure 3.14: Number of Nodes Alive Function of Amount of Data Received at the BS

Figure 3.13 shows the total number of nodes that remain alive over time. Figure 3.14 shows the total number of nodes that remain alive per amount of data received at the base station.

As shown by the figures above (Figure 3.11 - 3.14), our TinyOS LEACH implementation experiment results agree well with the ns2 LEACH simulation results. This verified our TinyOS LEACH implementation and simulation results. It also confirmed the conclusions of MIT's analysis. In addition, it proved the feasibility of hardware implementation of the LEACH protocol using Crossbow's sensor network hardware. To the best of our knowledge, our TinyOS LEACH implementation is the first implementation of the LEACH protocol on any hardware platform. It is good contribution to both the research and industrial community of wireless sensor networks.

Chapter 4

D-LEACH Protocol

4.1 Introduction

A wireless sensor network consists of sensor nodes deployed over a geographical area for monitoring physical phenomena like temperature, humidity, vibrations, seismic events, and so on. Typically, a sensor node is a tiny device that includes three basic components: a sensing subsystem for data acquisition from the physical surrounding environment, a processing subsystem for local data processing and storage, and a wireless communication subsystem for data transmission. In addition, a power source supplies the energy needed by the device to perform the programmed task. This power source often consists of a battery with a limited energy budget. In addition, it could be impossible or inconvenient to recharge the battery, because nodes may be deployed in a hostile or unpractical environment. On the other hand, the sensor network should have a lifetime long enough to fulfill the application requirements. In many cases a lifetime in the order of several months, or even years, may be required. Therefore, the crucial question is: How to prolong the network lifetime?

In any case in wireless sensor networks, energy is a critical resource and must be used sparingly. Therefore, energy conservation is a key issue in the design of systems based on wireless sensor networks.

4.2 D-LEACH Protocol Architecture

In order to improve the LEACH protocol performance, reduce energy consumption and prolong the network lifetime of the wireless sensor network system, we present a data-adaptive hierarchical protocol based on LEACH. We name it Dynamic LEACH, or D-LEACH.

In the original LEACH protocol, during each frame of the Steady-State Phase, each cluster member node sends its data to the cluster head at its assigned allocated TDMA time slot. However, in a real world environment, sensor nodes do not always need to send data in each frame. The D-LEACH's data sending mechanism takes advantage of this observation and consists of the following steps during the Steady-State Phase, within each cluster:

Step 1: Initially, all the cluster member nodes send data to the cluster head.

Step 2: The cluster head calculates the amount of similarity in the data that it received from the cluster member nodes in the first frame.

Step 3: Based on the amount of data similarity, the cluster head dynamically changes the number of cluster member nodes that send data to the cluster head in the next frames. Only some of the cluster member nodes need to send data to the cluster head, as opposed to LEACH where all the member nodes send data to the cluster head each frame. The more data similarity, the fewer nodes would send data to the cluster head next frame. We can reasonably get the aggregated data from the reduced number of cluster member nodes to represent the data of all the cluster member nodes in this cluster. Based on the amount of data similarity in the data received from its cluster member nodes, the cluster head dynamically gives each cluster member

node a probability to send data to the base station in next frames. This probability is derived from the similarity of data using the method presented later.

Step 4: After a period of time (for instance, when the current round is completed), repeat steps 1 to 3.

Next we explain how the cluster head calculates data similarity and the probability for the cluster member nodes to send data to the cluster head in the next frames.

The cluster head uses Standard Deviation of the data received from its cluster member nodes to measure the amount of data similarity. Then the cluster head conducts a Bootstrapping Simulation to calculate the probability that each cluster member node shall use to send data.

In the Bootstrapping Simulation, the cluster head randomly selects data from half of the total number of cluster member nodes in its cluster, and then calculates the Standard Deviation of the data selected. The cluster head runs this process 20 times to get 20 different standard deviations. After this, the cluster head calculates the distribution of these standard deviations. Then the cluster head compares the Standard Deviation of all the data received in its cluster against the derived distribution. Finally the cluster head maps this total distribution to a probability number between 0.5 and 1.0. Then the cluster head sends this probability to its cluster member nodes. The cluster member nodes use this probability to send data to the cluster head over the next frames. The cluster member node runs a random number generation function to get a random number between 0 and 1. If the random number it gets is less than the given probability, then the node sends data to the cluster head, otherwise the node does not send data. If there is no data similarity, the cluster member nodes always send data to the cluster head.

The rest of D-LEACH protocol is identical to LEACH. We implemented D-LEACH on the TinyOS platform, ran emulations on PowerTOSSIM to analyze its performance, and compared it against other major protocols including LEACH protocol and XMesh protocol, the default routing protocol of Crossbow MICA2 hardware platform.

Computing data similarity and the probability in D-LEACH introduces some additional energy consumption. However, in most of the cases D-LEACH reduces the amount of data packet transmission, which consumes much more energy than local computing does. Our experiment results show that the additional energy consumption that D-LEACH spends on computing data similarity and the probability is less than 2% of the energy that D-LEACH spends on sending and receiving data packets. Therefore, D-LEACH reduces the overall energy consumption much in the network.

The results of performance analysis are illustrated in Section 4.4. It shows that in many cases D-LEACH achieves much superior performance than both the LEACH protocol and the XMesh protocol, in terms of power consumption, total data received and network lifetime.

4.3 XMesh Protocol Overview

XMesh is a full featured multi-hop, ad-hoc, mesh networking protocol developed by MEMSIC for wireless networks. XMesh is the default protocol of Crossbow MICA2 hardware platform. An XMesh network consists of nodes (Motes) that wirelessly communicate to each other and are capable of hopping radio messages to a base station where they are passed to a PC or other clients. By hopping data in this way, XMesh can provide two benefits: improved radio coverage and improved reliability. Two nodes do not need to be within direct radio range of each

other to communicate. A message can traverse multiple nodes before reaching its destination. Likewise, if there is a bad radio link between two nodes, that obstacle can be overcome by rerouting around the area of bad service [55].

XMesh is a software library, written for the TinyOS operating system and runs on embedded devices such as motes. XMesh's distributed routing processes have three local processes: link quality estimation, neighborhood management, and connectivity-based route selections [55]. Each node has a link estimator which characterizes the link quality of its neighboring nodes. The neighborhood management process decides how each node chooses neighbors from its potential neighbors while under memory constraint. Together, link estimation and neighborhood management build a probabilistic connectivity graph of the network. The routing process then builds topologies on this graph based on a minimum transmission cost function. The resultant topology is a subgraph of the logical connectivity graph. These three processes together form a routing process with a goal to minimize total cost and provide reliable communications [55].

4.4 Experiment Results and Analysis

We implemented D-LEACH on the TinyOS platform, ran experiments to analyze its performance, and compared it against other major protocols including LEACH and XMesh protocols. The results of the performance analysis are summarized in this section.

The setup of the experiments is the same as the setup of experiments described in Section 3.4, where we conduct experiments for the LEACH protocol. In the experiments, the actual data

that are measured by the sensor nodes and transmitted in the network are temperature data, within a range between 20 °C to 50 °C.

The results are shown in Figure 4.1 to 4.4. In these figures, "LEACH" represents the result of our TinyOS implementation of the original LEACH protocol. "D-LEACH" represents the result of our TinyOS implementation of D-LEACH protocol presented in Section 4.2. "Xmesh" represents the result of the TinyOS implementation of XMesh protocol.

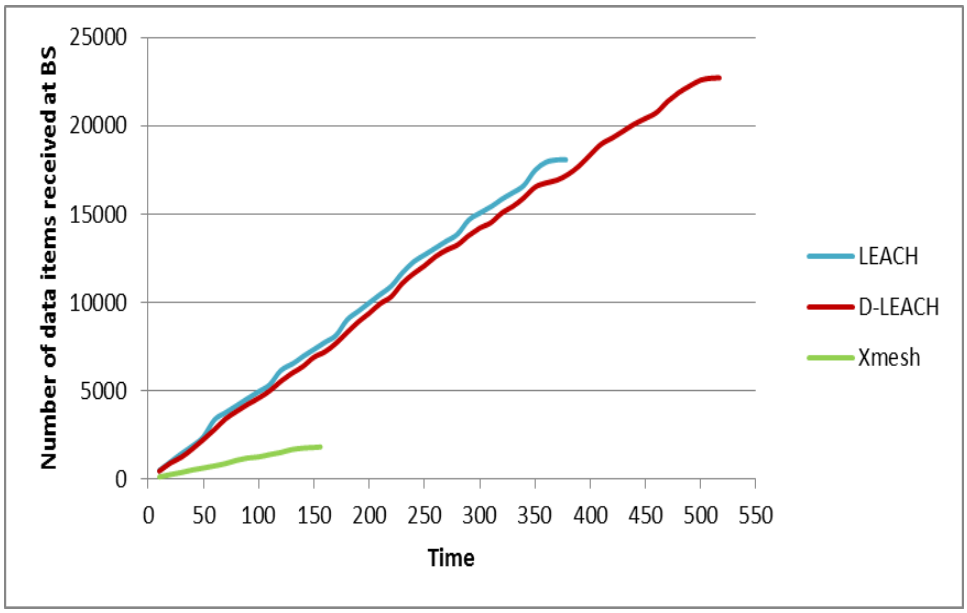


Figure 4.1: Total Amount of Data Received at the Base Station over Time

Figure 4.1 shows the total number of data signals received at the base station over emulation time, given the amount of total energy in all the nodes in the network are the same across the three different protocols. It shows that in total D-LEACH sends 26% more data packets to the base station than LEACH does (total number of packets sent: 22,705 vs 18,072),

while D-LEACH and LEACH both send much more data than XMesh protocol does in the experiment time. Moreover, D-LEACH achieves 37% longer network lifetime than LEACH does (network lifetime: 517 vs 378). Both LEACH and D-LEACH have much longer network lifetime than XMesh protocol.

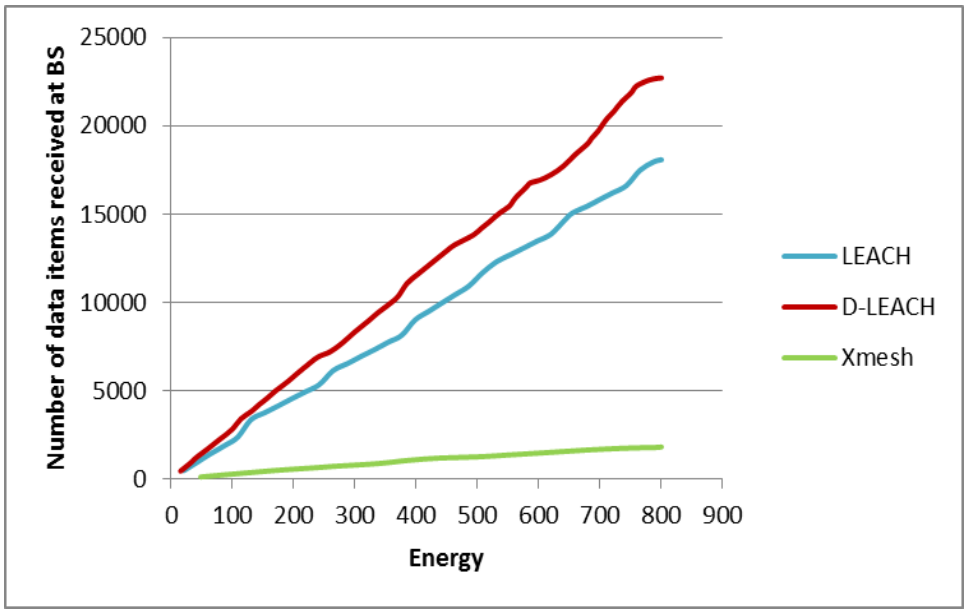


Figure 4.2: Total Amount of Data Received at the BS
per Amount of Energy Consumed by the Network

Figure 4.2 shows the total data received at the base station function of the amount of energy consumed by the network, for a given amount of total energy of the entire network. This figure shows that D-LEACH delivers the most data per unit energy, among the three protocols.

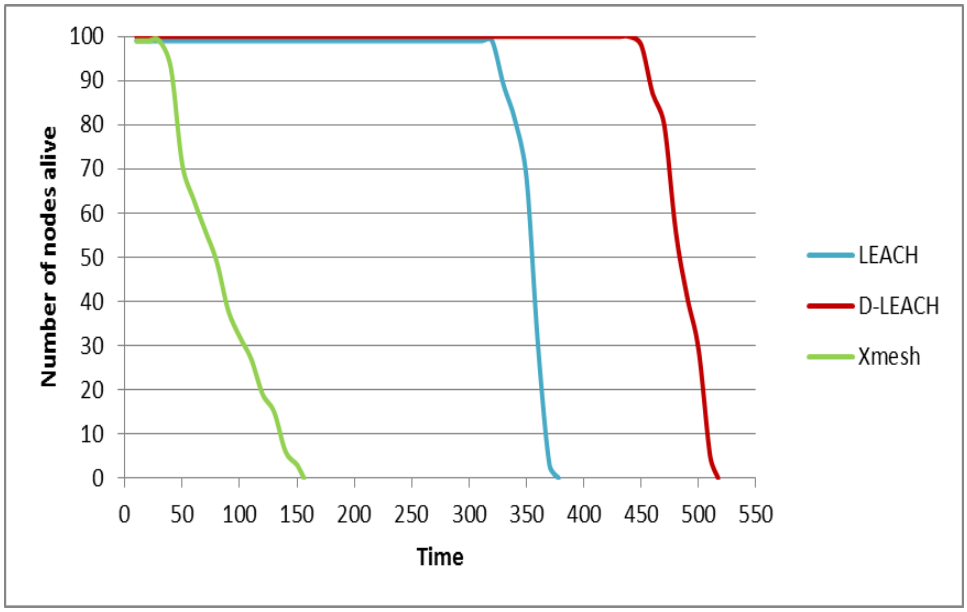


Figure 4.3: Number of Nodes Alive Over Time

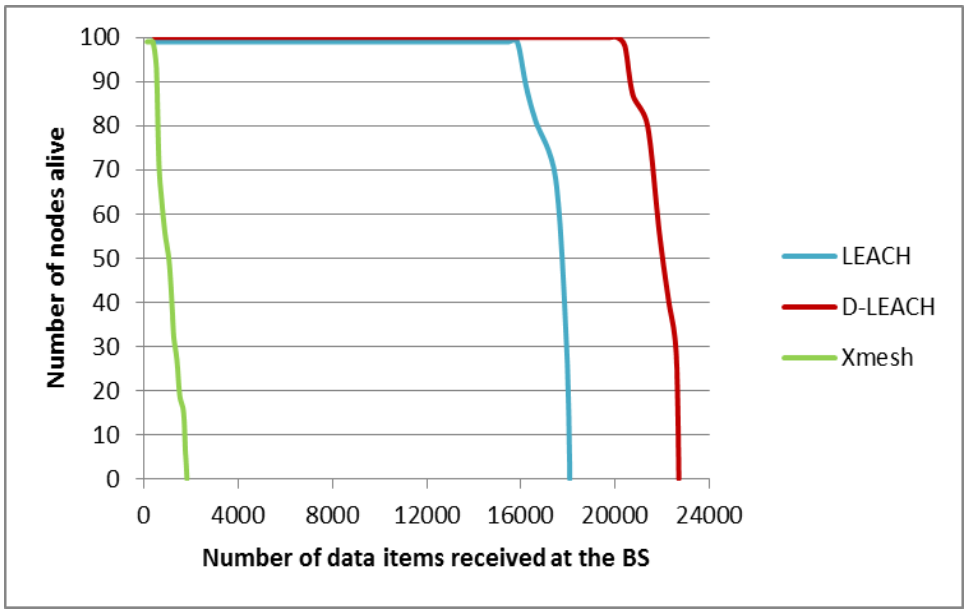


Figure 4.4: Number of Nodes Alive per Amount of Data Sent to the BS

Figure 4.3 shows the total number of nodes that remain alive over the experiment time. D-LEACH achieves better results than the LEACH and XMesh protocols. Because D-LEACH reduces the energy consumption of the nodes, nodes in D-LEACH have much longer lifetime. Therefore there are more nodes alive in the network at a given time point. D-LEACH achieves 37% longer network lifetime than LEACH does (network lifetime: 517 vs 378), and 231% longer network lifetime than XMesh does (network lifetime: 517 vs 156).

From Figure 4.4, the total number of nodes that remain alive per amount of data received at the base station, we see that nodes in D-LEACH have longer node lifetime, and can deliver more data than LEACH and XMesh protocols for the same number of nodes alive. The reason is that D-LEACH saved power consumption in the nodes thus is more energy effective in data transmission. We see that nodes in D-LEACH can deliver about 25% and ten times more effective data than LEACH and XMesh respectively, for the same number of node death.

There are two reasons why XMesh requires more energy to send data to the base station than D-LEACH and LEACH (hence, causing more node deaths for the same amount of data delivery, and a shorter network lifetime): Multi-hopping and lack of data aggregation. In the XMesh protocol, each message traverses several hops until it reaches the base station. Each hop requires additional energy consumption for sending and receiving data packets. In D-LEACH and LEACH, each message is transmitted over a single hop to the cluster head where data aggregation occurs. The aggregated signals are sent to the BS, greatly reducing the amount of data that needs to be transmitted to the base station.

As discussed above, Figures 4.1 - 4.4 show that the performance of D-LEACH protocol is superior to LEACH and XMesh protocol in all the four comparisons. As shown, D-LEACH

protocol improved the total number of data items received by 26% over LEACH protocol, and improved the network lifetime by 37% over LEACH protocol. Meanwhile it shows much better performance than XMesh protocol, in terms of total number of data items received, network lifetime, and energy consumption.

D-LEACH protocol's performance is related to the amount of similarity in the data that it receives from the nodes in the network. By changing the range of the actual data that are transmitted in the network, which are temperature data in our experiments, we are able to change the data similarity. We analyzed the impact of the data similarity on the D-LEACH's performance. The results are shown in Figure 4.5 and 4.6.

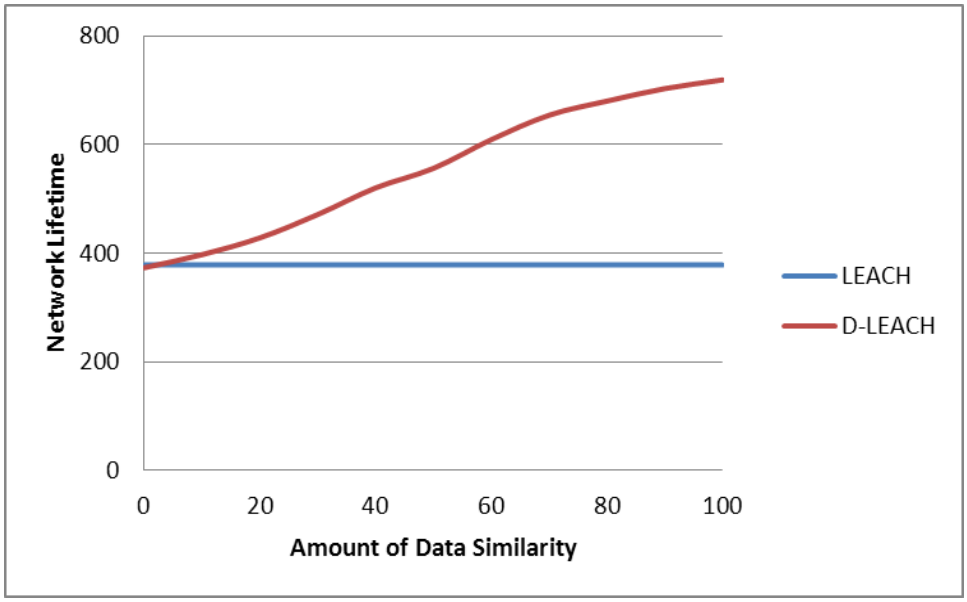


Figure 4.5: Network Lifetime over Amount of Data Similarity

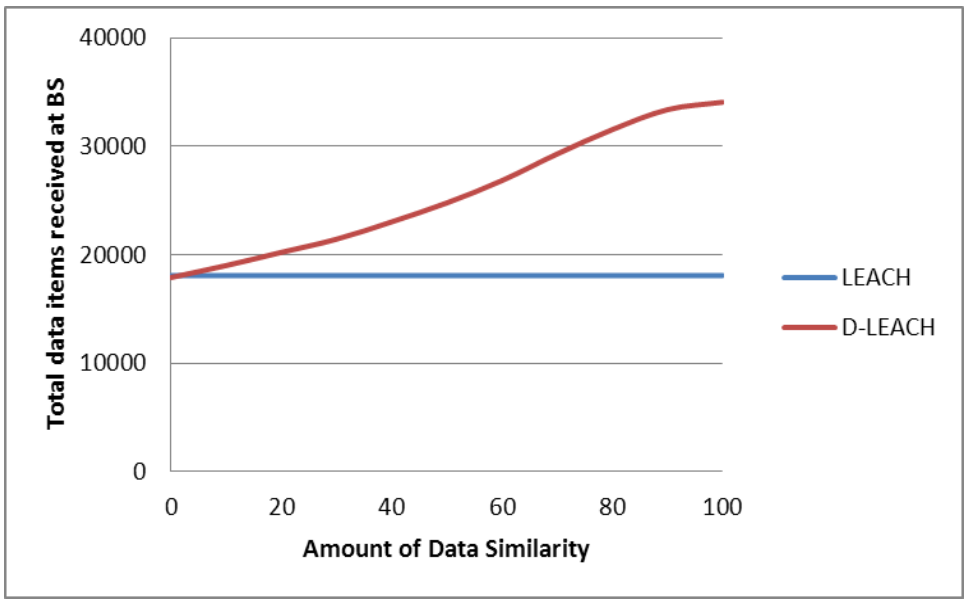


Figure 4.6: Total Amount of Data Received at the BS over Amount of Data Similarity

As shown in Figure 4.5 and 4.6, D-LEACH protocol achieves superior overall performance when compared with LEACH protocol, in terms of network lifetime and total amount of data items received. When the amount of data similarity is close to 0, D-LEACH protocol has similar performance as LEACH. As the amount of data similarity increases, D-LEACH protocol achieves better performance than LEACH. When the amount of data similarity reaches maximum, D-LEACH protocol achieves about two times of network lifetime and total amount of data items received when compared with LEACH protocol (network lifetime: 719 vs 378; total data received: 34,069 vs 18,072).

Moreover, our implementation of D-LEACH protocol on TinyOS platform worked flawlessly on Crossbow's sensor network hardware platform. It proves that D-LEACH has much perspective for real world applications of wireless sensor networks. Particularly, for wireless sensor network applications that are deployed at small to medium size fields, D-LEACH is an

excellent protocol to use. It achieves superior network performance in terms of network lifetime, power consumption, and number of data packets received in these applications.

Chapter 5

Application: A Wireless Health Monitoring System

In this research, our goal was to design a wireless sensor system, the Health Tracker 2000, that can monitors users' vital signs and notifies relatives and medical personnel of their status during life threatening situations.

The Health Tracker 2000 combines wireless sensor networks, existing vital sign monitoring technology to simultaneously monitor vital signs of the users. The use of wireless technology makes it possible to install the system in all types of homes and facilities. Radio frequency waves can travel through walls and fabric, sending the vital signs information to a central monitoring computer via a miniature transmitter network. Such information can easily be accessed from any location over the Internet. I was a member of the research and development team for this system and contributed to all the phases of system design and software/hardware development [63].

5.1 Introduction

The speed of change in the medical field has been overwhelming. Groundbreaking achievements have led people in the medical profession to have a greater understanding of the human body [60]. The average life expectancy in the United States has increased from 47.3 years in 1900 to 68.2 years in 1950 to 78.1 years in 2009 [61]. With such a high and continued increasing average life expectancy, medical care for senior citizens, age 65 and over, is becoming progressively more important.

The evolution of wireless technology is also extremely fast-paced. Wireless communications technology has become readily available for and widely used by the general public, with many million households in the U.S. using some form of a wireless network - the current wave of useful technology for home networking. The benefits of wireless technology are already apparent: portability, convenience, ease of installation, and decreasing cost.

What if wireless and medical sensor technology were combined? In this chapter, we discuss the design of a wearable device that can remotely monitor vital signs of users. This device is implemented using existing technologies. Our contributions are the idea of integrating the multiple technologies, and the actual implementation which did not exist before as of the time of this research [63].

The information from our device is sent to a base station which is connected to a computer. The information will be received by medical personnel and/or family members. Several patients may be monitored from a single base station. The system is designed so that it is easy to use and set up in medical facilities (such as hospitals) and residences. Fig. 5.1 is an overview of the Wireless Health Monitoring system.

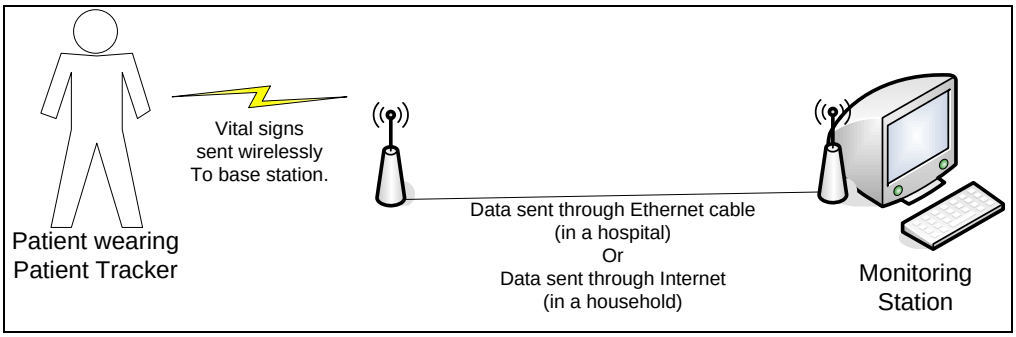


Figure 5.1: System Overview

The proposed Wireless Health Monitoring system will make it possible to wirelessly monitor vital signs of users in real-time and notify medical personnel and family members immediately in case of emergencies. With the aging of the US and the World population, the proposed system, the Health Tracker 2000, is expected to benefit the health care system. The research is sponsored by the Sensor Consortium whose goal is to introduce to and increase awareness of entrepreneurial skills for engineering students [53].

5.2 System Design

Figure 5.2 is the schematic diagram that shows the components of our Wireless Health Monitoring System.

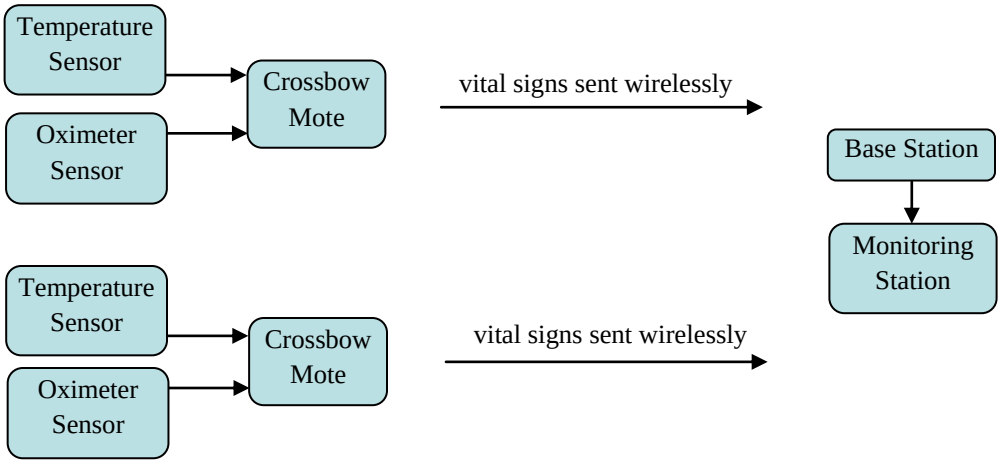


Figure 5.2: Schematic Diagram of our Wireless Health Monitoring System

The hardware design of our system is discussed in the following.

Crossbow Hardware

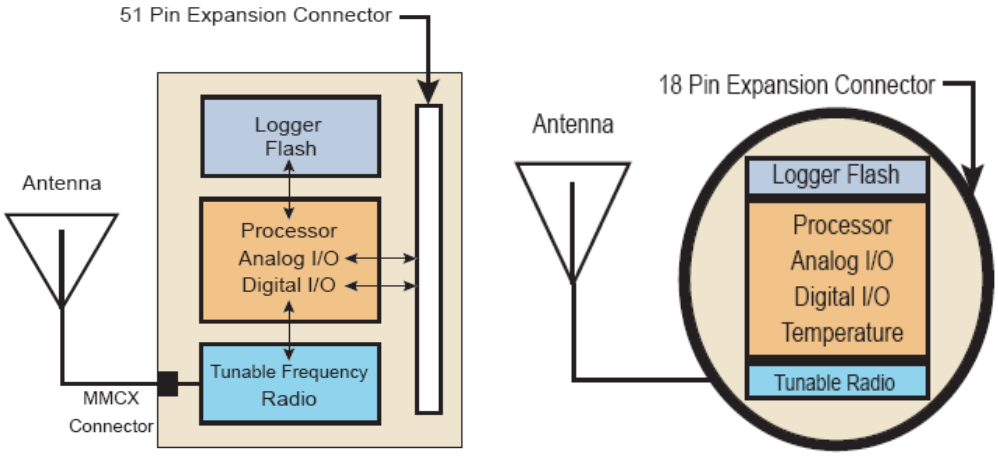


Figure 5.3: Block Diagram of a MICA2 (left) and MICA2DOT (right)

The MICA2 has a 51-pin connector and the MICA2DOT has an 18-pin connector. The MICA kit includes sensors that could interface each type of mote through these connectors. The MTS-510 sensor board is compatible with the MICA2DOT and includes a magnetometer, an accelerometer, and a microphone. The MTS-300 sensor board is compatible with the MICA2 and includes a magnetometer, an accelerometer, a microphone, a beeper, and a thermal sensor. These sensor boards are typically used for environmental monitoring. Crossbow also provides sensor acquisition boards that allow the designer to connect other sensors with the motes. The MDA510 is the sensor acquisition board compatible with the MICA2DOT. With this board, we have the ability to connect up to five sensors through the Analog-to-Digital Converter (ADC) inputs. The MDA310 sensor acquisition board is compatible with the MICA2 and has seven ADC inputs for up to seven sensors. The MICA2 and MICA2DOT motes are capable of transmitting information at frequency bands of 433.05MHz to 434.79MHz, and 902MHz to 928MHz with a data rate of 38,400 bps. The frequency band used is directly proportional to the effective transmission range.

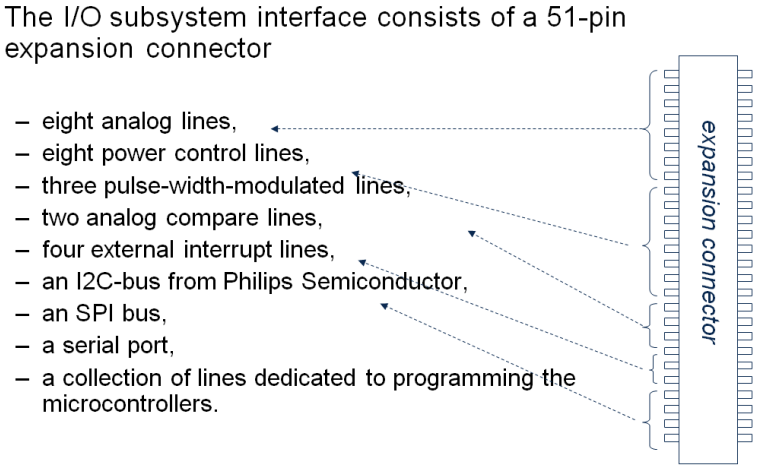


Figure 5.4: Crossbow MICA2 Pin Out

The sensor network transceivers selected for this research project, the Crossbow MICA2 and MICA2DOT motes, have embedded within a wireless transmitter/receiver, a microcontroller with analog and digital inputs, and a flash memory to store the sensor readings. These motes transmit data using RF communication, with the ability to transmit from 433MHz to 928MHz. The MICA2 and MICA2DOT motes are compatible with each other and can work together in a mesh network. Each of these motes can function as a router in a mesh network or a base station when plugged into an interface board which connected to a computer via a serial cable.

Figure 5.5: Crossbow MICA2 (left) and MICA2DOT (right)

Crossbow provides some sensor network programs for their network hardware, such as CntToLedsAndRfm, which demonstrates the motes ability to transmit data from one to another. Other programs provided by Crossbow such as Surge, allows viewing the actual network topology of the motes in use; while GoldenImage allows the motes to be updated without having to plug them into the base station.

Wireless Medical Sensor

Figure 5.6 shows the general layout of a wireless medical sensor [62]. The vital signs monitored shown in the diagram are heart rate/pulse, blood pressure, respiration rate, and body temperature. The sensors for heart rate/pulse, blood pressure, and respiration rate could be

implemented using pressure sensors. To ensure a proper reading of these sensor outputs, the signals must be amplified using op amps. The outputs from the temperature sensor and the three op amps are converted to a digital signal using an ADC (analog to digital converter). These signals are processed by a microcontroller or microprocessor and the resulting data is transmitted through a wireless module.

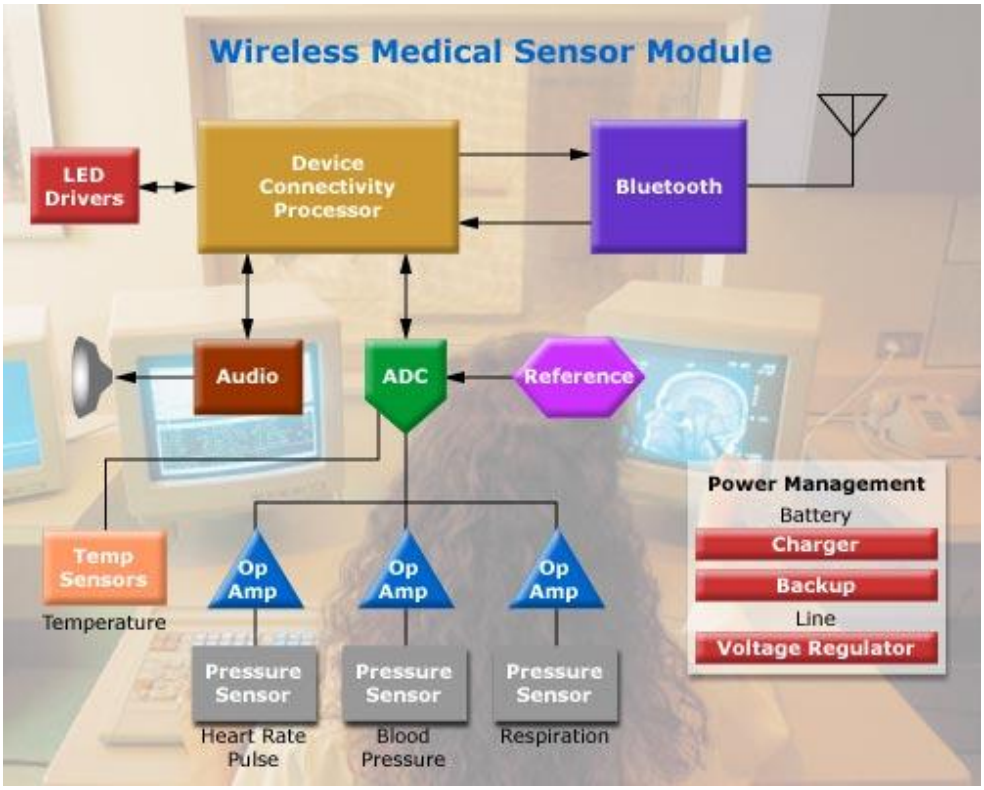


Figure 5.6: Wireless Medical Sensor Module

The wireless transceivers and receivers selected for this research project are the Crossbow MICA2 and MICA2DOT motes. These motes are connected to various vital sign sensors via the sensor acquisition boards, MDA310 and MDA510, also provided by Crossbow Technologies Inc.

The wireless medical sensor we are designing measures several vital signs of the user. Specifically, these vital signs are: temperature, pulse rate, and blood oxygen level.

Thermal

When selecting a thermal sensor, we first considered the National Semiconductor LM34 Precision Fahrenheit Temperature Sensor. However, there are several problems in using the LM34. First is the supply voltage. The LM34 requires at least 5V at the input. This requires us to add extra batteries and a voltage regulator which means extra components and larger size for the final product. The second problem is that the LM34 has a resolution of +/-1°F. This is too high for medical applications because body temperature of a healthy human is 98.6°F, the LM34 will accurately read this as either 98°F or 99°F.

We then considered the LM92 by National Semiconductors. The major difference is that the LM92 has a serial digital output (Fig. 5.7). Since the MDA510 does not have any digital inputs or outputs, we must use the LM92 with the MDA310 which is compatible with the MICA2. The LM92 has a minimum supply voltage of 2.7V and a resolution of 0.3°F, which is the best resolution among the commercially available National thermal sensors [50]. This component is clearly the ideal choice for our system. The maximum transmission frequency of LM92 is 2 readings per second. Temperature data is represented by 2 bytes.

An additional advantage in choosing the LM92 is that since it has a 2.7V minimum voltage supply, we are able to use the INT outputs on the MICA2. The INT outputs supply power to the sensors only when the mote is polling for data. Since the LM34 was connected to the 5V voltage regulator which is connected directly to the battery, it is always draining the battery. The INT outputs generate a 3V square wave (Fig. 5.8) that is asserted when obtaining

sensor data and unasserted when the mote is in sleep mode. This will considerably extend the battery life when used in comparison with the LM34.

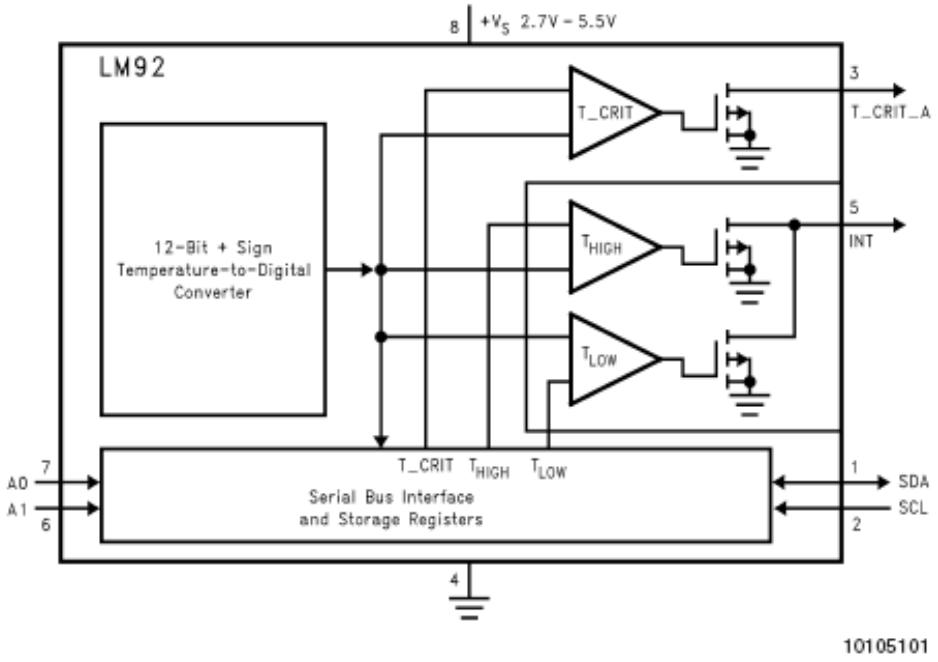


Figure 5.7: Schematic of LM92

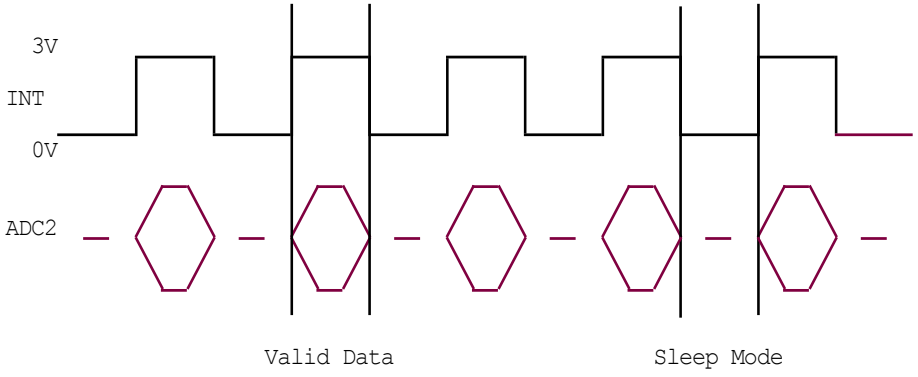


Figure 5.8: Timing Diagram for the Temperature Sensor

Oximeter

The Nonin ipod (Fig. 5.9) is a digital oximeter and sensor that goes over the user's finger [65]. The unit itself is simply a module that can be readily integrated into a host device. The module measures the blood oxygen level and pulse rate. The oximeter and sensor is contained in a single unit to save space in the final product. The ipod will be integrated in the system via a 3-wire interface with serial communications, similar to that of the LM92 temperature sensor. Since it is a digital output the unit will interface the MICA2 via its digital inputs. The ipod has a minimum operating voltage of 2V with a maximum of 6V, thus matching the specifications of our supply voltage of 3V.

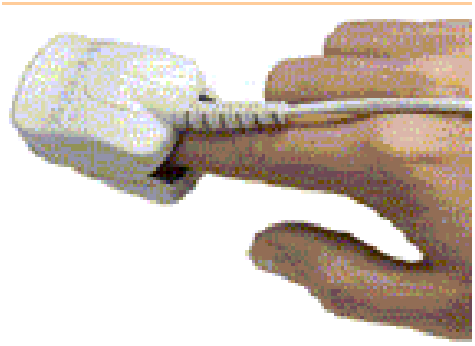


Figure 5.9: Nonin ipod

The ipod has a serial digital output. Table 5.1 shows the format of the output. The data from the device is sent at a rate of 3 bytes per second. The first byte is the status byte, the second is the heart rate, and the third is the blood oximetry. Some experimentation is needed to

correctly interpret the data sent to the mote. With serial data transmissions, timing is always an issue. Normally a serial device transmits data by latching each bit by means of a clock signal and a data signal.

	Status Byte	Heart Rate Byte	SpO2 Byte
Bit 0	Heart rate bit 7	Heart rate bit 0	SpO2 bit 0
Bit 1	Heart rate bit 8	Heart rate bit 1	SpO2 bit 1
Bit 2	Bad pulse, set if true	Heart rate bit 2	SpO2 bit 2
Bit 3	Marginal perfusion, set if true	Heart rate bit 3	SpO2 bit 3
Bit 4	Low perfusion, set if true	Heart rate bit 4	SpO2 bit 4
Bit 5	Out of track, set if true	Heart rate bit 5	SpO2 bit 5
Bit 6	Sensor disconnected, set if true	Heart rate bit 6	SpO2 bit 6
Bit 7	Always set to “1”	Always set to “0”	SpO2 bit 7

Table 5.1: Format of Nonin ipod Output

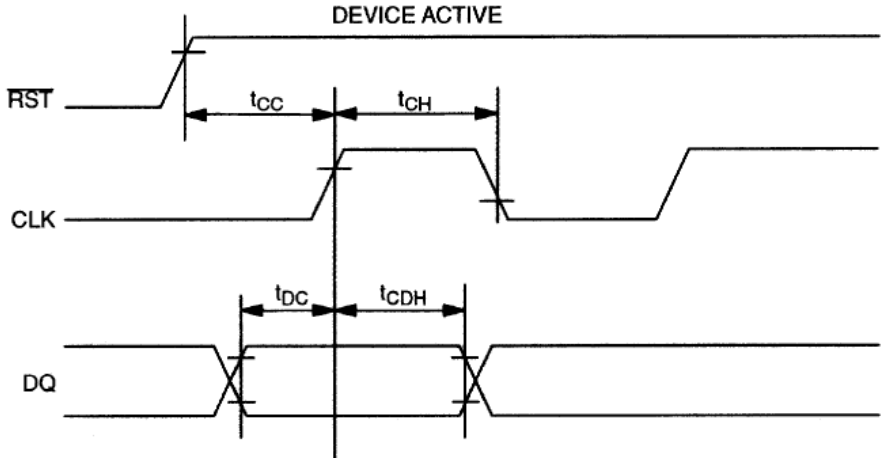


Figure 5.10: Serial Data Transfer Timing Waveform

Fig. 5.10 shows a timing waveform for a typical serial data device. The RST bit is unasserted to allow data to transfer. The DQ (data pin) displays the first bit and the bit is sent

when CLK goes high. The timing waveform for the device above is an active high latch, since the data is being sent when the CLK bit is high and not during the positive or negative edge. In other words, bit DQ is sent only when the CLK bit is in the t_{CH} region shown in Fig. 5.10. When CLK goes low, DQ changes to the next data bit and CLK goes high again to store the next bit. So in each data transmission, CLK goes high and low 24 times for 3 bytes of data. The Nonin ipod has two output wires: one for TTL and the other for RS-232. The TTL wire is equivalent to the CLK bit shown in the timing diagram and the RS-232 data bit is equivalent to the DQ bit.

For future extension, we researched the possibility of adding respiration rate sensors, and radio frequency identification tags (RFID) into our system, so that our system can monitor users' respiration rates, and track their location in real time. This work is not implemented in our current system, but can be added in future versions, as our system architecture is very scalable.

As mentioned, the maximum transmission frequency of LM92 temperature sensor is 2 readings per second, where temperature data is represented by 2 bytes. The data from the Nonin ipod oximeter is sent at a rate of 3 bytes per second. So the maximum data rate of total data collected by our system from one single user is 56 bps. The MICA2 and MICA2DOT motes are capable of transmitting information with a data rate of 38,400 bps. Therefore, our system has enough bandwidth to support hundreds of simultaneous users sending data at the maximum frequency.

5.3 System Implementation and Results

Once all the sensors are connected to the mote, a software system is developed to read the information and redirect the output to a database. From there, another software system that we developed reads the data from the database, interprets the data and determines whether the users of Health Tracker 2000 are in critical condition. If so, an alert will show and a message will be sent through the Internet.

A graphical user interface (GUI) was also developed to be used in the base station.

We have integrated the following parts in our system: the Crossbow MICA2 mote (MPR400CB), the Crossbow MDA310 Interface Board, the National Semiconductors LM92 Precision Centigrade Thermal Sensor, and the Nonin Integrated Pulse Oximetry Device (Ipod).

The LM92 thermal sensor was integrated in the MICA2 mote through the mote's I2C interface. The data read from the LM92 sensor was successfully sent to the base station wirelessly. The data was confirmed and displayed at the host computer. The temperature data from the LM92 sensor was tested for correctness by connecting the LM92 serial data (SDA) channel to an oscilloscope. The room temperature was verified by comparing the LM92's reading with another thermometer. Once the data generated by the thermal sensor was verified, the temperature data received at the monitoring station through wireless communication and our software interface was then verified. Some programming bugs, regarding the timing issue in Crossbow Technology's system program library were fixed in the process. The wave form in Figure 5.11 shows the temperature data recorded on the monitoring station in real time.

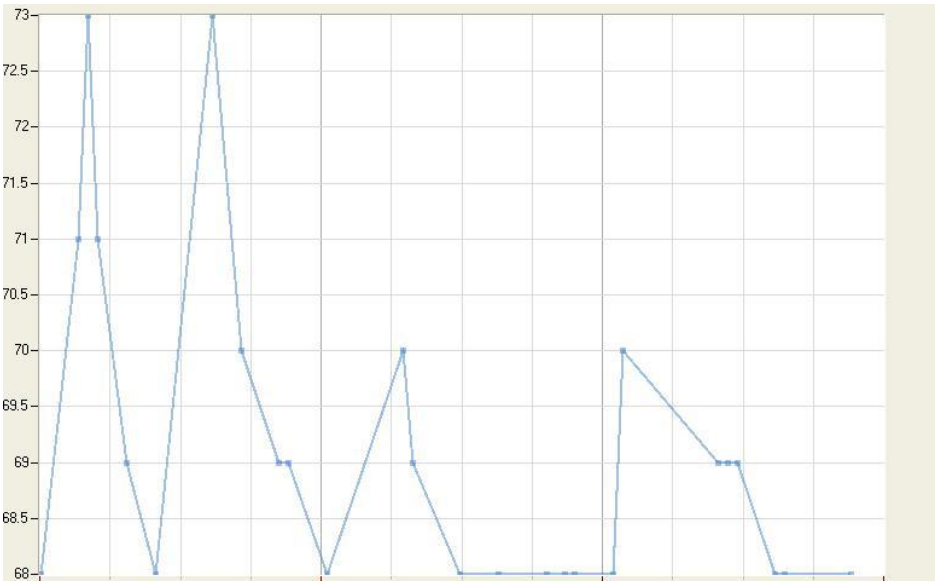


Figure 5.11: Temperature Data Recorded and Shown on the Monitoring Station

The Ipod oximeter was also integrated in the MICA2 mote through the mote's I2C interface. Figure 5.12 shows the blood oxygen concentration and heart rate of a user recorded and displayed on the monitoring station when the device clips onto a finger of the user.

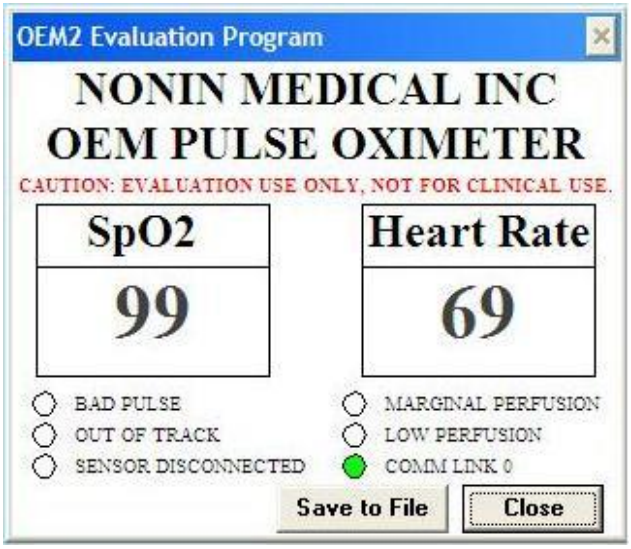


Figure 5.12: Blood Oxygen Level and Heart Rate Displayed on the Monitoring Station

5.4 Discussions

The goal of this research is to integrate various vital sign sensors with the motes provided by Crossbow Technology Inc. to provide a wireless health monitoring system. We identified the temperature sensor, the oximetry sensor, the respiration rate sensor and the RFID sensor as the key sensors in the system. We have integrated the temperature sensor and the oximetry sensor in our system. At a first glance, the project seems trivial as only system integration is involved. However, as we proceed with the project, we found that different sensors have different data formats. System integration has to accommodate these different formats and different hardware ports, and deal with many hardware and software technical challenges.

Chapter 6

Conclusions

Advances in micro-electro-mechanical systems technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power wireless sensor nodes that are small in size and can communicate untethered in short distances. These wireless sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on the collaborative effort of a large number of nodes. Networking together hundreds or thousands of wireless sensor nodes allows users to accurately monitor a remote environment by intelligently combining the data from individual nodes. These networks require robust wireless communication protocols that are energy and data transmission efficient with long network lifetime.

The work described in this dissertation has demonstrated the advantages of data adaptive hierarchical protocols for wireless sensor networks by designing and evaluating a new protocol - D-LEACH. In addition, a novel prototype application of wireless sensor network technology in real world health monitoring was implemented.

In the first part of this dissertation, we analyzed and implemented the LEACH protocol, an existing protocol architecture for wireless sensor networks that combines the ideas of energy-efficient cluster-based routing together with data aggregation to achieve desired performance. In

an effort to improve the LEACH protocol performance, the underlying motes power consumption and network lifetime, we proposed a data-adaptive hierarchical protocol based on LEACH, named Dynamic LEACH, or D-LEACH. In the D-LEACH protocol, the cluster head calculates the amount of similarity in the data it received from the cluster member nodes in each frame. Based on the data similarity, the cluster head dynamically assigns each cluster member node a probability to send data to the base station in the next frames. This probability is derived from the similarity of data using a Bootstrapping Simulation method. The amount of data similarity has direct impact on D-LEACH performance when compared with LEACH. In real world, particularly for wireless sensor network applications that are deployed at small to medium size fields, D-LEACH is an excellent protocol to use, as the data similarity is most likely significant in most of these cases.

We implemented LEACH and D-LEACH protocols on the TinyOS platform, ran experiments, analyzed the performance of D-LEACH protocol, and compared D-LEACH performance with other major protocols including LEACH and XMesh protocol. Our implementation is the first time that LEACH has ever been implemented on hardware motes. The results show that D-LEACH achieves much superior performance than LEACH and XMesh protocol, in term of power consumption, total data received and network lifetime. It proves that D-LEACH protocol has much perspective for real world applications of wireless sensor networks.

The second study proposed and developed a prototype for a novel vital sign monitoring system based on wireless sensor network technologies. In this research, our goal was to design a wireless sensor system that can monitor users' vital signs and notifies relatives and medical personnel of users' status during life threatening situations. The application combines wireless sensor networks, existing vital sign monitoring technologies to simultaneously monitor vital

signs. The use of wireless technologies makes it possible to install the system in all types of homes and facilities. Radio frequency waves can travel through walls and fabric, sending the vital signs information to a central monitoring computer via a miniature transmitter network. Such information can easily be accessed from any location over the Internet, thus greatly improving the health monitoring for the users.

For future work, we would like to further investigate the performance of the D-LEACH protocol under different traffic patterns, network sizes, and application types. We would like to provide a set of design guidelines for engineers to choose and set up the D-LEACH protocol based on the specifications of the applications such as expected traffic patterns, network sizes, and application types.

Another part of the future work is to add more features to our Wireless Health Monitoring System to support monitoring more vital signs and the user's physical location. Therefore, our system can monitor more vital signs and notify medical personnel of users' location and status during life threatening situations. In addition to adding these features, a research issue is the establishment of an efficient sensor network that can support large number of simultaneous users in a hospital or assisted living environment.

Bibliography

[1] Kenneth S. Johnson, Joseph A. Needoba, Stephen C. Riser, and William J. Showers, "Chemical Sensor Networks for the Aquatic Environment", *Chem. Rev.,* 2007, 107, pp. 623-640.

[2] S.N. Simić and S. Sastry, "Distributed Environmental Monitoring Using Random Sensor Networks", *Proc. 2nd Int. Workshop on Information Processing in Sensor Networks,* Palo Alto, CA, April 2003, pp. 582-592.

[3] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler and J. Anderson, "Analysis of Wireless Sensor Networks for Habitat Monitoring", *Wireless Sensor Networks,* 2004, VI, pp. 399-423.

[4] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira and J. Spletzer, "Distributed Search and Rescue With Robot and Sensor Teams", *Proc. 4th Int. Conf. on Field and Service Robotics,* Lake Yamanaka, Japan, July 2003.

[5] T. Schmid, H. Dubois-Ferrière, and M. Vetterli, "SensorScope: Experiences with a Wireless Building Monitoring Sensor Network", *Proc. of First Workshop on Real-World Wireless Sensor Networks,* Stockholm, Sweden, June 2005.

[6] "The Ultimate Diagnostic Solution", http://www.cancerdetector.com/, 2012

[7] K. Sharma and M. K. Ghose, "Wireless Sensor Networks: An Overview on its Security Threats", *IJCA Special Issue on MANETs,* Vol. 1, pp. 42-45, 2010.

[8] W. Heinzelman, "Application-Specific Protocol Architectures for Wireless Networks", PhD Dissertation, Massachusetts Institute of Technology, June 2000.

[9] W. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", *IEEE Transactions on Wireless Communications,* Vol. 1, No. 4, Oct. 2002, pp.660-670.

[10] "TinyOS Documentation", http://www.tinyos.net/tinyos-2.x/doc/, 2012

[11] B. Warneke, M. Last, B. Liebowitz, and K. S.J. Pister, "Smart Dust: Communicating with a Cubic-Millimeter", *Computer,* Vol. 34, pp. 44-51, 2001.

[12] Victor Shnayder, Mark Hempstead, Bor-rong Chen, and Matt Welsh, "Simulating the Power Consumption of Large-Scale Sensor Network Applications", *Proceedings of ACM SenSys04*, Baltimore, MD, November 2004.

[13] "The Network Simulator - ns2", http://www.isi.edu/nsnam/ns/, 2012

[14] Jason Lester Hill, "System Architecture for Wireless Sensor Networks", PhD Dissertation, University of California, Berkeley, 2003.

[15] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey", *Computer Networks*, Vol. 38, Issue 4, pp. 393-422, March 2002.

[16] J. Feng, F. Koushanfar, and M. Potkonjak, "System-Architectures for Sensor Networks Issues, Alternatives, and Directions", *Proceedings of the 20th International Conference on Computer Design*, 2002.

[17] "WPSM: Warfighter Physiological Status Monitoring", http://www.momrp.org/publications/WPSM_supp.pdf, 2012

[18] M.A.M. Vieira, C.N. Coelho. Jr., D.C. da Silva Jr., and J.M. da Mata, "Survey on Wireless Sensor Network Devices", *IEEE Conference on Emerging Technologies and Factory Automation*, 2003.

[19] J.M. Kahn, R.H.Katz, and K.S.J. Pister, "Next Century Challenges: Mobile Networking for Smart Dust", *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 271-278, Seattle, Washington, 1999.

[20] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, WA, August 1999.

[21] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00)*, Boston, MA, August 2000.

[22] D. Braginsky and D. Estrin, "Rumor Routing Algorithm for Sensor Networks", *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, October 2002.

[23] C. Schurgers and M.B. Srivastava, "Energy Efficient Routing in Wireless Sensor Networks", *MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force*, McLean, VA, 2001.

[24] M. Chu, H. Haussecker, and F. Zhao, "Scalable Information-Driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks", *The International Journal of High Performance Computing Applications,* Vol. 16, No. 3, August 2002.

[25] A. Manjeshwar and D. P. Agrawal, "TEEN: A Protocol for Enhanced Efficiency in Wireless Sensor Networks", *Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, CA, April 2001.

[26] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks", *SIGMOD Record*, September 2002.

[27] R. Shah and J. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Orlando, FL, March 2002.

[28] N. Sadagopan, B. Krishnamachari, and A. Helmy, "The ACQUIRE Mechanism for Efficient Querying in Sensor Networks", *Proceedings of the First International Workshop on Sensor Network Protocol and Applications*, Anchorage, Alaska, May 2003.

[29] S. Hedetniemi and A. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks", *Networks*, Vol. 18, No. 4, pp. 319-349, 1988.

[30] D. Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, WA, August 1999.

[31] A. Buczak and V. Jamalabad, "Self-Organization of a Heterogeneous Sensor Network by Genetic Algorithms", *Intelligent Engineering Systems Through Artificial Neural Networks*, C.H. Dagli, et.(eds.), Vol. 8, pp. 259-264, ASME Press, New York, 1998.

[32] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", *IEEE Journal on Selected areas in Communications*, Vol. 15, No. 7, September 1997.

[33] V. Chandrasekaran, "A Review on Hierarchical Cluster Based Routing in Wireless Sensor Networks", *Journal of Global Research in Computer Science*, 2012.

[34] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power Efficient GAthering in Sensor Information Systems", *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, March 2002.

[35] S. Lindsey, C. S. Raghavendra and K. Sivalingam, "Data Gathering in Sensor Networks using the Energy Delay Metric", *Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing*, San Francisco, CA, April 2001.

[36] A. Manjeshwar and D. P. Agrawal, "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks", *Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing*, Ft. Lauderdale, FL, April 2002.

[37] L. Subramanian and R. H. Katz, "An Architecture for Building Self Configurable Systems", *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing*, Boston, MA, August 2000.

[38] M. Younis, M. Youssef and K. Arisha, "Energy-Aware Routing in Cluster-Based Sensor Networks", *Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS2002),* Fort Worth, TX, October 2002.

[39] Y. Xu, J. Heidemann, and D. Estrin, "Geography-Informed Energy Conservation for Ad Hoc Routing", *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01),* Rome, Italy, July 2001.

[40] V. Rodoplu and T.H. Ming, "Minimum Energy Mobile Wireless Networks", *IEEE Journal of Selected Areas in Communications,* Vol. 17, No. 8, pp. 1333-1344, 1999.

[41] L. Li and J. Y Halpern, "Minimum Energy Mobile Wireless Networks Revisited", *Proceedings of IEEE International Conference on Communications (ICC'01),* Helsinki, Finland, June 2001.

[42] G. Finn, "Routing and Addressing Problems in Large Metropolitan-Scale Internetworks", *University of Southern California, Tech. Rep. ISI Research Report ISI/RR-87-180,* 1987.

[43] B. Nath and D. Niculescu, "Routing on a Curve", *ACM SIGCOMM Computer Communication Review,* Vol. 33, Issue 1, January 2003.

[44] Y. Yu, D. Estrin, and R. Govindan, "Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks", *UCLA Computer Science Department Technical Report UCLA-CSD TR-01-0023,* May 2001.

[45] J.-H. Chang and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks", *Proceedings of the Advanced Telecommunications and Information Distribution Research Program (ATIRP'2000),* College Park, MD, March 2000.

[46] K. Kalpakis, K. Dasgupta and P. Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks", *Proceedings of IEEE International Conference on Networking (NETWORKS '02),* Atlanta, GA, August 2002.

[47] K. Akkaya and M. Younis, "An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks", *Proceedings of the IEEE Workshop on Mobile and Wireless Networks (MWN 2003),* Providence, Rhode Island, May 2003.

[48] J.A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-time Communication in Sensor Networks", *Proceedings of International Conference on Distributed Computing Systems,* Providence, RI, May 2003.

[49] T. Murata and H. Ishibuchi, "Performance Evaluation of Genetic Algorithms for Flowshop Scheduling Problems", *Proc. 1st IEEE Conf. Evolutionary Computation*, Vol. 2, pp. 812-817, June 1994.

[50] "National Semiconductor", http://www.national.com, 2012

[51] Jason Hill, Mike Horton, Ralph Kling and Lakshman Krishnamurthy, "The Platforms Enabling Wireless Sensor Networks", *Communications of the ACM,* Vol. 47, Issue 6.

[52] P. Agarwal and C. Procopiuc, "Exact and Approximation Algorithms for Clustering", *Proc. 9th Annu. ACM-SIAM Symp. Discrete Algorithms*, Baltimore, MD, Jan. 1999, pp. 658-667.

[53] "Sensor Consortium", http://www.ee.sunysb.edu/~sensorconsortium/, 2012

[54] "ns2 Documentation", http://www.isi.edu/nsnam/ns/doc/node171.html, 2012

[55] "XMesh User Manual", http://www.memsic.com/support/documentation/wireless-sensor-networks, 2012

[56] "IEEE Standard 802.15.4d-2009", *IEEE Computer Society,* April 2009.

[57] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks", *Ad Hoc Networks,* March 2008.

[58] P. Kansal, and D. Kansal, "Compression of Various Routing Protocol in Wireless Sensor Network", *International Journal of Computer Applications,* 2010.

[59] L. Liu, F. Xia, Z. Wang, J. Chen, and Y. Sun, "Deployment Issues in Wireless Sensor Networks", *Proc. of First MSN International Conference*, Wuhan, China, December 2005.

[60] "History of Anatomy", http://www.intuitivebodywork.net/history-of-anatomy, 2012

[61] "The World Bank Report", http://www.worldbank.org/, 2012

[62] M. Welsh, D. Malan, and B. Duncan, "Wireless Sensor Networks for Emergency Medical Care", *GE Global Research Conference*, Harvard University, 2004.

[63] Edwards Teaw, Guofeng Hou, M. Gouzman, K.W. Tang, A. Kesluk, M. Kane and J. Farrell, "A Wireless Health Monitoring System", *Proc. of the International Conference on Information Acquisition,* pp. 247-252, June 2005.

[64] Guofeng Hou and K.Wendy Tang, “Evaluation of LEACH Protocol Subject to Different Traffic Models”, *Proc. of the Joint International Conference on Optical Internet and Next Generation Network*, pp. 281-283, July 2006.

[65] “Nonin Medical”, http://www.nonin.com, 2012

[66] “Crossbow Technologies”, http://www.xbow.com, 2012

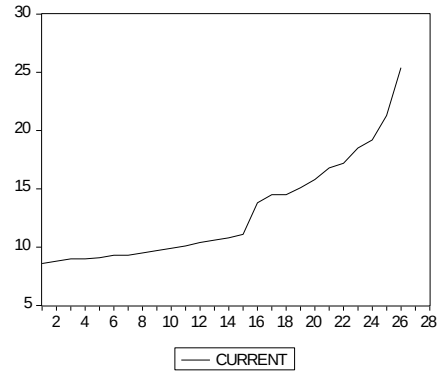
[67] Philip Levis, Nelson Lee, Matt Welsh, and David Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, 2003.

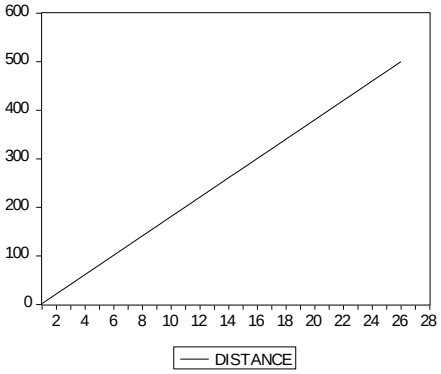
Appendix A

Energy Model Analysis

Data analysis

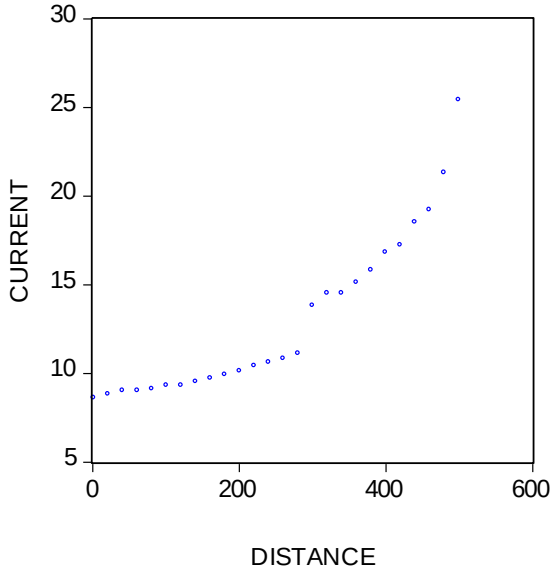
We used all the 26 points in Table 3.1 from current interval between 8.6 (mA) and 25.4 (mA), in accord with which we got distance level range from 2 (ft) to 500 (ft). Totally, we have 26 observations. First, we plot the graph of Current and Distance separately to see the trend. As can be seen from the figures below, both current and distance have an ascending trend.





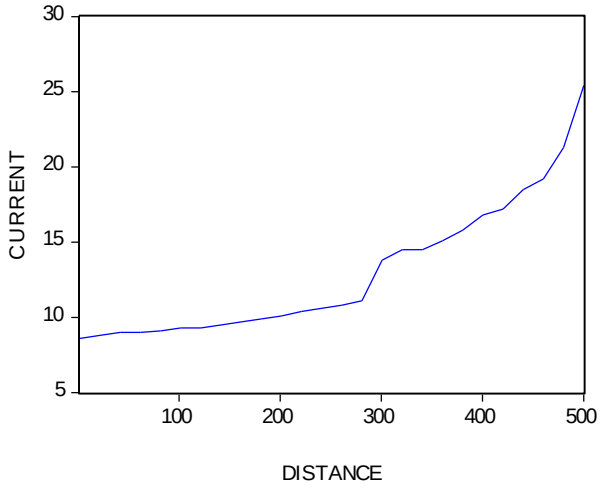
Second, we plot the scatter graph of the current versus distance and found that they have a positive relation. Particularly, there is a break point at observation 16 which coincides with the

theoretical reference mentioned in chapter 3.3. After the break point, the line suddenly becomes steeper.



Scatter Graph

Further, we draw the relation in line.



Regression and results

We use Eview 4.3 to analysis the relation between current and distance. Specifically, we apply the Least Square method. In order to get the most accurate results, we separate the samples at the break point and use over 30 models to test the relation within each part. The most satisfying 15 regression results are attached. We select from these 15 models via three criteria: One is the goodness-of-fit which test how well the regression values fit the actual values; the

second one is significant level, which tests whether the coefficient is significant; the last one is robustness.

First, we test the relation before the break point (observation 1 to observation 15) and found current is positively related to $Distance^2$ as shown in equation (1). Table A.1 shows that coefficient and constant are significantly over 1% level. Goodness-of-fit is 98%.

$$Current = 8.874057 + 2.94E\text{-}05 * Distance^2 \qquad (1)$$

Figure A.1 shows the actual, fitted and residual graph for equation (1), where the actual line and fitted line co-integrated well.

Next, we run the regression on the observations after break point. After comparing and analysis, we get equation (15).

$$Current = 11.61814 + 1.94E\text{-}10 * Distance^4 \qquad (15)$$

Table A.15 shows that coefficient and constant are significantly over 1% level. Goodness-of-fit is 95%. Figure A.15 shows the actual, fitted and residual graph for equation (15), where the actual line and fitted line co-integrated well.

The results shown in equation (1) and equation (15) agree well with the electronic communication theories. The first part relates to $Distance^2$, that corresponds to the Friss free space model. The second part relates to $Distance^4$, that corresponds to the two-ray ground propagation model. The communication theory indicates that if the distance between the transmitter and receiver is less than a certain cross-over distance ($d_{crossover}$), the Friss free space model should work, and if the distance is greater than $d_{crossover}$, the two-ray ground propagation model should work. As we can see from the results above, our analysis results agree with the theory.

The only difference is that the constant levels in equation (1) and equation (15) are not the same. Fortunately, these two values are close. The error may arise from experiments like measurement error, or it may come from experiment environment like temperature, humid and obstacles.

Furthermore, we tried to test the whole sample by one model. After comparing the criteria we select model shown in Table A.6.

$$Current = 8.178905 + 5.62E\text{-}05 * Distance^2 \qquad (4'')$$

The coefficient and constant are significantly over 1% level. Goodness-of-fit is 96%. Figure A.5 shows the actual, fitted and residual graph for equation (4''), where the actual line and fitted line co-integrated well. Even though the result sounds well, it against the theoretical reference which states that there should be a break point. And the Goodness-of-fit value is not as good as that of the way where there is a break point, as shown above. Also, the regression result on the second part of the samples using equation (4'') shows that this result (equation 4'') doesn't work well on second part of the samples.

As a summary of the analysis above, the Crossbow motes radio energy model could be summarized as below:

When transmitting data,

when distance < $d_{crossover}$, Current = 8.874057 + 2.94E-05 * $Distance^2$

when distance > $d_{crossover}$, Current = 11.61814 + 1.94E-10 * $Distance^4$

where $d_{crossover}$ = 295 ft.

When receiving data, Current = 9 mA.

Energy model analysis detail results are following:

Table A.1: Regression result for sample 1 - 15

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/27/09 Time: 18:25

Sample: 1 15

Included observations: 15

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	8.874057	0.039187	226.4529	0.0000
D2	2.94E-05	1.05E-06	27.94630	0.0000

R-squared	0.983627	Mean dependent var	9.680000
Adjusted R-squared	0.982368	S.D. dependent var	0.773859
S.E. of regression	0.102758	Akaike info criterion	-1.589310
Sum squared resid	0.137270	Schwarz criterion	-1.494904
Log likelihood	13.91983	F-statistic	780.9960
Durbin-Watson stat	0.806869	Prob(F-statistic)	0.000000

Current= 8.874057+2.94E-05 Distance*^{2} *(1)*

Figure A.1: Actual, fitted and residual graph for equation (1), observations 1 - 15

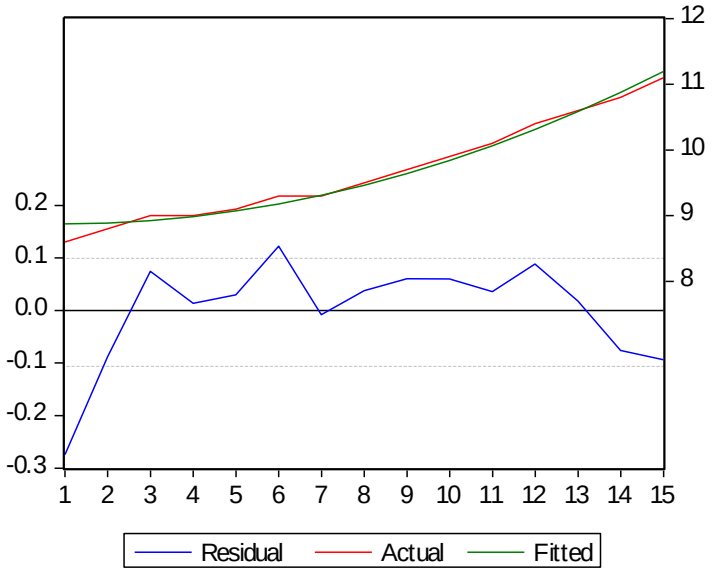


Table A.2: Regression results for sample 16 - 26 (1)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/27/09 Time: 02:05

Sample: 16 26

Included observations: 11

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DISTANCE	-6.969501	2.606651	-2.673737	0.0368

D2	0.027826	0.009946	2.797657	0.0313
D3	-4.90E-05	1.67E-05	-2.930968	0.0263
D4	3.22E-08	1.04E-08	3.088487	0.0214
C	661.7978	253.6843	2.608745	0.0402

R-squared	0.994526	Mean dependent var	17.46364
Adjusted R-squared	0.990877	S.D. dependent var	3.487771
S.E. of regression	0.333135	Akaike info criterion	0.942418
Sum squared resid	0.665874	Schwarz criterion	1.123280
Log likelihood	-0.183299	F-statistic	272.5280
Durbin-Watson stat	2.529155	Prob(F-statistic)	0.000001

*Current= -6.969501 * Distance +0.027826 * Distance^2 + -4.90E-05* Distance^3 + 3.22E-08 * Distance^4 + 661.7978* (2)

As shown in equation (2), the coefficient of *Distance*^{3} and *Distance*^{4} is extremely small. So we delete *Distance*^{3} and *Distance*^{4} from equation (2) and run regression again.

Table A.3: Regression results for sample 16 - 26 (2)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/27/09 Time: 02:12

Sample: 16 26

Included observations: 11

Variable	Coefficient	Std. Error	t-Statistic	Prob.

DISTANCE	-0.192032	0.050889	-3.773516	0.0054
D2	0.000301	6.34E-05	4.748985	0.0014
C	44.89195	10.01455	4.482675	0.0020

R-squared	0.964293	Mean dependent var	17.46364
Adjusted R-squared	0.955366	S.D. dependent var	3.487771
S.E. of regression	0.736851	Akaike info criterion	2.454137
Sum squared resid	4.343590	Schwarz criterion	2.562654
Log likelihood	-10.49776	F-statistic	108.0230
Durbin-Watson stat	1.440726	Prob(F-statistic)	0.000002

$$Current = -0.192032 * Distance + 0.000301 * Distance^3 + 44.89195 \qquad (3)$$

Figure A.2: Actual, fitted and residual graph for observations 16 - 26

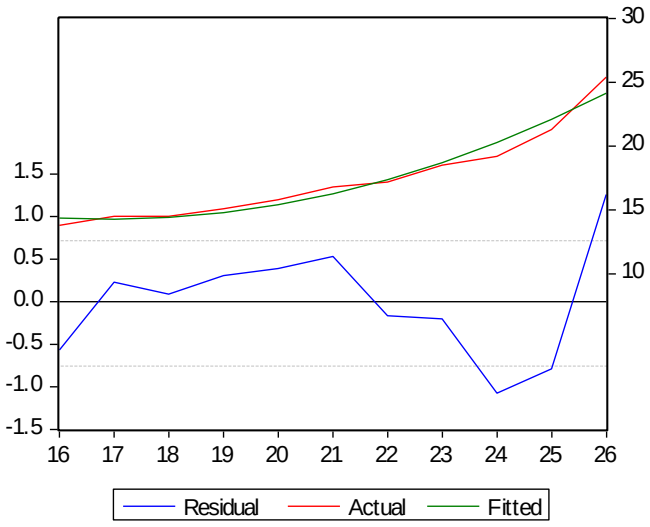


Table A.4: Regression results for sample 16 - 26 (3)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/27/09 Time: 13:35

Sample: 16 26

Included observations: 11

Variable	Coefficient	Std. Error	t-Statistic	Prob.
D2	6.24E-05	6.91E-06	9.037059	0.0000
C	7.209522	1.187204	6.072690	0.0002

R-squared	0.900737	Mean dependent var	17.46364
Adjusted R-squared	0.889708	S.D. dependent var	3.487771
S.E. of regression	1.158297	Akaike info criterion	3.294744
Sum squared resid	12.07487	Schwarz criterion	3.367089
Log likelihood	-16.12109	F-statistic	81.66843
Durbin-Watson stat	0.876901	Prob(F-statistic)	0.000008

$$Current = 7.209522 + 6.24E\text{-}05 * Distance^2 \qquad (4)$$

Constance value in equation (4) is close to that in equation (1).

Figure A.3: Actual, fitted and residual graph for observations 16 - 26

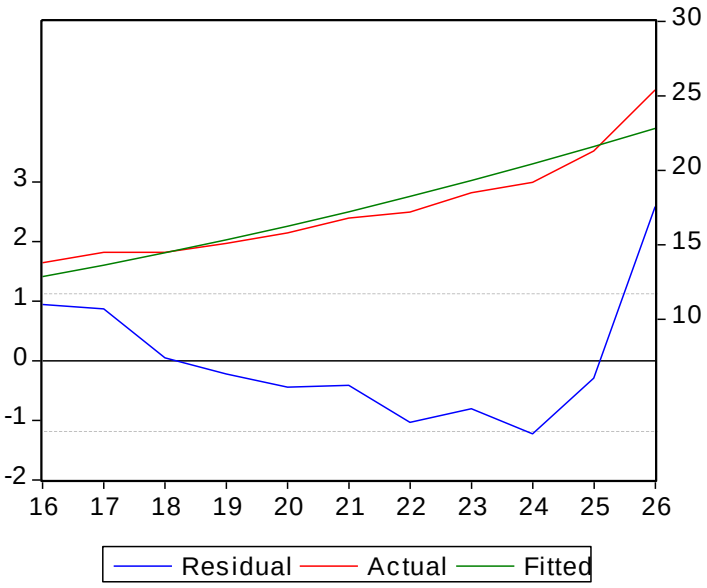


Table A.5: Regression results for sample 1 - 26 (1)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/27/09 Time: 14:17

Sample(adjusted): 1 26

Included observations: 26 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DISTANCE	-0.012715	0.003993	-3.184707	0.0041
D2	7.98E-05	7.69E-06	10.38772	0.0000
C	9.351729	0.432910	21.60199	0.0000

R-squared	0.972750	Mean dependent var	12.97308
Adjusted R-squared	0.970380	S.D. dependent var	4.536568
S.E. of regression	0.780764	Akaike info criterion	2.451080
Sum squared resid	14.02064	Schwarz criterion	2.596245
Log likelihood	-28.86404	F-statistic	410.5121
Durbin-Watson stat	1.055814	Prob(F-statistic)	0.000000

$$Current = -0.012715 * Distance + 7.98E\text{-}05 * Distance^2 + 9.351729 \qquad (4')$$

Figure A.4: Actual, fitted and residual graph for equation (4')

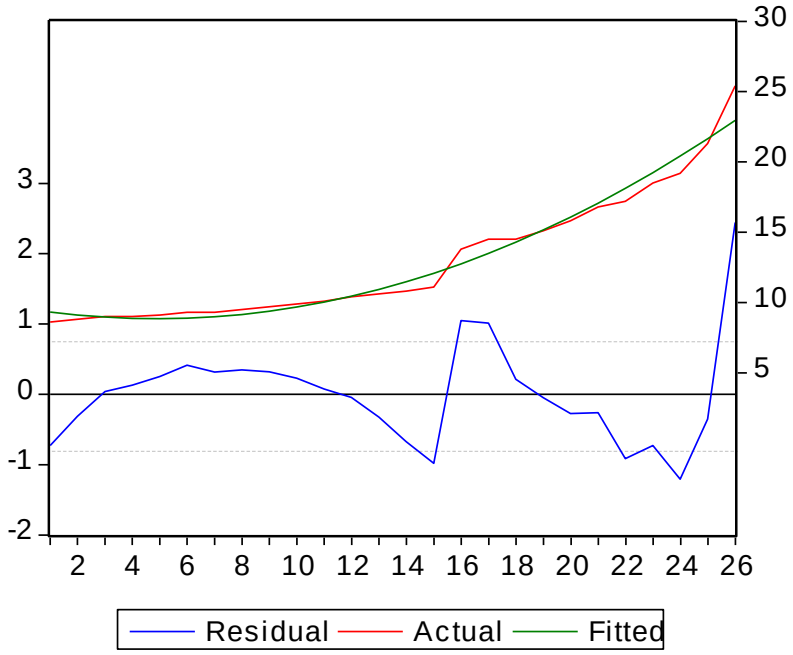


Table A.6: Regression results for sample 1 - 26 (2)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/27/09 Time: 14:21

Sample(adjusted): 1 26

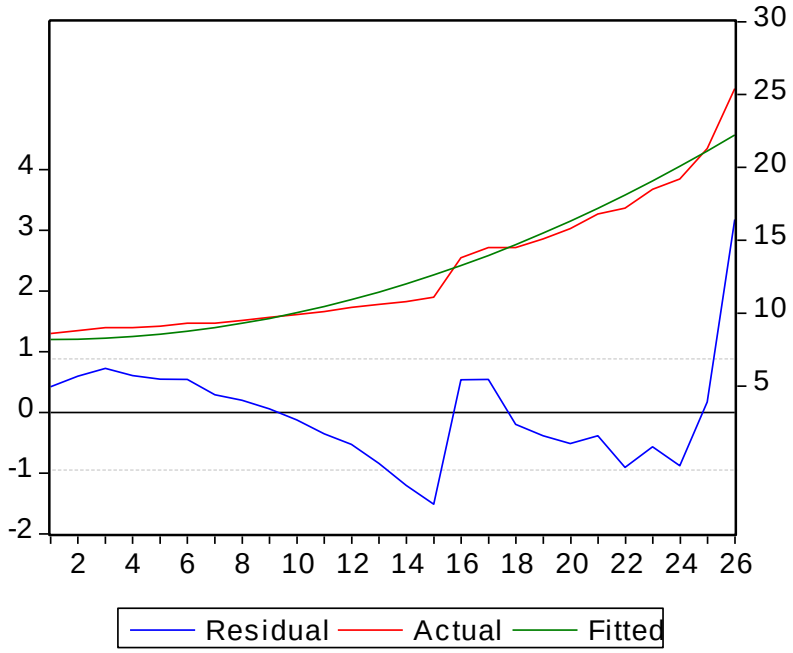
Included observations: 26 after adjusting endpoints

Variable	Coefficient	Std. Error	t-Statistic	Prob.
D2	5.62E-05	2.32E-06	24.23219	0.0000
C	8.178905	0.267430	30.58330	0.0000

R-squared	0.960733	Mean dependent var	12.97308
Adjusted R-squared	0.959097	S.D. dependent var	4.536568
S.E. of regression	0.917500	Akaike info criterion	2.739475
Sum squared resid	20.20335	Schwarz criterion	2.836252
Log likelihood	-33.61317	F-statistic	587.1991
Durbin-Watson stat	0.791991	Prob(F-statistic)	0.000000

$$Current = 8.178905 + 5.62E\text{-}05 * Distance^2 \qquad (4'')$$

Figure A.5: Actual, fitted and residual graph for equation (4'') (observations 1 - 26)



Since different current supports different distance, observation 17 and observation 18 have different Distance value but same Current value, one of them must be inaccurate. We then run regression with only one of them.

Table A.7: Regression results for sample 18 - 26 (1)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/28/09 Time: 13:45

Sample: 18 26

Included observations: 9

Variable	Coefficient	Std. Error	t-Statistic	Prob.

D2	7.13E-05	8.59E-06	8.299323	0.0001
C	5.410572	1.585311	3.412940	0.0112

R-squared	0.907748	Mean dependent var	18.20000
Adjusted R-squared	0.894569	S.D. dependent var	3.438023
S.E. of regression	1.116333	Akaike info criterion	3.251105
Sum squared resid	8.723390	Schwarz criterion	3.294933
Log likelihood	-12.62997	F-statistic	68.87876
Durbin-Watson stat	1.047353	Prob(F-statistic)	0.000072

$$Current = 5.410572 + 7.13E\text{-}05 * Distance^2 \qquad (7)$$

Table A.8: Regression results for sample 18 - 26 (2)

Dependent Variable: CURRENT

Method: Least Squares

Date: 07/28/09 Time: 14:37

Sample: 18 26

Included observations: 9

Variable	Coefficient	Std. Error	t-Statistic	Prob.
D3	1.13E-07	1.17E-08	9.663168	0.0000
C	9.428680	0.963641	9.784436	0.0000

R-squared	0.930263	Mean dependent var	18.20000
Adjusted R-squared	0.920300	S.D. dependent var	3.438023

S.E. of regression	0.970593	Akaike info criterion	2.971310
Sum squared resid	6.594352	Schwarz criterion	3.015138
Log likelihood	-11.37090	F-statistic	93.37681
Durbin-Watson stat	1.153578	Prob(F-statistic)	0.000027

Current= 9.428680+1.13E-07 Distance*^{3} *(8)*

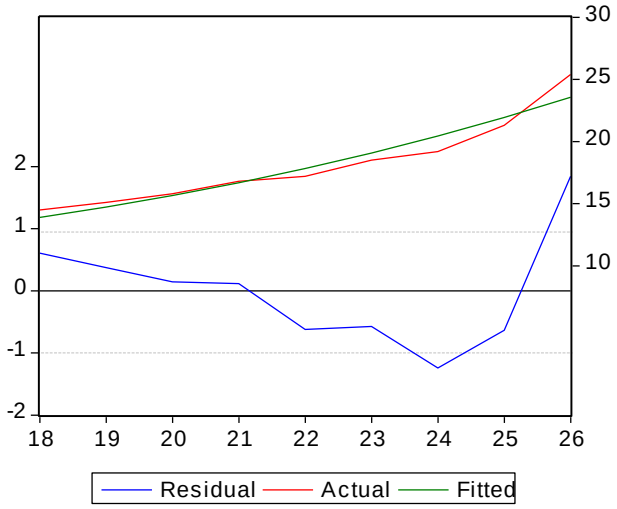


Table A.9: Regression results for sample 16 18 19 - 26

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 14:44

Sample: 16 25

Included observations: 10

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD2	6.49E-05	7.61E-06	8.528095	0.0000
C	6.704441	1.348810	4.970633	0.0011

R-squared	0.900902	Mean dependent var	17.76000
Adjusted R-squared	0.888515	S.D. dependent var	3.527416
S.E. of regression	1.177782	Akaike info criterion	3.341999
Sum squared resid	11.09736	Schwarz criterion	3.402516
Log likelihood	-14.70999	F-statistic	72.72841
Durbin-Watson stat	0.953031	Prob(F-statistic)	0.000027

Current= 6.704441+6.49E-05 Distance*^{2} *(9)*

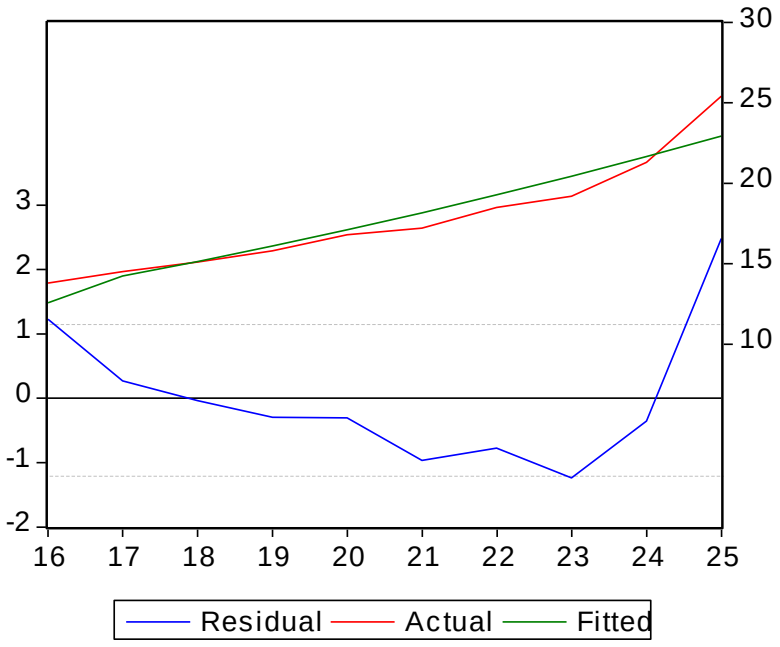


Table A.10: Regression results for sample 16 17 19 - 26

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:39

Sample: 16 25

Included observations: 10

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD2	6.25E-05	7.68E-06	8.136972	0.0000
C	7.187837	1.356103	5.300361	0.0007

R-squared	0.892198	Mean dependent var	17.76000
Adjusted R-squared	0.878723	S.D. dependent var	3.527416
S.E. of regression	1.228417	Akaike info criterion	3.426186
Sum squared resid	12.07207	Schwarz criterion	3.486703
Log likelihood	-15.13093	F-statistic	66.21031
Durbin-Watson stat	0.913696	Prob(F-statistic)	0.000039

$$Current = 7.187837 + 6.25E\text{-}05 * Distance^2 \qquad (10)$$

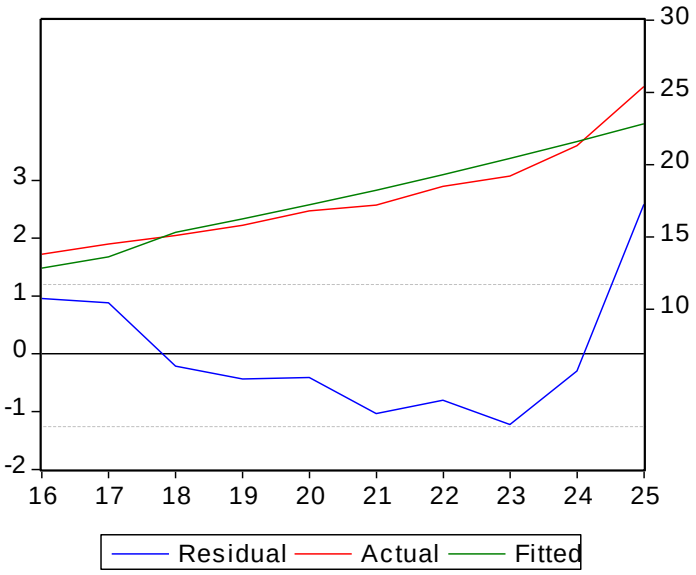


Table A.11: Regression results for sample 17 19 - 26

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:42

Sample: 17 25

Included observations: 9

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD2	6.70E-05	9.04E-06	7.413064	0.0001
C	6.287246	1.658922	3.789960	0.0068

R-squared	0.887012	Mean dependent var	18.20000
Adjusted R-squared	0.870871	S.D. dependent var	3.438023
S.E. of regression	1.235437	Akaike info criterion	3.453857

Sum squared resid	10.68414	Schwarz criterion	3.497685
Log likelihood	-13.54236	F-statistic	54.95352
Durbin-Watson stat	1.014403	Prob(F-statistic)	0.000148

Current= 6.287246+6.70E-05 Distance*^{2} *(11)*

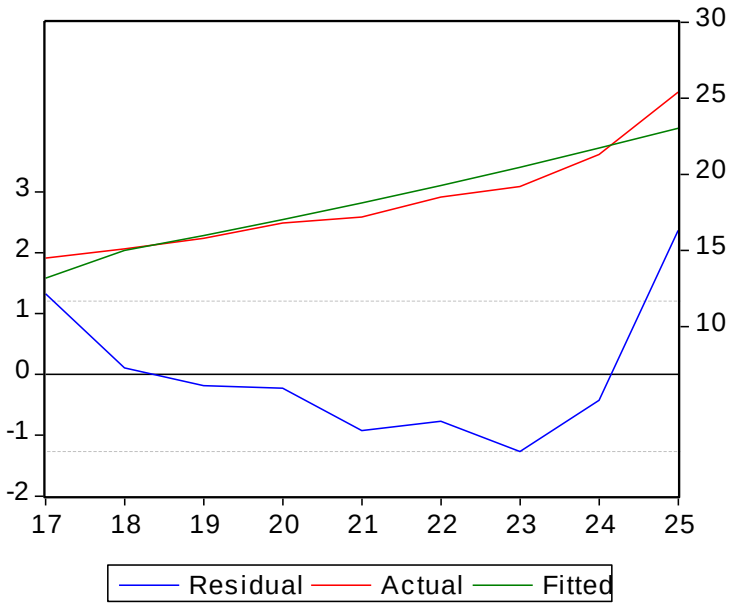


Table A.12, A.13 and A.14 show the regression results for a sample without observation 16, 17 and 18. After comparing the goodness-of-fit, we find that *Distance*^{4} has more power in fitting the data than *Distance*^{2} and *Distance*^{3}.

Table A.12: Regression results for sample 19 - 26 (1)

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:44

Sample: 18 25

Included observations: 8

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD2	7.60E-05	1.02E-05	7.457624	0.0003
C	4.432517	1.949739	2.273390	0.0634

R-squared	0.902623	Mean dependent var	18.66250
Adjusted R-squared	0.886393	S.D. dependent var	3.362795
S.E. of regression	1.133450	Akaike info criterion	3.300726
Sum squared resid	7.708247	Schwarz criterion	3.320587
Log likelihood	-11.20291	F-statistic	55.61615
Durbin-Watson stat	1.123844	Prob(F-statistic)	0.000300

$$Current = 4.432517 + 7.60E\text{-}05 * Distance^2 \qquad (12)$$

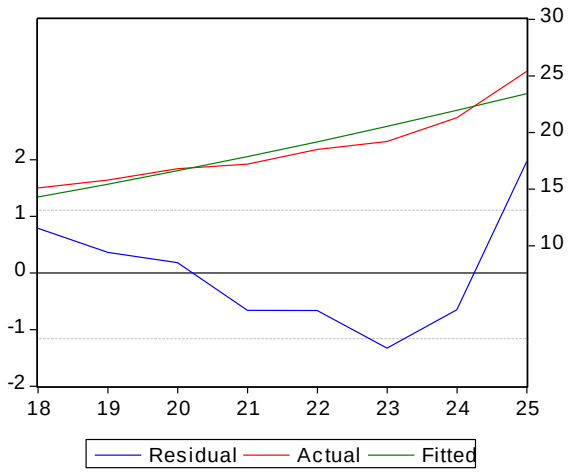


Table A.13: Regression results for sample 19 - 26 (2)

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:46

Sample: 18 25

Included observations: 8

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD3	1.18E-07	1.39E-08	8.508820	0.0001
C	8.948384	1.195651	7.484113	0.0003

R-squared	0.923469	Mean dependent var	18.66250
Adjusted R-squared	0.910714	S.D. dependent var	3.362795
S.E. of regression	1.004827	Akaike info criterion	3.059826
Sum squared resid	6.058067	Schwarz criterion	3.079687
Log likelihood	-10.23931	F-statistic	72.40001
Durbin-Watson stat	1.207599	Prob(F-statistic)	0.000144

Current= 8.948384+1.18E-07 Distance^3 (13)*

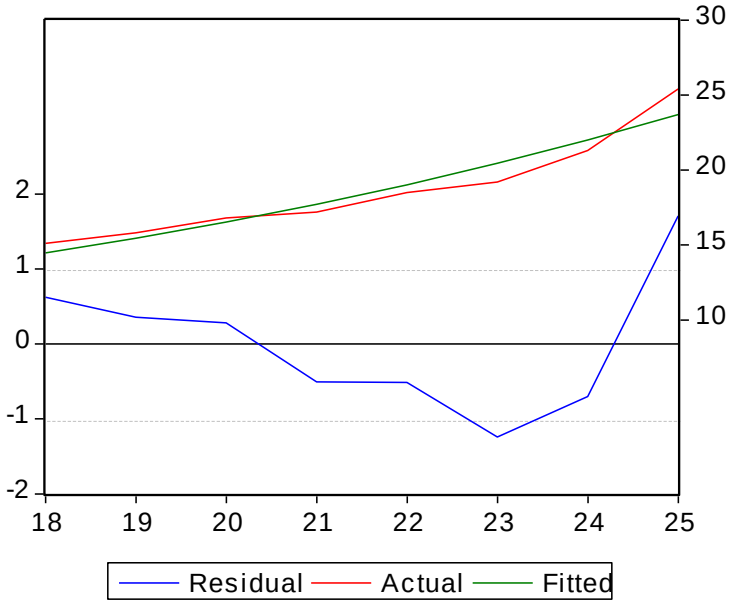


Table A.14: Regression results for sample 19 - 26 (3)

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:48

Sample: 18 25

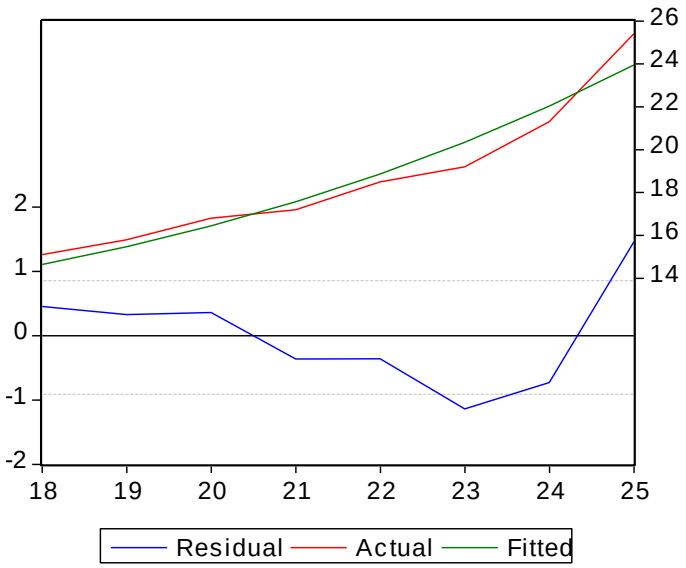
Included observations: 8

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD4	2.04E-10	2.08E-11	9.790270	0.0001
C	11.20248	0.823268	13.60732	0.0000

R-squared	0.941089	Mean dependent var	18.66250
Adjusted R-squared	0.931271	S.D. dependent var	3.362795

S.E. of regression	0.881597	Akaike info criterion	2.798155
Sum squared resid	4.663283	Schwarz criterion	2.818016
Log likelihood	-9.192621	F-statistic	95.84938
Durbin-Watson stat	1.310062	Prob(F-statistic)	0.000065

Current= 11.20248+2.04E-10 Distance*^{4} *(14)*



If we delete observation 17 and 18 from sample, as shown in table A.15 and A.16, *Distance*^{4} has more power in fitting the data than *Distance*^{2} and *Distance*^{3}.

Table A.15: Regression results for sample 16 19 - 26 (1)

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:51

Sample: 16 16 18 25

Included observations: 9

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD4	1.94E-10	1.72E-11	11.29463	0.0000
C	11.61814	0.643680	18.04955	0.0000

R-squared	0.947982	Mean dependent var	18.12222
Adjusted R-squared	0.940551	S.D. dependent var	3.538636
S.E. of regression	0.862798	Akaike info criterion	2.735857
Sum squared resid	5.210941	Schwarz criterion	2.779685
Log likelihood	-10.31136	F-statistic	127.5686
Durbin-Watson stat	1.381153	Prob(F-statistic)	0.000010

Current= 11.61814+1.94E-10 Distance^4 (15)*

Figure A.15: Actual, fitted and residual graph for equation (15)

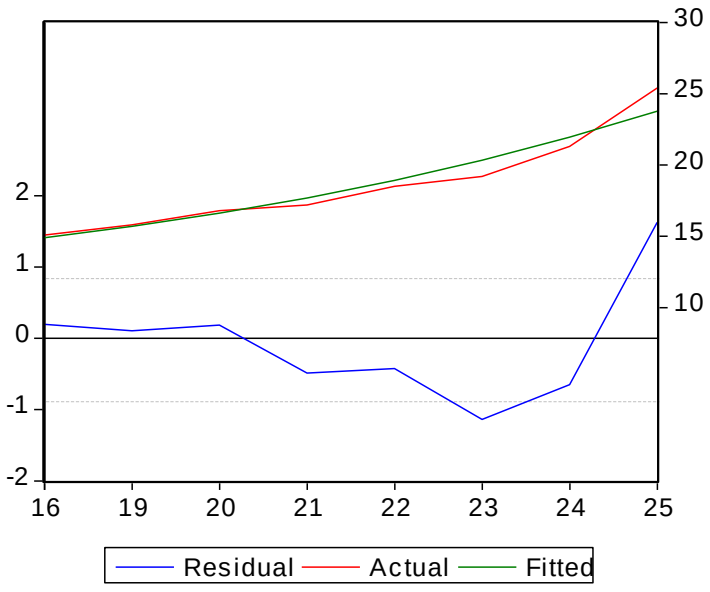


Table A.16: Regression results for sample 16 19 - 26 (2)

Dependent Variable: CC1

Method: Least Squares

Date: 07/28/09 Time: 16:54

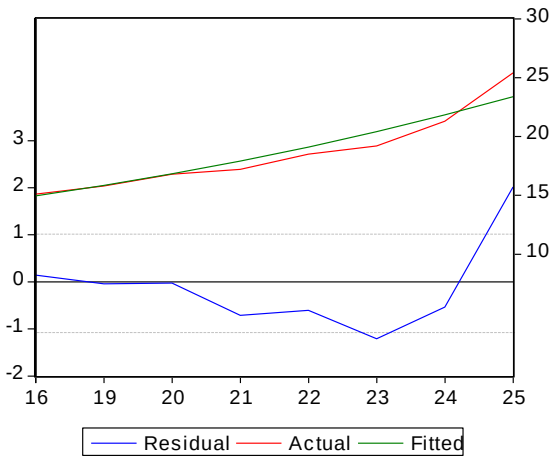
Sample: 16 16 18 25

Included observations: 9

Variable	Coefficient	Std. Error	t-Statistic	Prob.
DD3	1.08E-07	1.17E-08	9.208419	0.0000
C	9.901373	0.958261	10.33265	0.0000

R-squared	0.923743	Mean dependent var	18.12222
Adjusted R-squared	0.912849	S.D. dependent var	3.538636
S.E. of regression	1.044651	Akaike info criterion	3.118374
Sum squared resid	7.639077	Schwarz criterion	3.162201
Log likelihood	-12.03268	F-statistic	84.79498
Durbin-Watson stat	1.156482	Prob(F-statistic)	0.000037

*Current= 9.901373+1.08E-07*Distance*^{3} *(16)*



Appendix B

TinyOS LEACH Implementation Source Files

Some of the source code files of our LEACH implementation on the TinyOS platform are listed below:

Main Module:

Test_GroupBuilderM.nc

Cluster Head Selection Module:

ToBeSinkM.nc

Broadcast Module:

IDBroadcastM.nc

Cluster Setup Module:

GroupBuilder_memberM.nc (as a cluster member)

GroupBuilder_sinkM.nc (as a cluster head)

Time Synchronization Module:

Timesyn_memberM.nc (as a cluster member)

Timesyn_sinkM.nc (as a cluster head)

Data Transportation Module:

TransData_memberM.nc (as a cluster member)

TransData_sinkM.nc (as a cluster head)