SENSOR NETWORK DEPLOYMENT IN THE MCMASTER NUCLEAR REACTOR

SENSOR NETWORK DEPLOYMENT
IN THE MCMASTER NUCLEAR REACTOR

By

NICHOLAS MERIZZI, B.Sc. H

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (2005) McMaster University
Hamilton, Ontario
TITLE: Sensor Network Deployment in the McMaster Nuclear Reactor
AUTHOR: Nicholas Merizzi, B.Sc.H (McMaster University)
SUPERVISOR: Professor W.F.S Poehlman
NUMBER OF PAGES: xi, 123

Abstract

Lack of generality in deployment of *Wireless Sensor Networks* gives rise to many application specific research questions. In order to maintain safe levels of radiation, a Sensor Network can be used to provide greater flexibility within the McMaster Nuclear Reactor (MNR). Sensor Networks have conventionally been deployed in natural habitat areas, and to our knowledge, this is the first attempt to model its behavior inside a research reactor. This application specific scenario provides insight in determining the suitability of embedding Sensor Networks in nuclear reactors. Traditional networks have been designed to accommodate various applications. In this case, we believe that sensor networks, which serve a specific task, can be customized depending on the application. By tailoring a Sensor Network for the Nuclear Reactor, one will be able to maximize efficiency.

This thesis states a set of requirements for deploying a Sensor Network in the MNR. By using these requirements, the challenges surrounding Sensor Network communications were studied. These results are to provide McMaster's Health Physics department with the proper guidance if choosing to deploy such a network. The research defines the optimal MAC, and Sensor Network routing protocols for the reactor. The metrics used to determine optimality are reliability, latency, scalability, and lifetime. The approach to determine the suitability of sensor networks in the MNR is a discrete-event simulator called J-Sim. J-Sim is extended to simulate the various protocols that were studied in this research including One-Hop, Multi-Hop, LEACH, TDMA, and CSMA. Results indicate that a modified version of LEACH, called MNRLEACH, best suits the needs of the McMaster Nuclear Reactor.

Contents

List of Acronyns

ACA - Autonomous Component Architecture
ACK - Acknowledgement Packet
BS - Base Station
CD - Collision Detection
CDMA - Code Division Multiple Access
CH - Cluster-Head
CPU - Central Processing Unit
CORBA - Common Object Request Broker Architecture
CSMA - Carrier Sense Multiple Access
CTS - Clear-To-Send
DARPA - Defense Advanced Research Projects Agency
DCF - Distributed Coordination Function
DES - Discrete-Event Simulator
DML - Domain Modeling Language
DNS - Domain Name Service
DOC - Department of Communications
DSSS - Direct Sequence Spread Spectrum
EMI - Electromagnetic Interference
FCC - Federal Communications Commission
FDMA - Frequency Division Multiple Access
FHSS - Frequency Hopping Spread Spectrum
GUI - Graphical User Interface
IC - Integrated Circuit
IP - Internet Protocol
IFS - Inter Frame Spacing
IEEE - Institute of Electrical and Electronics Engineers
INET - extensible InterNetworking Framework
LAN - Local Area Network
LEACH - Low Energy Adaptive Clustering Hierarchy
MAC - Medium Access Control
MACA - Multiple Access with Collision Avoidance
MACAW - Multiple Access Collision Avoidance for Wireless
MCU - Micro-Controller Unit
MANET - Mobile Ad hoc NETwork
MNR - McMaster Nuclear Reactor
MNRLEACH - McMaster Nuclear Reactor LEACH protocol
PDA - Personal Digital Assistant
PCF - Point Coordination Function

OO - Object Oriented
OSI - Open System Interconnection
RF - Radio Frequency
RTS - Request-To-Send
TCP - Transport Control Protocol
TDMA - Time Division Multiple Access
T-R - Transmitter-Receiver
QoS - Quality of Service
WSN - Wireless Sensor Network

List of Figures

List of Tables

1 Introduction

Moore's law states that the number of transistors per square inch on an integrated circuit doubles every eighteen months. These rapid advances in hardware, with emerging wireless technologies, have enabled the development of low-cost, low-power, multifunctional sensor nodes. Traditional wireless networks function by connecting to some established access point that acts as a bridge between the wireless/wired worlds. The need to setup communication devices for possibly short-lived networks and networks with no fixed infrastructure is growing. A good example is the growth in demand for PDAs to be able to randomly connect for brief synchronization or a file transfer. These specific types of wireless networks are referred to as *Ad hoc* networks. *Ad hoc* is a Latin word which means unplanned, makeshift, or temporary [1]. These ad hoc networks are independent of any fixed infrastructure support and mathematically can be viewed as a constantly changing graph with possibly multiple components. The new challenge associated with ad hoc networks is determining ways that wireless mobile devices can perform network topology functions that are traditionally handled by routers. In other words, a key characteristic of an ad hoc network is that they must be capable of self-organizing themselves. This ultimately requires hosts to act as both routers and end nodes as efficiently as possible. Once nodes have discovered and created an ad hoc network, it must at all times be capable of allowing new nodes to join it or allow current nodes to withdraw from the network.

The ad hoc network space contains two major classifications based on the level of mobility and power constraints of the hosts [1]. Figure 1 positions ad hoc networks relative to other types networks and illustrates two of its subcategories. The first subcategory *Mobile Ad hoc NET*works (MANETs) are ad hoc networks that are made of highly mobile hosts. The main characteristics of these networks are their constantly changing topologies, which requires special attention in terms of routing algorithms. Examples of such networks are PDAs, laptops, and cell phones. The second classification of ad hoc networks are referred to as smart *Wireless Sensor Networks* (WSN) and have different characteristics overall. Traditionally, they are statically located over some geographical area and serve a specific purpose. These types of networks are extremely application specific and the nodes are normally energy constrained. This research will focus on the latter of the two, sensor networks.

1.1 Wireless Sensor Networks Overview

Previously, sensors were generally utilized in small quantities and were hard-wired back to a processing station for daily monitoring of some area. The research in sensor networks began in 1980 (during the Defense Advanced Research Projects Agency

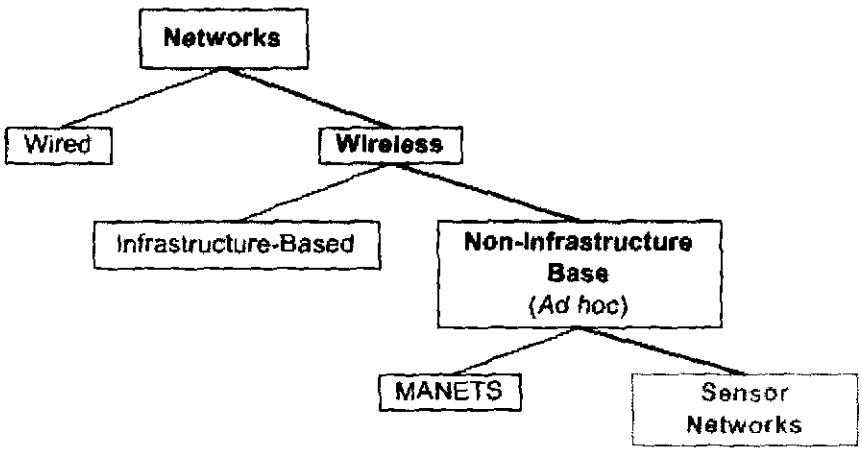

Figure 1: General Classification of Networks

(DARPA) period) with projects such as the *Distributed Sensor Networks* (DSN) and the *Sensor Information Technology* (SensIT) which are known as the first attempts to create sensor networks [2]. Today, the advances in micro-electro-mechanical systems (MEMS) and low power Integrated Circuits (IC) have led to the development of micro sensors with wireless capabilities. These sensors are able to perform a task (i.e. measuring weather, seismic activity, pressure, radiation, noise, light, etc..) and then eventually report back their findings to some base station. A network of such sensor nodes is called a Wireless Sensor Network (WSN). It is a tool for measuring a phenomenon in a given area, often times inhospitable areas such as volcanoes, combat zones, and disaster areas [3]. Figure 2 illustrates how sensors are randomly scattered throughout a sensing field. Each of these nodes has the capability to collect, and route the data back to the sink. This information is sent back via a wireless channel and from the sink may be communicated back to administration by Internet or satellite. These small sensors, often called *motes* (which is a term referring to a speck of dust [4]), collaborate together to present an accurate aggregate result of the sensing area [5]. The information is normally sent back via a low power radio transmitter to a base station using some defined routing protocol. With a wide range of sensors available today in the market, an increasing interest is being placed on determining suitable applications for these autonomous, disposable sensors. Efforts are being put towards determining how sensors should collaborate in both gathering and reporting back to a base station in order to minimize the energy cost, and yet maintain high reliability.

A sensor node is usually made up of four main components: a processing unit,

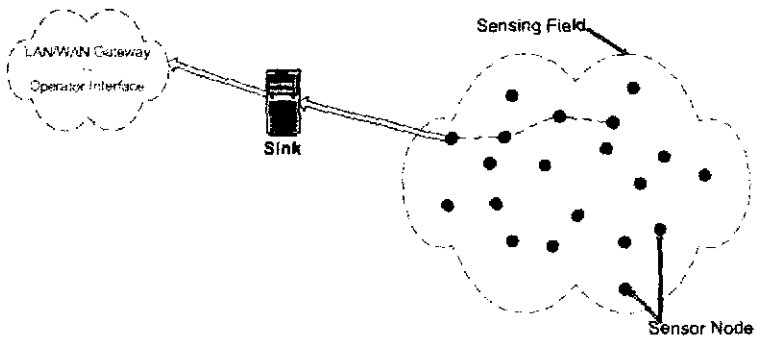

Figure 2: Sample sensor network scenario.

communication unit, sensor, and a power supply as shown in Figure 3 [6, 5]. With the developments in the three required technologies of sensor networks, (sensing, communication, and computing) one is now capable of integrating all the functions of a sensor onto a 5 mm^2 chip [4]. The processing unit normally consists of a micro-controller unit (MCU) in the range of 1-24MHz with 1Kb to 4Mb of on-board memory. The MCU provides the intelligence and is usually responsible for controlling when to sense the environment and has control over the radio component. The choice of MCU is generally determined by the application at hand, but one must be aware that the choice of MCU can drastically impact the node's power dissipatation. For example, MCUs such as Intel's StrongARM consumes 400mW of power while executing instructions, whereas Amtel's ATmeg103L ARM micro-controller only consumes 16.5 mW [7]. The communication component (the radio) essentially consists of a short-range radio which allows the sensor to communicate its data back to the base station. The radio is capable of running in one of the following states: transmit, receive, idle, sleep, and off. The energy source is normally a coin-like 3-4.5V battery (sometimes double A) with a capacity ranging from 1700mAh-2700mAh [1]. The sensor that is attached will vary depending on the application at hand. There exists many various types of sensors that can measure the environment, such as: temperature, light intensity, sound, magnetic fields, and image. An optional fifth component is an interface to allow the sensor to act as an actuator to a third party. A sensor functions by gathering sensed data locally and periodically communicating that data back to the base station. If high currents are drawn for a prolonged period of time from any of the above four components, the sensor will die quickly. Design methodologies need to be developed in order to maximize the lifetime and hence reduce the energy requirements. This means that the on-board operating system, the application layer, and the network protocols must all be designed to be

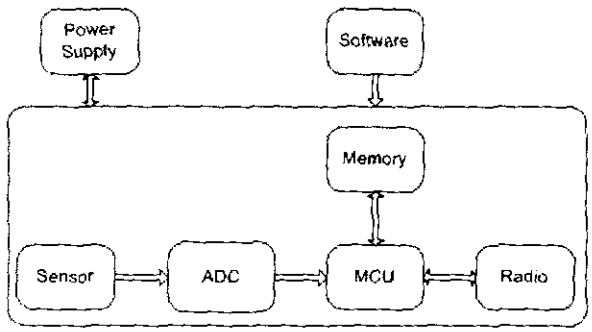

Figure 3: The internal components of a typical Sensor node

energy efficient. If attempting to prioritize the management of energy, one would place the cost of using the radio very high. The energy cost of a transmission (in an open area) is proportional to the distance squared, therefore, radio communication consumes a lot of energy during operation. To optimize the levels of wireless transmissions, specific protocols have been developed to reduce cost; some of these will be discussed in Section 2.2.

1.2 Sensor Network Applications

In the past, many environments were extremely challenging to monitor. This is mainly because they provided no infrastructure for either energy or communication. With a WSN we are able to step away from this limitation and only be bounded by environmental obstacles that impede Radio Frequencies (RF) emitted from the sensors. The concept of wireless sensor networks can be summarized by a simple equation [8]

$$\text{Sensing} + \text{CPU} + \text{Radio} = \text{Thousands of potential Applications}$$

We can classify these various applications into three main classes: environmental data collection, security monitoring, and sensor node tracking [8]. Applications that fall into *environmental monitoring* are ones wanting to collect sensor readings at various points in an environment over a period of time. The second class, *security monitoring*, are nodes placed at fixed locations that continually monitor for abnormalities (i.e. motion detectors). The last category, *tracking*, occurs when sensors are used to track the location of a specific object.

The wide range of applications for sensor networks is what is pushing the research forward. Commercial, residential, and military can all significantly benefit from this

technology. Below are specific examples that fall into one of the three application classes discussed above [5, 3, 9, 8, 10, 11, 1, 2]:

- In habitat monitoring, sensor networks have been used for monitoring environmental zones, biospheres, ocean activity, and even wildlife habitats. Weather forecasting and other natural events such as seismic movements, volcanic eruptions, and tornados have all used Sensor Network technology. Disaster monitoring, and prevention, of such natural forces is also a key area where WSN can aid in determining levels of threat.

- Smart Home/Office buildings can be built around sensor nodes that can monitor individual preferences for various environmental states such as humidity, temperature, lighting etc. Sensor networks can also be used to monitor the building infrastructure itself. So in areas of high seismic activity, any building irregularities can be reported to take preventative measures to avoid collapse.

- Military Communication networks, and specifically, ad hoc and sensor networks will become a key element to Future Combat Systems (FCS). By covering war grounds with sensors, the opponent's activity can be monitored from a safe distance. Other research such as shooter localization has been investigated where nodes sense the shock-wave and muzzle-blast that was given off by the shot to calculate its origin.

- Industries can also benefit from sensor networks. For example, retail can use sensor networks to maintain inventory. Paper Mills, or any other industry based on heavy machinery, can use sensors for diagnostics and preventative measures. For example, rolling machines in pulp and paper mills are very complex machinery and the slightest variations in speed, temperature or alignment of the rollers can have serious effects on production. Another example is the aircraft industry where sensing the noise level given by the engine and then counter-acting it by producing counter vibrations, can cancel outer noise.

- Many health applications are possible for sensor networks. For example, integrated patient monitoring, diagnostics, drug administration in hospitals, tracking and monitoring doctors and patients inside a hospital. Hospitals' current means of locating personnel and patients, are through intercom systems. By attaching sensors to doctors or patients their positions can be obtained more efficiently. Other possible opportunities that can be tracked and sensed by sensors include heart rate, oxygen saturation, end-tidal CO_2, and serum chemistries measurements, including serum glucose, with small sensors [4].

- In traffic control, sensors have been used at intersections either overhead or buried beneath the road. These sensors help in monitoring traffic and optimizing light changes accordingly. With the decrease in cost, and improvements in the technology, deployment of large scale sensors will be used for everything from video surveillance, to daily vehicle counts, and estimates of speeds with which people are driving into intersections. Other more advanced visions include integrating sensors into each vehicle and as vehicles pass each other they can exchange information. This information may consist of traffic jam locations, capable travelling speeds, traffic densities, and various other information like possible construction zones [12].

Clearly, there is a wide range of applications for sensor networks each of which require an application-specific model to maximize efficiency.

1.3 Challenges

Regardless of the application of a Sensor Network they all have a set of common operational challenges that require specific attention [6]. Specifically [5, 8, 9, 2]:

- Untethered Devices: This means that each individual sensor is disconnected from the world. This constraint requires energy efficiency to be studied carefully in order to maximize its use.
- Autonomous: Sensor Networks must function unattended, which means that maintenance and discovery must be done with minimal supervision.
- Adaptation: With the possibility of using sensor networks in an in-hospitable environment, one must realize that nodes can fail, die, or even be damaged. This threat means that nodes must adapt to topology changes as efficiently as possible.
- Ad hoc Deployment: There can either be deterministic placement of the nodes determined a priori or a random placement. Most scenarios do not have any fixed infrastructure (i.e. ocean, dense forest) and deployment can consist of simply throwing sensors from an airplane. After such a random deployment, it is up to the nodes to advertise their presence, establish connection, and eventually maintain it.
- Scalability: Traditional wired and wireless networks consists of 10's or even 100's of nodes. Sensor networks are normally deployed to study a phenomenon on the order of hundreds or thousands. This is again very application specific, but it can reach extremes as large as millions of nodes. Therefore, new models need

to be proposed to exploit such dense networks. The density can vary significantly with perhaps just a couple of nodes to a few hundred nodes in a radius of 5 meters.

- Cost: As stated, a Sensor Network consists of a large number of nodes, therefore the cost of a single node is very important to justify the overall cost. In order to see growth in deployment of such networks, the cost of individual nodes needs to be kept low. Today's technology allows a Bluetooth radio system to cost less then $10 [6]. To make the cost feasible, the target price should be closer to US$1 which makes reducing cost a challenging aspect to sensor networks.

- Diagnosing: Just taking into account the sheer number of sensors that is normally deployed makes it extremely difficult to pay attention to individual nodes. Traditional network monitors cannot be used any longer because no global naming scheme is present for these types of networks. Instead, in order to gather network statistics, such as power readings, location, communication patterns, one can inject queries and wait for responses.

- Security: In wired networks, one can easily secure the transmission medium and control the access points. This is not the same scenario for a wireless environment and even worse in a wireless Sensor Network environment. This is largely due to the fact that the medium is available to anyone within an acceptable communication range. Hence, to obtain security for sensor networks the use of encryption is normally required which comes as an expense due to the extra overhead and slower performance [13]. Since the environments in which sensors often operate is open security should be built into the design with high importance and not as an afterthought.

Therefore, several critical aspects exist and must continue to be studied in order for this technology to have future potential. In order to obtain efficiency, robustness, and scalability research in sensing, wireless communications, and low energy cost CPUs must continue to overcome the above challenges.

1.4 Thesis Contributions

McMaster's Nuclear Reactor (MNR) has many current means of obtaining accurate readings of radiation levels and verifying operator exposures. This research was geared to determine a new approach that can be layered on top of their traditional techniques in order to provide greater flexibility in monitoring radiation levels. The first contribution of this work was to identify the challenges and requirements of deploying such a network in the McMaster nuclear reactor, which to the best of our knowledge had not been

investigated elsewhere. Secondly, the research suggests a Sensor Network paradigm that will maximize lifetime while providing a certain degree of reliability for deploying such a network within the MNR. The protocol that best suits the requirements is a modified version of the Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol, which we called MNRLEACH. The work was tested using a discrete-event simulator (DES) called J-Sim where many new contributions were made to the open-source simulator. These contributions provide numerical results to specific protocol performance and allow for proper comparison amongst protocols.

1.5 Organization

This thesis consists of eight chapters and three appendices. Chapter 2 begins by providing general background knowledge into routing for sensor networks including a discussion on both MAC, and network layer paradigms. This chapter also looks into two conflicting models for wireless protocol stacks: the cross-layered design versus the traditional layered model. In addition an introduction on localization techniques for sensor networks is given at the end. Chapter 3 describes the McMaster Nuclear Reactor environment by explaining why sensors are appropriate for monitoring radiation, and the unique challenges involved in this application specific problem. This is followed by a chapter on J-Sim the simulator which was used for this research. This chapter explains J-Sim's architecture as well as the current Sensor Network framework that is available with its distribution. Chapter 5 explains all theoretical models that were used for these simulations. These models include radio propagation models, energy models, and radiation propagation models. Chapter 6 shows the simulation results of our research by comparing the issues of energy, latency, scalability, and reliability. The thesis concludes with Chapters 7 and 8 which provides a discussion on the results and future continuations of this research.

2 Wireless Sensor Networks

The deployment of a Sensor Network in a nuclear reactor demands the flexibility of being untethered while maintaining the same quality of service of a wired sensor network. One of the most challenging issues when attempting to deploy a Sensor Network in a specific environment is to determine the appropriate routing paradigm. This challenge is mainly due to the limitations and potential problems that can arise using a wireless channel. Therefore, to design an appropriate paradigm for the reactor, one needs to overcome the wireless limitations which include [14, 5]:

1. **Limited Channel Bandwidth** - The wireless channels are supervised by the Department of Communication (DOC) in Canada (or the Federal Communications Commission (FCC) in the US) which regulates frequency and bandwidth at which particular nodes can operate. Therefore, this limits the amount of bandwidth that can be used and hence efficient protocols must be developed to take advantage of these limitations.

2. **Limited Node Energy** - These are untethered devices and are normally self-powered by an attached battery which makes it very important to always minimize energy dissipation to maximize the lifetime of the sensor.

3. **Electromagnetic Wave Propagation** - These are short-range radios that experience very fast attenuation due to their small signal strength. Generally speaking an electromagnetic wave can be distorted quite easily by the environment in many ways. Inside the reactor, if lead blocks are moved for research purposes between a sender and a receiver it can cut off the line of sight and possibly stop communication from occurring.

4. **Non bi-directional traffic** - normally traffic is bi-directional in other types of networks. Here the traffic generally tends to all go in the same direction (towards the sink). This can put a heavy burden, and lead to large scale collisions near the base station. This necessitates the development of new MAC protocols, optimized for the application at hand.

To deploy sensors in the McMaster Nuclear Reactor these wireless challenges will need to be overcome. By developing reliable MAC protocols and energy-efficient network layer routing it is possible to extend the lifetime of a sensor and hence overcome problems 1 and 4. To keep tight control over power consumption new cross-layered architectures, where high level layers will communicate directly to their hardware, will be required. This will allow radio transmitters to be shut off when not needed and set to

send only as strong a signal as needed for the receiver to hear (addresses problem 2 and 3).

It is not the purpose of this chapter to make any particular method superior to another, but rather provide background knowledge into the concepts that were used to determine the optimal routing paradigm for the Reactor.

2.1 Methods of Evaluation

Having discussed possible applications and challenges linked with sensor networks, one must also determine the appropriate performance metrics which can be used in evaluating such networks. By keeping in mind the overall objective, which is to sense a well-defined area and report back findings to a base station, evaluation methods can determined. The key metrics for evaluating Sensor Network protocols as a whole are lifetime, coverage, latency, fault-tolerance, scalability, and security [3, 8]. Due to the nature of sensor networks these metrics are often interrelated. For example, by increasing security (i.e. including encryption) one has now increased the latency of each transmission due to the new overhead.

- *Lifetime,* for the majority of systems, would be classified as the most crucial indicator of the usefulness of a Sensor Network. By definition, it is the expected lifetime of the network as a whole. Since many applications involve a tremendous amount of nodes the time required for weekly or even monthly replacements would be unacceptable. Also, many ad hoc deployments of these networks are unreachable to humans. So the feasibility of replacement is not an option. It would be possible in some applications to have some sensors electrically wired to take advantage of external power. However, since ease of deployment is one of the key characteristics of such networks electrically wiring sensors in great quantities would remove the flexibility of such a network.

- *Coverage* which by definition is how well a sensing field is monitored or tracked by sensors [15] is another key metric which is of interest to many. Oftentimes a larger sensing area will want to be monitored depending on the application at hand. This requires great flexibility in the protocol's ability to scale. Other application specific scenarios that require a certain degree of fault-tolerance might demand a pre-determined density level in all sensing areas. This would require extra sensors to be placed which increases cost and traffic in the network. Therefore, proper analysis must be made to determine if the appropriate coverage demanded is being satisfied.

- *Latency* is the amount of time a user at the base station must wait until receiving the updates from sensors across the network. Latency will vary depending

on the in-house aggregation being done by individual sensors, and by the routing scheme. Multi-Hop schemes will generally increase the overall latency. In particular alarm applications, and disaster monitoring applications must have very quick response times.

- *Fault-tolerance* means how susceptible the network is to changes. A Sensor Network topology can vary over time significantly. Individual sensors have the risk of failing at any given point which means that gaps in the sensing field might occur. Another possibility is a variable environment which could have an affect on the wireless channel, or simply a node's battery lifetime. Various factors will affect the topology and hence the sensor networks must be fault-tolerant such that non-catastrophic failures are hidden from the application. Possible solutions for increasing levels of fault-tolerance is to perform data replication, or to use Acknowledgement (ACK) packets. The trade-off of both of these solutions is that they increase the already limited energy requirements.

- *Scalability* for sensor networks means performance degradation is not a result of adding more sensors to the network. With varying size fields, and changing environments, network protocols for such situations must be able to adapt. For example, a network might only be sensing one phenomenon, say temperature, but the users now would like to also monitor pressure. This would mean additional sensors would have to be deployed throughout the field scaling the network. For such large-scale networks the goal will be to localize traffic and perform data aggregation in order to reduce global collisions and large inflows of packets near the base station.

- *Security* of any open wireless transmission medium is always an issue. In the case of sensor networks large amounts of information regarding the status of a specific environment is being sent over a wireless medium. Although information such as temperature and light seems harmless, the accumulation of such information over time can lead to saboteurs attempting to exploit correlations in the data to determine patterns of weaknesses. Encryption and cryptography are obvious solutions to the problem but are costly in terms of both bandwidth usage and power consumption. For every packet, extra computation must be done to encrypt at the sender's end and then decrypt at the receiver.

2.2 Routing Architecture

Traditional routing has always been built around the concept of layers such as the four-layer TCP/IP model or seven-layer OSI model. Recent research shows that such a

layered model may not be the optimal solution for a Sensor Network [16, 17, 18, 19].

In the traditional layered model, the goal is to provide a modular design so that the protocol stack (i.e. OSI model) can adapt easily to specific hardware and operating system environments. Similar to other modern computer networks, a Sensor Network architecture can be organized as a series of layers, each one built upon its predecessor. The goal of each layer is to offer certain services to the higher levels while hiding its details and the details of the lower layers. A common protocol stack which can be used by sensors is shown in Figure 4 [6]. At the top is the *application layer* that is normally heavily dependent on the environment for which the sensor will be used. The *transport layer* is in charge of maintaining the flow of data when the application level requests it. The *network layer* is for upholding the routing of all data passed down to it by the transport layer. The *data link layer* is where MAC protocols control who has access

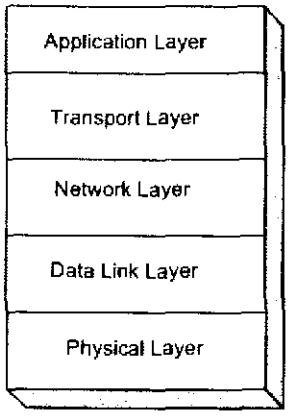

Figure 4: Traditional layered Protocol Stack.

to the shared communication channel that the sensors utilize. The last layer, *physical layer*, is in charge of modulation, transmission, and receiving the data.

This layered approach provides a clear separation of tasks and allows for independent design. For example, one can now implement a network layer relay (By definition a network layer relay is a function within the network layer which allows one network entity to forward data to another correspondent network entity) without being concerned with the upper layer protocols which a client may use. The downfall of this model is that it does not take into account the specific requirements of sensor networks. First, in WSNs we are generally concerned with developing complete end systems that normally flattens the layered structure. By integrating protocol data manipulations, one can combine a group of layers into a single pipeline to access data more efficiently (i.e. avoiding

the expensive costs of load and store operations at each layer see [16]).

To overcome the above downfalls, one must develop a reusable paradigm where application and domain knowledge are required at all layers. It can be seen that data traffic is largely based on higher layers such as data aggregation, user queries, and time synchronization [19]. It is therefore evident that a model, where more control is granted to the application layer, would be beneficial. This approach is often referred to as a *cross-layered design* where information can be shared between any combination of layers depending on the task for which it is being used. For example, allowing the application layer to directly communicate with the lower layers can lead to greater efficiency. A scenario which illustrates this advantage is in power management [18]. An application may be required to activate an 18 volt power source for a vibration sensor or maybe shut off the radio electronics immediately after a transmission. Recently there has been many suggested application-controlled routing protocols to expose the hardware layers to the requirements of the application. For example, TinyOS provides low-level hardware control through the use of a component model that eliminates the need for layers [18]. The downfall of flattening the layered model to this cross-layered design is that now Sensor Network software will largely be non-portable and non-reusable.

This research will test both approaches: traditionally layered and modern cross layered to determine which is best suited for the task at hand.

2.3 MAC Protocols

Sensor Networks attempt to modulate their information over some carrier signal and then transmit it back to the base station. Since sensors listen and transmit on a shared wireless transmission channel, a major obstacle is to reduce collisions. A collision is the result of two nodes sending data at the same time over the same channel [20]. *Medium Access Control* (MAC) protocols have been developed to aid in preventing collisions by using various techniques. MAC protocols are developed at the low level of the sensor protocol stack. It is located in the MAC layer which is normally considered a sub-layer of the data link layer. The MAC protocol for wireless sensor networks has two main objectives [6, 5]. The first is to create the network infrastructure. Since this layer is responsible for detecting incoming signals, error control, and sending information across the channel, it therefore needs to be able to establish communication links amongst high density sensor networks. Secondly, this layer has to fairly and efficiently share the transmission channel amongst all nodes. Sharing this channel normally falls into two categories: *scheduled protocols* or *contention based* [21].

Scheduled protocols (such as Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), Code-Division Multiple Access (CDMA), and Space-Division Multiple Access (SDMA)) avoid interference by scheduling nodes onto differ-

ent sub-channels that can be divided either by time, frequency, or orthogonal codes [22]. Due to these divisions scheduled protocols are collision free but generally not flexible enough for scaled situations. These methods normally require some global knowledge or control of the system. They are also inefficient if not all nodes have data to transmit since scarce resources are being allocated but not used [21].

The second class of MAC protocols, *contention based* (such as Carrier Sense Multiple Access (CSMA), Multiple Access Collision Avoidance for Wireless (MACAW) [23], Multiple Access with Collision Avoidance (MACA) [24], and IEEE 802.11 [25]), nodes compete for the shared medium. Unlike the scheduled based algorithms, the contention based protocols suffer from possible collisions of data since all nodes compete for the channel. These randomized protocols have the advantage of being simple and require no knowledge of the network topology or global control of the network. The downfall for using such a method with sensor networks is that, in very dense networks, the probability of collisions greatly increases. The other drawback is that these protocols normally assume that the radio is always on and listening to the channel. This is an expensive task and for untethered devices such as sensors their lives will be short lived.

Many observations can be made that separate the properties and challenges of contention based media access in wired LANs versus wireless LANs [24, 23]. First the contention occurs at the receiver and not the sender. This means that CSMA on its own will often be in-accurate. Secondly, congestion is location dependent which is highly dependent on the placement of the nodes. Third, collaboration amongst nodes in order to learn about congestion levels should be used rather then having each node independently determine the information. Fourth, for increased accuracy, synchronization information regarding media usage should be propagated across the network.

Due to the opportunity cost of each paradigm (scheduled versus contention based), oftentimes a hybrid approach is used by combining both contention based and scheduled based protocols together [14, 26]. The goal of such an approach would be to maximize throughput, minimize energy dissipation, and allocate as fairly as possible the medium amongst all nodes.

Two existing radio based infrastructures can be identified and shown why their MAC protocols are not suitable. First, in a cellular system, base-stations form a wired backbone and all mobile nodes are always a single hop away from a base station. In this paradigm, the main focus of the MAC layer is to provide high quality of services (QoS), and bandwidth allocation [6]. This makes energy consumption an un-important factor since base stations are permanently powered, and cell phones can easily be recharged. A second example is the Bluetooth and Mobile Ad hoc Network (MANET) technologies which closely resembles sensor networks. MANETS traditionally are highly mobile stations which can easily be recharged. Therefore, a MAC protocol in a MANET will be geared at maintaining the network infrastructure despite high mobility. In Bluetooth,

short-range wireless networks, referred to as piconets, are formed with the goal of replacing cables between electronic devices. Each Bluetooth piconet forms a centrally assigned TDMA schedule as a MAC protocol and utilizes FDMA in order to reduce collisions. Sensor networks differ from the above networks for multiple reasons. First, nodes are normally disconnected from the world and therefore are severely energy constrained. Secondly, unlike other networks sensor networks are traditionally very dense ranging in the magnitudes of tens of thousands. Third, the traffic is generated by periodic sending hence it is very sporadic.

The following sub-sections provide an overview on four approaches which were examined for this research including CSMA, TDMA, CDMA, and IEEE 802.11.

2.3.1 Carrier Sense Multiple Access - CSMA

Carrier Sense Multiple Access (CSMA) is a contention based protocol that allocates the channel on demand. Here every station senses the channel before transmitting; if the station is idle it transmits, otherwise it defers transmission [23]. Many variations of CSMA exist including non-persistent, 1-persistent, and p-persistent. These various flavors of CSMA are different by the way they defer before re-transmitting. For the purposes of the simulations in this research only non-persistent CSMA was used. In a 1-persistent CSMA network, when a node attempts to transmit, it keeps listening until the channel is free and then transmits its message [27]. In contrast, non-persistent CSMA senses the channel to see if it is free and, if so, it will send immediately. On the other hand, if the channel is busy, the sensor will back off for some random amount of time and try again when the timer expires. When sensing the channel, a sensor will attempt to calculate the signal strength in its general area to determine if its idle or not.

CSMA is dynamic and adaptable protocol which attempt to minimize collisions and requires no scheduling. However, by itself CSMA does not address some of the major obstacles associated with wireless media access such as the *hidden terminal problem*. As stated in Section 2.3 the contention occurs at the receiver and not the sender in wireless media which CSMA does not address. This has lead to a problem known as the *hidden terminal*, which occurs when a sensor is outside the transmission range of another sender and therefore leading it to think that the channel is idle and proceeds to sending (see Figure 5). For example, consider the scenario depicted in Figure 5. Here *radio A* can hear *radio B* but not *radio C*, and *radio C* can hear radio *B* but not *A* (hence by symmetry *B* hears both *A* and *C*). Now assume that *radio A* is transmitting information to radio *B*. If *C* is now ready to transmit, according to CSMA it will sense its channel and think the channel is idle and thus commence transmission. This clearly leads to collisions at *radio B* due to a hidden terminal (in this case *radio A* is hidden to *radio C*). Therefore, both signals will be received by any sensor that is mutually part of

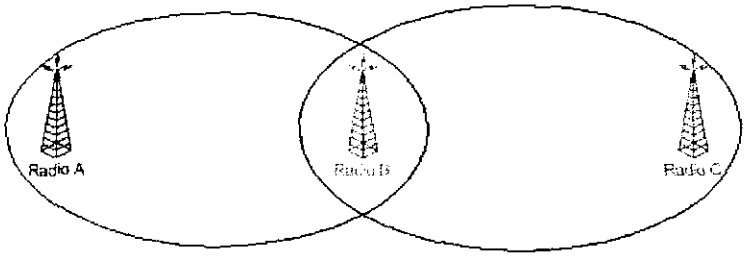

Figure 5: Hidden Terminal Problem

both transmission ranges.

Another scenario which can occur is known as the *exposed terminal* problem. Looking at Figure 5 again to illustrate this phenomenon assume now that *radio B* is transmitting to *radio A*. When *radio C* is ready to transmit it will sense the media and determine that it is busy and hence delay transmission. This is a flawed decision as there is no reason why *C* can not transmit to another station (with the exception of *radio B*). This shows how *radio C* was not given the proper information in order to make its decision.

Another drawback to CSMA is that it must also listen to the channel therefore loosing energy receiving packets that are for other destinations (known as the overhearing problem) [21].

2.3.2 Time Division Multiple Access - TDMA

Time Division Multiple Access (TDMA) is a scheduled based protocol that divides a channel into *X* time slots as shown in Figure 6. Sensors are granted access to the

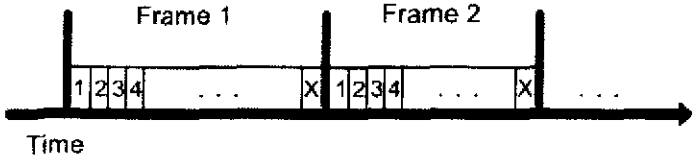

Figure 6: Sample TDMA frames are which are subdivided into *X* slots.

complete available spectrum, but they are assigned specific time intervals during which they are allowed access to it. Assuming there are *X* sensors that wish to transmit data, then each of these sensors is allocated a specific time slot. The *X* slots make up a frame

and these frames are continuously repeated. TDMA was mainly designed for a point-to-multipoint architecture which consists of multiple sensors communicating directly back to a base station (no peer-to-peer communication). If each frame is t_f seconds in duration and the channel bandwidth is B_w each sensor will receive $t_s = \frac{t_f}{X}$ seconds in which to transmit data. If we have a signaling scheme that is set to 1 bit/sec/Hz each node will be allowed to transmit $B_w t_s = B_w \frac{t_f}{X}$ bits per frame or in other words have a bandwidth of $R_b = \frac{B_w}{X}$ bps.

There are two main advantages of utilizing this model for sensor networks: simplicity and energy efficiency. When a sensor is not scheduled to transmit, it can shut down its radio components. This reduces common MAC problems such as *overhearing* and *idle listening*. *Overhearing* occurs when a node receives packets that are destined to other nodes [21]. TDMA lets a sensor shut down its radio component when it is not in its proper time slot eliminating the chance of overhearing traffic. *Idle listening* happens when a radio is listening to the channel to receive possible data. The cost of such a phenomenon has been studied and in the case of most radios its overall consumption is normally as high as a radio in receiving mode [7]. Since a sensor's radio component is only on during its assigned transmission time, it will not remain on listening in an idle state.

Despite its energy efficiency TDMA suffers from not being able to easily adapt to changing topologies. This is an expensive trade off which is leading to more research in contention based protocols. Additionally using TDMA requires that all sensors be time-synchronized and requires guard slots to separate sensors (prevent overlapping) [28]. These extra features increase the overhead of using such an approach.

2.3.3 Code Division Multiple Access - CDMA

Contention based protocols are often used in Packet Radio Networks (PRNs) instead of scheduled protocols due to their greater degree of flexibility. The challenge is that narrow-band random access protocols do not allow multiple radio signals to overlap which leads to higher rates of collisions [29]. By using approximately orthogonal (low cross-correlation) spread-spectrum waveforms multiple sensors are simultaneously capable of occupying the same bandwidth. Shared access to the medium using CDMA is based on various coding techniques such as direct-sequence pseudo-noise, frequency hopping, time hopping or combinations of these techniques [29]. This is advantageous in comparison to techniques such as FDMA or TDMA since sensors will be granted the whole spectrum.

The *spectrum* by definition is the range of frequencies that a signal contains. *Spread spectrum* (SS) is the concept of sacrificing bandwidth to gain better signal-to-noise performance. SS systems utilize modulation techniques where the signal of interest,

with an information bandwidth R_b, is spread to occupy a much larger transmission bandwidth R_c.

There are two basic spread-spectrum techniques which are *direct sequencing (DS)* and *frequency hopping (FH)*. With Direct-sequence spreading the signal is multiplied by a known signal of much larger bandwidth. On the other hand frequency hopping, the center frequency of the transmitted signal varies in a pseudo-random pattern [22].

Direct-sequence spectrum (DS-SS) is a form of spread spectrum which attempts to encode the signal to be sent by using a specific spreading code to distinguish it from other signals [22]. This allows users to freely use the full available radio spectrum over as large an area and as many times as possible. Spreading the signal across a wider band exposes it to potential channel degradations and to interference. The transmitted energy required remains the same but, since the signal is spread on a larger bandwidth, it is often below the noise floor of receivers. In other words, the signal looks like noise to any receiver that does not know the signal's structure [22]. This was the key reason for its initial development by the military since any signal would be difficult to detect and jam by adversaries. Successful reception of a signal will occur only if the receiver has the same spreading code which was used by the sender. Any other signal encoded by a different spreading code will appear as noise to a sensor. As the total number of nodes in a network increases the Signal-To-Noise (SNR) of each transmission will be reduced. Therefore, as far as performance CDMA networks will loose quality as the total number of nodes increase. The SNR directly depends on the amount of spreading and the number of interfering signals by the following equation [14]:

$$N = \eta_b c_d * \frac{B_w}{R_b(\frac{E_b}{I_o})} \tag{1}$$

where η_b is the bandwidth efficiency factor, c_d is the capacity degradation factor, B_w is the total bandwidth, and $\frac{E_b}{I_o}$ is the bit energy to interference ratio required to achieve an acceptable probability of error. Under ideal conditions and when the minimum SNR required ($(\frac{E_b}{I_o}) = 1$=0 dB which is achieved by minimizing the spreading of the data) the equation simplifies to

$$R_b = \frac{B_w}{N} \tag{2}$$

Therefore with minimum spreading each of the *N* sensors can transmit at the equivalent bit rate of TDMA and FDMA systems.

Several advantages exist when using spread spectrum techniques:

- Increased resistance to interference

- Signal becomes very hard to intercept or detect which prevents any sort of jamming.
- Greater tolerance to certain propagation effects (i.e. multi-path interference and narrow band Electromagnetic Interference (EMI).
- Higher ranging capabilities.

2.3.4 IEEE 802.11

This protocol, like CSMA, is a randomized contention based paradigm where sensors compete for the channel. Like other MAC 802.x protocols, the 802.11 protocol spreads across both the MAC and Physical layer. The MAC layer which is discussed in this section is designed to interact with three different types of technologies at the Physical layer: Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), and Infrared.

The MAC 802.11 standard has two medium access methods: *DCF (Distributed Coordination Function)* and *PCF (Point Coordination Function)* [27]. DCF is used when no central control system exists, or is desired, and it utilizes CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). PCF uses a centrally controlled polling method to support synchronous data transfer and would therefore not be appropriate for very large sensor networks. In order to use 802.11 DCF is required but PCF is optional. DCF is designed for ad hoc networks, while PCF (sometimes referred to as infrastructure mode) is mainly used where designated access points can manage wireless communications.

DCF sits on top of the physical layer and is essentially based on CSMA as discussed in Section 2.3.1. When the network load is low CSMA is a very effective solution for sharing the medium, but as traffic increases collisions must be detected. In traditional Ethernet networks CSMA/CD (Carrier Sense Multiple Access with Collision Detection) is used where the CD (Collision Detection) allows the MAC layer to re-transmit after an exponential random back-off takes place. Allowing the MAC layer to perform the Collision Detection and retransmission is advantageous since it prevents the overhead of having upper layers deal with such a problem. Unfortunately, Collision Detection is not feasible in wireless LANs such as 802.11 for two particular reasons [27]:

- To have sensors perform Collision Detection would require full duplex radios capable of both transmitting and receiving concurrently. This would increase the cost of sensors which is not a feasible option.
- Unlike wired LANs where all nodes hear one another, in wireless LANs one can not assume that all sensors can hear one another. Due to the hidden terminal

problem situations will arise that a sensor might think the channel is free but is in fact busy.

Therefore, for the 802.11 protocol it uses a combination of CA (Collision Avoidance) and Acknowledgment packets (ACK packets) to perform the equivalent. So when a node desires to use the channel it verifies the medium and if it is busy then it defers (like CSMA). On the other hand, if the medium is free it will wait a specific amount of time (this time is called the Inter Frame Spacing (IFS)) before actually transmitting. The IFS is simply a period that a station is required to wait after sensing an idle channel. If the channel successfully remains idle for a time no less then the specified DCF IFS (DIFS), the station can begin transmission.[1] Once the receiving station starts to receive packets it sends Acknowledgment packets (ACK) back to the source. Once the sender receives these acknowledgments it serves as a confirmation that no collisions occurred. If the sender does not receive an acknowledgment packet then it will retransmit the fragment until it receives one or reaches the maximum number of retransmissions.

Since the DCF approach is still vulnerable to collisions do to various wireless phenomena such as the hidden terminal problem this protocol attempts to reduce the probability of collisions by using *Virtual Carrier Sensing*. Hence in 802.11, carrier sensing is accomplished at both the air interface, known as physical carrier sensing (CSMA/CA), and at the MAC sub-layer (Virtual carrier sensing) [13]. The *Virtual Carrier Sensing* is based on the MACAW (Medium Access Control with Collision Avoidance for Wireless) protocol [23] which is ultimately an extension of the original work done by Karn in [24] in proposing MACA. This method reduces the number of collisions by establishing a brief handshake between a sender and receiver before attempting to transmit. The handshake is initiated by the sender who sends out a *Request-to-Send (RTS)* packet to whom it wants to communicate. If the receiver is capable of accepting data, it replies with a *Clear-to-Send (CTS)* packet. This brief exchange serves as an announcement to others that the channel will be occupied and not idle. For example, in Figure 5 although node c cannot hear the RTS from a, it can hear the CTS from b. Once a neighboring node overhears either an RTS or CTS packet not destined for itself it will back-off without sending its own packet. By adding this additional step one now reduces the chance of having collisions with large data packets which will increase performance. The standard specifies the RTS threshold which allows packets that are smaller then the threshold to be sent without the initial RTS/CTS exchange [25].

In wired networks Ethernet packets can be as large as 1518 bytes long which in wireless LANs would be problematic for several reasons [27]:

[1] The 802.11 standard defines four different types of Inter Frame Spaces which allows for a certain prioritization. These types are: Short Inter Frame Space (SIFS), Point Coordination Frame Space (PIFS), Distributed Inter Frame Space (DIFS), and Extended Interframe Space (EIFS). For details on when each are used and their duration refer to [25].

- As the packet size increases the probability of a packet getting corrupted also increases due to higher bit error rates in wireless media.
- If transmitting small packets and a collision occurs the overhead in re-transmitting the packet is much smaller.
- When using FHSS at the Physical layer, the medium is interrupted often therefore the smaller the packet the smaller the chance that the retransmission will be postponed after dwell time.

To allow for both environments (wired/wireless) to communicate efficiently together the IEEE committee added the ability for the MAC layer to perform fragmentation and re-assembly allowing wireless packets to remain small.

The added control over the shared medium comes at a tremendous cost on energy consumption. The MAC 802.11 protocol has now three types of packets: management, control, and data with all data exchange being performed with acknowledgment packets as well. Hence studies show that the energy costs associated with this protocol is very high when nodes are in idle mode. For battery powered devices, such as sensors, the radio power drawn in idle mode is almost as high as when receiving [21] which is an unacceptable situation. The 802.11 protocol does have a power saving mode but it relies on an Access Point (AP) to maintain continual records of the stations incoming packets and to notify it that there is information waiting to be received. When the power save mode is activated, the 802.11 protocol indicates its desire to enter a sleep state to the access point by toggling the power save bit in the header of each 802.11 frame from a 0 to a 1 [25]. When the access point receives such frames it will begin buffering packets for the client while it is asleep. When the sleeping sensor wakes again it will attempt to establish communication with the access point to determine if packets were waiting to be delivered to it. In an ad hoc network such as sensor networks no such hardware exists.

The IEEE 802.11 protocol initially developed in 1990 had the objective of developing an international WLAN standard. Although it has well suited traditional wireless networks over time it has paid little attention to energy constrained devices and hence is less suited for sensor networks.

2.4 Routing Protocols

In order to maximize efficiency in application specific sensor nodes, one must not only look at the possible MAC protocols but also the higher network level options. Higher level protocols are capable of optimizing transmission by maintaining efficient routes, shutting down the radio, keeping alternate routes, and act as routing intermediaries.

Generally all routing protocols fall into one of the following categories [30, 31]:

- **One-Hop Model:** A routing protocol uses this model if all sensors attempt to establish a direct connection back to the base station.
- **Multi-Hop Model:** This paradigm is followed whenever multiple hops are used to send back to a base station.
- **Hierarchical Model:** These protocols allow sensors to form clusters where each cluster has a local base station (known as a cluster-head) which can perform some data aggregation to reduce the amount of data heading to the base station and also conserves energy.
- **Data-Centric:** These are query based protocols and function based on the naming of the requested data. This helps eliminate redundant transmissions.
- **Location-based:** This family of protocols utilize the position information to relay the data to the desired regions rather than the whole network.

For the purpose of this research, Data-Centric and Location-Based protocols were not examined. Data-centric is when the sink sends out a query to a certain region and waits for data from the sensors located in that region [30]. Since the goal was to develop a model for sensing the radioactivity in a reactor, operators working in the reactor would prefer to have the sensors automatically send out warnings without waiting for the base station to query for it. Therefore, Data-Centric protocols were not studied in this research. Location-Based protocols will often be used to minimize energy by efficiently directing queries in specific regions. Although the protocols tested for our simulations are location aware through the use of beacons they would not be considered Location-Based protocols. This research also does not look into location based protocols for two reasons. First, the goal was not to rely on GPS capabilities since it does not function indoors. Secondly, one of the design criteria was simplicity, so it was felt that location based protocols would add extra overhead which was not needed for the purposes of the reactor. Below is an overview of the protocols analyzed and simulated for this research.

2.4.1 Direct Method

The *direct method*, sometimes referred to as the *One-Hop* method, occurs when all sensors communicate directly to the base station [31]. This paradigm's simplistic nature provides a routing algorithm with no setup cost and tremendous ease of deployment. As shown in Figure 7(a) all sensors (represented by circles) send directly back to the base station providing very low latency between the sensed phenomenon and the updated

status at the sink. Figure 7(b) shows the low levels of latency throughout a simulation of 55 sensors using the One-Hop method and IEEE 802.11 as the MAC protocol.[2] The y-axis in Figure 7(b) represents the latency where, with the exceptions of a couple peaks the average is always near zero. The advantages of this model are quickly overlooked

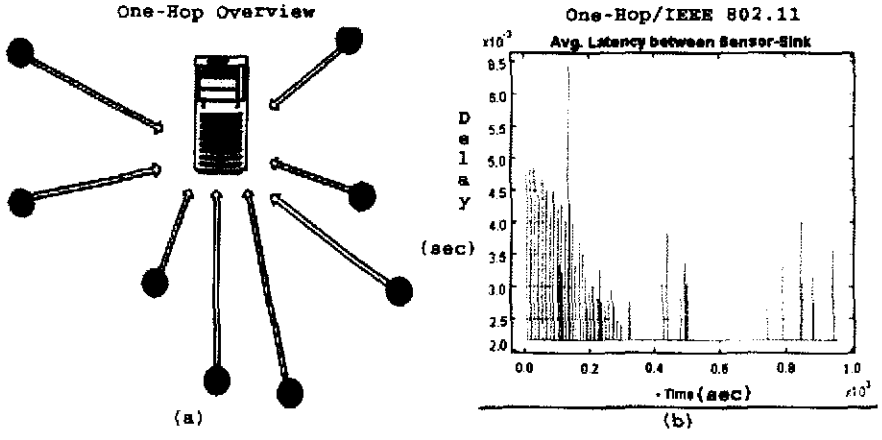

Figure 7: (a) Sample One-Hop Paradigm (b) One-Hop model latency

by its drawbacks. The direct method makes the base station a bottleneck point. This problem is known as the implosion problem, because it involves a high number of collisions, there is a degradation in the packet delivery ratio [32]. Avoiding the implosion problem can be accomplished through either a very strict media access control or a sparse network containing a smaller number of nodes.

A second disadvantage of this method is the cost in terms of energy for packet transmission. The cost of transmission is proportional to the distance squared. By making the realistic assumption that our sensors are equipped with radio transmission power that is based on software control we will be able to maximize lifetime. Therefore, by using known propagation models such as Friis freespace equation (details of the equation are explained in Section 5.1) [22]:

$$P = \frac{P_t G_t G_r \lambda^2}{4\pi d^2 L} \tag{3}$$

and knowing our sensors have a certain receiving threshold, we can calculate the minimum radio amplification power required. Applying this radio model to the direct method leads to distant nodes dying out much quicker than closer nodes [33, 31]. Figure

[2]See appendix A.7 for values used in obtaining the above results.

8 illustrates this phenomenon over time. This simulation shows sensors (star shaped) which are still alive and a black X for sensors which have died. As can be seen sensors which are located at greater distances die first which quickly reduces the size of the sensing field.[3] The third disadvantage of this method lies strictly in basic radio transmis-

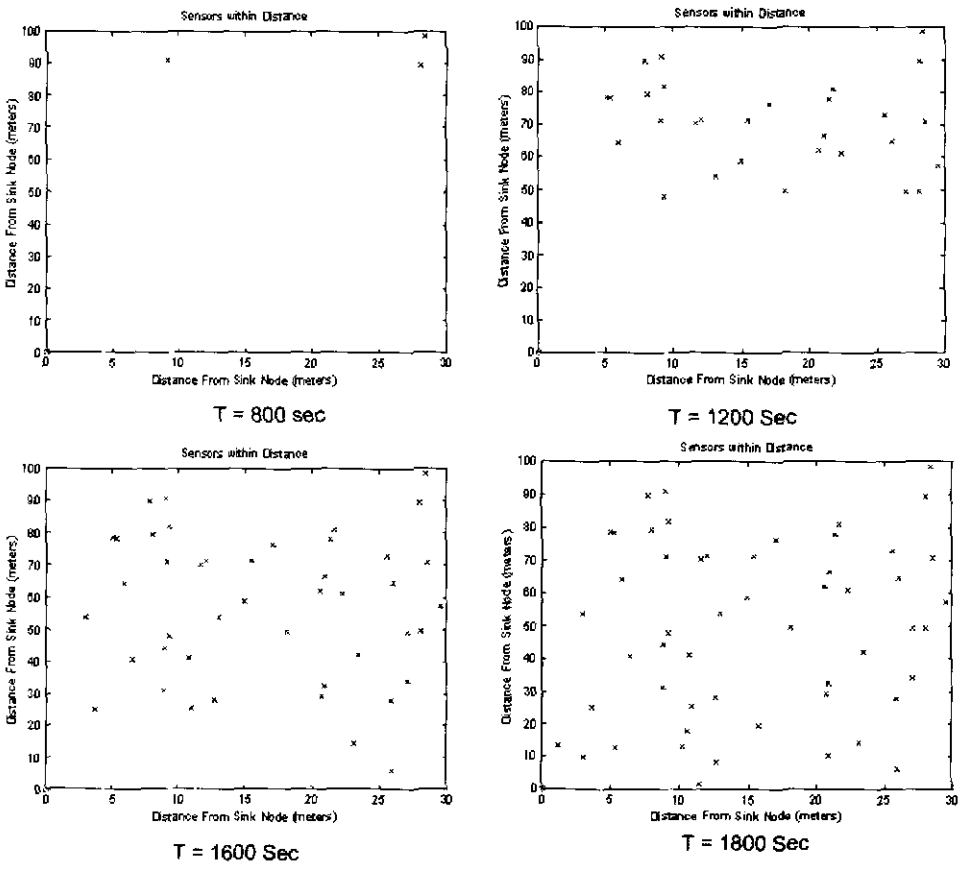

Figure 8: Sensor death over time using One Hop Model.

sion challenges. Reflection, diffraction, and scattering are the three basic propagation phenomena that will affect a sensor from communicating back to its base station. Since the direct method makes no effort in communicating with a neighbour node depending on the application environment some sensors may never reach the sink.

[3]See appendix A.7 for values used in obtaining the above results.

2.4.2 Multi-Hop Method

This model attempts to maximize lifetime by reducing transmission cost and making sensors form linear chains back to the base station [31]. By assuming software controlled radio transmit power and location aware sensors, each sensor only sends to its closest neighbor that is located one step closer to the base station. Figure 9 illustrates how the information travels from source to destination by hopping from one node to another until reaching the base station. This method prevents the implosion problem and utilizes distance to its advantage to prolong overall lifetime.

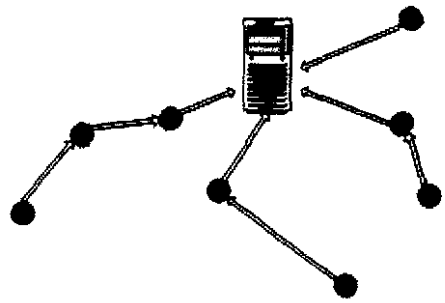

Figure 9: Sample Multi-hop Paradigm

The failure of this method lies in its dependency on sensors located closer to the base station. Since sensors closer to the sink are constantly acting as routing intermediaries their power is quickly drained. This phenomenon is known as the *self-induced blackhole* effect. Figure 10 illustrates this effect through simulation. The star shapes are for those sensors that are still alive, and a black *X* represents sensors that have died.

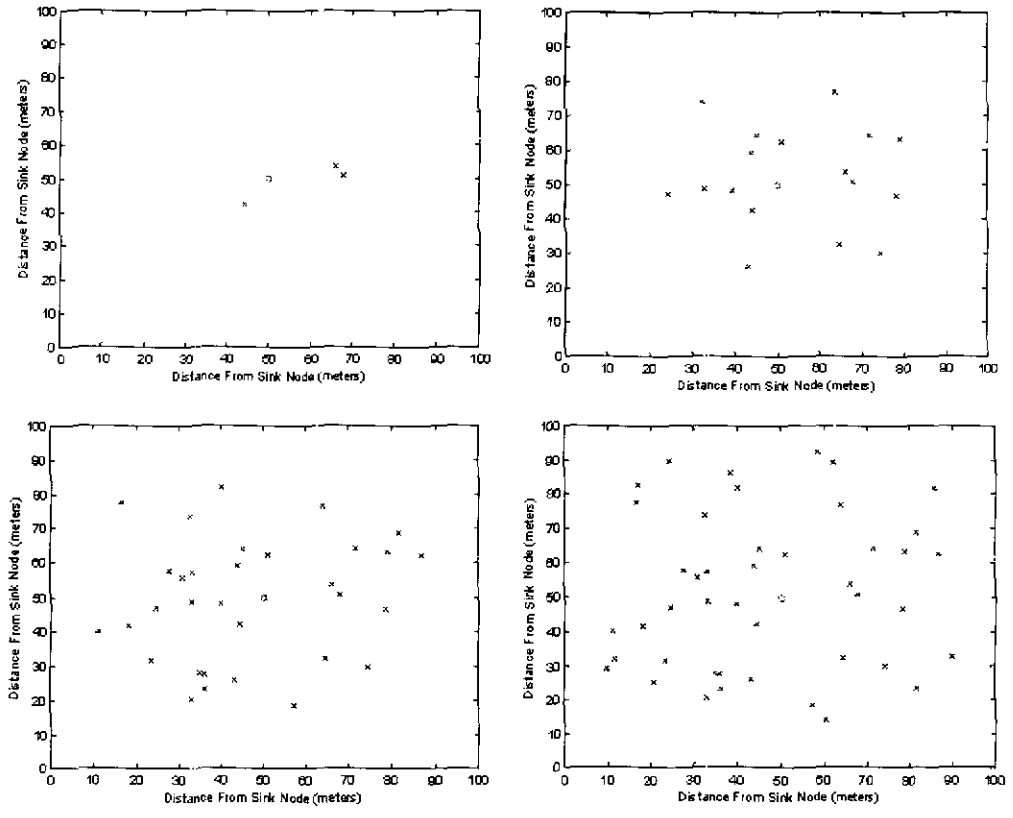

Figure 10: Multi-Hop scheme where inner sensors act as routing nodes.

As can be seen sensors which are located closer to the sink die before those at greater distances. This leads to sensing gaps and hence shows that this protocol leads to an un-even distribution of sensor usage.[4]

A second problem lies in the challenge of wireless communication. A problem already discussed in Section 2.3.1 called the hidden terminal, occurs when a sensor is outside the transmission range of another sender and therefore leading it to think that the channel is idle and proceeds to sending (see Figure 5). This problem is predominantly present in this paradigm hence close attention needs to be paid to the MAC protocol that will be used with the Multi-Hop scheme.

Although forming chains allows sensors to have lower transmission cost one is now faced with higher delays before data reaches back to the base station. When compared

[4]See appendix A.7 for values used in obtaining the above results.

to the delays that a One-Hop method would have it is evident that as a scenario scales and more hops are needed the latency in a Multi-Hop scheme will become higher. Depending on the application this may be unacceptable.

2.4.3 LEACH

Low-Energy Adaptive clustering hierarchy (LEACH) is a self-organizing, adaptive clustering protocol [33, 14]. This protocol is classified as a hierarchical protocol in which sensors organize themselves into clusters. Each cluster has one node acting as a cluster-head, which acts as a local base station. Figure 11 illustrates a small LEACH network.

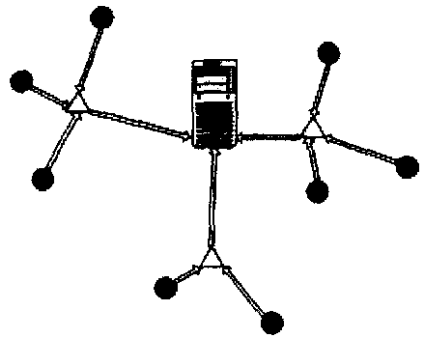

Figure 11: Sample LEACH paradigm.

Here the CPU represents the base station, triangles are the elected cluster-heads for a round, and the circles are the sensors that make up the three clusters in this scenario. The cluster-head organizes a sending schedule for all nodes in its cluster and periodically sends the aggregated data in one larger transmission back to the base station. To avoid draining the battery of the cluster-heads LEACH is broken into rounds where

Figure 12: LEACH is broken down into two distinct phases: Set-up and Steady-state phase.

each round consists of a setup phase followed by a longer steady-state phase as shown in Figure 12.

During the setup phase each node independently decides whether or not to become a cluster-head based on some threshold

$$T_n = \begin{cases} \frac{p}{1-p*(r \bmod (1/p))} & n \subseteq G \\ 0 & otherwise \end{cases} \quad (4)$$

where p is the desired percentage of cluster-heads, r notes the current round, and G is the set of nodes that have not been cluster-heads in the last $1/p$ rounds [33]. It has been determined that the optimal number of cluster-heads is 5% of the total number of live sensors according to [14].

During the setup phases all nodes must remain on and CSMA is used as a media access protocol. To avoid collisions and the hidden terminal problem, the transmission power is set so that all nodes can hear. Once the cluster-head has heard back from all joining sensors it creates a TDMA schedule which allocates a time slot to each sensor. This allows for both energy saving (nodes can go to sleep when it is not their turn to transmit) and eliminates inter-cluster collisions.

In order to address the issue of intra-cluster collisions, each cluster communicates using different CDMA codes. Therefore, a cluster-head chooses a unique spreading code and filters all received energy using that spreading code. This leads to filtering out traffic from other clusters. Furthermore, cluster-heads use a dedicated spreading code to communicate back to the base station.

In both the direct and Multi-Hop schemes, minimal overhead is experienced in network maintenance. Forming clusters in LEACH means an overhead in energy consumption, and short periods of no updates to the base station. The setup cost will vary with the size of the broadcast packets used during this phase (for a detailed discussion on the energy costs associated with this setup phase please read Appendix B.6). Given a dense topology one can assume, on average, properly spaced cluster-heads. Despite the overhead complexity and cluster setup costs, LEACH shows many promising advantages. LEACH reduces collisions by combining several MAC techniques and attempts to evenly distribute energy consumption. Another advantage of this paradigm is its ability to scale which is not present in the Multi-Hop and direct methods (due to latency and congestion respectively). Many other paradigms have emerged including protocols such as PEGASIS, APTEEN, and TEEN (see [34, 35, 36, 37] for details on those protocols) but due to time constraints they are not investigated in this research.

2.5 Localization

Sensors can either be deployed in a deterministic or non-deterministic fashion throughout a sensing field [9]. A deterministic placement would involve operator control over sensor placement, whereas a non-deterministic distribution is when no order in the

placement of the nodes exists. Non-deterministic placement can occur in several various applications, for example ocean surfaces, war zones, and even inside volcanoes. Due to reachability these nodes are oftentimes dropped from above and must automatically form their ad hoc network. If the sensors are aware of their positions many benefits are possible such as improving routing efficiency [38], and identifying the coordinates of a hot spot in a reactor.

Localization is by definition the problem of having a sensor estimate its spatial-coordinates to determine its approximate physical location after deployment [38]. There are four broad categories for localization: (1) in-building Infra-Red (IR) networks (2) wide-area cellular networks (based on RF), (3) Global Positioning Systems (GPS), and (4) RF-based beacons. The first category using IR technologies leads to many limitations such as poor scalability, high maintenance cost, and reduced performance in certain conditions [39]. The next category, wide-area cellular networks, uses aspects such as measuring signal attenuation, the Angle Of Arrival (AOA), and the time difference of arrival (TDOA). These methods are known to be effective for outdoor larger areas but have been shown to be inaccurate due to the difference between indoor and outdoor wireless propagation effects. For multiple reasons Global Positioning System (GPS), the third category, is also not an appropriate alternative for sensors. First, GPS can work only in outdoor environments and since we are deploying sensors inside a reactor it would not accurately function. Secondly, sensors must remain small, inexpensive, and disposable; equipping them with GPS receivers would increase unit price significantly. A GPS capable sensor, according to [38], can make a sensor be two orders of magnitude more expensive then non-GPS sensors. The fourth type, RF-based beacons, can provide many benefits since radio frequency devices are widely available, cost is much less (no GPS), and no specialized hardware is required.

Localization techniques within each of the above categories can be further subdivided into into two categories either fine-grained, or coarse-grained. Fine-grained techniques are based on timing and signal strength whereas coarse-grained methods are based on proximity to reference points. Fine-grain methods include [5]:

- **Timing:** When the distance is calculated based on the amount of time it takes for a signal to get from a sender to a receiver using some reference point.
- **Received Signal Strength Indicator (RSSI):** Radio propagation signals attenuate over distance; therefore by using certain models one can calculate the distance between the sender and receiver.
- **Signal Pattern Matching:** This method utilizes a pre-generated database which creates a grid of certain dimensions and assigns a unique signature to each square of the grid. This central system then matches a transmitting signal from sensors

with the pre-constructed database and determines its location. This is not an optimal method for sensor networks since it is based on a centralized paradigm and it is counter to the concepts of ad hoc deployment.

On the other hand, we have coarse-grained localization techniques which include proximity based localization. Proximity based algorithms use an approach referred to as trilateration. Assume we have a fix number of sensors which are already pre-programmed with their spatial coordinates called *beacons*. These beacons will not take any role in monitoring but rather will be used to aid the other sensors in determining their locations (These beacons are similar to the 27 GPS satellites orbiting the earth). These beacons will remain permanently on and will periodically broadcast out their coordinates with a signal strong enough to travel the whole reactor. Therefore, when a regular sensor node starts-up it will listen for broadcasts from these beacons and calculate its own position based on incoming RF signal-strength (SS).

The details of this localization algorithm is based on determining the intersections of circles in 2D space or spheres in 3D space. Consider a sensor, X, that wants to know its location in a 2D space. If X knows it is 20 meters from beacon A then sensor X knows that it is anywhere along a circle of radius 20 meters from A shown in Figure 13(a). Now if sensor X also hears a periodic broadcast from beacon B it can combine

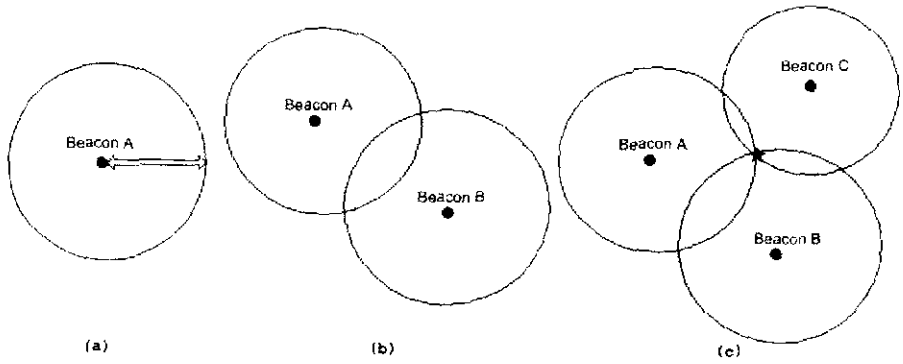

Figure 13: Sensor Localization using Trilateration in 2D space.

the information from both beacons A and B and determine it is at one of the overlapping points (Figure 13(b)). With the additional help of a third beacon, beacon C, sensor X now approximates its position in space (Figure 13(c)). Therefore, trilateration is a geometric principle which can be used to approximate sensor location. The above example was strictly in 2D space, but the principle extends to three-dimensional space where spheres are used instead.

For the purposes of the McMaster Nuclear Reactor there will be two types of nodes deployed: sensors with no a priori knowledge of location, and beacons with knowledge of their position. Therefore, the sensors are capable of being freely placed wherever within the reactor. Once the sensor is powered it will enter an initial setup phase which will include determining its coordinates. This approach is similar to others which have been deployed and tested such as RADAR [39]. RADAR is an RF-based system for locating and tracking users inside a building. Another similar system which approximates user locations through RF signal strength is the Cricket system discussed in [40].

3 MNR Environment

This chapter starts with a discussion on the differences between a research reactor, like the MNR, and power reactors. It illustrates key areas in which sensor networks can be of great help in both day-to-day safety and disaster prevention. This is then followed by a survey on the current competing technologies that the McMaster Health Physics department have at their disposal. The chapter finishes with a discussion on the wireless communication challenges due to the building structure and a listing of routing requirements for such an environment.

3.1 Research Reactor and Sensor Networks

There exist two classifications of nuclear reactors. The first is called a power reactor and is used to generate electricity. The second category of reactors are research reactors that are built for the purposes of nuclear research. Such research includes, but is not limited to, agents for cancer detection and treatment, stronger materials, and better electronics. The two types of reactors differ in various aspects such as size, purpose, and heat generation. In terms of size a typical power reactor core can roughly fill the size of a two-car garage, whereas a research reactor's core is the size of a medium-capacity home water heater [41]. In power reactors tremendous amounts of heat and high pressures are required for electricity generation. Therefore, such reactors have large cores and are normally complex. The MNR is classified as a research reactor and can be said to be a low-temperature, low-pressure reactor operating on a daily base of 2MW [42]. Three particular aspects that make the MNR a research reactor are *beamports*, *incore irradiation systems*, and *rabbit irradiation systems*.

The six *beamports* provide a direct path to the core to perform experiments where materials can be studied under direct radiation. A beamport at its simplest level can be considered an aluminum can with air in it which shields the radiation from the water. To reduce the strength of the radiation beams various insulations and filters are placed within the beamport to attenuate the signal strength. The most common use of these beamports in the MNR is for radiography. This is normally performed on three of their six beams producing the highest radiation signals and are the most rearranged beamports. Each reconfiguration (i.e. rearrangement) is dictated by the experimental requirements for a particular use. The danger with such beamport experiments is that radiation keeps going and gets scattered by the materials being studied. To prevent human contact, shielding in the form of concrete blocks are placed all around the beamports. The problem with this solution is that if a shift in the blocks occur then a crack may be created exposing operators directly to radiation. Their current means of safety for this scenario involves an area radiation monitor and Health Physics personal sur-

veying the area with hand-held instrumentation. A Sensor Network in this case would be able to surround the inside and outside of the shielded wall performing a continuous data report while the beamport is open. This would create a blanket around the concrete shielding structure which would provide historical, and real-time data logging which is more flexible as well as time efficient than their current means.

A second common scenario in the MNR is the use of the open core to perform research. This is known as an *incore irradiation system* which is typically not available in power reactors. With the low pressure, low temperature, and flexible core, operators can insert items in or next to the core. Results of performing such tests are that researchers can study new radioactive isotopes for potential cancer therapy. The danger with such experiments is when a radioactive element is pulled out of the water. If the level of radioactivity is too high the element should not come near or in contact with the operator and should remain below the water surface or put in an isolated, shielded environment. The use of sensor networks deployed along the bridge and inside the reactor pool would allow for operators to test and determine the level of radioactivity of the object as it is being pulled from the core. This would allow for a more pro-active approach when compared to their current means which is a single surface area radiation monitor.

The third aspect of the MNR which illustrates some of the unique studies that occur in the reactor is the use of *rabbit irradiation systems*. This is a pneumatic tube system which allows one to insert various samples to be exposed to neutron radiation for short durations (minutes to fractions of minutes). The plastic tubing goes through the core and the material that is inserted will capture radiation, hence becoming radioactive (highly material dependent). Currently a radiation monitor is placed at the exit where the rabbit comes out. If the radiation monitor senses high radiation it is sent back off. An advantage of the Sensor Network would be to place them back up the line outside the plastic tubing before the exit. Therefore, when samples come by, the sensors can act as actuators and if the radiation is too high divert the material to a shielded area.

With a focus on deploying sensor networks within the reactor there is also a heavy need for determining a method to accommodate potential outdoor radiation hazards. This is classified as a *type D* emergency for which very few means of sensing exists if such a scenario were to occur. Presently the Health Physics department would have to send their scarce resources (operators) driving around in cars obtaining measurements to determine which direction the plume is and how strong it is. Understanding the direction of the wind would be critical in such a scenario and could be accomplished by using sensors. An outdoor wireless monitoring station that can quickly be deployed across the university campus which would greatly improve their ability to handle such an emergency.

This section summarized the key areas in which this reactor separates itself from common power reactors. Three particular day-to-day research scenarios are discussed

beamports, incore irradiation systems, and the rabbit irradiation system, and how sensor networks can lead to a more flexible and pro-active safety environment.

3.2 Competing Technologies

In order to justify the deployment of sensor networks in a reactor, a survey on the current methods used by Health Physics officials in the MNR was completed. Many of their current means do not involve a tremendous amount of technology. These techniques have withstood the test of time, are well understood, and have proven to be reliable. The major downfall of upgrading or using newer methods of surveying radiation is the cost of such tools which often ranges in the thousands of dollars. The *detection* of radiation requires the identification of a physical or chemical change in some medium through which the radiation passes. The actual *measurement* of such a detection requires the ability to quantify such physical or chemical changes [43]. This research looks at a tool that encompasses both the ability to detect and measure the radiation in its surroundings.

The primary tool used by the Health Physics department are *Thermal Luminescent Dosimeter* (TLD) badges. This is considered a passive detection method, since these badges do not require battery power. They are used to assess an individual's cumulative external radiation exposure. They are small chips of material (i.e. LiF or CaF_2) which, when exposed to heat after being penetrated by radiation will give off light proportional to the dose received [43]. The advantage of such a device is that it is very flexible, cheap, reliable, and can last a very long period of time. The downfall of this tool is it is unable to obtain time-stamped data or perform any sort of historical data collection (i.e. storage) of time-series data. These badges simply provide the dose accumulation over the time period it was active. The other crucial downfall is the down-time that is experienced before obtaining results from these badges. They must be sent out for analysis, and results come back within approximately three weeks which creates large gaps between the actual operator exposure and preventative measures that can be taken.

A second tool is the *Geiger-Mueller* (GM) detector which is a portable and hand-held detector. It functions with a gas-filled chamber with a voltage applied so that a central wire becomes an anode and the chamber wall a cathode. Any incoming radiation will interact with the chamber and create an electronic pulse which can then be measured. Due to its dependability the GM detectors are the most widely used surveying tools used in the industry [43]. This surveying approach provides very reliable data gathering since experienced personal are on-site using their own domain knowledge to locate possible hot spots. The downfall is the potential danger to operators performing the survey, the down-time, cost, and the formal survey report that must accompany all on-site surveys. It is also considered a *point-in-time* analysis of the current conditions in a specific area of the reactor which provides no historical or cumulative data.

historical data collection capabilities, and reliability.

3.3 MNR Wireless Challenges

It is a challenging task to build an RF friendly building that is free from aspects such as reflections, diffractions, and scattering. Between the reactor core being in the middle of the building, thick walls, and floor surfaces of the reactor electromagnetic waves can attenuate rapidly. In order for any sort of wireless radiation monitoring to be done in the reactor the wireless system must be robust and free of as much erratic behavior as possible. If the radio system fails then the overall system will also fail endangering the lives of the operators in the reactor.

To alleviate many of the RF multipath problems we must determine where to locate the base station for optimal performance and how to position the nodes to maintain strong radio signals. For accurate simulations of deploying a Sensor Network in the reactor, proper dimensions, and an understanding of the layout of the building is needed to prevent having any dead spots. The building is constructed with 1.5 meter thick reinforced concrete slabs all around and its shape is illustrated in Figure 15. It has 4 floors with many possible obstructions that exists [42] one such floor (the experimental floor) is shown in Figure 16). This floor is where most operators work and visitors normally are located. A primary concern will be radiation at lower floors especially near the beam ports since high activity can normally be found there.

3.4 Application Specific Requirements

Unlike traditional networks which are built to accommodate a wide variety of applications and protocols sensor networks are built for a specific task. In order to maximize efficiency, reliability, and lifetime each WSN should be tailored for the specific application at hand. In this research, the feasibility of sensors in the McMaster Nuclear Reactor is being investigated. The following sections discuss the various issues that need to be addressed for a successful deployment in such an environment.

3.4.1 Mobility & Placement

Multiple types of mobility exists in sensor networks including: mobile base station(s), mobile sensors or even a combination of both. Sensor mobility can range from very frequent to infrequent and this plays an important role in determining the proper protocol choice. In the case of the MNR, the solution consists of a single non-mobile base station with non-mobile nodes to report back the information to the operators.

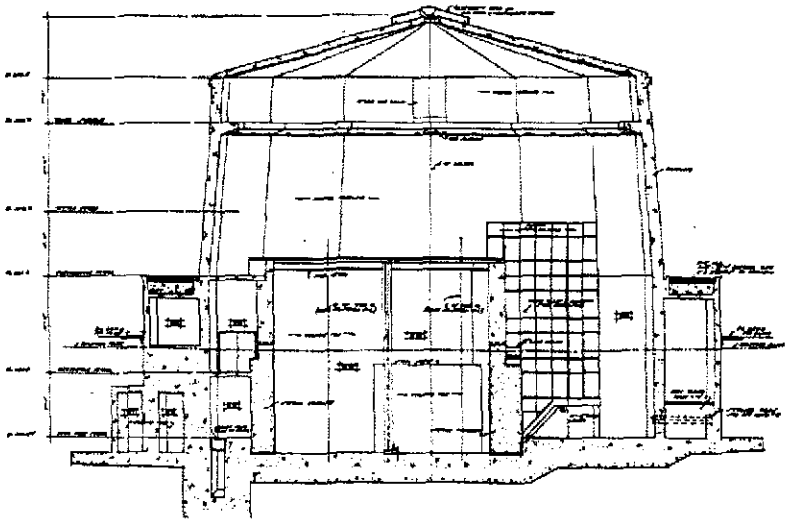

Figure 15: The North to South Cross section of the MNR.

The actual deployment of sensors can either be *pre-determined* (i.e. deterministic) or *randomized* (i.e. non-deterministic). In a pre-determined network sensors can be strategically placed such that routing protocols can use set routes. In a randomized layout sensors are in a non-optimal, non-uniform distribution. In a more realistic Sensor Network environment sensors are deployed in a randomized fashion where sensors are simply scattered randomly. This oftentimes results in a non-optimal, non-uniform distribution of numerous sensors. For the MNR sensors will be manually placed but in a randomized fashion with no assumptions being made on placement. This will allow for greater flexibility if one will eventually need to move a sensor.

3.4.2 Data Aggregation

In optimizing protocols for WSN a concept that has been introduced several times is the use of *data aggregation*. Data aggregation is the combination of data from different sources, and can be implemented in a number of ways. One can attempt to classify a Sensor Network as being one of the following types [45]:

1. All sources send completely different information (no redundancy)
2. All sources send identical information (complete redundancy)

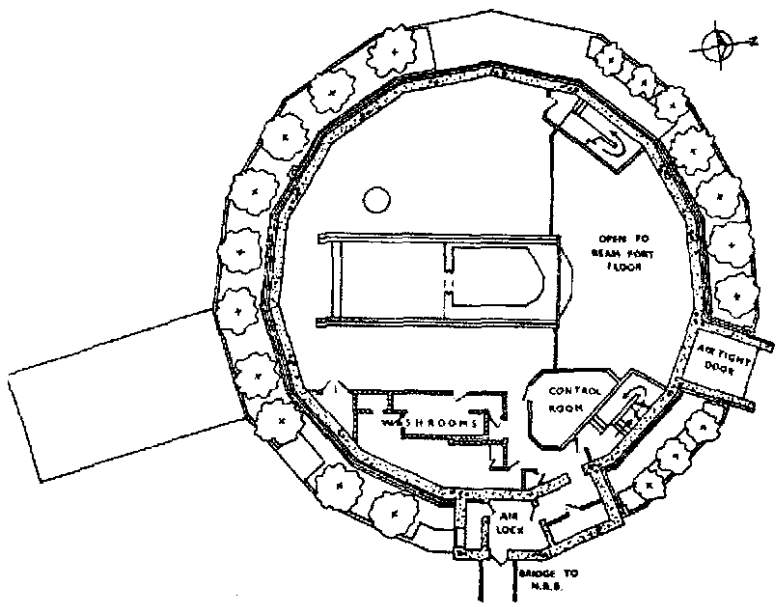

Figure 16: This is the skyview of the experimental floor (Third Floor).

3. The sources send information with some intermediate, non-deterministic, level of redundancy.

In the context of monitoring a reactor all data gathering from monitoring an event can be classified as critical and yet identical. For example, if two sensors are placed within five feet of each other and radiation levels are being monitored on three second intervals chances are a tremendous amount of similarity will be seen.

The opposite of data aggregation normally leads to a bottleneck at the base station. This problem is often referred to as the *response implosion problem* where the base station requests a certain piece of information and gets answers back from all nodes in the network [32]. Since sensor networks rely on wireless communication the results of such large quantities of replies will lead to several collisions reducing the overall efficiency of the network. Although if the WSN is relatively small then this is traditionally the method of choice.

Therefore, to a degree a routing protocol should for this scenario be modified to reduce duplication to an acceptable level while still maintaining accurate readings for operators. For the reactor a balance will have to be found between how much aggregation will be performed and accuracy.

3.4.3 Scalability & Density

Scalability of sensor networks is an extremely important factor. Deployment of sensors in a reactor will have to take into account scalability for several reasons. If radiation activities increase, structural changes to the reactor occur, or a new type of sensed data is desired then added sensors will be required. By scaling a network upwards, its density is increased. The density of a network can be defined as the number of nodes per nominal coverage area. In [46], density is defined as

$$\mu(R) = \frac{N\pi R^2}{A} \tag{5}$$

where N is the number of nodes distributed in a region of area A, and R is the nominal range of each node. The proper levels of density required will be determined by the sensing area a sensor can sense and by the Health Physics safety standards. The deployment of such a network should attempt to mimic a grid that is surveyed by humans using instruments such as Geiger-Mueller detectors.

3.4.4 Propagation Model

Here a propagation model that can accurately describe the behavior of the phenomenon will be required. A primary concern in the McMaster Nuclear Reactor is the level of radioactivity which is what the sensors will be attempting to gather. In order to accurately study their behavior, simulation will be used since analytical studies of such a large system would be inefficient. Therefore, a model was created to simulate the behavior (i.e. attenuation) of radioactivity in the reactor.

3.4.5 Network-Layer Routing

Several routing protocols have been developed in recent years and almost all of them fall into one of the following five categories: Direct, Multi-Hop, Data-Centric, Hierarchical, or Location-Based. Within each category a large number of varying protocols have been proposed each with a particular aspect in mind. Despite all the suggested protocols to maximize efficiency one must exploit the specific environment that will be monitored. Hence, for the reactor a careful study was made on various suitable protocols and a modification to the hierarchical LEACH protocol, which was called *MNRLEACH* is proposed.

3.4.6 Fault Tolerance and Robustness

Wireless Sensor Networks are built around inexpensive, disposable sensors that can frequently die. When measuring radioactivity in a reactor we are attempting to measure the

level of safety for various regions in the building (and potentially around a set perimeter outside the building). Therefore, accuracy and continuity of service is of tremendous importance. An ideal solution to be presented to the MNR committee must guarantee a certain degree of reliability, and provide a means to complement current measuring techniques without being invasive. This leads to a protocol having to be capable of handling everything from node failures, to cluster failures, and the possibility of new obstructions preventing previously used paths. Providing analysis on the fault-tolerance of an application is highly dependent on the particular scenario being monitored and for the purposes of this research will not be studied.

3.4.7 Prioritized Data

In some Sensor Network setups Multi-Hop or hierarchical schemes are used where data aggregation is normally performed. This can be done to prevent overloading the base station and essentially filtering out any potential outliers along the way. Methods used can range from taking the sum, average, or minimum of a set of incoming messages and only sending the calculated result onwards. In many networks this is a feasible solution because it minimizes the packet size by performing in house calculations. If occasional data are dropped, or certain periodic events do not make it back to the base station when monitoring radioactivity this can be potentially hazardous. Therefore, a means of distinguishing the importance of a message must be dealt with. For example, data aggregation can be allowed on low-priority messages (i.e. acceptable levels of radiation), but when passed a certain defined threshold a high priority message delivered in the most optimal fashion will be needed. For that reason a need to distinguish between high and low priority data will be required. Alternatives that were studied are data replication (sending duplicate packets), and the use of Acknowledgment packet (ACK packets).

3.4.8 Latency & Accuracy

In terms of radioactivity it is important for operators to be notified of high emissions as promptly as possible. Any proposed model will have to describe the degree of latency between actually sensing the area, processing the input, and the transmission time before reaching the base station. Hence, given a maximum level of acceptable delay how do you efficiently transmit data to always be kept under that limit.

Obtaining accurate information is the primary objective of a sensor. Here accuracy of a sensor will have to measure up against Geiger counters and other current means that the Health Physics team has at McMaster University (see Section 3.2) . These characteristics will be highly dependent on the quality of the sensing hardware. Due to

the limitations of this research the accuracy of the sensing units included in a sensor is not under consideration.

4 J-Sim Architecture

Even with the growth of interest in sensor networks there are still not many tools available for their study [47]. J-Sim was one package which at the present time came closest to capturing all aspects of sensor networks. J-Sim, version 1.3, a component-based, discrete-event simulator was used for the experiments in this research [48, 26].

Modifications to the simulator were required for two main reasons. The first was that the simulator had two conflicting, and inconsistent, energy models. The first energy model was located in the wireless package, and the second in the sensorsim package which was incorrect. The second reason for modifying the simulator was that only one routing protocol for sensor networks had been created (specifically directed diffusion [49]). These modifications, and others, were done in Java and the details can be found in Appendix A. The executable scripts were written in Tcl, and some of the graphs that are used to illustrate the gathered results were done in Matlab.

4.1 Background on J-Sim

J-sim is a discrete-time event-driven simulator which is built around a component based paradigm called *Autonomous Component Architecture (ACA)*. The goal of this simulator package was to bring simulation development one step closer to hardware modularity. As stated in [48] it is generally understood that one does not have good control over software development when compared to hardware development. What J-Sim attempts to do is mimic the integrated circuit (IC) design in order to obtain truly autonomous components.

To understand the advantage of using a component based simulator, one must look at previously used simulation architectures such as object-based designs. The Object-Oriented based paradigm was an attempt to realize hardware modularity in software. As software projects have scaled upwards evident problems have surfaced over time [48, 50, 51]:

- **Code Tangling** - Over time as a project grows several features will want to be added which lack a single composing unit (i.e. cross-cutting concerns), but instead are found in small code fragments scattered throughout several components. The problem is that some concerns which are very hard to encapsulate using a single entity oftentimes lead to tangled code. This tangled code makes reading, maintaining, extending, and even reusing the code extremely challenging.

- **Strong Bindings** - Normally objects are too tightly coupled to other objects. This high coupling exists due to the direct referencing of other class instances which

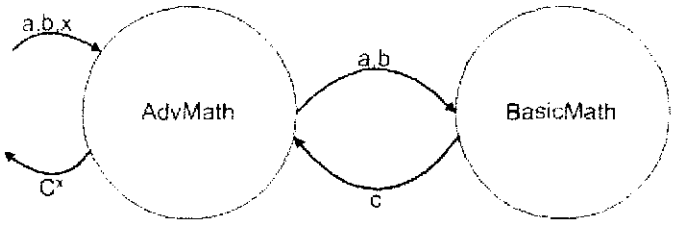

Figure 17: The Object-Oriented Approach to calculating $(a+b)^x$.

should be avoided. This leads to software codes which are prone to the hyper-spaghetti phenomenon, and are therefore difficult to reuse. For example, in Figure 17 each circle represents a class. In this case the user wants to calculate $(a+b)^x$ by calling *AdvMath*. If looking at the *AdvMath* object it takes three parameters (a, b, and x) and it will output $(a+b)^x$ as its return value. *BasicMath* works by accepting two integers (a and b) and outputting their sum in c. This is a clear example of strong bindings where one class *(AdvMath)* directly references (and depends) on the *BasicMath* class in order to calculate $(a+b)^x$.

- **Unseen Bindings** - This occurs when additional unknown bindings are performed without the caller's knowledge. For example, when attempting to calculate $(a+b)^x$ the caller simply passes a, b, and x to the *AdvMath* class as shown in Figure 17. What the caller does not know is that *AdvMath* needs to call a method in *BasicMath* in order to return a value. This forces one to look at the implementation and trace through what can be many layers of code making it challenging, and time consuming, to do code reuse.

Currently many Object-Oriented based simulators exists, such ns-2, which provides enormous support for various types of simulations [52, 53]. One of the major downfalls of such a large simulator is that it is oftentimes challenging to extend and reuse their codes. The two apparent design problems are that *bindings are too strong*, and *bindings are unseen*.

Software components by definition are independent binary units that are used in composition. Procedures, classes, and modules can form components as long as they are in binary form and remain composable. These two constraints: binary and composable assure that robust integration can occur over time. Each component accepts data only through its entry points, called *ports*, and produce output data at some of its ports which is based on some agreement, called *contracts*, beforehand. If we look at Figure 18(a) it represents a software component in J-Sim whereas Figure 18(b) represents a typical IC (Figure duplicated from [48]).

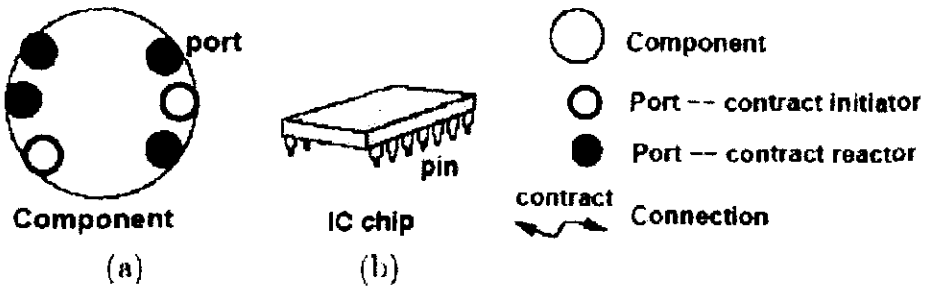

Figure 18: How J-Sim mimics the integrated circuit (IC) design

There exists a one-to-one relationship between Figure 18(a) and Figure 18(b): components versus IC chips, ports versus pins, data versus signals, and contracts versus specifications in the IC cookbook.

Above the ACA lies a generalized framework for modelling packet switched networks. This framework is based on the abstract components extracted from the Internet and is called *extensible Internetworking framework (INET)* [48]. This package provides the base classes for constructing nodes, network links, protocols, and packets all of which can easily be used in a plug-and-play fashion. It is a general enough package so that other specific network architectures can easily be derived on-top of this layer such as IETF Integrated/Differentiated Services architecture, mobile wireless networks, and WDM-based optical network architectures.

J-Sim has also adopted the *split-programming model* (similar to ns-2) to provide a greater degree of flexibility. In large-scale simulators it has been found that different simulations require different programming models [54]. So low-level tasks such as event processing or packet forwarding which requires high levels of performance and are rarely modified are implemented in a compiled language such as Java. On the other hand, tasks that require dynamic configurations such as node placement, traffic types, or traffic sources are often modified thus they are best suited by scripting languages such as Tcl. Both ns-2 and J-Sim support this dual language environment however in J-Sim no explicit code needs to be specified in order for Tcl to access classes, methods, or fields in Java. Additionally the split-programming model allows for cleanly separating the core simulation design, maintenance, and extension from the actual research experiments which requires an easy-to-use, reconfigurable environment.

Therefore, with characteristics such as autonomous components, platform independence, and extensibility J-Sim provided a strong foundation for the basis of this research. J-Sim was chosen for the above reasons and was extended at appropriate loca-

tions to create a family of routing protocols that have already been suggested.

4.2 Sensor Network Framework

J-Sim was chosen due to its efficient design but as well as its completed Sensor Network package which is included in version 1.3. [55, 56] constructed a simulation framework for WSN on top of both the ACA and INET architectures. A WSN simulation consists of three types of nodes: a *target node* (the event generator of the phenomenon being sensed), *sensor node* (the actual sensor which senses and transmits back to a base station), and a *sink node* (acts as the base station). For example, if a high dose of gamma radiation is being emitted from the core, it may generate emissions that can be detected by radiation sensors in the area. This emission in J-Sim would be simulated by the use of target nodes and then sensed by sensors as shown in Figure 19. Once the sensor has sensed a new phenomenon it will proceed to sending it back down to the wireless channel and ultimately to the sink node.

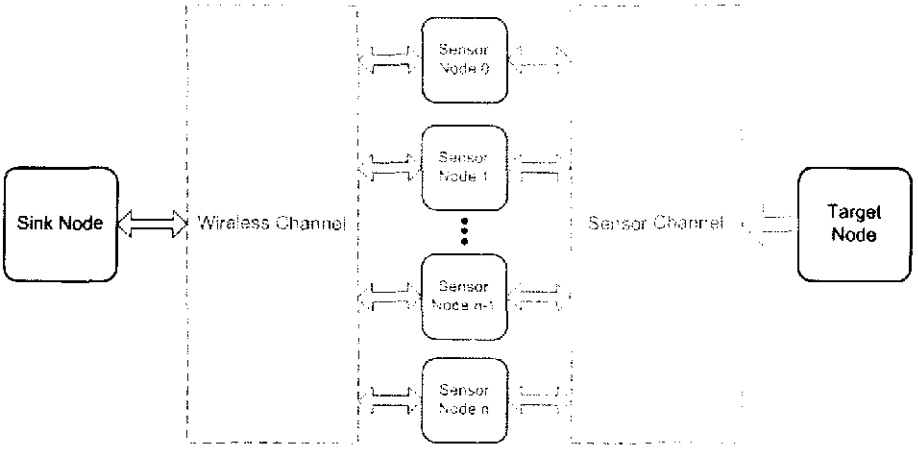

Figure 19: Typical sensor network environment in J-Sim.

Therefore, the role of a sensor is to detect the stimuli on the sensor channel which is generated by target nodes. In this case the sensor determines what level of radiation is in the area and proceed to sending periodic updates to the base station.[5]

[5]The words base station and sink will be used inter-changeably in this research.

4.2.1 Target Node Overview

The target node which can be considered the event generator is in charge of periodically generating a stimulus (i.e. seismic, pressure, radiation etc...) and propagate it over a sensor channel. As can be seen in Figure 20 the *Target Agent Layer* generates the random event and the lower *Physical layer* sends it over the sensor channel for neighboring sensors to receive. The target agent is only equipped to send down data and not receive any information. Currently J-Sim's target nodes only support two types of phenomena

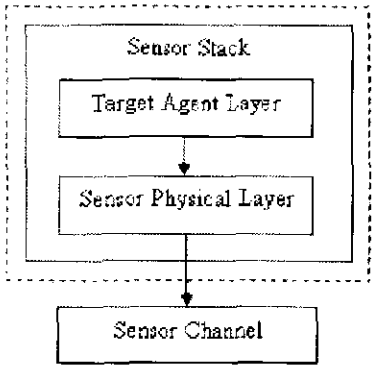

Figure 20: Internal components of a Target Node

seismic propagation and acoustic propagation.

4.2.2 Sensor Node Overview

A sensor node will listen on a sensor channel for sensed data generated by target nodes. Since any phenomena (i.e. radiation, light, temperature, infra-red etc...) attenuates over distance if the sensor is too far from the source of the emission then the incoming signal strength will be below a pre-determined comprehensible threshold. The received signal power is calculated by the sensor propagation model used in [55]. The type of medium being sensed can be different depending on the application being simulated. For example, with this configuration the sensing stack can be used to capture seismic, acoustic, or even ultrasonic behavior. Once a sensor has successfully received information over the sensor channel it must forward that information to the base station. As can be seen in Figure 21 the information travels up the *Sensor stack* and ultimately to the *Sensor Application Layer*. The *Sensor Application Layer* acts as the coordinator between the sensor stack and the network stack. It is an application specific module that normally must be extended for the task at hand. The *Application Layer* for example can send

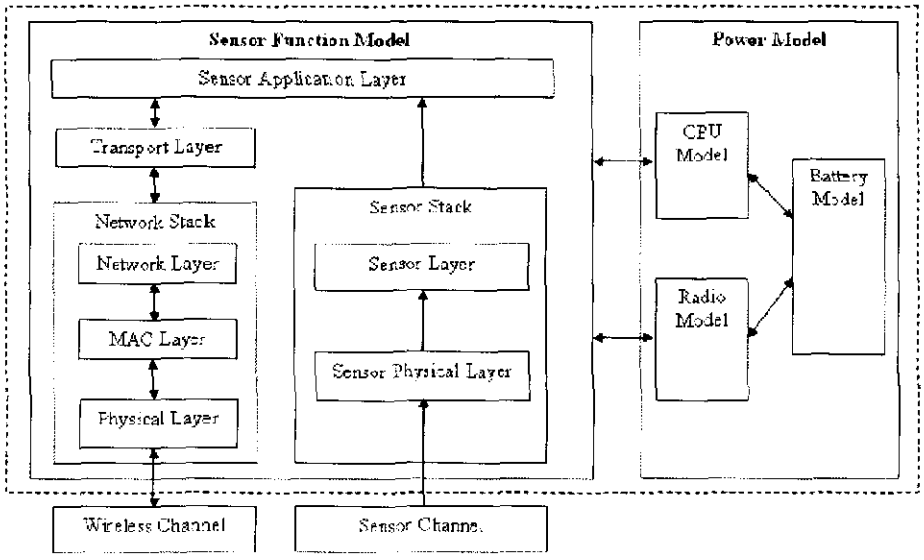

Figure 21: Internal components of a Sensor Node

immediately after receiving information from the sensor channel, or buffer incoming data to perform calculations on them over a period of time (i.e. average, maximum, minimum). Since any given sensor can also act as a routing node the application layer also can control how, and if, it forwards on packets that come up the wireless channel. Currently two ad hoc protocols (AODV and GPSR) have been implemented and are being distributed with the package [55]. Directed Diffusion is also available as a patch online which presently is the only WSN specific protocol available.

Once the application layer decides it wants to forward the data, it is passed down the wireless protocol stack. The *Network Layer* takes care of various IP layer functionality including sending/delivering services to other upper layer protocols. The *MAC layer* determines if the channel is clear to send and if so it is sent down to the physical layer for sending. Currently only IEEE 802.11 MAC protocol is implemented. The *Physical Layer*, manages the actual sending of the data over the channel. Included at this layer are the wireless propagation models. Signal attenuation over the wireless channel is calculated by using one of the three propagation models which are included with the wireless package these are the free space model, the two-ray ground model, and the irregular terrain model (discussed in Section 5.1) . Since the size of the MNR is relatively small, this research will only use the free space model for calculating incoming signal strength at receiving ends.

The framework includes a power model that can be utilized for every sensor. This power model defines energy-producing components (i.e. battery) and energy-consuming components (i.e. CPU and radio). These energy-consuming components can be in several various states (i.e. idle, sleep, off, transmit, receive) and depending on their state will consume various amounts of energy from the battery. When the battery drains, an immediate message can be sent to the application layer shutting the sensor down.

4.2.3 Sink Node Overview

The sink node acts as the base station and is essentially the central hub for where human operators can monitor the sensors measurements. The internal components of a sink are identical to the sensor node with the exception of the sensor stack since it does not perform any sensing. Information that is received over the wireless channel can be further processed by a control server for long term storage or reviewed by an operator.

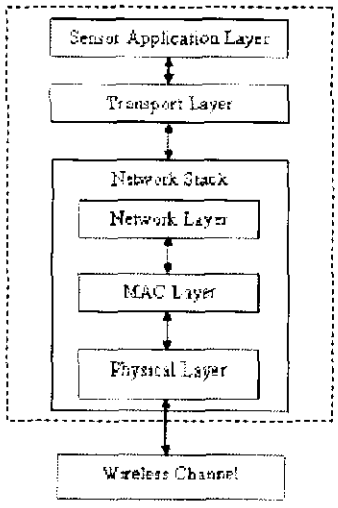

Figure 22: Internal Components of a Sink Node

Once the data has been reviewed, the sink can (optionally) send a command/query to the sensor nodes to perform certain tasks. This is why there exists doubly linked arrows between the the physical layer and the wireless channel in Figure 22.

4.3 Alternative Simulators & Technologies

In order to develop Sensor Network applications a number of simulation environments have been created to provide various insights into issues such as scalability, communications, lifetime, and feasibility. This section provides a brief overview into some of the other general-purpose network simulator packages that are available as well as similar programming paradigms.

Ns [52], whose current version is 2.28, started in 1989 as a variant of the general-purpose network simulator called REAL. Through public support ns-2 has grown and now provides substantial support for TCP, routing, and multi-casting over wired and wireless networks. Ns is probably the most widely accepted and used simulator in the network research community with tremendous variety of protocols available for simulation. It supports a dual-language environment similar to J-Sim but instead uses a C++ Object-Oriented approached combined with Tcl. Unfortunately, this Object-Oriented design has lead to the challenges discussed in Section 4.1 where objects are tightly coupled, and bindings are hiding from developers who want to extend the simulator. This makes it more challenging for code reuse and extending the simulator for new protocols. Through experiments according to [57] ns-2 is faster then J-Sim but requires significantly more memory. Therefore, when scaling to thousands of nodes as sensor networks often require, memory consumption might become problematic in ns-2. Efforts have been made to extend ns-2 to model Sensor Network behaviors, specifically the SensorSim project [58, 59]. SensorSim extends ns-2 by providing power and communication models as well as a new Graphical User Interface (GUI). This package is similar to J-Sim's sensorsim package which is used in this research with the exception of the GUI. Unfortunately, at the present time the SensorSim extension code is unavailable since it hadn't been maintained by the developers of the project.

As networks continue to grow and protocols and technologies continue to diversify it will be critical for simulators to scale towards larger topologies and run for longer periods of time. SSF (Scalable Simulation Framework) [53] is another general-purpose network simulator focused on scalability. Here scalability includes simulating large numbers of nodes, traffic flows, heterogeneous systems, and performance scalability with a number of processors. Their approach is to define simulation primitives to facilitate construction of large scale simulations over parallel and distributed systems. The construction of these primitive objects is not done in a script (like ns-2 and J-Sim) instead they have created their own Domain Modeling Language (DML) to create topologies. This provides for tremendous ease in developing large networks, but is not flexible enough for providing an interactive environment as J-Sim does [48]. Some of SSFNets' goals are to understand global network conditions including scaled conditions and predicting future behaviors. To achieve this scalability SSFNets' framework, simi-

lar to J-Sims' INET framework which is laid ontop of ACA, sits directly above the SSF API. So SSF provides the core simulation APIs while SSFNet was developed to offer modelling of internet protocols such as: IP, TCP, UDP, OSPF, and BGP. In terms of performance the C++ implementation of SSFNet makes up for speeds that are approximately the same as ns-2, and when compared to ns-2 and J-Sim uses the least amount of memory [57].

In efforts for striving towards accurate simulations of sensor networks another discrete-event simulator called TOSSIM has been designed [60, 61]. TOSSIM which comes included in distributions of TinyOS[6] allows one to feed unchanged TinyOS applications directly into the simulator. The simulator compiles the TinyOS application into native executable code which then runs on the simulator host [63]. By simply modifying some of the low level TinyOS commands which interface with the hardware into discrete simulator events TOSSIM can accept programs and simulate thousands of motes. TOSSIM has support for several radio-propagation models and more importantly provides individuals with the ability to simulate their network of potentially hundreds of motes before actually deploying. Also available for TOSSIM is a graphical frontend called TinyViz for aiding in the simulation creation and visualization. Various extensions to this project have already been seen such as PowerTOSSIM [64, 63] which focuses on modelling energy as best as possible for TinyOS applications. Despite TinyOS's popularity as a primary candidate for an Operating System for sensors, it is not the purpose of this research to deploy and validate TinyOS applications on motes therefore a more general tool would be appropriate.

Many other simulators such as OPNET [65], GloMoSim (Global Mobile Information Systems Simulation Library) [66], and MaRS (Maryland Routing Simulator) [67] are also available but were either outdated, not available, or not flexible enough for extending and developing a proper framework for studying the behaviors of sensor networks.

J-Sim's ACA can also be compared to other competing software paradigms. One such example is component based middlewares such as CORBA (Common Object Request Broker Architecture) [68] which enables invocation of operations on objects anywhere on a network as if it were locally executed [69]. In such technologies each object resides on various machines and are well bounded through defined interfaces (called Interface Definition Language (IDL's)). This interface defines to the clients what the server has to offer and hence any clients wanting this service will directly interact with this IDL. Therefore, in this model the caller and callee of a method are both bounded to some interface (specifically the stub and skeleton codes) and at run time these interfaces

[6]Sensor networks must have special embedded Operating Systems (OS) in order to fully control and maximize their lifetimes. One such operating system is called TinyOS which is an open-source, component-based OS written in nesC for sensors [62].

are linked together to complete the bindings. This separation of implementation and interface resembles J-Sims' architecture in that components are developed separately and only at run-time are these components (possibly on different machines) linked. The difference as discussed in [48] are:

- The late binding of the stub and skeleton codes at run-time implies tightly coupled components at compile time. This means that components are now bounded to the underlying communication interface.
- The function calls are now strongly typed. A client has to specify the function signatures exactly, and the other side must agree. This means that the parameters and return values of function calls are verified at compile time. Although this provides for an increase in error checking (possible reduction in debugging time required) its opportunity cost is the loss of flexibility to monitor, debug, or reconfigure components at run time.

On the other hand, the ACA in J-Sim hides all the communication among components in mechanisms called *ports* and *wires*. This allows for the components themselves to be reused easily for different purposes.

As one can see many alternative simulators and technologies are available. For reasons such as portability, documentation, design, or availability J-Sim offered significant advantages and was chosen for this research.

5 Simulation Models

The growth in the interest of sensor networks have given rise to many new protocols and algorithms to meet changing operational and technological needs. For verifying even moderately-sized networks it is very challenging to analytically model such interactions between all nodes. Simulation has become a vital tool to quickly and inexpensively study behaviors of various paradigms on the Internet [70]. Common alternatives to our approach such as test-beds and laboratory experiments have proven to be expensive to build, difficult to reconfigure, share, and have very limited flexibility [54]. Therefore, simulation was used to determine the optimal protocol stack required for the MNR. MAC algorithms were developed and extended at appropriate locations within the package as well as network level protocols. In this research three Sensor Network protocols are tested (Direct, Multi-Hop, and LEACH) along with three MAC algorithms (TDMA, CSMA, and IEEE 802.11) in order to determine which in terms of system lifetime, scalability, latency, and reliability would be best suited given the MNR requirements discussed in Section 3.4.

For successful comparisons of various routing protocols it is imperative to have solid theoretical models for all aspects of these simulations. The battery is the sole energy source contained in each sensor with two main consumers: the radio and the CPU. For this reason models which explain the depletion of energy from the battery source must be accurate. This section describes the models that were used for energy consumption, radiation propagation, wireless propagation, and CPU energy consumption.

5.1 Radio Propagation Model

In order to accurately model the sensor networks, the wireless channel is equipped with certain propagation models which allows sensors to determine the strength of the incoming signal. These models are integrated in the physical layer (lowest level) of the wireless protocol stack in J-Sim and are automatically called upon when a sensor receives information.

Propagation models attempt to predict the average received signal strength at a given distance from the transmitter, as well as the variability of the signal strength in close proximity to a particular location. There are three main phenomena that affect wireless communication: *Reflection*, *diffraction*, and *scattering*. *Reflection* occurs when a propagating wave reaches an object which has very large dimensions when compared to the wavelength of the wave [28]. *Diffraction*, on the other hand, occurs when the radio path between the Transmitter-Receiver (T-R) is obstructed by a surface that has sharp irregularities (edges). The outcome of the wave passing through this obstruction is strongly dependent on the geometry of the object. These secondary waves are present

throughout the space and even behind the obstacle, giving rise to a bending of waves around the blockage, even when a line-of-sight path does not exist between transmitter and receiver. The third concept is *scattering* which occurs when the medium through which the wave travels consists of objects with dimensions that are small compared to the wavelength, and where the number of obstacles per unit volume is large [28]. Objects such as lamp posts and trees will absorb radio waves and diffuse the waves out all directions (scattering) thereby providing additional radio energy for a receiver.

Along with the above three concepts another important concept that will need to be taken into account is that of *multipath* waves. *Multipath* occurs when a propagating signal has taken more then one path (caused by any of the three phenomena - Reflection, Diffraction, and Scattering) which makes the received signal the vector sum of all the signals incident from any direction or angle of arrival. Depending on the phase of the wave some signals will either aid the direct path, while others will subtract from the main signal.

A standard model used to predict received signal strength when the transmitter and receiver have a clear, unobstructed line-of-sight path between them is the *Friis free-space equation*. This approximates the received signal by using the following equation:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{4\pi d^2 L} \tag{6}$$

where:

P_t=Transmitting Power (watts)
P_r=Receiving Power (in watts)
λ= Wavelength
G_t= Transmitter Antenna Gain
G_r= Receiver Antenna Gain
d = Distance
L = System Loss (L>= 1)

For the purpose of this research the following values were used: 914 MHz radio, $G_r = G_t = 1$, and we ignored system loss (i.e. L = 1). This gives a wavelength (λ) value of approximately 0.328 meters (see Section 5.2 for derivation). These values were based on the LEACH experiment done in [14].

5.2 Radio Energy Consumption

The radio on a sensor is the major energy consumer and therefore must be carefully monitored. The cost of keeping a short-range radio on (i.e. in *IDLE* mode) can be on the same levels of magnitude of receiving and transmitting [21, 71]. For example, on a

Mica2 mote, the ratios for radio power drawn are 1:1:1.41 (idle:receiving:transmission) at 433MHz with RF signal power of 1mW in transmission mode [72]. Traditional network abstractions usually do not grant software precise control over hardware. This research looked at two alternatives: software controlled (cross-layered design) and non-software controlled (traditional-layered approach) radios as discussed in Section 2.2. We believe that embedded software must have precise control over their hardware in order to maximize lifetime.

Figure 23 illustrates the key components of the radio models used in this research including the *transmitter* and *receiver electronics*, and the *amplifier*. Amplifiers are

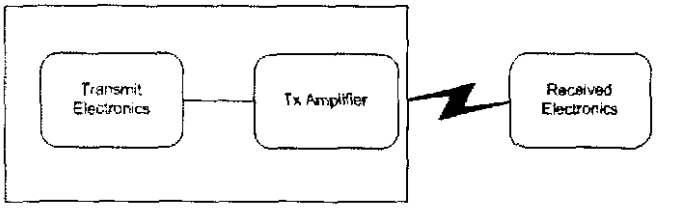

Figure 23: Radio Model

used to boost the power of the outgoing signal (in watts) to a level needed to communicate with other transceivers. The greater the amplifying power the greater the energy consumption. For the purposes of this research the energy required to run either the transmitter or receiver circuitry will be the same.

5.2.1 Non-Software Controlled Radio Model

In this model the application layer of a sensor is not able to shut down or control at what transmission power the radio should send. Therefore, consumption is strictly based on what operating mode the radio is in and on how much current that mode draws as a function of time. For the purposes of this research the amplifying power for this model was constant and set to 0.096 watts which allowed a sensor to communicate to any other sensor across the set dimensions of the MNR.

Energy consumption for Sensor Network simulation is based on realistic values and electrical physics. In order for these untethered devices to obtain power, a battery must be connected which will provide the potential difference through a chemical reaction. This potential difference (i.e. the voltage) is what will push current through the sensor giving the electrical components the ability to function. If we ignore resistance in the simulations and set constant values for current one can calculate precisely how much energy each component requires. By definition,

$$I = \frac{q}{t} \tag{7}$$

where I is the electrical current (measured in Amperes), q is the charge (in Coulomb's), and t is the time (in seconds). One also knows that voltage (V) is the amount of energy each electron is carrying therefore

$$V = \frac{W}{q} \tag{8}$$

where W is the energy expended in joules, and q is the charge in Coulomb's. By combining Equations 7 and 8 one obtains Equation 9 which was used by sensors to determine how much energy was consumed with respect to current and time.

$$W = V * (I * t) \tag{9}$$

Table 1 summarizes the current values that were used for this research.[7] In order to

Table 1: Current that is Drawn from Radio Components Depending on their States.

Radio Component	**Current (Amperes)**
Radio Transmitter	0.016 Amps
Radio Receiver	0.008 Amps
Radio Idle	0.005 Amps
Radio Sleep	0.000008 Amps
Radio Off	0.000001 Amps

compute the amount of energy expended it was also required to determine the time it took to send and receive packets of certain sizes. To determine transmission time (Tx_{time}) the following equation was used

$$Tx_{time} = \frac{DataSize}{Bandwidth} \tag{10}$$

Since the transmission rate (bandwidth) is set to 1.0 Mb and packets are of fixed size it was possible to calculate the actual transmission time required. This transmission time determined how much power needed to be taken away from the battery model during simulation. Equation 10 signifies how important it will be to keep data being sent over

[7] In reality the behavior of a battery is based directly on the type of battery the sensor is using and more specifically the materials that make up the battery and the environment in which they are deployed [47].

channels small, since the greater the packet means longer transmission time. The longer a sensor transmits the more power is drawn from the limited battery.

Being able to determine the wavelength was also required for the simulator to calculate the attenuation of a signal. The equation which links frequency (f) with the wavelength (λ) is

$$f = \frac{c}{\lambda} \tag{11}$$

where c represents the speed at which the light travels. The velocity c of electromagnetic waves traveling through air is approximately $3 * 10^8$ m/s. The antennas used for this research are omni-directional and are using a 914 MHz channel to communicate. This means that the wavelength is approximately 0.328 meters.

5.2.2 Software Controlled Radio Model

Sensors which allow high-level applications direct and efficient control over their hardware are called cross-layered designs. Unlike the non-software controlled radio this model removes energy based on wireless propagation models. Studies have shown that as a power amplifier sends with higher transmission rates it leads to a direct increase in power consumption [71]. Therefore, in battery powered sensors it is necessary to minimize the amount of power required by the power amplifier at all times. In other words, if we know that a receiver's listening threshold is at a certain power level then we do not need a sender to send with anymore power than is required. So given that the sender knows the location of the receiver it can approximate the distance and by using a model such as Friis free space equation (see Section 5.1) one can determine how amplified the outgoing signal should be. This allows for a sender to always minimize the outgoing energy and hence prolong the lifetime of a sensor. Figure 24 shows a detailed view of this model (this model is based on previously done work from [33, 35, 14, 37]). Table 2

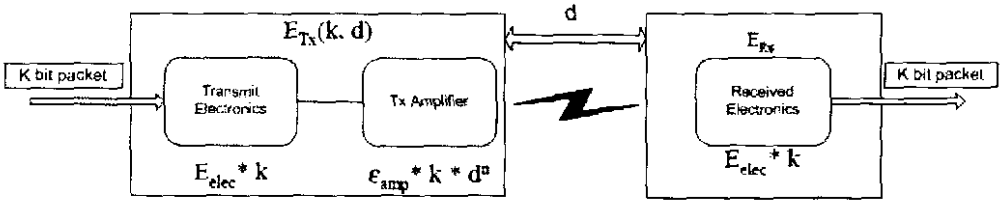

Figure 24: Detailed View of cross-layered sensor radio model.

is a summary of the upcoming notation that is used to describe the values used in these

Table 2: Notation used for describing Radio Characteristics

Operation	Description
E_{elec}	Energy for radio electronics (i.e. just to activate radio without amplifier)
$E_{Tx}(k,d)$	Energy required for Transmission
$E_{Tx-elec}(k)$	Transmitter Electronics as a function of k bits
$E_{Tx-amp}(k,d)$	Amplifying energy consumed to transmit k bits a distance d
$E_{Rx}(k)$	Energy required for receiving
$E_{Rx-elec}(k)$	Receiver electronics as a function of k bits
ε_{amp}	Transmitter amplifier energy used for transmission
$\varepsilon_{friss-amp}$	The amplifying energy which is dependent on the receiver threshold
P_r	Signal power received at the receiver
P_t	Transmission Power of outgoing signal
k	The size of the message (k bits)
d^n	The T-R distance to the n^{th}power
R_b	Radio bit rate

simulations. If one desires to transmit a message that is k-bits in size, a distance d the radio will expend

$$E_{Tx}(k,d) = E_{Tx-elec}(k) + E_{Tx-amp}(k,d) \tag{12}$$

which reduces to

$$k * E_{elec} + k * \varepsilon_{friss-amp} * d^2 \tag{13}$$

For receiving such a message the cost in terms of energy is

$$E_{Rx}(k) = E_{Rx-elec}(k) = k * E_{elec} \tag{14}$$

for the purposes of these simulations the electronics energy (E_{elec}) was set to 50 nJ/bit. This energy is consumed due to factors such as modulation, digital coding, and signal filtering that occurs at both receiving and transmitting ends.

The parameter $\varepsilon_{friss-amp}$ directly depends on the receiver threshold. Therefore, the transmitting signal strength will need to be set so that the receiving power (P_r) is above some threshold $P_{r-thresh}$. By knowing a sensor's receiver threshold one can work backwards to determine the transmitting power (P_t) which must be attained. If a radios bit-rate is set to R_b then the transmission power is set to the amount of energy required to transmit one bit times the bit-rate. Therefore,

$$P_t = E_{Tx-amp}(k,d) * R_b \tag{15}$$

which can be simplified by inserting the value for $E_{Tx-amp}(k,d)$ times the bit-rate:

$$P_t = \varepsilon_{friss-amp} * R_b * d^2 \tag{16}$$

and by combining Equation 16 with the Friis free space equation (Equation 6) one gets

$$P_r = \frac{\varepsilon_{friss-amp} * R_b G_t G_r \lambda^2}{(4\pi)^2} \tag{17}$$

and when setting Equation 17 equal to $P_{r-thresh}$ one obtains a value for $\varepsilon_{friss-amp}$

$$\varepsilon_{friss-amp} = \frac{P_{r-thresh}(4\pi)^2}{R_b G_t G_r \lambda^2} \tag{18}$$

Therefore, the minimum transmission power required for a successful reception is

$$P_t = \frac{(4\pi)^2 * P_{r-thresh} d^2}{G_t G_r \lambda^2} \tag{19}$$

$P_{r-thresh}$ is the receiving threshold which for this research was set to -52dBm (equivalent to 6.3 nW). By inserting the values shown in Table 9 into Equation 19 $\varepsilon_{friss-amp}$ simplifies to 10 pJ/bit/m^2.

5.3 CPU Energy consumption

Although the CPU consumption is much smaller in comparison to radio consumption to be accurate it must be taken into account. If heavy amounts of data aggregation and memory usage is required then the CPU can affect the overall lifetime of a sensor. A sensor's CPU can be in one of the following four states: *CPU_Active*, *CPU_Idle*, *CPU_Sleep*, and *CPU_off*. Each state draws its respected amount of current as shown in Table 3. The energy is drawn in a similar fashion to the non-software controlled radio

Table 3: Current that is Drawn from the CPU Components Depending on their States.

CPU Component	Current (Amperes)
CPU Active	2.9e-3Amps (2.9mA)
CPU Idle	2.9e-3Amps (2.9mA)
CPU Sleep	1.9e-3Amps (1.9mA)
CPU Off	1e-6Amps ($1\mu A$)

model as discussed in Section 5.2.1.

5.4 Radiation Model

Radiation refers to the propagation of waves and particles through space which includes both electromagnetic radiation and atomic particle radiation. Electromagnetic radiation has a wide spectrum (see Figure 25) of energy which includes visible light, radio waves, microwaves, x-rays, gamma rays, infrared and ultraviolet radiation.

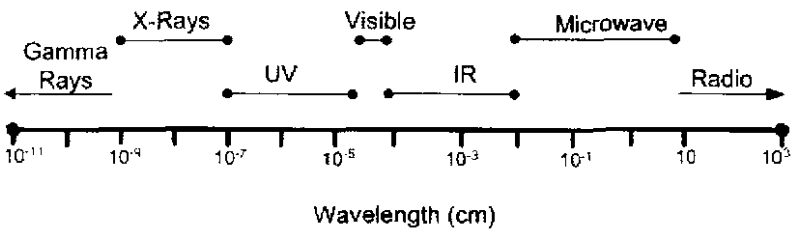

Figure 25: The Radiation Spectrum

Particle radiation includes alpha and beta particles, protons, and neutrons. The speed of electromagnetic radiation travels at the speed of light, whereas the speed of particle radiation is dependent on the source and interaction of the particle with other matter. For the purposes of this research a simple free-space attenuation factor was used to model the behavior of radiation as shown in Equation 20.

$$\frac{1}{d^2} \tag{20}$$

The propagation of radiation is inversely proportional to the distance squared. This resembles the attenuation of wireless communication since by definition all electromagnetic waves shown in Figure 25 travel at the speed of light. This model assumes that air is negligible and that no solid impediments are blocking the sensor (i.e. visible line of sight).

6 Analysis of Deploying Sensor Networks in MNR

This chapter discusses the results of the simulations which provide numerical reasoning to re-enforce the conclusions of this thesis. Sections 6.1 and 6.2 discuss the routing architecture and protocols that were chosen. In order to fairly compare all protocols a parser was created to read in a set topology. A Graphical User Interface (GUI) was developed in order to facilitate creation and viewing of the sensors as shown in Figure 26.

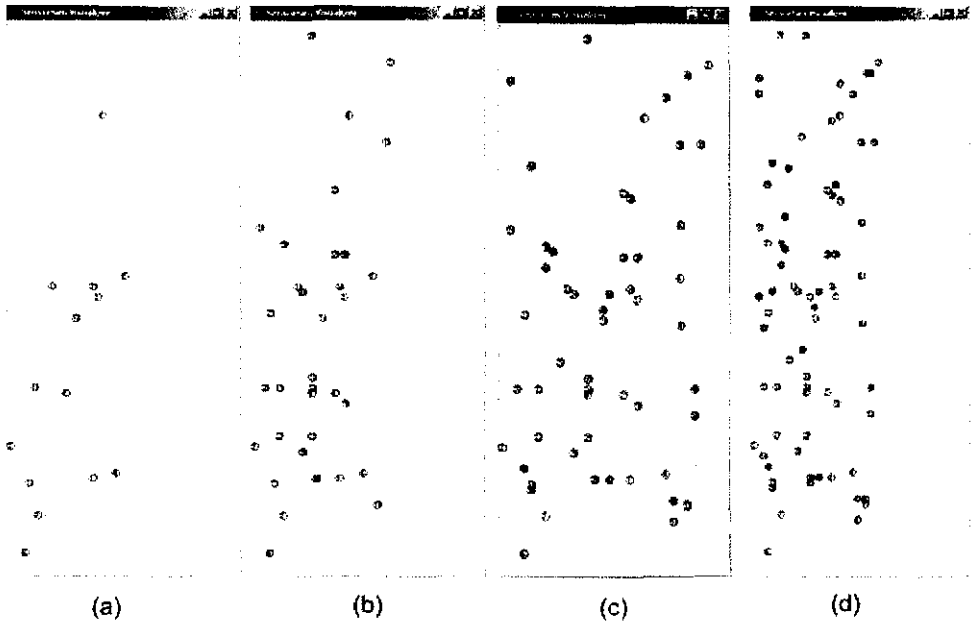

Figure 26: Sensor network GUI.

The four windows illustrate the sensor placements which were used for some of the various sized topologies in this research including: 15 (Figure 26(a)), 35 (Figure 26(b)), 55 (Figure 26(c)), and 75 (Figure 26(d)) sensors. The window plots the base station in the top left (0,0) and then places the other nodes accordingly across the field.

In order to meet the requirements of deploying a Sensor Network in the McMaster Nuclear Reactor as discussed in Section 3.4 many decisions were made. The first was to eliminate mobility within the network. This meant that once a base station or sensor was placed and powered they should not be mobile. If a reactor operator wishes to move a sensor for more accurate readings the sensor should first be powered off, repositioned,

and then turned back on. The powering off of the sensor before manual displacement is required because the sensor location will be re-calculated upon being powered up again. This re-calculation of the new position is accomplished by using fixed beacons which are strategically placed beforehand to allow for triangulation (see Section 2.5). Since no mobility occurs the focus of the protocol is not based on maintaining frequently changing routing tables.

The next decision was to determine if any data aggregation was to be performed. Data aggregation would allow for fewer packets to be transmitted to the base station which is beneficial, but its cost is the delay in packet delivery and higher CPU and memory utilization. Since historical data is a requirement for the reactor the sensors should not attempt to calculate various in-house results such as averages over time, or summations. Rather they are required to send their raw data directly back to the sink. This means that the only form of data aggregation that can be performed is buffering packets and performing one transmission for multiple sensed data. In models such as the One-Hop and Multi-Hop scheme it was determined that data aggregation was not suited for such paradigms, mainly due to delay and packet size. Aggregating the data means they are not sent immediately which means that there is an increased delay between when the actual environment was sensed and when the sink receives the packet. More importantly one must now have sensors transmit larger packets on a shared transmission medium which is prone to collisions. Collisions of larger packets is expensive and should be avoided. For LEACH with the use of spread-spectrum and cluster-heads one can permit data aggregation since only a small percentage of the sensors actually send to the base station. So within a cluster, sensors send periodically their data and cluster-heads can accumulate this information to prepare one larger packet. Since cluster-heads CPUs have to always remain on, there is no increase in energy utilization. The larger packets which the base station receives from cluster-heads are sent using a specific spreading code only known between base stations and cluster-heads. Therefore, collisions with intra-cluster traffic is minimal and prevention of the implosion at the base station is accomplished.

Another decision which was required was to enhance the LEACH protocol in order to provide capabilities for prioritized data. Here prioritized data are packets which contain sensed radiation levels above a certain threshold which is considered to be a threat to operators. The original LEACH protocol (as discussed in Section 2.4.3) was modified to allow acknowledgment packets between base stations and cluster-heads when sensed data were above a certain mark. This meant that cluster-heads received packets as usual, but in the event that a sensor in its cluster sends a packet with a high radioactivity the cluster-head turns the PRIORITY flag on. When the cluster-head decides it is time to send to the base station, a new flag is inserted signaling the base station to reply with an Acknowledgment (ACK) packet. If the cluster-head does not

receive an ACK packet from the sink it will proceed to sending again until successful reception of an ACK packet is heard (or until MAX_RETRANSMISSION (a parameter that is set by the user) is reached).

Following several experiments, it was determined that this modified version of LEACH, which was called MNRLEACH, would be the optimal protocol to be deployed in the reactor. For reasons such as reduction in operator overload, greater energy distribution, scalability, and reliability LEACH exceeded the other simulated protocols. The following sections discuss the results obtained and the reasoning behind this choice.

6.1 Routing Architecture Selection

As discussed in Section 2.2 two design approaches are possible for the protocol stack in sensor networks. The first, known as the traditional layered architecture provides too much overhead and not enough control over the hardware. The second approach, called the cross-layered design flattens traditional models (i.e. OSI model) to allow non-adjacent layers to communicate. An example of this would be the application layer which communicates directly with the hardware layer. For the deployment of sensor networks in the McMaster Nuclear Reactor, it has been found that a cross-layered design would be highly beneficial. Two different simulations were done each using one base station, and twenty-five sensors. The first ran the One-Hop model using the cross-layered approach and CSMA as the MAC protocol. The second simulation consisted of the same approach except no cross-layered communication was allowed and the standard 802.11 protocol controlled the radio and hardware. The goal of running each simulation was to understand which model would preserve energy, and spend it in a proper fashion. The results of the aggregate energy consumption (i.e. all nodes combined) of each simulation after the first 1,000 seconds are shown in Figure 27.

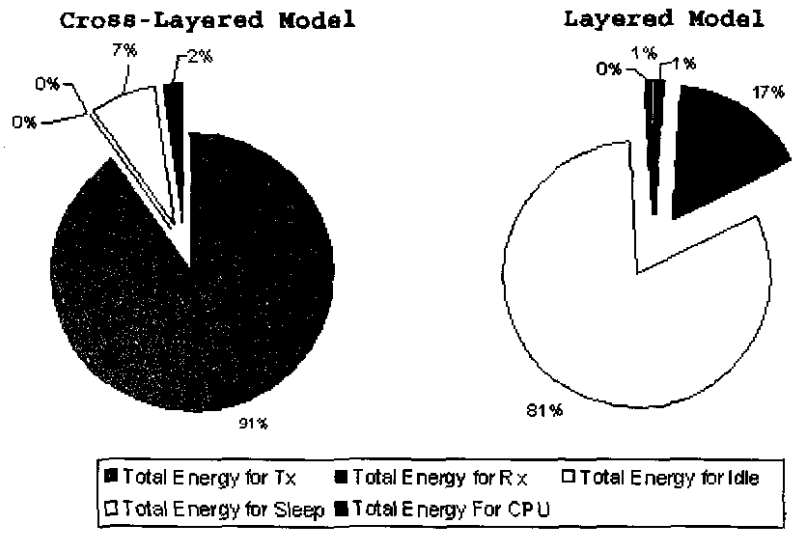

Figure 27: Energy distribution of cross-layered design (left-hand side), and the layered architecture (right-hand side).

The pie chart on the left hand side of Figure 27 represents the cross-layered design. This model shows that the majority of energy consumption, 91%, was spent for transmitting packets. This is precisely where energy in the One-Hop model should be spent since sensors have no need to be receiving any information. The second major consumer were sensor radios in sleep mode. This is because the application layer can immediately shut down the radio components upon completing a transmission. Hence, reducing power consumption since the current drawn in sleep mode is small. Each packet generation involves the CPU being active for a period of time which must also be taken into account in the overall analysis of energy consumption. The 2% is the percentage of CPU energy drawn from all sensors during the simulation. The graph clearly shows that no energy was spent on receiving or remaining in idle mode which are both high consumers of energy.

The second pie chart on the right hand side of Figure 27 represents the layered approach using MAC 802.11. The majority of energy consumption using this paradigm is used by the radio remaining in idle mode. To be specific idle radios consumed 81% of all energy during the length of the simulation. This is because the MAC 802.11 protocol does not have any method of putting the radio circuitry to sleep accurately in an ad hoc environment. The power save mode which is present in this protocol relies on an access point. This process is suited for infrastructured wireless networks and

not ad hoc networks such as the one on which this research is focusing. Therefore, a tremendous amount of energy is lost due to the hardware not being able to shut down like the cross-layered design.

The second major consumer is the 17% which is consumed by receiving packets. In the One-Hop model no packets should be received by sensors since they are not acting as routing intermediaries. Since this model does not provide the application layer with the control over the hardware the radio does not know when to receive and when to shut down. This leads to the problem of idle listening (discussed in Section 2.3.2) which is a tremendous loss of energy that can be avoided.

Another interesting result in comparing these two approaches is to see how much longer the Sensor Network lives using the cross-layered design. The cross-layered design lives 13 times longer than the layered design when comparing the One-Hop/CSMA (cross-layered) versus One-Hop/802.11 (layered) combination. To be specific the cross-layered model lives approximately 11, 808.0 seconds whereas the layered design lives only 854.0 seconds. This shows how significantly more energy was spent in the traditional layered model that could have been avoided. This has lead to the decision that in the reactor a cross-layered approach will lead to prolonged lifetimes and be beneficial.

6.2 Routing Paradigm Selection

This section looks into four performance metrics: latency, lifetime, reliability, and scalability to determine which overall protocol provides the most efficient paradigm. These four metrics were defined and discussed in Section 2.1.

6.2.1 Latency

Latency was measured to understand the travel time behavior between the packet creation at a remote sensor and the received time at the base station. This network must continuously update the base station with critical information for operators. Hence keeping time delays as small as possible is a necessity. It was found that certain protocols had many frequent peaks of high latency while others did not.

The One-Hop models (CSMA, 802.11, and TDMA) all had slightly different results. Their latency graphs are shown in Figure 28 which were obtained by running simulations of 75 sensors.

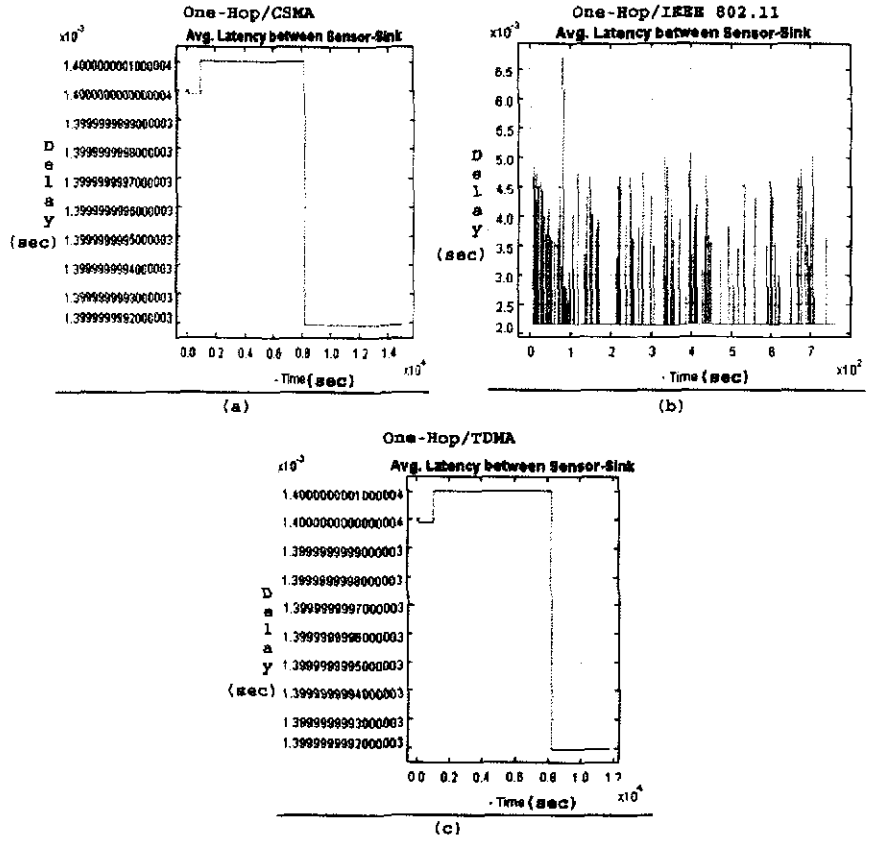

Figure 28: Latencies of (a) One-Hop/CSMA, (b) One-Hop/802.11, and (c) One-Hop/TDMA.

Two immediate conclusions can be drawn from these graphs. The first is to notice the similarity between Figure 28(a) and Figure 28(c). The behaviors of both the One-Hop/CSMA and One-Hop/TDMA schemes are extremely similar. They both communicate one packet back to the base station. The difference is that in the One-Hop/CSMA scheme sensors have to contend for channel access which should ultimately give a slightly higher latency. In this case the overall average latency for both the One-Hop/CSMA and One-Hop/TDMA schemes were 0.001408 and 0.001400 seconds respectively. Although their average times were very close the TDMA base scheme always results in a slightly lower delay since the full bandwidth is given and no back-offs are required. This pattern is discussed further in Section 6.2.4 where scalability at dif-

ferent sensor densities are studied.

The second observation that can be made from Figure 28, is when looking at Figure 28(b) which illustrates the One-Hop/802.11 combination. Here the average overall latency is 0.002198 seconds which nearly doubles the One-Hop/CSMA and One-Hop/TDMA models. The frequent spikes are due to the overhead in reserving the channel, and using ACK packets which maintain high reliability rates but comes at the cost of higher delays. The spikes are also due to the nature of the sporadic Sensor Network traffic.

A more subtle point is understanding the sudden drop in the graph of Figure 28(a). When the simulation started there were 75 sensors competing for the channel. When reaching 800 seconds into the simulation less than 10 sensors remained. These 10 sensors were located within a 15 meter proximity of the base station. This means the travelling distance is shorter and with very few nodes competing for the channel no collisions are occurring (hence no backing-off). This leads to slightly faster delivery times.

The beliefs for latency in the Multi-Hop schemes were that they would take significantly longer to be received by the sink in comparison to One-Hop models. This was verified as shown in the graphs of Figure 29. This shows latency being plotted from simulating 75 sensors for both the Multi-Hop/CSMA and Multi-Hop/802.11 protocols. The calculated average latency for both the Multi-Hop/CSMA and Multi-Hop/802.11 were 0.0655 and 0.0488 seconds respectively. The Multi-Hop/802.11 takes approximately 18 times longer for a packet to be received when compared to One-Hop/802.11 protocol. This overhead is strongly due to the long chains that are formed which increases the number of media access. Instead of having to request the media once per packet for a chain of n sensors it must be requested n-1 times to get to the base station. There must also be n ACK packets sent instead of one from the base station which consumes extra energy. For the same reasons the latency in the Multi-Hop/CSMA scheme is also significantly higher then its One-Hop scheme. Due to these prevalent delays Multi-Hop schemes should not be used in the reactor.

The LEACH latencies were measured strictly between the base station and clusterheads. Within a cluster the delays will be extremely small and similar to the One-Hop/TDMA scheme in Figure 28(c). The more important packets that are larger and carry information for the whole cluster must be received in reasonable amounts of time. Figure 30 shows the latency measured for a simulation of 75 sensors running LEACH.

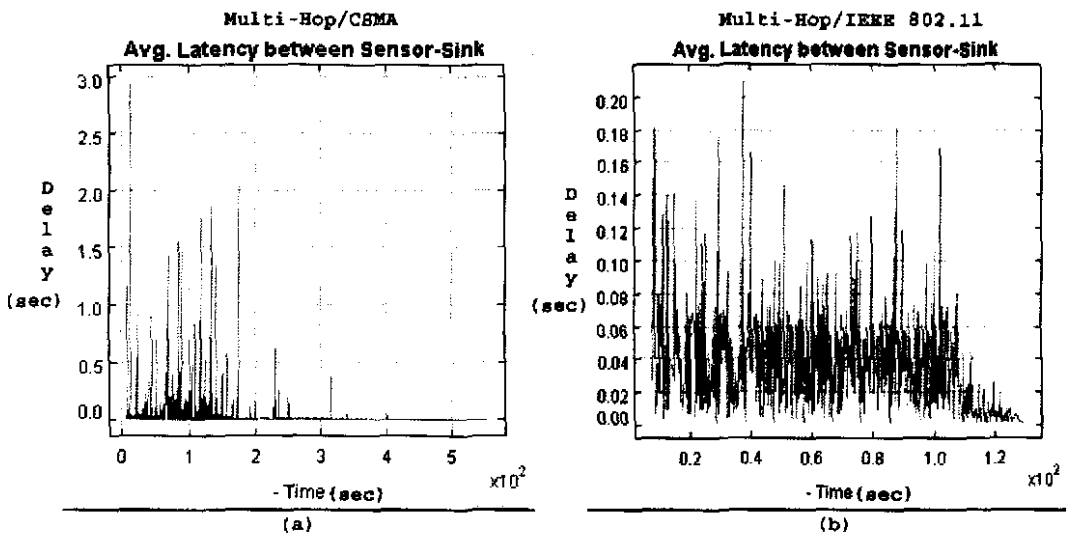

Figure 29: Multi-Hop/CSMA latency graph in figure (a) and Multi-Hop/802.11 graph in figure (b).

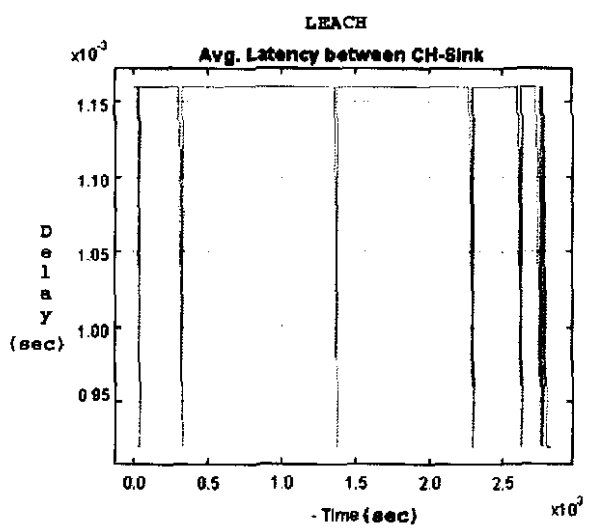

Figure 30: Latency for LEACH simulation of 75 sensors.

The calculated average for the latency is 0.00116 seconds for cluster-heads to deliver their data to the base station. This is slightly smaller then the delay experienced in the One-Hop/CSMA model because there are fewer nodes, and hence less traffic competing for the channel. With roughly only 5% of sensors per round that elect themselves as cluster-heads the collisions towards the base station is very little.

Figure 30 has various troughs which can be linked to the setup periods that were occurring. This means that at certain periods of time no traffic was heading to the sink, or that certain steady-state phases had very few cluster-heads so the delay was dropped.

Overall latency by itself was not a strong enough indicator to eliminate specific protocol combination. The worst performers with respect to latency were the Multi-Hop schemes. Furthermore in Section 6.2.4 one will see that the delays become significantly higher as the network scales to hundreds of nodes with the Multi-Hop models. The fastest was the One-Hop/TDMA and LEACH models. Other factors must be taken into account to conclude an accurate decision.

6.2.2 Lifetime

Probably the most important factor for a successful deployment in the MNR is to be able to prolong the lifetime of the network. The objective here is to determine the protocol which would provide the best coverage for the longest period of time. The 55 node simulation, with an initial energy of 0.25 joules for the One-Hop/CSMA and TDMA models are shown in Figure 31.

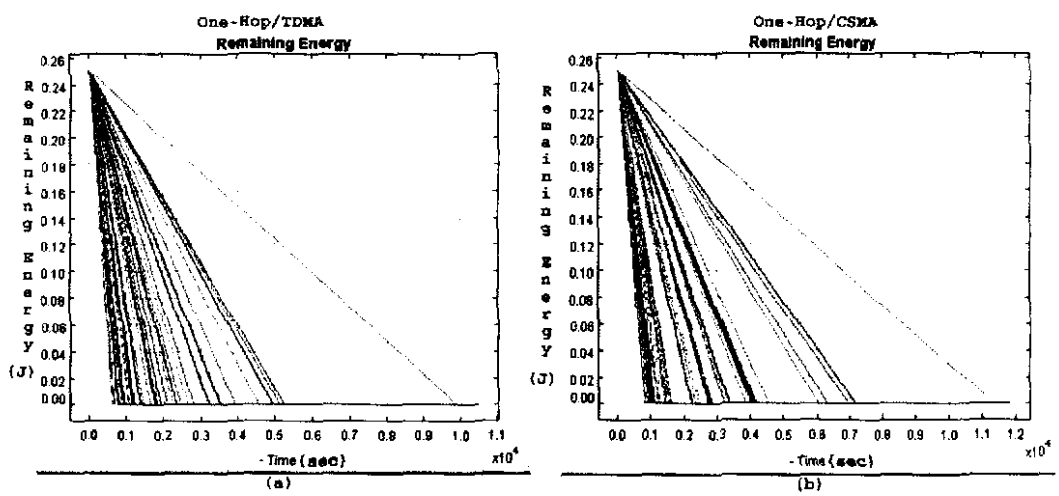

Figure 31: The lifetimes for (a)One-Hop/TDMA scheme and (b) the One-Hop/CSMA.

The y-axis represents the sensor's remaining energy (in joules), and the x-axis depicts the time (in seconds). Each line represents a different sensor. As one can see there is strong resemblance between the two models. This is accurate since both of these models follow the same behavior. The only difference is that CSMA has to occasionally contend for the bandwidth whereas TDMA is always granted full access at the cost of receiving periodic scheduling broadcasts. The approximate lifetime of the One-Hop/CSMA scheme was 11,808.01 seconds, and the One-Hop/TDMA approach lasted 10,480.1 seconds. The reason that the TDMA network died slightly quicker is due to the overhead related to the scheduling cost. Periodically all sensors must turn on their radio and listen till they receive a schedule from the base station. The period of idle listening and reception of the scheduling packet over the lifetime of the sensor takes its toll. Each line in both graphs of Figure 31 represents a different sensor. The lines with the steepest slope are the sensors which are further away from the sink. Those are the sensors which die first since their transmission cost is higher.

Despite its tremendous life span the energy distribution is not evenly spread and these paradigms will lead to what can be referred to as *sensing gaps*. Sensing gaps are areas in the sensing field which are no longer being monitored. Figure 32

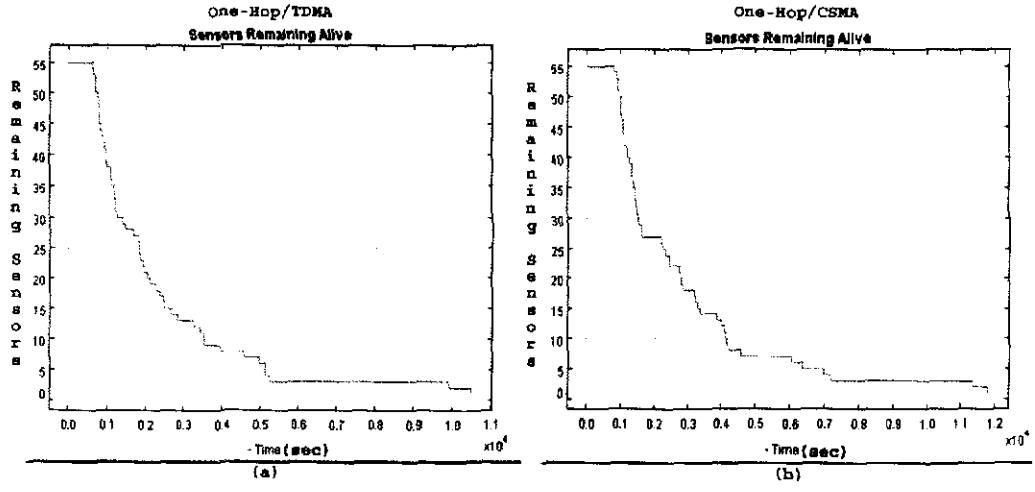

Figure 32: Remaining sensors for both the (a)One-Hop/TDMA and (b)One-Hop/CSMA schemes.

shows the remaining sensors over time for both the One-Hop/TDMA model (Figure 32(a)) and the One-Hop/CSMA model (Figure 32(b)). If we look at Figure 32(a) at 25% of its total life time (i.e. time (t) = 2620.0) there is only roughly 15 of the original

55 sensors left, and at 50% of its total lifespan (time (t) = 5240.05 seconds) there are only 4 sensors remaining in the sensing field. With so many sensors that die quickly this means that certain areas of the reactor will not be monitored for a very long period of time which creates a hazard. If a hot spot emerges at a distance from the sink it is most likely that those sensors will have already died and therefore no updates at the base station will be received for the status of that area. Due to the uneven distribution of energy, which results in sensing gaps, the above two models would be problematic if deployed in the reactor.

When using the 802.11 protocol combined with either the One-Hop or Multi-Hop schemes their lifetimes are considered to be very short in comparison to the others. Both the One-Hop and Multi-Hop schemes which are layered on top of the 802.11 protocol die quickly due to the cost of overhearing and idle listening. The graph in Figure 33

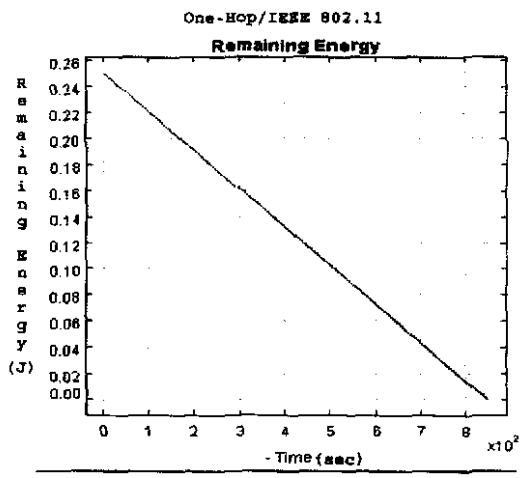

Figure 33: Remaining energy when using the One-Hop scheme and MAC 802.11

illustrates this point by showing the total lifetime of all sensors using the One-Hop scheme to simply being 854.0 seconds. A similar graph can also be produced using the Multi-Hop/802.11 MAC protocol as well. Due to their short lifespan both of these protocols in terms of lifetime are not suitable for the MNR.

The Multi-Hop/CSMA framework suffers from similar problems which are overhearing, and idle listening. Since a sensor does not know when it will receive packets from upstream neighbors it remains on listening to the channel permanently. Figure 34 shows the lifespan of such a network of 55 sensors.

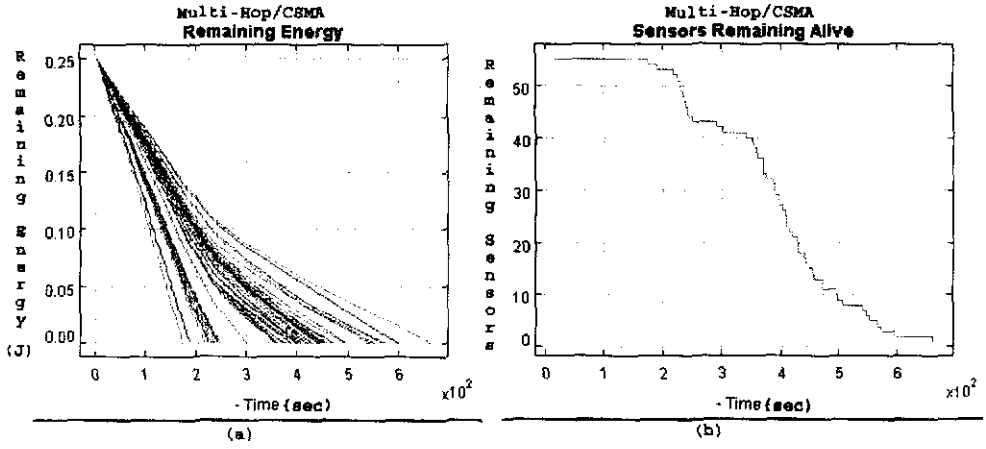

Figure 34: Multi-Hop and CSMA lifetime graph.

The y-axis of Figure 34(a) represents the remaining energy in joules (initial energy was 0.25 J) and the x-axis represents the time. Each line in Figure 34(a) coincides with the energy level of one of the 55 sensors. Figure 34(b) shows the death-rate of the sensors with the y-axis representing the number of sensors left. The uneven death is linked to the constant re-use of nodes near the sink as routing intermediaries. This problem is known as the black-hole effect and was discussed in Section 2.4.2. This combination will therefore again create a routing gap near the base station which must be prevented in order to have a successful Sensor Network.

The lifetime of both Multi-Hop schemes are drastically affected by the radio remaining in idle mode. If one looks at the scenario where no packets are transmitted and all sensors are simply turned on the maximum lifetime of the network can be calculated. For example, the sensors in these simulations all started with an initial energy of 0.25 joules and the radio being in idle consumes 0.0002 amps/second. Therefore, solving for t (time) in Equation 21

$$R_{idle} * t = E_{initial} \tag{21}$$

will give the maximum lifetime a sensor can live with no radio transmissions or CPU activity occurring. By replacing R_{idle} with 0.0002 A, and $E_{initial}$ with 0.25 joules, t yields 1,250 seconds. In reality the simulations lifetime will always be less then 1,250 seconds because transmissions, receptions, and CPU consumption will also take more energy from the battery making the sensor die even quicker. This signifies that such a Multi-Hop scheme is not suitable because it demands too much energy from the radio. Unless this paradigm is optimized so that sensors can go to sleep at certain points during

the transmissions while packets get buffered, this model will not last long enough to be appropriate for the MNR.

The last protocol which was studied in this research, LEACH, has a lifetime which is dependent on the size of the network. Figure 35 shows LEACH's lifetime lasting roughly 3000 seconds with an initial energy of 0.25 joules.

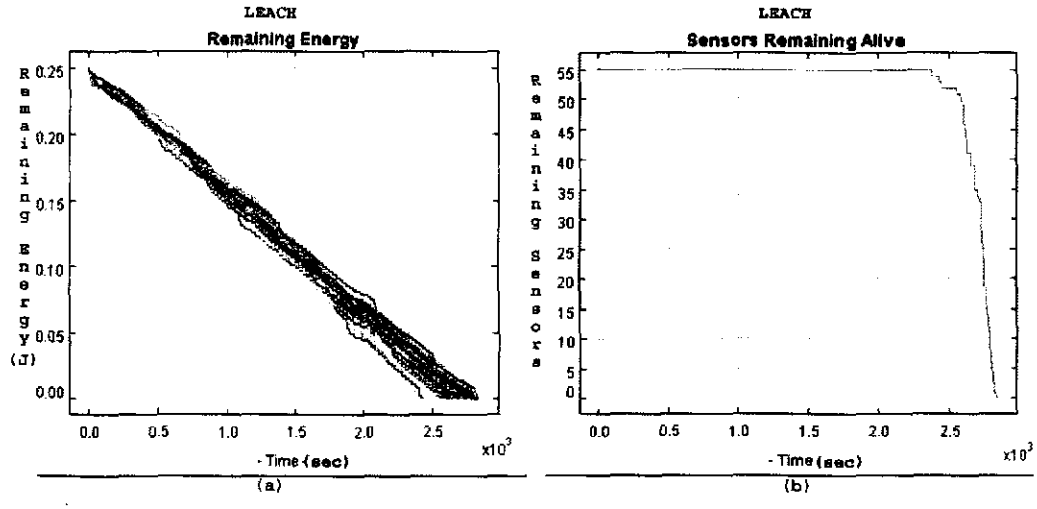

Figure 35: The remaining energy (a) and remaining live sensors (b) when using LEACH.

The y-axis of Figure 35(a) represents the remaining energy in joules, and the y-axis in Figure 35(b) represents the number of live sensors. Despite not having a total lifetime which last as long as the One-Hop & CSMA/TDMA models its even spread of energy across all nodes makes LEACH a much better protocol. Due to LEACH's rounds and constant changing of cluster-heads one will not run into the problem of having sensing gaps. As shown in Figure 35(b) all nodes remain alive and they all die in the same short time frame no matter their distance from the base station. Therefore, unlike the One-Hop model distant nodes are not prone to dying quicker and unlike the Multi-Hop model nodes near the base station are not constantly being used as routers.

6.2.3 Reliability

Since the objective of this Sensor Network is to provide a safer environment for studying nuclear related aspects, the ability to deliver up-to-date data is critical. Therefore, any paradigm that is chosen must be robust and capable of handling changes to both

the environment and topology efficiently. One method of obtaining how reliable the protocols were was to determine how many packets were sent in total and compare that with the number of packets that were received by the sink.

In the One-Hop/CSMA combination the following Tcl output illustrates the success of the protocol:

```
Base Station Received: 13658
Total Packets sent from all nodes: 13697
Collisions at Base Station: 38
Total packets dropped at Application layer: 0
Total packets dropped at physical layer: 1
Drops due to collisions (discovered at MAC layer): 38
Number of Dropped Packets: 39
Success Rate (Reliability): 99.71527
```

The above output is using 55 sensors and it shows the quantity of packets that the base station received versus that the total number of packets sent by all nodes. It also tells the user where the packets were being dropped. In this case, the source of many lost packets was due to collisions at the base station (specifically 38 packets collided at the sink). Overall its reliability is solid at a 99% successful delivery rate.

The One-Hop/TDMA scheme will provide a 100% reliability rate simply because it is not a contention based MAC protocol but rather a scheduled protocol which prevents any collisions. As shown by the following:

```
Base Station Received: 14175
Total Packets sent from all nodes: 14175
Collisions at Base Station: 0
Total packets dropped at Application layer: 0
Total packets dropped at physical layer: 0
Drops due to collisions (discovered at MAC layer): 0
Number of Dropped Packets: 0
Success Rate (Reliability): 100.0
```

the base station received exactly the right number of packets which was transmitted by all nodes (14,175 packets to be specific). Despite this protocol's ability to provide 100% reliability it is based on a central scheme. If the sink is incapable of providing proper scheduling, or if a sensor doesn't receive the schedule then the protocol's ability to function drops. It also relies on a synchronized clock system which adds complexity and would be hard to maintain. Therefore, the protocol is reliable but does not provide a great enough degree of fault-tolerance and flexibility. Another example, would be if the operator decides to add new sensors somehow the sink needs to be notified of these new members as well which makes the whole network dependent on this one station possibly creating a bottleneck point.

Both of the One-Hop and Multi-Hop schemes functioning atop of IEEE 802.11 provide solid reliability and have proven to be extremely robust. Table 4 shows how both protocols deliver all packets successfully due to the precautions taken in the 802.11 protocol.

Table 4: One-Hop/IEEE 802.11 Reliability and the Multi-Hop/IEEE 802.11 Reliability Ratios.

	One-Hop/802.11	Multi-Hop/802.11
Base Station Received	4162	944
Total Packets sent from all nodes	4163	951
Collisions at Base Station	0	0
Total Packets Dropped At Application Layer	0	0
Total Packets Dropped at Physical Layer	1	7
Number of Dropped Packets	1	7
Success Rate (Reliability)	99.98%	99.26%

These precautions include the use of ACK packets and the four-way handshake using RTS-CTS packets. These extra packets provide strong reliability but come at the cost of limited energy. Every extra packet transmission consumes memory and should be avoided.

The Multi-Hop/CSMA provided numerical results for the well known hidden-terminal problem. As discussed in Section 2.3.1 the CSMA protocol does not sense at the receiver and hence can naively conclude the channel to be free. This has lead to the Multi-Hop/CSMA scheme to provide varying results and its reliability can vary depending on the sensor placement. Significantly lower reliability rates were experienced once approaching 175 sensors for example with 200 sensors the following Tcl output shows the low reliability:

```
Base Station Received: 2457
Total Packets sent from all nodes: 3277
Collisions at Base Station: 321
Total packets dropped at Application layer: 0
Total packets dropped at physical layer: 0
Drops due to collisions (discovered at MAC layer): 820
Number of Dropped Packets: 820
Success Rate (Reliability): 74.97711
```

It was concluded that this combination would be un-reliable and hence inappropriate for deployment in a data-critical environment such as the MNR.

The LEACH protocol, which uses specific spreading codes allows for a very robust and flexible environment that easily adapts to dynamic network topologies. Since it forms clusters which reduces long distance transmissions it provides stronger reliability than the One-Hop model. At the same time the cost of providing acceptable reliability does not come at such an energy cost like the 802.11 based protocols. The following is an extract of the Tcl output at the end of a simulation:

```
Total packets dropped at Application layer: 0
Total packets dropped at physical layer: 0
Drops due to collisions (discovered at MAC layer): 23
Base Station Received 322
Total Packets sent from all nodes: 11314
Total Packets sent from CH to BS: 345
 -------
Number of Dropped Packets: 23
Success Rate: 93.33333
```

Note here that there are two different total transmitted packets being accounted for. The line which states *Total Packets sent from all nodes* represents all non-broadcast packet transmissions (i.e. only data packets) which includes all cluster related traffic, and traffic going back towards the base station. The following line, *Total Packets sent from CH to BS* is strictly the data packets going from cluster-heads to the sink. Figure 36 shows how the implosion problem is reduced in LEACH by the use of clustering. The y-axis in the diagram represents the total packets received. The figure has two lines: the blue one represents the theoretical number of packets the sink should have received if it were following a One-Hop/Multi-Hop scheme (i.e. when every sensor sends to the sink).

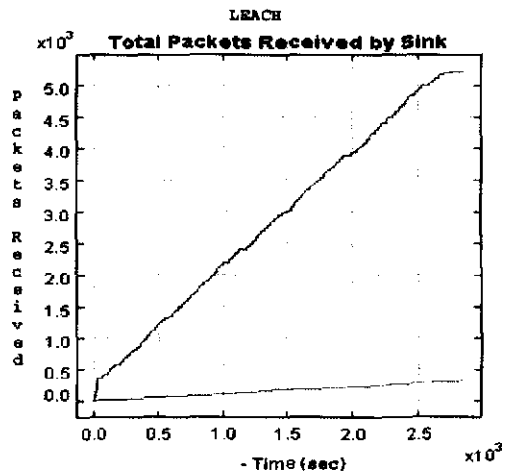

Figure 36: Theoretical and actual packets received when using LEACH.

The second line is the actual number of packets that the sink received using LEACH. This shows how the LEACH protocol reduces the traffic bursts that occurs near the sink. If we compare Figure 36 with Figure 37 one can see the difference in the number of packets that are transmitted.

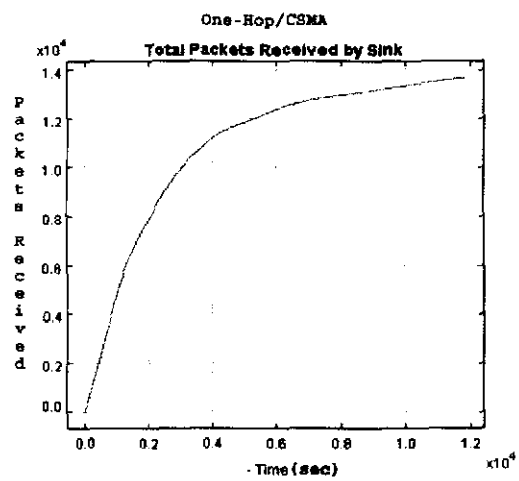

Figure 37: Packets received by sink with One-Hop mode and 55 sensors

If LEACH had no cluster-heads it essentially would be the One-Hop model where every node is sent to the base station (and hence receiving 11,314 packets in this scenario as opposed to 345). This would result in large numbers of packets being directly sent to the sink which would increase collisions. Through numerous simulations LEACHs reliability rate consistently remained in the 90 percentile.

6.2.4 Scalability

A critical factor in determining the suitability of a protocol is to determine if it can successfully scale depending on the reactor's needs. In the case of sensor networks scaling refers to the protocol's ability to function both when the network is sparse and when it is dense. To see if these protocols were scalable two groups of simulated data were gathered. The first set represents a dense network where simulations were run for each protocol with 200, 500, and 1000 sensors. The second set of data was collected to study how the network behaved when it was sparse. This involved running the simulation with 15, 35, and 75 sensors. The following sections discuss the results that were obtained.

The first group of results was to test highly dense networks. Table 5 shows the results of running the network with 200, 500, and 1000 sensors for each protocol combination.

Table 5: Reliability and latency results for dense sensor networks.

	200 Sensors		**500 Sensors**		**1000 Sensors**	
	Relia.	**Avg. Lat**	**Relia.**	**Avg. Lat**	**Relia.**	**Avg. Lat**
OH/802.11	100.0%	0.00225	100.0%	0.00269	99.94%	0.00343
OH/TDMA	100.0%	0.0014	100.0%	0.0014	100.0%	0.0014
OH/CSMA	98.81%	0.00142	97.71%	0.0015	99.50%	0.00168
MH/802.11	88.01%	11.469	4.88%	12.11455	1.187%	12.44625
MH/CSMA	74.97%	0.45577	43.95%	0.26615	27.99%	0.06428
LEACH	96.55%	0.00116	96.84%	0.00121	97.56%	0.00124

The first result that is interesting to note in these scaled environments is how the TDMA scheme consistently delivers with identical delays no matter the network size. This trend is also visible in sparse TDMA networks as shown in Table 7. Every node is given the full bandwidth and hence latency will remain the same as long as the bandwidth is consistent.

Another result shown in Table 5 is an interesting relationship between the reliability and latency for the Multi-Hop/CSMA scheme. The reliability of the network drops

significantly from 74.97% down to 27.99% with network sizes of 200 and 1000 sensors respectively. This is due to the increase number of collisions at every hop as the density of the network grows the chains formed are longer. The longer the chains the higher the probability of loosing a packet along the way. The interesting point to note is when looking at the latency of this combination. It seems that as the network grows the latency is reduced. For example, with 1000 sensors an average latency of only 0.06427 seconds is experienced whereas with 200 sensors a much larger latency of 0.455577 seconds is obtained. This information is false and is due to the reliability. As the network grows the sensors that are further away will have a much smaller chance of having their information delivered. This means that the only packets actually being received when simulating this protocol combination with 1000 sensors are sensors that are very close (i.e. under 10 meters) from the base station. The sensors that are close obviously have fewer hops, if any, and are quickly delivered which provides for decreasing latencies as the reliability decreases. The meaning of this correlation is that packets will not be delivered from distant nodes which is not an optimal solution for the MNR.

In order to obtain the most optimized results for both of the Multi-Hop schemes the *stop_threshold* value was modified depending on the simulation. The *stop_threshold* (see Appendix B.5) represents the point at which a sensor has almost no energy left. At that point a sensor should not attempt to generate new packets but rather clear its queue of current packets and help upstream neighbors deliver their packets before dying. This is a Multi-Hop scheme where priority is placed on the routing of already created packets rather then the creation of new packets. Since the reactor deals with sensing critical data heavy priority is placed on reliability. Therefore, for small networks (15, 35, 55, 75 Sensors) when sensors reached an energy level of only 0.016 joules they were prevented from sending anymore data. For larger networks (200, 500, 1000 Sensors) the *stop_threshold* was set to values of 0.09 and 0.1 to get better results. The increase in the *stop_threshold* is necessary for larger networks since more packets are queued along the chain and waiting to be delivered. This problem of having to use such a stop_threshold value shows how these two combinations are not suited for scalability. When simulating a 1000 sensors one must set half of the total energy of a sensor to simply clear the network of packets that have already been generated and are trying to make their way back to the sink. This prevents new data from being sensed and hence would not be able to alert operators of any radioactive dangers.

An interesting result was how the life of LEACH increases as the network grows. In LEACH, sensors elect themselves to be a cluster-head. Once a cluster-head has been elected it will not be elected again until all other peers have had a chance. This means the greater number of sensors the fewer number of rounds over a period of time that the sensor will elect itself as cluster-head. This is advantageous since being a cluster-head drains high amounts of energy. So the fewer number of times a sensor is a cluster-head

Table 6: LEACH lifetime using various network sizes.

Network Size	Avg. CH occurrence	Lifetime
15 Sensors	21 Times	2243.0 seconds
35 Sensors	11 Times	2631.0 seconds
55 Sensors	8 Times	2837.0 seconds
75 Sensors	6 Times	2918.0 seconds
200 Sensors	3 Times	3650.0 seconds
500 Sensors	1 Times	3849.0 seconds
1000 Sensors	<1 Time	4416.0 seconds

the longer the overall network lifetime. The scalability of LEACH in terms of lifetime is shown in Table 6. This is a tremendous benefit of the protocol for the MNR since it will be able to meet any plans for future network expansions.

The second group of results tested the protocol's ability to function in sparse environments. Table 7 shows the results that were obtained.

Table 7: Reliability and latency results for sparse sensor networks.

	15 Sensors		35 Sensors		75 Sensors	
	Relia.	**Avg. Lat**	**Relia.**	**Avg. Lat**	**Relia.**	**Avg. Lat**
OH/802.11	100.0%	0.00217	100.0%	0.00218	99.94%	0.00219
OH/TDMA	100.0%	0.00140	100.0%	0.0014	100.0%	0.00140
OH/CSMA	99.79%	0.00140	99.72%	0.0014	99.50%	0.00141
MH/802.11	99.77%	0.01513	99.57%	0.02614	99.75%	0.04878
MH/CSMA	99.90%	0.0074	99.09%	0.02080	98.90%	0.06553
LEACH	88.30%	0.00115	89.66%	0.00115	94.53%	0.00116

The first pattern which should be pointed out from Table 7 are the jumps in latencies for both of the Multi-Hop schemes as the number of sensors change. These jumps are due to the multiple hops that each packet has to go through to get to the base station. Every hop increases the possibility of a collision as well as increases latency as the table shows. The Multi-Hop/802.11 combination provided the greatest degree of latency and in comparison to the One-Hop models it is significantly slower. The Multi-Hop/CSMA is consistently faster than the Multi-Hop/802.11 since less media reservation is performed with this MAC protocol.

Another interesting trend is how LEACH's reliability increases as the network density increases. When LEACH is running with only 15 sensors an accuracy of roughly 88% is obtained which increases to over 96% as the network is scaled towards 500 sensors. A possible explanation for this is due to the larger physical dimensions of the cluster sizes that are formed when fewer nodes exists. With a greater number of nodes there will be more CHs and therefore sensors will be able to join clusters that are nearby. This reduces the chance of loosing packets and increases the intra-cluster reliability.

An additional observation that was noted for both Multi-Hop schemes were the regressing lifetimes as the networks were scaled upwards. Both the Multi-Hop/CSMA and Multi-Hop/802.11 protocols when simulated with very few nodes live approximately 1000 seconds. As the network's density increases, the lifetime of both networks decrease rapidly. Table 8 shows how the lifetime

Table 8: Multi-Hop Lifetime (seconds) when Scaled

	Multi-Hop/CSMA	**Multi-Hop/802.11**
15 Sensors	950.5	741.75
35 Sensors	680.25	366.25
55 Sensors	657.50	261.00
75 Sensors	553.0	187.75
200 Sensors	535.0	80.25
500 Sensors	465.5	45.50

is affected as the network grows for each of the Multi-Hop schemes. The Multi-Hop/CSMA combination dies more slowly than the 802.11 protocol because it has less overhead. For example, if an average of 5 hops must be done for a packet to be transmitted in the CSMA model, it only involves 5 transmissions. Whereas in the 802.11 protocol it would involve 4 packets per hop (RTS, CTS, DATA, ACK) and hence a total of 20 transmissions for a successful reception. Although the extra packets are used to increase reliability for the application at hand the cost in terms of energy is too much.

To summarize, the One-Hop models suffer from the implosion problem when scaled and do not provide enough flexibility. The Multi-Hop models loose their edge with respect to latency when scaled having to make too many hops. The Multi-Hop models also suffer in terms of lifetime as the network grows. CSMA becomes affected by the wireless phenomenons such as the hidden terminal problem, and the 802.11 protocol is an energy expensive alternative. LEACH results in a protocol which is the closest to successfully achieving great scalability capabilities.

7 Conclusion

The key to deploying a successful Sensor Network is to identify the application specific requirements. It is the belief of this research that general purpose protocols and applications will need to be tailored to specific environments in order to maximize efficiency. The MNR requirements were to have a protocol that is robust to changing environments, scalable, minimal delays, and provide reliability (i.e prioritized packets). With these requirements six combinations of protocols were evaluated to determine the appropriate model to be used. By extending, and modifying, J-Sim this study has provided many numerical insights on the combination of various protocols, and the use of certain architectures. First, it is clear that traditional layered routing approaches, such as OSI, are no longer appropriate for sensor networks. Instead the use of a cross-layered design, which would be built in accordance to the application, should be used to prolong life. Second, a clustering protocol such as LEACH was able to outlive other protocols and maintain strong reliability. A modified version of LEACH, called MNRLEACH, where ACK packets are used when sensors sense higher radiation levels was recommended for this environment. This study has shown that MNRLEACH overall provided the reactor with the proper balance of simplicity, latency, scalability, and reliability.

8 Future Work

This research explored the possibility of deploying a wireless Sensor Network in the McMaster Nuclear Reactor. An interesting research topic which would complement this work would be the study of Sensor Network coverage and placement. Any event monitoring system must be fault-tolerant and provide a certain degree of coverage. Here coverage means how well a sensing field is being monitored. This work would mathematically model the number of sensors per area the reactor would need to obtain a certain degree of coverage. This would provide insight into how dense the network should be, how to optimize densities for hot spots, and ultimately answering how many sensors in total the reactor requires.

Another possible avenue for extending this research is to test other protocols. This work only tested a total combination of six routing protocols. Many proposals for both MAC and application layer protocols have been suggested in recent papers. An obvious extension to this work would be the addition of these protocols in J-Sim and a comparative analysis between them. This would lead to a more complete study of suggested routing protocols and possibly give insight into a new protocol which would be tailored strictly for monitoring nuclear reactors.

In order to obtain accurate simulations one should attempt to mimic the environment being studied as closely as possible. An important extension to this work would be to develop an indoor radio propagation model which characterizes RF behavior in the MNR similar to the one proposed in [73]. Understanding the physics of the indoor MNR environment will allow one to determine how the locations and number of sensors will be affected. Therefore, measuring attenuation factors to take into account the building structure (i.e. floors, walls, the reactor core) and developing an empirical model specific to the environment will lead to greater accuracy than simply using the Friss free space equation.

A final extension is clearly the deployment of actual sensor motes in the reactor. By purchasing motes, and programming them using an embedded operating system such as TinyOS, one can actually attempt to deploy the network. This would lead to possible discoveries that might have been overlooked by having only used simulation.

A Modifications to J-Sim

To implement the various routing protocols and determine the optimal routing paradigm for the McMaster Nuclear Reactor several modifications were made to J-Sim. As discussed in Section 4.2, J-Sim already had a basic Sensor Network package layered on top of the INET architecture. Four primary objectives were desired:

1. **A Cross-Layer Design** - Allowing application layers to closely interact with hardware layers to gain more efficiency.
2. **A new structured MAC layer to facilitate expansion** - J-Sim does not support many new proposed MAC protocols for sensor networks which can partly be due to the lack of easily being extensible.
3. **Application level routing packages** - To develop LEACH, One-Hop, TDMA, and the Multi-Hop Scheme.
4. **Visual Frontend package** - A graphical frontend for seeing simulation (i.e. node placement and possible node movements)

The above four objectives were accomplished by introducing new components (as defined by J-Sim's ACA) and creating new packages. The definition of a *package* for the purposes of this research is the one used in [74]. Specifically packages are used to group similar objects together which provides encapsulation to outside objects, and visibility within the package. The first modifications that were made were to integrate the two battery models that were present and conflicting (wireless package vs. sensorsim package). Once this was completed new contracts and ports were made to support a cross-layer design in order to increase efficiency. This meant that instead of having a strict hierarchy, such as the OSI model, higher layers were directly able to communicate with the hardware. Once the proper framework was in place four new application level packages were created: *LEACH, multihop, OneHop,* and *OneHopTDMA* each of which contained their respected Java code to simulate the task at hand. The other modifications were to extend the lower layers to add MAC protocols such as TDMA, CSMA and keep IEEE 802.11 which was already implemented. The following sections discuss the details of the code which was used in this research.

Our field dimensions were set to the same size of the McMaster Nuclear Reactor which is approximately 30m x 100m. By picking the two furthest points in the reactor one could roughly calculate its distance to be 104 meters. If the base station is optimally placed then that maximum distance can be roughly divided in half. Due to the dimensions of this application, we felt that strictly using the free-space wireless propagation model would accurately model attenuation.

The radio which was used in these simulations is a 1Mb radio that functions on the 914MHz frequency band. This makes for a wavelength of roughly 0.3282 meters. In J-Sim's wireless package there are three thresholds which must be set to have accurate simulations: *RXThresh*, *CSThresh*, and *CPThresh*. If the incoming signal is not greater then *CSThresh* the signal is too weak to understand and hence the packet is discarded. For our simulations *CSThresh* was set to $1 * 10^{-9}$ Watts. *RXThresh* on the other hand is when the signal was strong enough to hear (i.e. greater then *CSThresh*) but unfortunately too weak to demodulate. This value is set to $6 * 10^{-9}$ Watts. The final threshold is the *CPThresh* which stands for the capture threshold. If a sensor is currently in the process of receiving a packet with a power of $P_1$and another packet arrives with power $P_2$the following comparison is made to see if a capture occurs $\frac{P_1}{P_2} > CPThresh$. *CPThresh* is set to 10 in our experiments. These selected values are based on research from [14] and the threshold values are also the default values used in both ns-2 and J-Sim.

A.1 Package Integration & Overview

In J-Sim all classes fall into specific Java packages as discussed in [48][8]. The core package which implements the autonomous component architecture is called *drcl.comp*. This package implements general components, ports, and generic queues. The implementation of the event-driven and real-time process-driven simulation engines are located in the *drcl.sim* package. Two other important packages are the *drcl.net and drcl.inet* packages. The first, *drcl.net*, includes all the necessary generic classes and components for packet-level network simulation. The second, *drcl.inet*, implements the INET framework discussed in Section 4.1. The drcl.inet package is where the extensions of this work takes place. This package includes implementation of various Internet protocols, and several network applications.

To further classify these packages and illustrate their dependencies a layered diagram as shown in Figure 38 can be used.

[8]As of version 1.0 over 38 packages were included with over 351 classes. This number has grown with version 1.3 released. This section will only discuss the relevant packages. To obtain details on other packages see [48].

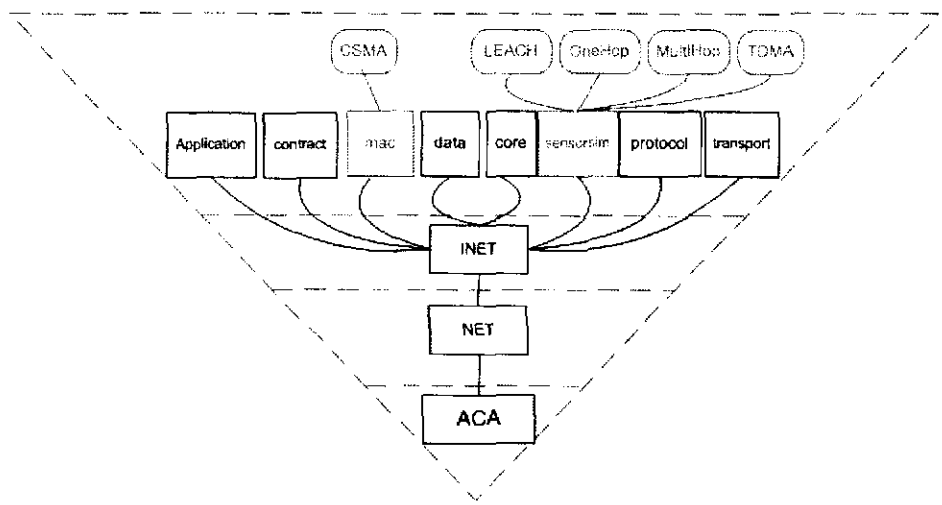

Figure 38: J-Sim's layered packages and the extensions that were added to the simulator.

This upside-down pyramid illustrates the organization and extensions of the work in this thesis. At the bottom of the pyramid lies the *ACA* layer which implements the autonomous component architecture. Above that is the *NET* layer, which is essentially the drcl.net package, and it defines the components required for a packet-level network simulator. Above this layer lies the *INET* framework which defines many various abstract networking components such as nodes, links, protocols, and many contract classes. Layered above this are the protocols which were developed for this research. The drcl.inet.mac package was extended to include a package called *drcl.inet.mac.CSMA* which defines all the appropriate classes related to this MAC layer protocol. Similarly the drcl.ruv.sensorsim package was extended to have the *LEACH*, *OneHop*, *OneHopTDMA*, and *MultiHop* packages available.

Another package which was created, not shown in the above diagram, is the visualSensorNet package. This package contains all the classes that were required to build the simulation using purely Java and no Tcl. This package also provides the classes for the visual frontend which was used to graphically represent the network being studied.

A.2 User Manual

This small manual will explain how to use the many modifications that were done to J-Sim in this research. A user has the option of strictly running the simulator using Java with no Tcl or with Tcl. Appropriate classes were created in order to construct

sensors, targets, and sink components using strictly Java. Tcl sample scripts are included with the package and are located at *$J-Sim/script/drcl/inet/sensorsim/*. By installing J-Sim correctly (see instructions at J-Sim's homepage) one can simply type *java drcl.ruv.System <filename.tcl>* and the script will be executed. To modify the settings of the simulation simply modify the desired Tcl script (i.e. simulation time, number of sensors, target, mobility etc...)

The second option, using strictly Java, is to navigate to the $J-Sim/classes/visualSensorNet/ directory and type *java visualSensorNet.SensorSim*[9] and a window that looks like Figure 39 will appear.

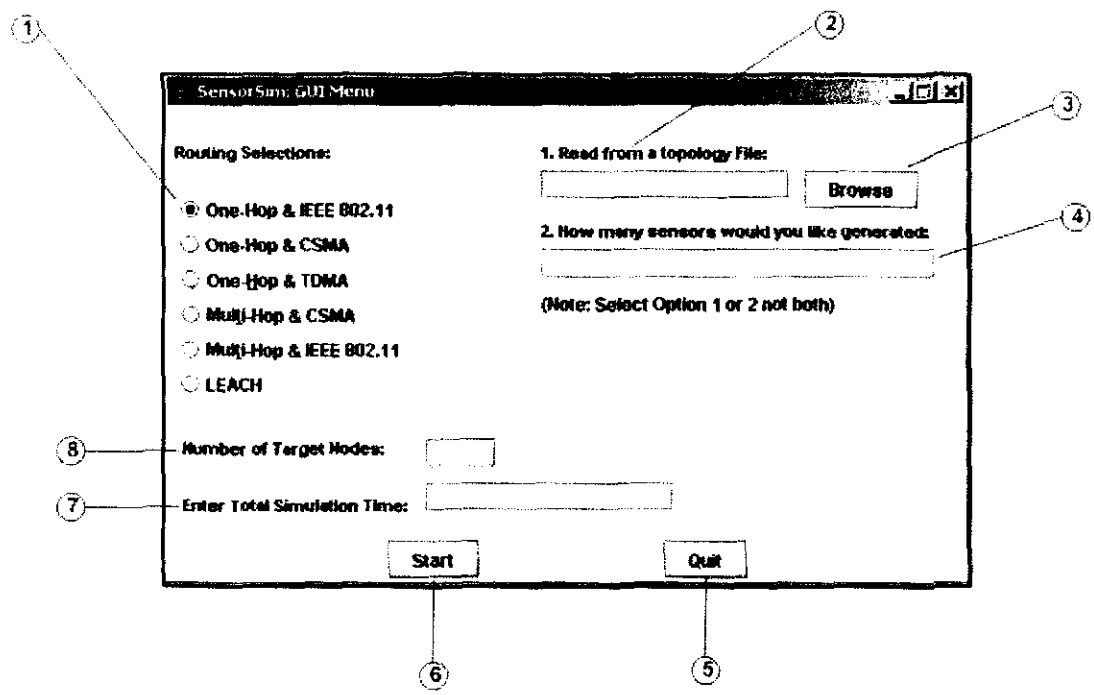

Figure 39: Main menu window for setting up and starting a simulation.

This window allows you to customize and easily compare various routing protocols for a Sensor Network. Explanation for the various parts of the screen are given below:

[9]If running very large simulations one will have to increase the heap size to prevent receiving run time errors. To do so change the executing command to *java -Xmx512M visualSensorNet.SensorSim* which will increase the maximum heap size to 512Mb. For more information type *java -X*.

1. This is the Routing Selection menu which allows a user to choose which routing paradigm he/she would like to simulate. Currently there are six choices to choose from as illustrated in the figure. These are mutually exclusive and one would not be able to select 2 or more protocols at a time. Currently the design assumes that all sensors will run the same protocol.

2. Since a parsing package was added one can now use topology scripts to feed the simulator with the sensor location, and sensing field dimensions. If the user knows the path he/she can type it directly into the text-box. A sample topology script is included in appendix C.

3. To facilitate the task of locating a topology script a Browse button was placed in the GUI which lets the user quickly locate the file.

4. If the user does not have a script file that can be used one can instead have the simulator randomly generate the positions of all the sensors that wish to be generated. Note if there is content in textbox 1 then you cannot enter information in text box 2. Simply type in an integer in this textbox (no doubles) and the simulator will create that many nodes.

5. To quit the simulator and close the program window press this *Quit* button.

6. Once having entered all the required information click on *Start* which will setup and begin the simulation. This main menu window will automatically close and several other windows will appear illustrating the state of the Sensor Network and constantly updating it.

7. This is where one enters the total simulation time that is desired. Note that this field is required and must be an integer. Any other characters will cause an erroneous halt.

8. This textbox is used to enter the total number of target nodes wanted. These target nodes will be randomly scattered across the sensing field. Currently this Java based frontend does not allow changes to the type of target node to be created. They are all set to generate seismic events.

A.3 Energy Model Modification

J-Sim came with two major contributions, the Wireless Sensor Network package and the Wireless package. In both scenarios energy is a key resource that for many applications is critical. Both of these packages implemented their own energy models which were

not linked together. In order to overcome this problem a new combined model needed to be developed. Figure 40 shows the internal components of the original sensor node package. In blue are the components (the arrows are the corresponding contracts) that

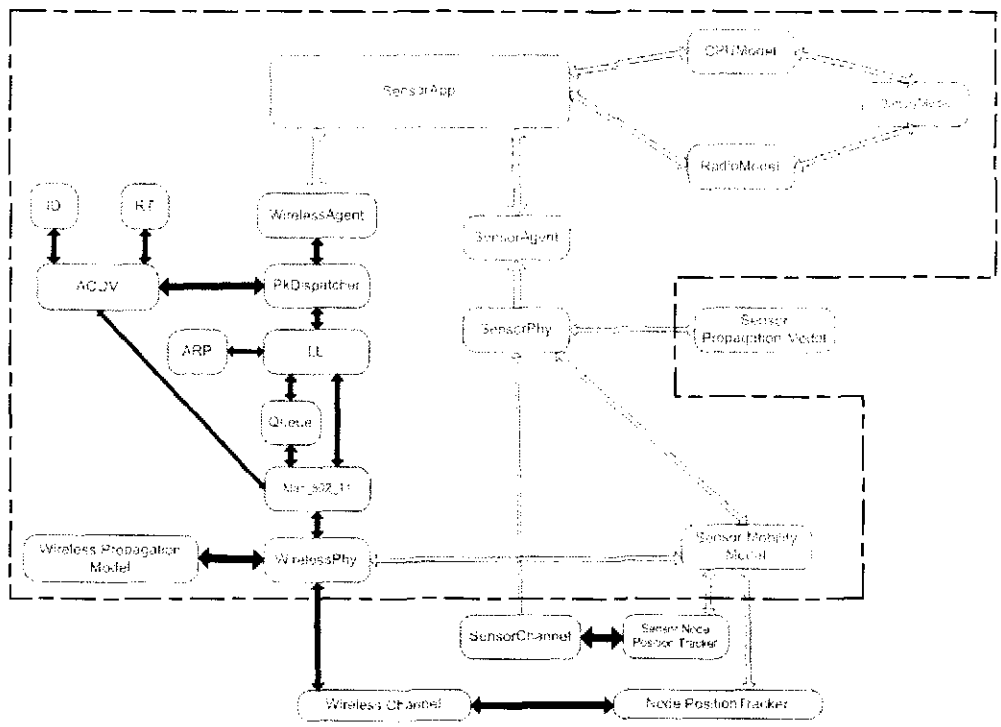

Figure 40: Original Sensor Node detailed view.

belong to the wireless package of J-sim, whereas the green boxes are the Sensor Network components. The *WirelessPhy* component has a built-in battery object which gets updated as time goes on depending on the state of the radio. The second battery module was attached further up off the *SensorApp* layer (called *BatteryMode*). These two models were combined into one as shown in Figure 41. Now a contract exists between the *SensorApp* and the *CPUModel* such that, depending on the CPU state, an appropriate amount energy is removed from the energy model which lies within the lower hardware components (i.e. in *WirelessPhy*). The ports used between the application layer and *CPUModel* are *.cpu* and *.reportCPUMode* respectively. These ports communicate by sending integers which represents the CPU status. The status integer can be one of the following values: *CPU_IDLE, CPU_SLEEP, CPU_ACTIVE,* and *CPU_OFF*. Periodi-

cally it is *CPUModel*'s job to deduct the consumed energy from *EnergyModel* located within *WirelessPhy*. This is done through the *.battery* port located in *CPUModel* which connects to the *.cpuEnergyPort* located in *WirelessPhy*. The contract which defines this link is called *BatteryContract* and is located in the drcl.inet.sensorsim package.

To test both traditional and cross-layer designs an additional contract was created to allow the application layer to communicate directly to the hardware layers. The model that was already in place supported the multi-layer approach very well. In order to support a cross-layer design another connection shown in Figure 41 which connects *SensorApp* to *WirelessPhy* component was created. This allows for the application layer to directly control the status of the radio and obtain the latest information from the battery. With both a cross-layered and traditional architectures in place direct comparisons were able to be made of their effects on performance.

A.4 Application Level Modifications and Additions

The One-Hop model, where every sensor must send directly to the base station, is shown in Figure 41. Three different One-Hop scenarios were created: One-Hop with CSMA, One-Hop with 802.11, and One-Hop with TDMA. Below we discuss the details of each combination.

A.4.1 One-Hop and CSMA

Inside the *sensorsim* package a new sub-package was created called *OneHop*. This package contains 3 Java files: *OH_Packet*, *OneHopApp*, and *SinkAppOH*. *OneHopApp* and *SinkAppOH* both extend *SensorApp* for specialization purposes and code reuse. *OH_Packet* extends *Packet* and serves as a packet type which sensors running the *OneHopApp* send to a base station.

The simulation begins by starting the sink followed by the sensors. The sinks only role in this scenario is to listen at all times for incoming packets on the shared channel. Once the sink receives a packet it will post results and update graphs to show the latest status of the Sensor Network. When each sensor starts, a send timer begins and is continuously called upon until it no longer has enough energy to send. Each sensor generates a random number between 8 - 12 to prevent collisions amongst other nodes. Therefore, every 8-12 seconds each sensor sends a packet directly back to the base station. The packet, which is sent from the OneHopApp, is called a *OH_Packet* and contains the following information:

- ***sid***: The sensors unique ID

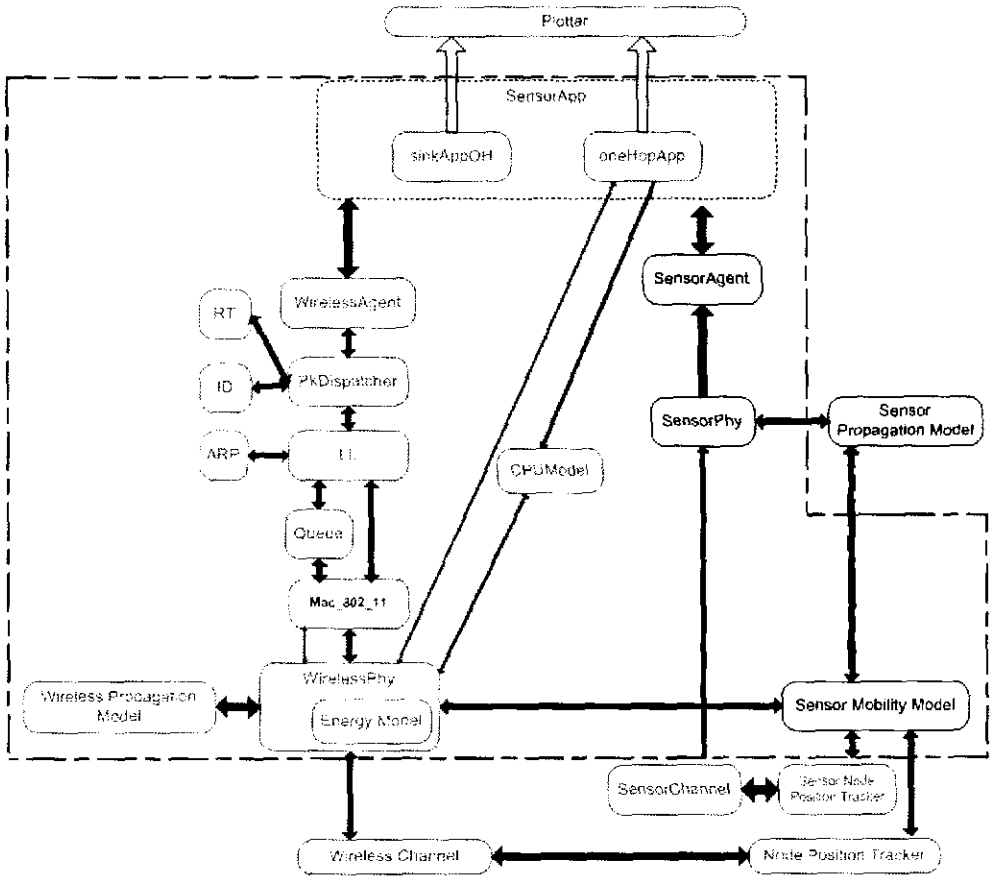

Figure 41: Overview of the One-Hop Method as an extension to J-Sim.

- **s_pos**: The sensors positions in the (x,y,z) plane.
- **timeStamp**: A timestamp which indicates when the packet was sent from the sensor. This is used to study latency.
- **phenomenon_**: This is simply an Object which represents the phenomenon you are attempting to monitor.

Each sensor's routing table has one route entry that points directly to the base station with no intermediate hops. This entry is permanently in the table with a negative time-out value set with it (value of -1). The sensor's CPU status starts off in sleep mode and

is always set to active mode whenever it is time to send an update. Between transmissions the CPU is always returned to sleep to reduce the amount of energy that is being consumed.

To minimize energy consumption, the lower layers were modified to prevent overhearing and leaving the radio in idle state. The *wirelessPhy* component has a new flag called *oneHopMode;* if it's set to true then the radio will be shutdown between sends. In other words, in the oneHopMode the radio is turned on from the application layer before every send and is shut back down after the transmission. Once the radio is turned on the MAC CSMA layer begins sensing if it is suitable to send otherwise it backs-off for a random period of time. This will prevent sensors from having to constantly listen since idle listening is costly and will drain the battery quickly. The implementation of the MAC CSMA is contained in the file called Mac_CSMA.Java. With these modifications one now has a cross-layered design where the application layer directly controls the hardware (i.e. when the radio changes state).

A.4.2 One-Hop and 802.11

Here the same *OneHopApp* is used from Figure 41 except no sleeping can be utilized and the traditional layered network with no flattening of the architecture was modelled. IEEE 802.11 functions by using both physical and virtual carrier sensing. The physical carrier sensing is provided by the physical layer (i.e. CSMA) while the virtual carrier sensing is done by the MAC layer called Network Allocation Vector (NAV). Due to NAV and the RTS/CTS packets that are used as announcements all nodes must remain on at all times. Thus, this model has a sensor which sends every 8-12 seconds, and has a permanent route entry to the sink in its routing table. Like the One-Hop and CSMA combination (see Section A.4.1) the packet that is sent is an *OH_Packet* and contains the same information.

A.4.3 One-Hop and TDMA

To combine the One-Hop model with a TDMA scheme a new package called *OneHopTDMA* was created that contained three classes: *OH_TDMA_Packet*, *OneHopAppTDMA*, and *SinkAppTDMA*. *OH_TDMA_Packet* is similar to *OH_Packet* and it also extends J-Sim's *Packet* class and serves as the type of packet that is sent back to the base station. *OneHopAppTDMA* has two roles (1) to listen for broadcasts from the sink to determine when to send and (2) to periodically send back to the base station during its time slot. The *SinkAppTDMA* has global knowledge of the network and creates a global sending schedule which allocates every sensor a specific time slot to which they are allowed to use the channel. The *SinkAppTDMA* sends out this schedule once at the start of the simulation and then re-sends a new schedule every five frames. The re-send of a schedule

allows for re-allocation of the time-slots if sensors have died. Note that this scheme assumes that all clocks are synchronized by some mechanism.

The simulation begins by the sink starting and immediately afterwards the sensors. When the sensors begin, they simply listen on the channel for a broadcast which is sent from the sink within the first second after commencing. The timer that controls how often to re-send the schedule is called *adv_sch_timer* (which is an instance of an *ACATimer* object). The first frame begins two seconds after the broadcast is sent from the sink at startup. From that point on, every frame is repeated (Frame length depends on the number of nodes and duration of each time slot) and a 10 second gap (stored as *gap_time_TDMA*) between each frame is inserted. This gap allows one to space out how regularly sensors send packets to a base station.

So instead of having the TDMA protocol implemented in the lower layers it is actually coordinated by the application layer. This simulation once again stress the importance of how software can have direct control over the hardware in order to increase performance. The actual MAC layer is simply a CSMA based protocol which never experiences a busy channel since nodes only open up for transmission when it is their pre-assigned time slot. Guards between time slots are simulated by making each sensor believe that the packets that are sent are actually larger then they appear. Therefore, each sensor calculates a longer transmission time required at the application level but are received and transmitted at a faster rate by the hardware. This prevents any overlapping and adds an assurance that no collisions will occur. In our simulations the application level calculates transmission times around packets that have a header size of 25 bytes and a body size of 170 bytes. The actual size of the packet is only 175 bytes and is set in the *wirelessPhy* component.

A.4.4 Multi-Hop and CSMA

The Multi-Hop scheme was partly implemented in J-Sim and finished off in the Tcl script which is associated with it. This scheme functions by making each sensor find a neighbor that is One-Hop closer to the base station. In order words chains are formed where each sensor has an up-link and a down-link. A sensor can only have one down-link but can have multiple up-links. The simulation does not show the cost of maintaining these chains but will be developed in future work. Since our sensor networks are immobile once a sensor reaches a certain minimum energy threshold it should send out packets to all its up-link neighbors signaling to use its down-link neighbor. This will maintain the chain and prevent any packets from being dropped.

The MultiHopApp which is shown in Figure 42 illustrates the key modifications. All Multi-Hop specific classes can be found in the *MultiHop* package located in a sub-directory in *the drcl.inet.sensorsim* package. The *MultiHopApp* subclasses *SensorApp*

and is in charge of generating data packets and maintaining neighbor and sink information. When a sensor starts, its neighbor is set to the sink until it has determined who its

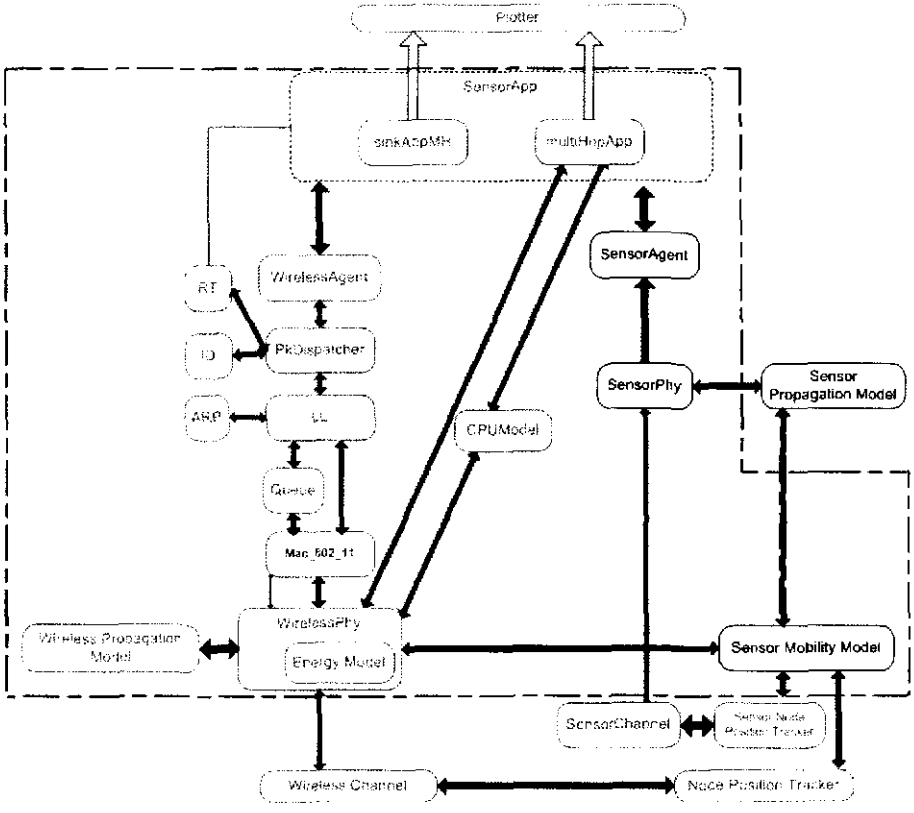

Figure 42: Overview of Multi-hop paradigm implemented in J-Sim

closest neighbor is. The sinks position is by default (0,0,0) but can be changed either in the Java file or through Tcl. Each sensor is equipped with a function called *EuclideanDist()* which is capable of calculating the distance to the sink or to a neighbor. After receiving who its neighbor is, a timer (called *sendTimer*) begins and when it expires the sensor will send a packet to its neighbor (which can possibly be the sink). The *sendTimer* is based on a random number between 6 and 10 to prevent collisions (future work hopes to have a better defined interval setting). There are two sending functions: *SendMyData()* and *SendDataNextHop()* the first is for sending a packet whose source is itself the latter is when a sensor is forwarding on data down the chain.

Another new feature here is letting the application level directly modify the routing

table as shown in Figure 42. A new port called *.setRoute* was created to connect the application to the *RT* component (at the *.service_rt* port). Whenever a new neighbor is set, an entry is inserted into the route table so a successful transmission can occur.

The *NeighborQueryContract* which can be found in this *MultiHop* package is a contract which allows a sensor to query the *SensorNodePositionTracker* component to determine the current location of a node. The contract is used in *MultiHopApp* in the method *setNeighbor()* which connects to the tracking component to determine who the closest node is. Therefore, the model assumes sensors have global knowledge of all nodes. In practice an initial setup phase at startup would replace this assumption. In order to accommodate this query from *MultiHopApp* a new port, specifically *.multiHop* was created in *SensorNodePositionTracker* along with a couple new methods (*EuclideanDist*, and *closestNeighbor*).

The maintenance work to keep these chains of sensors connected together is done in the Tcl script. The *setNeighbor()* and *neighborUpdate()* subroutines successfully maintain neighboring sensors over time (A Java based implementation can also be found in the *visualSensorNet* package in a file called *SensorSim.java*). *setNeighbor()* chooses the closest node that is in the direction of the base station and sets it as the current neighbor. As stated earlier, in practice this method would require some initial set-up phase where global information would be sent across the network in order for nodes to self-adapt to changes in topology. *setNeighbor()* also creates a global list called *sensorList* which is used by *neighborUpdate()* when sensor dies. *neighborUpdate()* is called upon periodically and checks to see if any sensors have died. If no sensors have died then no topology changes have occurred and all chains are still intact. On the other hand, if a sensor has died then all its up-link neighbors need to have new neighbors associated to them.

The MAC layer which is used is CSMA and is called Mac_CSMA (located in the drcl.inet.mac.CSMA package). The CSMA simply senses the channel and if clear will proceed to sending to its neighbor. This combination is vulnerable to the hidden terminal problem.

A.4.5 Multi-Hop and IEEE 802.11

This scheme is identical to the Multi-Hop and CSMA combination except the MAC layer used was the IEEE 802.11. Therefore, the traditional network abstraction is used and no precise control over the hardware is utilized in this model. The only exception to this rule is the application layer being able to communicate with the routing table to update neighboring entries. A flag has also been created in the wirelessPhy component to identify to the hardware that these sensors are running in Multi-Hop mode. The use of RTS packets can be set through the Tcl script as well.

One more aspect to note is the transmission timer halting when the sensor's energy reaches a certain threshold. A variable called *stop_threshold* is the breakpoint to which no more packets are sent from a sensor (This feature is also present in the Multi-Hop/CSMA combination). The rest of the energy is to be used for acting as a router and forwarding the packets that are waiting in the queue. This is critical in order to keep the reliability high or else when a sensor dies several packets that were waiting to be forwarded on will die since the sensor was busy generating new packets to be sent. The threshold for small network sizes could set as low as 0.015 joules, but for networks of 200+ sensors a comfortable buffer of 0.05 joules should at least be given in order for all packets to be successfully received towards the end of the network's life.

A.4.6 MNRLEACH

This implementation is a variation of the original LEACH protocol which has been ported from an old version of ns-2 (specifically version ns-2.1) which was developed at MIT by the $\mu - AMPS$ (Micro-Adaptive Multi-Domain Power-Aware Sensors) project [14]. The MNRLEACH capable sensor is shown in Figure 43. To implement this hierarchical routing protocol a new package was created called LEACH inside of drcl.inet.sensorsim. This package contains nine classes:

- **LEACH_ACK_Packet** - This class extends the *Packet* class and is instantiated when a sink receives a message from a cluster head with a priority flag turned on. This is not part of the original LEACH package but is an additional feature to increase the reliability of the protocol when dealing with critical data such as radiation. This packet will be sent from the sink to the cluster head who sent the priority packet to acknowledge its reception. The *packet_id* field is the unique packet number which was used in the transmission of the data from the cluster head. This is required to be sent back so that the cluster head knows which packet the sink is acknowledging.

- **LEACH_CH_Packet** - This class extends the *Packet* class and defines the body of a message for a cluster head advertisement message. When a sensor elects itself as a cluster head for a round it must broadcast to the others its status. This packet is used to do this and contains three fields: (1) *cluster_head* which is the ID of the cluster head (2) *sender_pos* this is an array holding the coordinates of the CH in three-dimensional space and (3) *code* which is just an integer that represents the spreading code to be used for the DS-SS for this cluster.

- **LEACH_DATA_Packet** - This class represents the body of a message for a unicast packet that is either being sent from a sensor to a CH or from a CH to a base

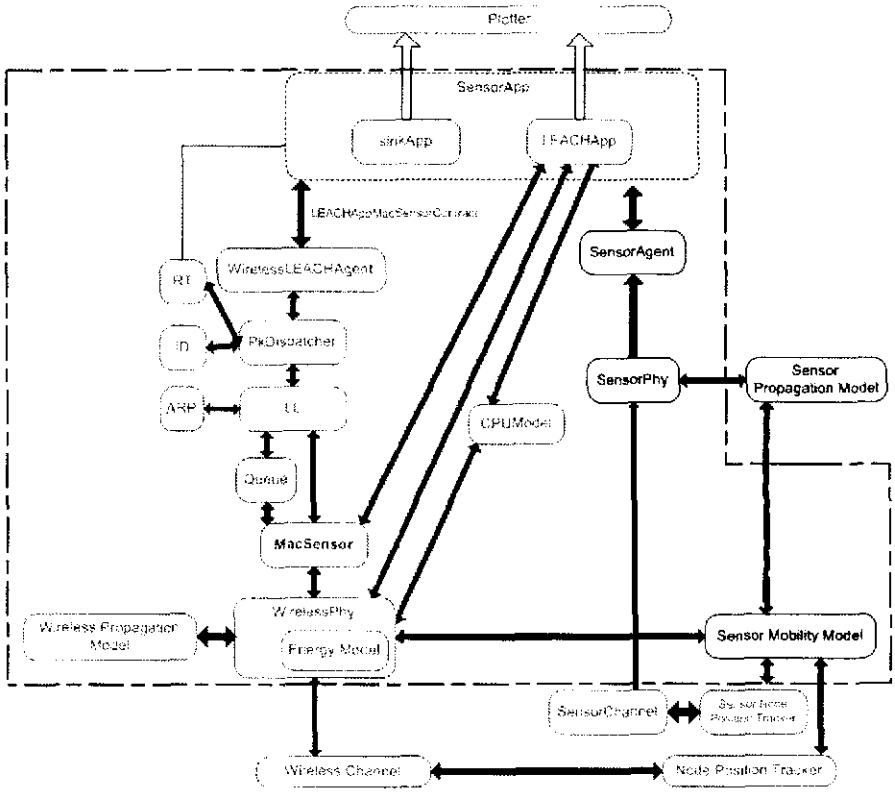

Figure 43: Overview of the LEACH extension implemented in J-Sim

station. This class also extends the *Packet* class. The primary fields in the body of this packet are (1) *sender_id* which is who sent the packet (2) *CH_id* the ID of the CH for this sensor (3) *phenomenon* an *Object* that can hold the value of any phenomenon being sensed by the sensor (4) *timestamp* is the time at which the packet was created (5) *code* which again is the spreading code to be used for this transmission and (6) priority_flag which signifies the sink that there is dangerous activity being reported in its cluster.

- **LEACH_Join_Packet** - Another type of packet which defines the body of the message that all non-cluster heads send back to the CH that they have chosen to join. This packet is sent as a broadcast with enough power so that every other sensor can hear. This prevents from having the hidden-terminal problem. The only fields in this message are (1) *src_id* the ID of the sensor wanting to join (2)

chID which cluster head it has selected and (3) *code* which is again the spreading code to minimize inter-cluster collisions.

- **LEACH_SCH_Packet** - Once a CH has heard back from all the sensors which want to join its cluster it creates a TDMA schedule during the setup phase. This schedule is then put into the body of *LEACH_SCH_Packet* and is broadcasted out to all sensors to create the appropriate TDMA schedule.

- **LEACHApp** - This is the sensors application layer which controls everything from radio status, CPU status, determining cluster head, and communicating back to either CH or a base station. It is a subclass of SensorApp.

- **SinkAppLEACH** - This is the base station application layer which essentially just listens over the shared medium for a specific spreading code and accepts packets from CHs only.

- **WirelessLEACHAgent** - Acts as the transport layer between the sensor application layer and the wireless protocol stack. This layer receives application specific data from LEACHApp and encapsulates it into a LEACH_SensorPacket which is then passed down into the wireless protocol stack. This class is a subclass of WirelessAgent which ultimately subclasses Protocol, an important J-Sim class that implements transport, routing and various other signaling protocols.

- **LEACHAppMacSensorContract** - This contract defines the communication between the LEACHApp component and the wirelessPhy component. The LEACHapp component is equipped with a port called *macsensor* to which it can attempt to set spreading code, notify hardware that it is acting as a CH for this round, and perform other administrative tasks. This contract allows for sharing status information quickly between application layer and lower MAC and data link layers.

LEACHApp which represents the application layer of each LEACH enabled sensor implements the core of all the algorithms. Sensors decide whether or not they should elect themselves as cluster heads when calling the *decideClusterHead()* function which implement the statistical algorithm discussed in Section 2.4.3. The TDMA schedule which is used within each cluster is created and advertised using the same approach as the *OneHopTDMA* scheme. Basically every sensor has a pre-determined time-slot to which their packet can be sent. The default spreading code for cluster heads to communicate back to a base station is zero otherwise the spreading code is simply the ID of the cluster to which they belong. Java *Vectors* are used by cluster heads for two reasons (1) to maintain the list of who has join their cluster and (2) to maintain the received data from each sensor until it is time to package it and forward it to a base station.

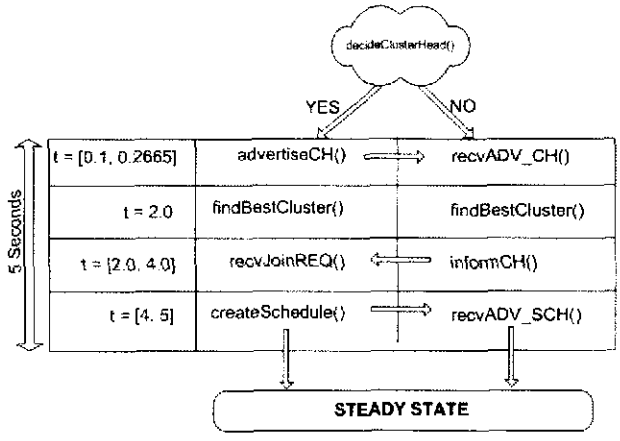

Figure 44: A LEACH setup phase breakdown.

The setup phase can be broken down into further sub-phases as shown in Figure 44. The whole phase last 5 seconds and can be broken down into 4 parts as shown in Figure 44. The setup phase is initiated by a call to *decideClusterHead()* which determines who the CHs will be for the current round. From there if the sensor elected itself as a CH then it will advertise its new status to all others. This occurs between the time interval [0.1,0.266] seconds and is done by calling the *advertiseClusterHead()* method. Therefore, all sensors are aware of all the possible clusters to which it can join before the 1 second mark hits. At this point both non-CH and CH call *findBestCluster*() where the non-CH determine to which cluster it should join (this occurs at time = 2.0 seconds into the setup phase). Then between time interval [2.0,4.0) the non-CHs send out broadcasts informing the CHs which cluster it is joining. For non-CH the method *informCH()* is invoked and when the packet reaches the CHs application layer the *recvJoinREQ()* is called. The final phase which begins 4 seconds into the setup phase is for the CHs to create a local TDMA schedule and broadcasts it out to the sensors. This is done by calling the *createSchedule()* function and is processed upon arrival by the *recvADV_SCH*() procedure. This makes the whole setup phase last exactly 5.0 seconds and it occurs every 20 seconds (hence a steady-state phase is 20 seconds long). All broadcast packets used during the setup phase are 20 bytes long and are sent out with enough power so that all sensors in the sensing field are capable of hearing the message. This prevents the hidden terminal problem from occurring.

LEACHApp also checks for dangerously high sensed values from the information it receives from its sensors. If at any point a cluster head receives a data update from one of its cluster members whose sensed phenomenon is higher then *danger_threshold*

(user defined value) then a priority flag is turned on when the aggregated update is sent back to the sink. The packet is also inserted into a vector called PriorityPackets and is removed only upon receiving a *LEACH_ACK_Packet* or when the *retransmission_timer* has reached its maximum number of retries (user defined value).

A.5 MAC level Modification and Additions

J-Sim has a wireless protocol stack which has clearly defined layers and binding contracts. At the lowest level it has one core MAC protocol implemented, the IEEE 802.11 standard. Although this protocol has proven to be extremely reliable it's concern for energy is neglected making it an unfavorable candidate for Sensor Network simulation. Therefore, this layer needed to be modified in a way to easily allow extensibility for future development and research of MAC protocols. What has been created is an abstract *Mac* object which only holds what can be considered essential fields and procedures to any MAC protocol. This abstract class can then be consistently sub-classed to implement any MAC protocol a user would desire to study. For the purposes of this research the *Mac_802_11* component had fields extracted from it and inserted into the *Mac* component. From there a new component called *Mac_CSMA* was created and sub-classed *Mac* as show in Figure 45. It is essentially a CSMA MAC protocol which does not have the overhead of ACK, RTS/CTS, and DIFS delays. These modifications are all found in the *mac* package which is part of the *INET* package. Since the contention based MAC protocols such as CSMA involve backing off new timers were implement (as shown in Figure 45). There are three timers *MacSensor_TxSensorTimer*, *MacSensor_RxSensorTimer*, and *MacSensor_DeferSensorTimer* which all subclass an object called *Mac_Sensor_Timer*. They basically are all similar except that they each have different handlers for when the timer expires. After *Mac_CSMA* determines that the channel is free to send it sends off the packet and begins the transmission timer. This timer changes the state of the card to busy preventing it from hearing incoming messages or sending others. Once it expires the state is reset back to its standard idle mode and we check the queue if more packets need to be sent off. The duration of both the transmission and receiving timers are based on the packet size and the bandwidth (determined in *DATA_Time(..)*).

A.6 Radiation Propagation Model Additions

In order to study the behavior of the radiation the *RadiationProp* class was created in the drcl.inet.sensorsim package. This class extends *SensorRadioPropagationModel* and attempts to calculate the radioactive strength of the sensed signal as shown in Figure 46. The behavior of this class is similar to those of *SeismicProp* (for seismic activity),

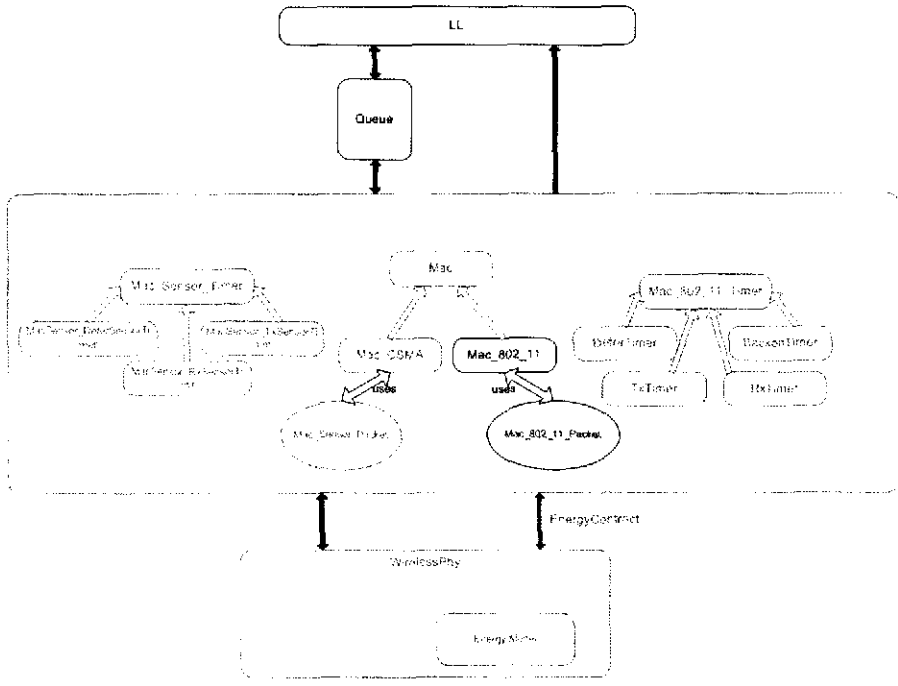

Figure 45: New MAC Hierarchy for easier extensibility.

and *AcousticProp* (for noise detection). The propagation model that this object uses is described in Section 5.4.

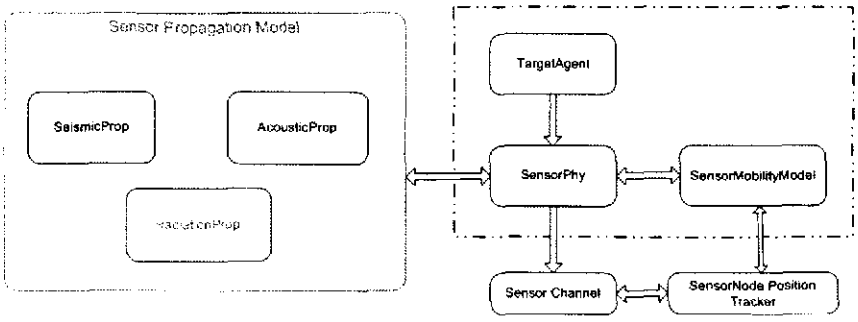

Figure 46: RadiationProp class in J-Sim

A.7 Simulation Parameters

Table 9: The significant variables used in this research for some of the various simulations that were performed in J-Sim.

Parameter	Description	Value
Dimension	Field Dimensions (in meters)	(30x100)
sinkPos[3]	Sinks Location (Array of Doubles)	(0,0,0)
sink_nid	Sink's ID	0
nid	The current Sensors ID	$1 \leq s_i \leq n$
RXThresh	Signal Threshold	$6 * 10^{-9}$W
CPThresh	The Capture Threshold	10 dB
CSThresh	Signal Detection Threshold	$1 * 10^{-9}$W
bandwidth	The radio bit-rate	1Mb
Efriss_amp	The amplifying energy required (J/bit/m^2)	2.408^{-10}
Gt	The Transmission Antenna gain	1.0
Gr	The receivers Antenna gain	1.0
Ht	Height of the transmitting Antenna	1.5 m
Hr	Height of the receiving Antenna	1.5 m
frequency	The radio frequency used by the sensors	914 MHz
L_	System loss factor	1.0
nn_	Total number of nodes	55
num_clusters	Total number of clusters desired for LEACH	5
cpu_electronics	CPU overhead for being active	$5*10^6$
stop_threshold	For MH; To clear lingering packets before death	0.016 J
EXcvr_	The energy consumed to run the radio Electronics	$50 * 10^{-9}$

B Validation

In order to verify that the simulations were accurate, analytical scenarios were calculated and then compared with simulation results. The test cases for all combinations used in this research are tested below. This includes the following combinations:

- Direct Method & CSMA
- Direct Method & TDMA
- Direct Method & IEEE 802.11

- Multi-Hop Method & CSMA
- Multi-Hop Method & IEEE 802.11
- LEACH

These scenarios look at validating the energy models, radio models, and packet delivery rate which were used.

B.1 One-Hop with CSMA Validation

Here we assumed that only one stationary sensor was in the network at location (25.0, 25.0, 0.0) with the base station located at (0.0, 0.0, 0.0). This means that the Euclidean distance between the sensor and sink was approximately 35.355 meters. To verify the energy and radio models for this combination of application and MAC protocols we must first determine how much energy the CPU consumes in the various states it can be in and then do the same for the radio component. In the One-Hop mode the CPU can

Table 10: Values used for the One-Hop and CSMA Validation Experiment.

Description	Value
Sensor Coordinates (x,y,z)	(25.0, 25.0, 0.0)
Sink Coordinates(x,y,z)	(0.0, 0.0, 0.0)
Sending Interval	10.0 seconds
Packet Size	175Bytes
Radio Bandwidth	1 Mb
Initial Energy	0.25 Joules
Radio Electronics	$50*10^{-9}$
$E_{friss-amp}$	$2.408 * 10^{-10}$
CPU Electronics	$5 * 10^{-6}$
Radio Frequency	914MHz

only be in one of two states: *CPU_ACTIVE* and *CPU_IDLE*. *CPU_ACTIVE* consumes $2.9 * 10^{-3}$mA per second and *CPU_IDLE* only consumes $1.9 * 10^{-6}$mA. The CPU is only active when it is time to generate and send the packet to the base station. The generation of the packet and the transmission time required to send the packet can be calculated by:

$$TxTime = \frac{8 * packet\ size}{bandwidth} \tag{22}$$

we multiply the packet size by 8 to convert to bits. Using the values from Table 10 we can calculate TxTime to be 0.0014 seconds and by taking the CPU electronics delay into account we conclude that the CPU is only active for 0.00145 seconds/transmission. The rest of the time it is in sleep mode which means that if the sensor sends once every 10 seconds it spends 9.9986 seconds in sleep mode. By multiplying time*current we obtain the energy consumption (here the sensors run at 1 Volt) which is $4.0745 * 10^{-6}$ joules for being active and $1.899734 * 10^{-5}$ for the CPU being in sleep mode.

Now we must consider the radio's consumption which again can only be in SLEEP or TX mode. When transmitting the radio has two main energy consumers the radio electronics (this includes the energy to modulate, encode etc...), and the amplifying power required to transmit. To calculate the amplifying power we use the following equation

$$P_t = E_{friss-amp} * bandwidth * d^2 \tag{23}$$

this represents the transmission power required per bit. For our simulation this results in an energy drain of

$$P_t = (2.408 * 10^{-10}) * (1 * 10^6) * (35.355)^2 = 0.30099 \tag{24}$$

The second energy consumption is the radio electronics which is defined per bit in Table 9 as EXcvr_ and if multiplied by the bandwidth we obtain 0.05. Therefore the total energy requirements for one single transmission can be modelled by

$$E_{Tx} = \frac{(8 * packet\ size) * (P_t + (EXcvr * bandwidth))}{bandwidth} \tag{25}$$

$$= \frac{(8 * 175) * (0.30099422682 + 0.05)}{(1 * 10^6)} = 4.9139 * 10^{-4}$$

Therefore, every transmission cost 4.914 joules to transmit. The rest of the time the radio is in sleep mode which still consumes a slight amount of energy. Again if we do one transmission every 10 seconds and one transmission takes 0.0014 seconds then 9.997 seconds are spent with the radio in sleep mode. Then this means the radio will consume an additional $7.99888 * 10^{-5}$joules of energy.

If one combines the CPU and radio energy consumptions for a period of 10.0 seconds together it becomes

$$E_{round} = (E_{radio-sleep} + E_{cpu-sleep}) + (E_{radio-tx} + E_{cpu-active}) \tag{26}$$

$$\approx 9.899 * 10^{-5} + 4.955 * 10^{-4} \approx 5.945 * 10^{-7}\ \text{joules} \tag{27}$$

Knowing that every 10 seconds we consume roughly $5.9445*10^{-7}$of energy and that a sensor starts with an initial capacity of 0.25 joules one can approximate how many

packets it will send and when it will die.

$$\frac{E_{starting}}{E_{round}} = \frac{0.25}{5.9445 * 10^{-7}} \approx 420.56 \text{ packets} \tag{28}$$

This means that roughly 421 packets will be able to sent back to the base station over a lifetime of 421*10 = 4210 seconds. The following is a sample output which correlates with the above calculations:

```
TCL0> create sink 0
create sensor 1
No target agents ....
Sensor1 will now send every: 10.0 seconds
Simulation begins...
Sensor1 Location: (25.0, 25.0, 0.0)
Sensor1 is dead at 4211.000001 its distance from the sink was: 35.35534
All nodes dead at 4211.0
---------------------------------------------
Simulation Terminated
Results:
Base Station Received 421
Sensor1 Sent 421 Packets to BS
Total packets dropped at Application layer: 0
Total packets dropped at physical layer: 0
Total Packets sent from all nodes: 421
Number of Dropped Packets: 0
Success Rate: 100.0
```

B.2 One-Hop and TDMA Validation

This following scenario was created in order to verify the validity of using the One-Hop application layer protocol with a TDMA based scheme. Here we created a Sensor Network consisting of only two nodes, each equipped with radio bandwidths of 1Mb. The position of $sensor_1$ was (25.0, 25.0, 0.0), and $sensor_2$ was (25.0, 95.0, 0.0) this means that their distances from the sink were 35.355m and 98.2344m respectively. The scenario starts with the base station sending out a broadcast with the schedule to the nodes at time (t=) 0.2 seconds. The size of the broadcast schedule packet is 90 bytes. The first frame begins at t=2.0 seconds and $sensor_1$ sends first followed by $sensor_2$. The goal of this verification is to determine if both packets get to the base station and that their concluding energies for sending, sleeping, and receiving broadcasts schedules are successful for the duration of one *period*. By definition a *period* is the length of time between two consecutive schedule broadcasts. In this case a new schedule is broadcasted from the sink every 5 frames. We also verify that the timings of all transmissions to make sure they are accurate (i.e. sensors send in their allocated time slots only and sleep otherwise unless waiting for an updated schedule).

Table 11: Values used for the One-Hop and TDMA Validation Scenario

Description	Value
$sensor_1$	(25.0, 25.0, 0.0)
$sensor_2$	(25.0, 95.0, 0.0)
Packet size	175 Bytes
Schedule Broadcast Packet Size	90 Bytes
Gap Time	10.0 seconds
Frame Time	0.0032 seconds
P_{t1} (amplification cost for $sensor_1$)	0.30099422682
P_{t2} (amplification cost for $sensor_2$)	2.3237
Px_{cvr} (radio electronics for Tx and Rx)	0.05
Bandwidth	$1 * 10^6$Mb
Slot Time	0.0016 seconds

In a frame each sensor is allocated a time slot. This time slot is calculated by the following

$$\text{slot time} = \frac{(8 * packet\ size)}{bandwidth} = 0.0014 \text{ seconds} \tag{29}$$

Therefore, each sensor will be allocated a time slot of 0.0014 seconds to send their information to the sink. To prevent any sort of overlap extra guards are used in the simulations. To create these guards *virtual packet sizes* are used in order to determine the allocated slot time. These *virtual packet sizes* are always slightly larger then the actual packet to be transmitted is. This makes the TDMA schedule allocate longer periods of time for transmission, hence inserting gaps between sends. The virtual packet size used in our simulations is 200 bytes making the *virtual slot time* 0.0016 seconds. Therefore the virtual frame time is:

$$frameTime = slot\ time * nn = 0.0016 * 2 = 0.0032 \tag{30}$$

With this information we can calculate the duration of one period by the following equation

$$period = (5 * frame\ time) + (4 * gap\ time) + T_{initial-setup} \tag{31}$$

$$= (5 * 0.0028) + (4 * 10) + 2.0 = 42.014 \tag{32}$$

this means that one should be able to run a simulation for 42.014 seconds and have completed one full period. This means that two successfully transmitted packets should have been sent within that period of time.

We first look at the radio energy consumption for $sensor_1$ immediately after receiving the second schedule at time 42.014. To determine the total cost of using the radio for this scheme we must solve

$$E_{total} = E_{Tx} + E_{Rx} + E_{sleep} \tag{33}$$

for each of the two sensor. The cost of one transmission is the sum of the amplifying power and the radio electronics per bit which results in

$$E_{Tx} = \frac{(8 * packet\ size) * (P_{Tx1} + EX_{cvr}}{(1 * 10^6)} \tag{34}$$

where $P_{Tx1}$is the amplifying power for $sensor_1$ and EX_{cvr} is the radio electronics consumed during any transmission. Therefore for every transmission $sensor_1$ consumes in total $4.91391917*10^{-4}$joules of energy. For for $sensor_1$ to receive a packet (the schedule from the base station) it will consume

$$E_{Rx} = \frac{(8 * packet\ size) * (EX_{cvr})}{bandwidth} = \frac{(8 * 90) * (0.05)}{1 * 10^6} = 3.6 * 10^{-5} \tag{35}$$

The last component of the equation is to determine how much energy the radio consumes in sleep mode. This can be characterized by

$$E_{sleep} = E_{frameSleep} + E_{gapSleep} \tag{36}$$

where $E_{frameSleep}$ is the amount of time the radio spends sleeping during a frame and $E_{gapSleep}$ is the 10.0 second gap between frames when no activity is occurring. Therefore if we are trying to determine the total amount of sleeping a radio did during one period which consists of 5 frames and 4 gaps we can solve the following

$$E_{sleep} = [E_{frameSleep} * Frame\ count] * 0.002] + [Gap\ duration * Gap\ count * 0.002] \tag{37}$$

where GapDuration is 10 seconds, GapCount is the number of gaps that occur in one period, and frameCount in the number of frames in one period.

$$E_{sleep} = [(0.0014 * 5) * 0.002] + [40 * 0.002] = 1.12 * 10^{-6} \tag{38}$$

Therefore since during one period a sensor will send 5 packets and receive 2 schedules we calculate E_{total} for sensor 1 by

$$E_{total1} = (5 * E_{Tx}) + (2 * E_{Rx}) + E_{sleep} \tag{39}$$

$$\approx 2.5301 * 10^{-3}$$

So for 1 period $sensor_1$ will consume roughly 2.53 μJoules. The above steps were also be applied for $sensor_2$ and Table 12.

Table 12: Radio Consumption of each Sensor in joules for a duration of one period.

Radio State	Consumption
$sensor_1$Tx	0.00024569596
$sensor_1$ Rx	$3.6 * 10^{-5}$
$sensor_1$Idle	$1.2*10^{-6}$
$sensor_2$Tx	0.0166159
$sensor_2$Rx	$3.6*10^{-5}$
$sensor_2$Idle	$1.2*10^{-6}$

So we can approximate the amount of energy each sensor will have remaining following one period by doing

$$E_{rem} = E_{initial} - E_{onePeriod} \tag{40}$$

for our analysis we obtained (throught similar calculations as above) that $sensor_1$ would have roughly $2.475 * 10^{-2}$ joules remaining and $sensor_2$ would have $2.331 * 10^{-2}$ joules remaining. Below are the actual simulated results that were obtained:

```
TCL0> create sink 0
create sensor 1
create sensor 2
No target agents ....
Simulation begins...
Sensor1 Location: (25.0, 25.0, 0.0)
Sensor2 Location: (25.0, 95.0, 0.0)
******* Sink is now Sending out Schedule using Broadcasting at time 0.201 ********
-------------------
BROADCAST PACKET
The Packet size is: 90
The Tx power was: 2.65482the radio electronics was: 0.05
Sensor0 is removing:0.0019474704 at time: 0.201
******* Sink is now Sending out Schedule using Broadcasting at time 42.717 ********
-------------------
BROADCAST PACKET
The Packet size is: 90
The Tx power was: 2.65482the radio electronics was: 0.05
Sensor0 is removing:0.0019474704 at time: 42.717
Sensor1 has 0.24703438289 joules remaining
Sensor2 has 0.23287532705 joules remaining
```

The variations in our result is mainly due to the fact that we did not take into account CPU consumption. Since we analyzed CPU consumption and saw that it was accurate for the One-Hop/CSMA combination we did not re-calculate all the values again here.

B.3 One-Hop and IEEE 802.11

Here the goal was to verify the results when combining the One-Hop and IEEE 802.11 protocols together. Their behaviors were analytically studied by using one sensor located at (25.0, 25.0, 0.0), and one sink located at the origin. The software running at the application level had no control over the hardware (i.e. the radio components) therefore following the traditional network protocol stack. We first analytically obtain the results after one transmission and make a prediction as to how long the sensor should live for. These results are then compared with the simulated results to determine its validity.

Each packet was transmitted with its maximum power based on the worst-case distance scenario (i.e. the distance between the furthest two points) which for the reactor is roughly 104 meters. Given that distance the transmission power $P_t$was 2.6045 watts. The transmission intervals for the sensor are set to send to the base station every 9.8 seconds. The 802.11 protocol is running in DCF (Distributed Coordination Function) mode for ad hoc networks. Table 13 shows the detailed breakdown of the cost of transmission and receiving for the sensor. Therefore one can conclude by summing the various times

Table 13: Transmission and Receiving costs for each transmission using the 802.11 and One-hop protocols.

Packet Type	Packet Size	Transmission Time
RTS Frame	44 Bytes	$\frac{8*44}{1*10^6} = 3.52 * 10^{-4}$sec
DATA Frame	175 Bytes	$\frac{8*175}{1*10^6} = 0.0014$ sec
ACK Frame	38 Bytes	$\frac{8*38}{1*10^6} = 3.04 * 10^{-4}$sec
CTS Frame	38 Bytes	$\frac{8*38}{1*10^6} = 3.04 * 10^{-4}$sec

in Table 13 that the radio will be in transmission mode for 0.001752 seconds and receiving mode $6.08 * 10^{-4}$seconds. We can then say that in the first 10 seconds the radio is in idle mode for

$$R_{idle\ time} = total_{time} - [Tx_{time} + Rx_{time}] \tag{41}$$

$$R_{idle}\ \text{time} = 10.0 - [0.001752 + 6.08 * 10^{-4}] = 9.99764\ seconds$$

One can now ultimately calculate the transmission cost for every one packet send (in other words radio transmission cost for every 10 seconds). The following illustrates this

$$E_{radio} = E_{Tx} + E_{Rx} + E_{idle} = [radioTx_{Time} * i_{tx}] + [radioRx_{time} * i_{rx}] + [radioIdle_{time} * i_{idle}] \tag{42}$$

$$= 2.48032 * 10^{-5} + 4.864 * 10^{-6} + 0.001999528 \approx 0.00203\ \text{joules}$$

To calculate the CPU consumption for this same 10.0 second period of time consult Appendix B.1 since the same size data packets and activity are measured and explained there. The CPU energy consumption can be calculated to be $E_{CPU} = 2.289{*}10^{-5}$joules for that 10.0 second period of time. Thus the total energy consumed for one round (where a round is 10.0 seconds) is approximately:

$$E_{one-round} = E_{radio} + E_{CPU} \approx 0.00203 \tag{43}$$

This means that roughly 124 rounds may occur given an initial energy of 0.25 joules and one transmission per round. The code below shows the simulated output using the code developed for this research. The results clearly coincide with the calculated ones from above.

```
TCL0> create sink 0
create sensor 1
Sensor1 will now send every: 9.8seconds
Simulation begins...
Sensor1 Location: (25.0, 25.0, 0.0)
Sensor1 is dead at 1217.1 its distance from the sink was: 35.355339
All nodes dead at 1219.0
---------------------------------------------
Simulation Terminated
Results:
Base Station Received 124
Collisions at Base Station: 0
Sensor1 Sent 124 Packets to BS
Total packets dropped at Application layer: 0
Total packets dropped at physical layer: 0
Total Packets sent from all nodes: 124
Number of Dropped Packets: 0
Success Rate: 100.0
```

B.4 Multi-Hop and CSMA

The methods used to validate the One-Hop models were extended to be used for the Multi-Hop paradigms. Here three sensors are used and one sink. The goal are two fold: first verify the energy levels consumed after each sensor has sent 3 packets and secondly verify that all 3 packets were received by the sink on time. The detailed values used can be found in Table 14. So based on their locations one can deduce that s_1 neighbor will be the sink (since its the closest), then $s_2$will send to $s_1$and likewise $s_3$will send down to s_2. The following calculates the radio costs for each sensor.

- s_1Transmission cost is: $P_{Tx1} = (2.408{*}10^{-10}){*}(1{*}10^{6}){*}(14.1421^2) = 0.04816$ watts of amplifying power required. Then when combining with the costs of the radio electronics and the packet size you get

$$E_{totalSensor_1} = \frac{[packet\ size * (PXcv + P_{Tx1})]}{1 * 10^6} \approx 1.37424 * 10^{-4}\ joules \tag{44}$$

Table 14: Values for analytical study of using the multi-hop and CSMA protocols together.

Description	Value
$sensor_1$Coordinates (s_1)	(10.0,10.0,0.0)
$sensor_2$Coordinates (s_2)	(15.0,35.0,0.0)
$sensor_3$Coordinates (s_3)	(20.0, 75.0, 0.0)
sink Coordinates	(0.0, 0.0, 0.0)
$s_1$distance to sink	14.1421 meters
$s_2$distance to sink	38.0789 meters
$s_3$distance to sink	77.6209 meters
packet size	175 bytes
bandwidth	1 Mb
Packet Tx time	0.0014 seconds
Simulation Duration	30.9 seconds
$s_1$sending interval	10.0 sec
$s_2$sending interval	10.1 sec
$s_3$sending interval	10.2 sec

therefore since the first sensor acts as an intermediate routing node it will ultimately forward 9 packets for this simulation costing in total $9*E_{totalSensor_1} \approx 0.00124$ joules of energy.

- s_2's transmission cost is: $P_{Tx2} = (2.408*10^{-10})*(1*10^6)*(25.4950^2) \approx 0.15652$ watts of amplifying power required. Then when combining with the costs of the radio electronics and the packet size you get

$$E_{totalSensor_2} = \frac{[packet\ size * (PXcv + P_{Tx2})]}{1 * 10^6} = 1.37424 * 10^{-4}\ joules \quad (45)$$

and since s_2 acts as an intermediate routing node for s_3 and it will ultimately forward 6 packets for this simulation costing in total $6*E_{totalSensor_2} = 0.001735$ joules of energy.

- $s_3's$ transmission cost is: $P_{Tx3} = (2.408 * 10^{-10}) * (1 * 10^6) * (40.311289^2) = 0.3913$ watts of amplifying power required. Then when combining with the costs of the radio electronics and the packet size you get

$$E_{totalSensor_3} = \frac{[packet\ size * (PXcv + P_{Tx3})]}{1 * 10^6} = 6.1782 * 10^{-4}\ joules \quad (46)$$

and since $s_3$does not have to act as an intermediate routing node for any other sensors it only sends off its 3 packets which gives a total transmission cost of $3*E_{totalSensor_3} = 0.0018535$ joules of energy.

- For both $s_1$and $s_2$which have to listen to receive a node the cost of doing that is

$$E_{rx} = \frac{[(8*175)*(0.05)]}{1*10^6} = 7*10^{-5}\ joules \tag{47}$$

and since $s_1$receives 6 packets $E_{rx1} = 6*E_{rx} = 4.2*10^{-4}$joules and similarly for $s_2$we have $E_{rx2} = 3*E_{rx} = 2.1*10^{-4}$ joules.

- Calculation of radio idle time (since it cannot be shut down unlike the One-Hop models) is simply taking the total simulation time and subtracting the amount of time the radio is spent in transmitting and receiving modes. Therefore $s_1$is non-idle for $9*0.0014$=0.0126 seconds and $s_2is$non-idle for $6*0.0014$= 0.0084 and likewise $s_3$is non-idle for 0.0042. This means that on a 30.9 second simulation $s_1is$idle for 30.8874 seconds, s_2 is idle 30.8916 seconds, and finally $s_3$is idle for 30.8958 seconds. Since the current drawn in idle state is 0.0002 Amps this gives energies of 0.00617748, 0.00617832, and 0.00617916 joules drawn from s_1, s_2, s_3respectively.

Combining the above information we can take the sum of energy spent in transmission, receiving, and idle mode to obtain the total energy consumed by the radio after each sensor transmits 3 packets (i.e. in 30.9 seconds).

$$E_{radioS_1} = 0.25 - [9.535457*10^{-5} + 0.0012368 + 4.2*10^{-4} + 0.00617748] \approx 0.24207 \tag{48}$$

$$E_{radioS_2} = 0.25 - [8.312199*10^{-5} + 2.8913*10^{-4} + 2.1*10^{-4} + 0.00617832] \approx 0.24324$$

$$E_{radioS_3} = 0.25 - [7.09236*10^{-5} + 0.0018535 + 0 + 0.00617916] \approx 0.24313$$

In order to be as accurate as possible the CPU consumption (although small) was also calculated. Here a discussion on CPU consumption for s_1 is shown. To obtain the CPU consumption for s_2 and s_3 one would also follow the same procedure.

Sensor 1's CPU is active only 9 times during the 30.9 seconds in order to transmit and receive the packets that are being routed to the sink. Each active period last

$$E_{CPUtime} = [\frac{8*packet\ size}{bandwidth}] + cpu_{electronics} = 0.001405\ seconds \tag{49}$$

which means that the energy consumed for each active period is $E_{CPUtime}*I = 4.074*10^{-6}$ joules. Since s_1 goes active 9 times this consumes a total of $3.66705*10^{-5}$

joules. The rest of the time the CPU is in sleep mode which means it sleeps for a total of 30.886355 seconds during the 30.9 second simulation consuming roughly $5.868407*10^{-5}$ joules. By combining its sleep energy and active energy we get that the CPU for s_1 consumed roughly $9.5354*10^{-5}$ joules of energy.

Likewise one can calculate that the total energy consumed is $8.312199 * 10^{-5}$ and $7.09235 * 10^{-5}$ joules for s_2 and s_3 respectively.

Below is the simulation output which verifies the above work. Due to overhead of using various components and rounding some variations exists but overall the figures are acceptable and well within the calculated range.

```
TCL0> create sink 0
create sensor 1
create sensor 2
create sensor 3
No target agents ....
Sensor1 will now send every: 10.0seconds
Sensor2 will now send every: 10.1seconds
Sensor3 will now send every: 10.2seconds
simulation begins...
Sensor1 Location: (10.0, 10.0, 0.0)
Sensor2 Location: (15.0, 35.0, 0.0)
Sensor3 Location: (20.0, 75.0, 0.0)
Sensor1 Neighbor is: 0
Sensor2 Neighbor is: 1
Sensor3 Neighbor is: 2
-----------------------------------------------
Sensor1 has 0.242100478525 joules remaining
Sensor1 idle radio used: 0.00593752 joules
Sensor1 Radio was idle for: 29.6876 sec
Sensor1 CPU active used: 3.66705e-05 joules
Sensor1 CPU was active for: 0.012645 sec
Sensor1 CPU sleep used: 5.85149745e-05 joules
Sensor1 CPU was sleep for: 30.797355 sec
Sensor1 CPU Total used: 9.51854745e-05 joules
-----------------------------------------------
Sensor2 has 0.241404742017 joules remaining
Sensor2 idle radio used: 0.00593752 joules
Sensor2 Radio was idle for: 29.6876 sec
Sensor2 CPU active used: 2.4447e-05 joules
Sensor2 CPU was active for: 0.00842999999999 sec
Sensor2 CPU sleep used: 5.8522983e-05 joules
Sensor2 CPU was sleep for: 30.80157 sec
Sensor2 CPU Total used: 8.2969983e-05 joules
-----------------------------------------------
Sensor3 has 0.241088265508 joules remaining
Sensor3 idle radio used: 0.00593752 joules
Sensor3 Radio was idle for: 29.6876 sec
Sensor3 CPU active used: 1.22235e-05 joules
Sensor3 CPU was active for: 0.004215 sec
Sensor3 CPU sleep used: 5.85309915e-05 joules
Sensor3 CPU was sleep for: 30.805785 sec
Sensor3 CPU Total used: 7.07544915e-05 joules
```

B.5 Multi-Hop and IEEE 802.11

After validating the Multi-Hop method in Appendix B.4 and the IEEE 802.11 protocol in Appendix B.3 no further testing was done. This is also partly due to the fact that the IEEE 802.11 component had already been contributed and tested by others using J-Sim over time.

B.6 LEACH

For validating LEACH we mainly focused on verifying the setup phase to make sure that it was accurately simulating the protocol. The goal here is to determine the cost of performing this setup every 20 seconds and verify our computed results with simulated ones. In any setup phase one knows that each CH will send out 2 packets one for advertising that its a CH and another to advertise the schedule. On the other hand the non-CH only have their radios enter transmit mode once to notify which cluster they wish to join. Since all radios are on and listening during the setup phase all sensors overhear the broadcasts. This means that for a non-CH it will receive

$$Packets_{received} = (2 * CH_{total}) + ((S_{total} - CH_{total}) - 1) \tag{50}$$

where CH_{total} stands for the total number of CHs for that round, and S_{total} represents the total number of nodes left in the simulation. Each transmission takes approximately

$$Rx_{time} = Tx_{time} = \frac{(8 * packet\ size)}{bandwidth} = 1.6 * 10^{-4} \text{ seconds} \tag{51}$$

and this is also how long receiving a packet will take. The total packets transmitted and received combined for a sensor is equal to the total number of sensors participating in the election process (say S_{total}). Since the setup phase lasts roughly 4.0 seconds[10]we can conclude that the radio spends

$$R_{idle} = 4.0 - (S_{total} * Tx_{time}) \tag{52}$$

in idle mode waiting to receive further instructions before entering steady state phase. To verify the above we constructed a small LEACH simulation of 5 sensors and ran the simulation from 0.0 to 3.99 seconds. This means that during this time period all sensors sent 1 packet and received 4 for a total of 5 packets. Therefore, the radio was in non-idle state for approximately $R_{idle} = 4.0 - (5 * Tx_{time}) = 8 * 10^{-4}$ seconds. Hence the radio

[10]Note that the setup phase last 5.0 seconds but that during the last second non-CH begin to go to sleep once they receive their schedules. So to measure radio consumption we will only look at the first 4.0 seconds.

was in idle mode the rest of the time which is 3.9892 seconds which consumes 0.0002 amps/second. So the amount of energy drawn from being in idle mode was $7.9784*10^{-4}$ joules and the transmission cost was $4.327712*10^{-4}$ and receiving costs was $8*10^{-6}$ joules per message. So the total energy consumed (ignoring CPU utilization) during the setup phase is approximately $E_{setup} = 7.9784*10^{-4}+4.327712*10^{-4}+(4*(8*10^{-6})) \approx 0.00126$ joules of energy. By subtracting the initial energy of 0.25 joules by E_{setup} we obtain that all sensors should have 0.248737388 joules remaining. Below are the simulated results which coincide with the above discussion.

```
TCL0> create sink 0
create sensor 1
create sensor 2
create sensor 3
create sensor 4
create sensor 5
No target agents ....
Simulation begins...
Sensor1 Location: (2.80123746153, 34.6600531296, 0.0)
Sensor2 Location: (9.45388483324, 38.1413074388, 0.0)
Sensor3 Location: (12.2862371208, 15.9576288033, 0.0)
Sensor4 Location: (29.960189159, 69.6639849197, 0.0)
Sensor5 Location: (12.7783638298, 86.5362937965, 0.0)
******** Round 0 Sensor1 Is a cluster head at time 0.1 **********
******** Round 0 Sensor3 Is a cluster head at time 0.1 **********
Sensor1 is advertising its CH status at time: 0.12957091827916728
Sensor3 is advertising its CH status at time: 0.21524028535833
I am CH1 and I will create a schedule in:2.9167774112141873 sec. The time is:2.1
Sensor2 Current Cluster head is Sensor1 whose distance is: 7.508451812300823
I am CH3 and I will create a schedule in:2.3750465680846027 sec. The time is:2.1
Sensor4 Current Cluster head is Sensor1 whose distance is: 44.304445579100545
Sensor5 Current Cluster head is Sensor1 whose distance is: 52.82695709861167
Sensor5 is sending a Join-REQ to Sensor1 at distance: 52.826 at time 2.25551672
Sensor2 is sending a Join-REQ to Sensor1 at distance: 7.5084 at time 2.66452806
Sensor4 is sending a Join-REQ to Sensor1 at distance: 44.304 at time 3.78177548
last update time was at: 3.99 seconds
-----------------------------------------------
Sensor1 has 0.2487373888 joules remaining
Sensor1 idle radio used: 0.00079784 joules
Sensor1 Radio was in idle mode for: 3.9892 sec
-----------------------------------------------
Sensor2 has 0.2487373888 joules remaining
Sensor2 idle radio used: 0.00079784 joules
Sensor2 Radio was in idle mode for: 3.9892 sec
-----------------------------------------------
Sensor3 has 0.2487373888 joules remaining
Sensor3 idle radio used: 0.00079784 joules
Sensor3 Radio was in idle mode for: 3.9892 sec
-----------------------------------------------
Sensor4 has 0.2487373888 joules remaining
Sensor4 idle radio used: 0.00079784 joules
Sensor4 Radio was in idle mode for: 3.9892 sec
-----------------------------------------------
Sensor5 has 0.2487373888 joules remaining
Sensor5 idle radio used: 0.00079784 joules
Sensor5 Radio was in idle mode for: 3.9892 sec
```

C Sample Topology Script

```
# ************************************************************
# 35 Randomly placed sensors that can be read into
# the simulator by using a file parser
# It can be used to define global topology and node locations
# Comments begin with a '#'
# Topology definitions must begin with keyword Topo
# ************************************************************
# Topo  xmin    xmax     ymin        ymax     dx       dy
Topo     0        30          0        100        100      100
#These are the node locations
# NB : !! no verification is made on locations . Be sure they fit the grid
# Node number should begin from 0
#node_number            positionX               positionY           positionZ
0                       18.7151583232           85.5531244471           0.0
1                       0.0281954230872       79.6015860884           0.0
2                       19.1572160829           51.1023490928           0.0
3                       23.1543606348           84.4639643023           0.0
4                       25.7544086714           47.8218010849           0.0
5                       12.3032503865           69.0974875209           0.0
6                       6.44182931932           92.7512325778           0.0
7                       20.9897807152           17.4816010136           0.0
8                       3.98047047387           99.2241810538           0.0
9                       18.2432913586           49.9928786652           0.0
10                      9.09351787022           49.1828159658           0.0
11                      4.67638123998           86.4650012862           0.0
12                      5.18298527002           68.1114438307           0.0
13                      14.7109388023           55.8281686417           0.0
14                      1.20910829455           38.2770217202           0.0
15                      6.57121543613           41.3927833742           0.0
16                      26.5530512699           90.4423090585           0.0
17                      19.1665037485           71.4283350256           0.0
18                      28.8080327813           22.0231822329           0.0
19                      13.087136379            85.0037405198           0.0
20                      17.3600746446           69.2485065522           0.0
21                      17.8948867963           31.2079521507           0.0
22                      3.61553907563           54.550813583            0.0
23                      10.6571667551           50.00551173             0.0
24                      12.7906937351           77.2986883658           0.0
25                      17.7166089545           43.488992119            0.0
26                      5.84716305875           77.5650944922           0.0
27                      10.9629390999           80.391504327            0.0
28                      12.0039672554           2.25886921504           0.0
29                      19.4444691294           43.975526534            0.0
30                      29.0023371293           7.60044171829           0.0
31                      12.1871877798           66.8833826514           0.0
32                      2.70366680003           68.4263603149           0.0
33                      12.5513438101           68.1180528217           0.0
34                      18.0341320988           32.1939481572           0.0
```

References

[1] Sachin Mujumdar. Prioritized geographical routing in sensor networks. Master's thesis, Vanderbilt University, Nashville, Tennessee, 2004.

[2] Chee-Yee Chong and Srikanta Kumar. Sensor networks: Evolution, opportunities, and challenges. volume 91 of *8*, pages 1247–1256. Proceedings of the IEEE, IEEE, August 2003.

[3] Sameer Tilak, Nael Abu-Ghazaleh, and Wendi Heizelman. A taxonomy of wireless micro-sensor network models. *Mobile Computing and Communications Review*, 6(2):28–36, April 2002.

[4] Konrad Lorincz, David Malan, Thaddeus Rulford-Jones, Alan Nawoj, Anthony Clavel, Victor Shnayder, Geoffrey Mainland, and Matt Welsh. Sensor networks for emergency response: Challenges and opportunities. *Pervasive Computing*, 3(4):16–23, Oct-Dec 2004.

[5] Archana Bharathidasan, Vijay Anand, and Sai Ponduru. Sensor networks: An overview. Department of Computer Science at the University of California.

[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, March 2002.

[7] Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[8] Jason Lester Hill. *System Architecture for wireless Sensor Networks*. PhD thesis, University of California Berkeley, 2003.

[9] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002. Georgia Institute of Technology.

[10] N. Dhyanesh and S. Raghavan. Sensors on sea (sos): a simple novel sensor-based best-effort system for ocean related disaster management. In *Proceedings of International Conference on Processing Intelligent Sensing and Information Processing (ICISIP 2004)*, pages 206–211, Chennai, India, Junuary 4-7 2004 2004.

[11] Cory Kidd, Robert Orr, Gregory Abowd, Christopher Atkeson, Irfan Essa, Blair MacIntyre, Elizabeth Mynatt, Thad Starner, and Wendy Newstetter. The aware

home: A living laboratory for ubiquitous computing research. In *In the Proceedings of the Second International Workshop on Cooperative Buildings*, Pittburg, USA, October 1-2 1999.

[12] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking archive*, pages 263–270, Seattle, Washington, United States, 1999. ACM Press.

[13] Brian P. Crow, Indra Widjaja, Jeong Geum Kim, and Prescott T. Sakai. Ieee 802.11 wireless local area networks. *IEEE Communications Magazine*, 0163-6804:116–126, September 1997.

[14] Wendi Heinzelman. *Application-Specific Protocol Architectures for Wireless Networks*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, United States, June 2000.

[15] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. In *In Proceedings of the Workshop on Wireless Sensor Networks and Applications*, pages 115–121, San Diego, CA, USA, 2003. ACM Press.

[16] Mark B. Abbott and Larry L. Peterson. Increasing network throughput by integrating protocol layers. *IEEE/ACM Transactions on Networking*, 1(5):600–610, October 1993.

[17] David D. Clark and David L. Tennenhouse. Architectural considerations for a new generation of protocols. *ACM*, 1990. 089791-405-8.

[18] J. Suh and M. Horton. Powering sensor networks. *IEEE Potentials*, 23(3):35–38, August-September 2004.

[19] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: An environment for developing wireless embedded systems software. Technical Report 0034, CENS, December 2003.

[20] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *In Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, pages 1567–1576, New York, NY, USA, June 2002. Vol.3.

[21] Wei Ye and John Heidemann. Medium access control in wireless sensor networks. Technical report, USC/ISI, October 2003.

[22] Simon Haykin and Micheal Moher. *Modern Wireless Communications*. Prentice Hall, 2005.

[23] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. Macaw: A media access protocol for wireless lan's. pages 212–225, London, UK, August 1994. In Proceedings of the IGCOMM'94 Conference on Communications Architectures, Protocols and Applications, ACM Press.

[24] ARRL/CRRL Amateur Radio 9th Computer Networking Conference. *MACA - A New Channel Access Method for Packet Radio*, London, ON, Canada, September 22 1990.

[25] IEEE-SA Standards Board. Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *ANSI/IEEE Standard*, 1999 Edition Reaffirmed 12 June 2003, 2003.

[26] J-Sim Homepage. http://www.j-sim.org. 2005.

[27] Pablo Brenner. A technical tutorial on the ieee 802.11 protocol. Technical report, Director of Engineering for BreezeCOM, California, USA, 1997.

[28] Theodore S. Rappaport. *Wireless Communications-Principles and Practice*. Prentice-Hall, 1996.

[29] Limin Hu. Distributed code assignments for cdma packet radio networks. *IEEE/ACM Transactions on Networking*, pages 1500–1509, 1993. 1(6):668-677.

[30] Kemal Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks. *Elsevier Ad Hoc Network Journal*, 3(3):325–349, 2005.

[31] Jamil Ibriq and Imad Mahgoub. Cluster-based routing in wireless sensor networks: Issues and challenges. In *Proceedings of the 2004 Symposium on Performance Evaluation of Computer Telecommunication Systems*, San Jose, California, United States, July 25-29 2004.

[32] Chaiporn Jaikaeo, Chavalit Srisathapornphat, and Chien-Chung Shen. Diagnosis of sensor networks. In *IEEE International Conference on Communications (ICC 2001)*, pages 1627–1632, Helsinki, Finland, June 11-14 2001.

[33] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *In Proceedings of 33rd Annual Hawaii International Conference on System Sciences*, pages 1–10, Hawaii, United States, 2000. IEEE.

[34] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, 11(6):6–28, December 2004.

[35] Stephanie Lindsey and Cauligi S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *IEEE Aerospace Conference*, Big Sky, Montana, United States, March 2002.

[36] Arati Manjeshwar and Dharma P. Agrawal. Apteen: a hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. *Parallel and Distributed Processing Symposium*, pages 195–202, April 2002.

[37] Arati Manjeshwar and Dharma P. Agrawal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings 15th International Parallel and Distributed Processing Symposium*, pages 2009 – 2015, Washington, DC, United States, April 2001. IEEE Computer Society.

[38] Mihail L. Sichitiu and Vaidyanathan Ramadurai. Localization of wireless sensor networks with a mobile beacon. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 174–183. IEEE, 25-27 October 2004.

[39] P. Bahl and V. Padmanabhan. Radar: An in-building rf-based user location and tracking system. pages 775–784, Tel Aviv, Israel, March 2000. In proceedings of Infocom'2000, ACM Press.

[40] Nissank Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. Boston, MA, USA, August 2000. Proceedings of the sixth ACM International Conference on Mobile Computing and Networking (ACM MOBICOMM), ACM Press.

[41] Bill Appleton and Collin West. Research reactors of the future: The advanced neutron center. http://www.ornl.gov/info/ornlreview/rev27-12/text/ansmain.html, 2005.

[42] McMaster Nuclear Reactor. Safety analysis report. Technical report, McMaster University, Hamilton, Ontario, Canada, February 2002.

[43] Radiation safety training guide for radionuclide users. Environmental Health and Safety, Florida Atlantic University, 2003.

[44] MGP Instruments Homepage. http://www.mgpi.com/. 2005.

[45] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. Modelling data-centric routing in wireless sensor networks. In *IEEE Infocom*, New York, NY, USA, June 23-27 2002. IEEE.

[46] Bulusu et Al. Scalable coordination for wireless sensor networks: Self configuring localization systems. In *ISCTA 2001*, Ambleside, U.K., July 2001.

[47] Sung Park, Andreas Savvides, and Mani Srivastava. Simulating networks of wireless sensors. In *Proceedings of the 2001 Winter Simulation Conference*, Arlington, Virginia USA, December 9-12 2001. IEEE Computer Society.

[48] Hung ying Tyan. *Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation*. PhD thesis, Ohio State University, 2002.

[49] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *ACM/IEEE Transactions on Networking*, 11(1):2–16, February 2002.

[50] Krzysztof Czarnecki and Ulrich W. Eisenecker. Components and generative programming. volume 24 of *6*. ACM SIGSOFT Software Engineering Notes, November 1999.

[51] Krzysztof Czarnecki. *Generative Programming Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models*. PhD thesis, Technical University of Illmenau, October 1998.

[52] ns2 Homepage. http://www.isi.edu/nsnam/ns/. 2005.

[53] SSFNet Homepage. http://www.ssfnet.org/homepage.html. 2005.

[54] Lee Breslau et al. Advances in network simulation. *IEEE computer*, (0018-9162):59–66, May 2000.

[55] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang. J-sim: A simulation and emulation environment for wireless sensor networks. http://www.j-sim.org, 2005.

[56] Sung Park, Andreas Savvides, and Mani B. Srivastava. Sensorsim: A simulation framework for sensor networks. from http://www.j-sim.org/v1.3/sensor/JSim.pdf, 2005.

[57] Proceedings of the Communications Network and Distributed Systems Modelling and Simulation conference. *Scalability of Network Simulators Revisited*, Orlando, Florida, USA, February 2003.

[58] Sung Park, Andreas Savvides, and Mani Srivastava. Sensorsim: A simulation framework for sensor networks. pages 104–111, Boston, MA, August 11 2000. In the Proceedings of MSWiM.

[59] SensorSim Homepage. http://nesl.ee.ucla.edu/projects/sensorsim/, 2005.

[60] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulatino of entire tinyos applications. In Proceedings of the first ACM conference on Embedded Networked Sensor Systems (SenSys), ACM, Nov 2003.

[61] TOSSIM Homepage. http://www.cs.berkeley.edu/ pal/research/tossim.html, 2005.

[62] TinyOS Homepage. http://www.tinyos.net, 2005.

[63] Victor Shnayder, Mark Hempstead, Bor rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. Baltimore, MD, USA, 2004. Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM Press. Pages 188-200.

[64] PowerTOSSIM Homepage. http://www.eecs.harvard.edu/ shnayder/ptossim/, 2005.

[65] OPNET Homepage. http://www.opnet.com/, 2005.

[66] GloMoSim Homepage. http://pcl.cs.ucla.edu/projects/glomosim/, 2005.

[67] MaRS Homepage. http://www.ccs.neu.edu/home/matta/software.html, 2005.

[68] CORBA Homepage. http://www.corba.org/, 2005.

[69] Andreas Vogel and Keith Duddy. *Java Programming with CORBA*. Wiley Computer Publishing, second edition edition, 1998.

[70] Sandeep Bajaj et al. Improving simulation for network research. Technical Report 99-702b, USC Computer science Department, March 1999.

[71] Jean-Pierre Ebert et al. Measurement and simulation of the energy consumption of an wlan interface. Technical report, Technical University of Berlin, June 2002.

[72] Crossbow Technology Inc. Homepage. http://www.xbow.com/. 2005.

[73] Scott Seidel and Theodore Rappaport. 914 mhz path loss prediction models for indoor wireless communications in multifloored buildings. *IEEE Transactions on Antennas and Propagation*, 40(2):207–217, February 1992.

[74] Barbara Liskov and John Guttag. *Program Development in Java Abstraction, Specification, and Object-Oriented Design*. Addison-Wesley, October 2000.