Power Conservation in Energy Harvesting Sensor Networks

Tim Roberts

September 26, 2011

1 Abstract

We examine energy harvesting sensor networks, more specifically, a sensor network using the Geographic Routing with Environmental Energy Supply (GREES) algorithm. We start with a discussion of other sources of energy conservation both in energy harvesting and non-energy harvesting sensor networks. Ideas presented in these works are combined where possible with the GREES algorithm. A sensor network was actually built to test and (if possible) improve the algorithm. There were problems along the way, but they were overcome to produce a functioning energy harvesting sensor network that used solar cells as the energy harvesting unit. Tests were run on the network by giving a consistent light and battery supply, and then changing parameters of the algorithm to see their effect on the lifetime of the network, indicating the network's sensitivity to individual parameters. These results are presented, along with their interpretation, as well as an error analysis detailing the behaviour of the algorithm. We discuss how sensitive the network is to each parameter, indicating which parameters are more important to calibrate or measure correctly.

Contents

List of Figures

2 Introduction

A sensor network is a conglomeration of micro-controlled nodes that can function autonomously yet still communicate with each other. They are often used for collecting data over large geographic areas. Often the area is so large that data may not always be transmitted directly between a transmitter/receiver pair of nodes. As a result the data must be routed through intermediate nodes.

A challenge arises because nodes have limited power supplies. Logically, nodes that are closer to the receiving processor will be busier, and therefore use up their power supplies faster. Some networks have a finite, fixed power supply. These nodes require very rigid programming structures in order to ensure that each node broadcasts at the correct time. The rigid structures are put in place to get the maximum lifetime out of a battery, and thus maximize the network lifetime. If a node is out of sync, then there is the chance that its data will not be received and the node is as good as lost (until it can re-sync).

Energy harvesting sensor networks do not just have a fixed supply. They have some type of attached device that allows them to extract energy from the environment. This gives them the ability to stay active for much longer, but the complexity of the programming involved is much greater. Each node has a constantly changing amount of energy coming in to replenish the fixed supply, and the communication protocol needs to be able to reflect that. There are a few

different energy harvesting methods, each with their own unique advantages and challenges. Included in the different types of energy harvesting are: vibration or kinetic energy, wind energy, thermal energy, and solar energy (which is examined in this thesis).

How can a message be routed efficiently through a network to a destination taking into account how much energy each node has remaining? In an attempt to solve this problem, the Geographinc Routing with Environmental Energy Supply (GREES) protocol is used. In addition to using the protocol, the sensitivity of the protocol to several of the protocol's parameters is also be examined. Although this protocol could be used with other methods of energy harvesting, solar energy is the easiest method to exert control over in order to view the effects of the protocol's different parts.

Many papers on the subject of sensor networks look at different aspects like battery life, throughput, different networking techniques, etc. However, most of these papers only simulate the sensor network and the performace or power savings. We wanted to build a sensor network, in order to see in the physical realm the effects and sensitivities of an energy harvesting sensor network.

This thesis is broken down into different parts. After this introduction, there is a literature review outlining what other groups have done with regards to sensor networks. Following the literature review, there is a section detailing the exper-

imental setup we used. After that there is a discussion of the algorithm that we used to control the traffic flows within the sensor network. Then we present the results and interpretation of our experiments, followed by an analytic approach to the senstivity of the algorithm to various parameters, and finally a conclusion with ideas for future research.

3 Literature Review

This section of the thesis will present what other research groups have done in the area of energy harvesting sensor networks. The theme throughout this thesis is power conservation. Anastasi et al. [2] have created a diagram that presents various methods of power conservation in a tree diagram (see Figure 1).

This literature review will discuss each of the power conservation methods presented in Figure 1, along with information on what other research groups have done in each of these areas. In addition, it will present research on power conservation through network coverage, ending with a discussion of the parts extracted from each paper and compiled into our sensor network study.

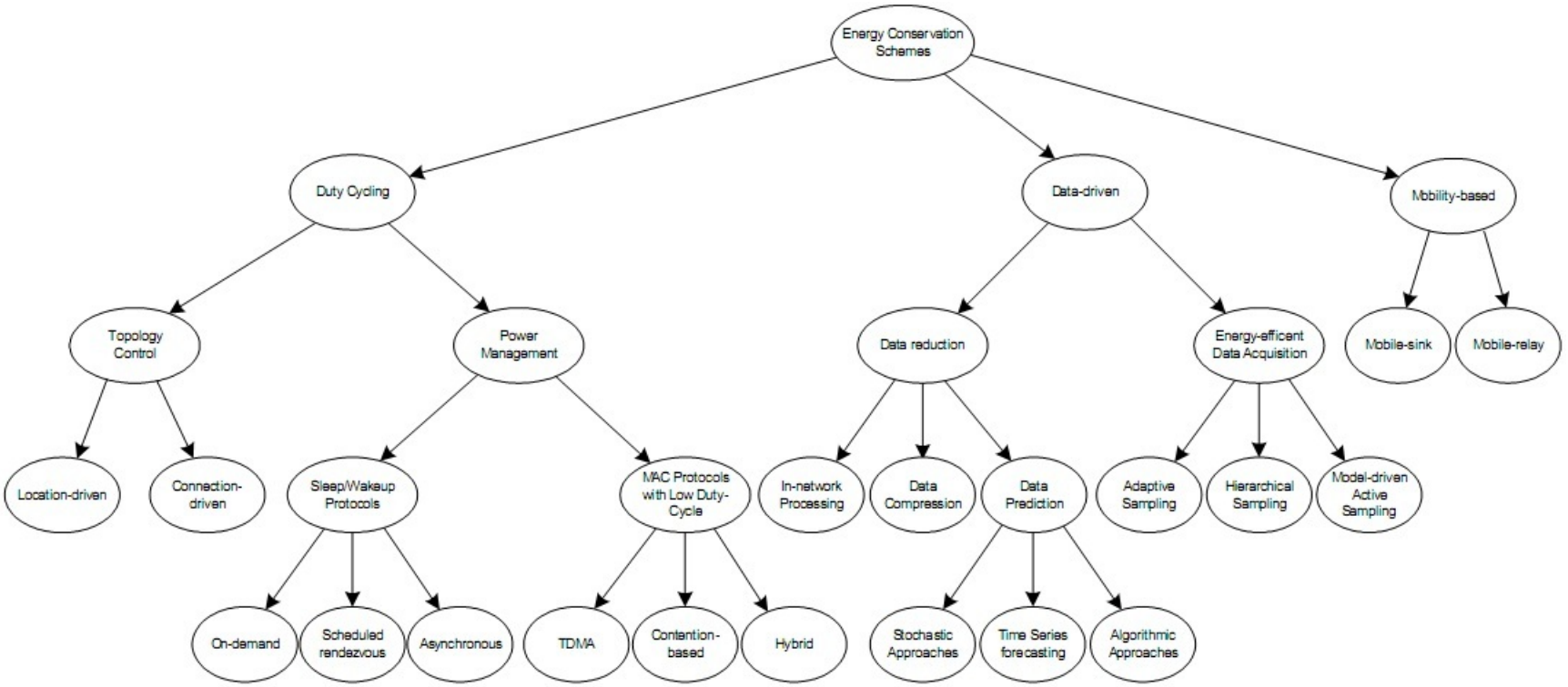

Figure 1: Energy conservation schemes and methods

3.1 Mobility

Conserving energy through mobility is fairly straightforward. All that conserving energy through mobility does is either move the sensor node or the base station to change the route traffic flows to reach the base station, balancing how much power each node has left, and therefore improving performance [2, 8, 14]. For the purposes of our experiment, we will only be considering fixed networks.

3.2 Topology Controls

Controlling the topology usually means changing the duty cycle of a network. The duty cycle of a network is how much time is spent transmitting or receiving data, processing data or sleeping. There are two options when it comes to changing the duty cycle as it pertains to the topology of the network. The first option is location driven. A location driven topology control assumes that the location of sensors in a given network is known and activates the sensors when that node's turn has come to achieve full (or partial as requested) sensor coverage [2, 35].

Connection driven topology control dynamically activates or deactivates nodes within a network to achieve the desired coverage or network performance [2, 5]. Zeng et al. [39] propose the Geographic Routing with Environmental Energy Supply (GREES) algorithm which accounts for how much energy is being harvested from the environment as well as remaining battery life to determine which nodes

to use and to dynamically route messages to the base station. A node that is harvesting more energy will receive more messages to route as that will have less effect in the long term with regards to battery life of the node.

Further research into topology controls was done by Giridhar and Kumar [11]. This paper takes two network topologies and provides an upper bound on the functional lifetime for a linear topology, and finds upper and lower bounds for a 2-D grid. The paper defines functional lifetime as the number of times that a sensor network can perform a specified task without running out of energy. This concept of functional lifetime was used to create our battery model.

Giridhar and Kumar attempt to prove how long a network will last, however, the proofs do not hold in all cases. Giridhar and Kumar show that the lifetime of a network will decrease outside of the bounds etablished if the network traffic volume is higher than accounted for when building the network [11]. In our experiment however, our network requirements are not big enough to be outside these bounds, and therefore the conditions Giridhar and Kumar established will hold.

3.3 Power Management

There are a few different power management techniques, also referred to as sleep/wake-up protocols. These protocols are: on-demand, scheduled rendezvous, and asynchronous. These protocols are presented in brief below.

On demand protocols request information from certain nodes when needed by the base station. The rest of the time nodes are sleeping, collecting data, or periodically scanning for their identifiers to be called [2, 28, 36].

Scheduled rendezvous protocols wake each node at set times in order for that node to have brief communication with the base station. The rest of the time the node is sleeping or collecting data. This method requires fine grained synchronization [2, 22, 23].

Asynchronous protocols have a node wake when it has pertinent information for either its neighbouring nodes or for the base station. This method requires some sort of collision detection or contention based scheme when in use [2, 25, 29].

3.4 MAC Protocols

There are a few options for energy conservation over MAC protocols. A MAC protocol is the method by which the Medium Access Layer (MAC) provides addressing and makes a physical connection between two computers. Existing power conservation methods include time division multiple access (TDMA), contention-based schemes, and hybrids of the two.

With a TDMA scheme, each node is given a certain amount of time to broadcast its message. It must broadcast within this time frame in order for its message to not conflict with other messages. This protocol requires fine grained synchronization

[2, 4, 15].

A contention-based scheme does not require fine grained synchronization. Anastasi et al. [2] discuss a few options for contention based schemes, but they rely on carrier sensing the medium to see if it is in use. If not the node will broadcast [2, 26, 37].

A hybrid protocol is fairly straightforward, combining a TDMA scheme with a contention based scheme [2, 9, 38]. With this scheme, a node will only broadcast if the medium is not currently in use by another node.

Van Dam and Langendoen [32] propose a method of energy conservation that dynamically changes the length of the duty cycle. The paper claims that this will outperform a fixed length duty cycle in certain situations by a factor of five. The motivation for this scheme is to solve the problem of "idle listening". This happens when a node is listening for a message to be transmitted to it, when the message that is sent is not for that particular node, or there are no messages being transmitted at all. They propose to solve this problem in two parts: they suggest that there should be two types of message to broadcast, and that the duty cycles of the nodes should be modified. First off, they propose one message type to communicate between one or all of the nodes in the network, then a different message type to the base station. To increase the power savings, the different types of messages are of different lengths. Obviously, the shorter message will use less

power. Shorter messages are used to communicate between nodes, while the longer messages are used to communicate with the base station. The second method of power saving is modifying the duty cycle. If there is nothing important happening in the network, the duty cycle becomes longer. If the duty cycle is longer, there is less time spent transmitting and receiving, and therefore less power is used. When a node event occurs, the rate at which a node broadcasts increases to provide better network response.

There were a few problems with the protocol discussed by van Dam and Langendoen that needed to be overcome. Firstly, in order to make sure that as many messages were delivered as possible, a request-to-send (RTS) and clear-to-send (CTS) scheme was implemented. This is a fairly standard networking method for collision avoidance. Secondly, when they implemented this protocol on a physical sensor network they discovered that clock drift became a huge issue, taking the network down in as little as 10 minutes. As a result they discovered that this protocol needs fine grained synchronization. To synchronize, they simply broadcast a synchronization signal and each node would adjust its own clock to 50% of the difference between its current clock and the synchronization signal. This ensured a converging time difference and allowed the nodes to stay synchronized over long periods of time [32].

3.5 Data Prediction

Data prediction schemes are a subset of data reduction schemes. We discuss three data reduction schemes: in-network processing, data compression, and data prediction. In-network processing (each node processes the data and transmits only the final result) and data compression (compress the data before transmitting) are self explanatory, so this subsection will focus on different types of data prediction: algorithmic approaches, with a focus on a stochastic approach and time series forecasting.

An algorithmic approach uses a defined mathematical model to predict future data. One technique is to treat an incoming value as a pixel and use mpeg4 prediction to estimate future values [12]. Another example would be to compare the current value to the expected value and only transmit data if the current value is outside of a given tolerance of the expected value [27].

A stochastic approach to data prediction uses probabilistic methods to estimate what will happen next in the network. Ideally, the stochastic approach would eliminate the need for sensors, but of course that can never happen. A stochastic approach can be used in conjunction with a sensor network to cut down on energy used by sensors if the stochastic approach is able to model the network correctly at least some of the time [7, 17]. One potential method for doing this would be to take into account the past history of the network to see if a value is outside of a

plausible range.

A time series forecasting approach bases future predictions on previous data. It will keep track of data flow and correlate it with the times of transmission. This requires being able to run the program and record data multiple times or at least recognizing patterns within data. If the second run through is the same as the first, or at least similar, it can make an estimate about what will be transmitted and save processing time by transmitting the estimate [20, 31].

3.6 Energy-Efficient Data Acquisition

Energy Efficient Data acquisition uses adaptive sampling, hierarchical sampling, or model based active sampling in order to cut down on the power used by sensors.

An adaptive sampling approach uses similarities over time or space to send less data. If a base station can categorize and store received data, sometimes a node may not need to transmit data because the base station already has the data stored (this would most likely be used in conjunction with something like time series forecasting). Depending on sensor node density it can read from only one node, because that node is able to speak for several of its neighbours (for example a number of temperature sensors spread out over a small area) [1, 16, 33].

A hierarchical sampling approach has a network overseer that knows what node has what sensors and how much power they use. A more accurate sensor

will typically use more power. If a node has a power draining sensor and a power efficient but less accurate sensor, it can compare the results of the two. If the less accurate sensor is reading the same as the more accurate one, it will only access the power efficient sensor. It will periodically check the two (sometimes a node will have more than two) sensors to make sure that they are getting consistent readings. The node will balance the power draining and accurate sensor with the power efficient and less accurate sensor [2, 19, 30].

A model based active sampling approach is a similar idea to the data prediction algorithmic approach, but rather than controlling whether a node sends a message, a base station controls whether a sensor will use energy to read its sensor. It uses a mathematical model to come up with a best guess of what the sensor input could be. If it can predict what the sensor will read, the node does not need to sample the sensor [10, 24]. This value would have to fall within a pre-defined range in order to be considered for use.

3.7 Network Coverage

One more possible method for increasing a network's uptime is by increasing the network coverage. If multiple nodes cover the same area, then it is a waste of power to be using all of the nodes at the same time. By shutting down nodes that are not necessary at that point in time and only waking nodes as the nodes

currently being used die off, the uptime of the network is increased.

Gupta et al. [13] discuss how to organize a number of sensors into as small of a network as possible but still be able to respond to a query. The paper contains multiple algorithms to organize the sensors to use as few sensors as possible and therefore save energy.

First off, they describe a centralized approximation algorithm. This algorithm runs in polynomial time, and returns a coverage area that is $O(r \log n)$ where n is the network size and r is the communication distance between two nodes. This is not an optimal algorithm, however it is easy to distribute over large networks by creating a network of networks, and has very low communication overhead.

The second approach that this paper discusses is called the Distributed Self Organization Algorithm. This method is essentially a greedy algorithm that connects one sensor to another sensor, one at a time, until the entire desired region is covered. The algorithm has required nodes that need to be included for full coverage, as well as intermediate nodes to facilitate communication between all nodes [13].

More research into network coverage was performed by Xing et al. [34]. This paper discusses the relationship between connectivity of a sensor network and the coverage that the network provides. The paper also talks about the Coverage Configuration Protocol (CCP) and combines it with the Span technique [6] in

order to guarantee connectivity as well as coverage. The Span method dynamically maintains network topologies and ensures that nodes of a network are connected. CCP is a protocol that allows a wireless sensor network to control how much sensor coverage the network provides. If fewer gaps in the sensor coverage are required, then logically the network needs more coverage. If an important event is missed, then it is called a fault. In order for there to be fewer faults, more network coverage is required. The Span technique is a method for ensuring that the network nodes are able to remain in contact with each other, even if the network size is changing dynamically [6]. Xing et al. [34] combine CCP with SPAN to create a network that can dynamically control how much network coverage is provided while ensuring (with SPAN) that the network stays connected.

One last paper by Giridhar and Kumar [11], tries to generate a generic definition of network lifetime using formal methods. Their definition uses a number of existing concepts as well as adding a few new ones. The paper combines connectivity and coverage to form something called connected coverage. It also incorporates a concept known as graceful degradation, which is that the performance of the network gradually decreases as more nodes fail rather than catastrophic failure once one (or a small number) of nodes fail. One of the new concepts in this definition is expressing how well the network can cope with disruptions or interference. It also introduces time-integration, which is ensuring that the network completes a given

task before a given deadline, rather than making sure that the program performs the task each time the program iterates [11].

3.8 Energy Harvesting

A second issue that deserves some extra consideration would be how a network that harvests energy from the environment deals with power conservation. A network that extracts energy from the environment will have different considerations in how to conserve its supply. If a node extracts power from the environment it needs to be able to keep track of how much power it has harvested and communicate that information to other nodes. The other nodes can make a decision from there whether to use that node or a different node that has harvested more energy.

With that in mind, Kansal et al. [18] take into account two design considerations: energy neutral operation, and while in energy neutral operation, getting maximum performance from the network. Energy neutral operation means that a network is using the same amount of energy in processing and transmitting as it harvests from the environment. It is not gaining energy, but it is not losing energy either.

To harvest as much energy as used and achieve energy neutral operation, they discuss a few energy storage options: no energy storage, ideal energy buffer, and non-ideal energy buffer (a battery or capacitor). Going further in depth, the pa-

per discusses energy storage and related concerns, achievable performance level, and measurement support. The paper then goes on to discuss energy usage algorithms, including energy prediction models, energy optimization, dynamic duty cycle adaptation (this is discussed in van Dam and Langendoen [32]), and evaluates all of them, both theoretically and experimentally through simulation [18].

Zeng et al. [39] devise a method of routing messages through a network to a final destination while keeping track of and using an environmental energy supply. The algorithm they develop, the Geographic Routing with Environmental Energy Supply (GREES) protocol, is the main focus of this thesis. The protocol aims to balance how much energy is coming in to a given node with how much charge is left in a battery and distributing that balance across the entire network to extend the network's lifetime. More information about the GREES algorithm will be given in later sections.

3.9 Culmination

For the purposes of our experiments, various pieces were extracted from the different papers. Firstly, our sensor network uses a mobility based scheme, ensuring that each sensor node is close enough to either other nodes it can talk to, or the base station. Each node has a fixed power output through its transmitter, but each node can be placed in different positions to allow for either more dense or

more sparse coverage.

The network also uses a crude form of TDMA. Each node is given a block of time to transmit its message. There are sometimes conflicts between nodes due to clock drift, but the amount of time between broadcasts compared to the length of the broadcast message mitigates this problem.

To create our network, we also use a modified energy efficient MAC protocol. Only certain nodes are able to communicate with the base station. However, information from the nodes that can communicate with the base station still needs to be able to reach all of the other nodes. To get around this, all nodes can eavesdrop on information bound for the base station, but as was stated earlier, only certain nodes can communicate with the base station.

Due to the small size of the network, attaining connected coverage is fairly easy. The papers discussed above talk about formal methods for proving this, but in a network with only four nodes and a fixed topology (see Figure 2) attaining connected coverage is not difficult.

This network also attempts to achieve energy neutral operation using a non-ideal energy buffer. This is an admirable pursuit; however it is not successfully implemented due to a shortage of energy from the energy harvester. The solar cells are simply not able to harvest enough energy to power the network or recharge the energy buffers. This means that although the solar cells will do their best

to ensure that the network stays up and running for as long as possible, it will eventually die. This gives us the oppourtunity to look at different parameters of the GREES algorithm to see the effect and sensitivities of each parameter.

4 Experimental Setup

The basic hardware used in this experiment is made by Libelium [21]. The nodes used in this experiment are the Squidbee model and there are four of them. The base station is the Libelium Squidbee Gateway. The datasheet for the Squidbee nodes is included in Appendix A.

There were several reasons behind choosing this hardware platform. First off, it is the least expensive platform still in production, and it is open source, allowing full access to all of the features.

The Squidbee platform is based on the Arduino duemilanove with an XBee shield on top. Arduino is an open source embedded programming platform with a myriad of add-ons and modifications. An Arduino development board gives a programmer access to all features included in the board without charging any licensing fees. XBee is a wireless communications protocol that conforms to the 802.15 wireless standard [3] published by the Institute of Electrical and Electronics Engineers (IEEE). The XBee shield allows the nodes to transmit and receive information wirelessly. For the purposes of this experiment, a data transmission rate of 19200 baud is used. Any baud rate could be used, ranging between 1200 to over 200000. 19200 was selected due to it having a reasonably high data transmission rate without the synchronization issues that arise from higher baud rates. If a higher baud rate was used, and the clocks of two different nodes were out of

sync, the two nodes would be unable to communicate with each other, except for very brief periods. The higher the baud rate, the more sensitive a node is to its synchronicity with other nodes.

A small experimental setup of four nodes and a base station is used for the research in this thesis. This was because of the cost of the nodes, as well as for reasons of simplicity. Fewer nodes allows for better observation of the effects of any changes to the network.

In this sensor network, a basic node consists of a temperature sensor, an Arduino Duemilanove processor, an XBee shield for Arduino Duemilanove, a solar cell and a 9V battery. The lighting in the room is provided by 40 watt fluorescent tube lights for controlled experiments, sunshine was early afternoon during the months of April and May.

4.1 Basic Setup

The basic setup of the sensor network is as illustrated in Figure 2 below. Due to the limited number of nodes available, the nodes are arrayed in a more linear fashion than they would be in a typical field application.

A node will calculate which node to broadcast its message to, taking into account the location of the node, as well as the power the node has left and how much power it is receiving from the environment. The messages that are being

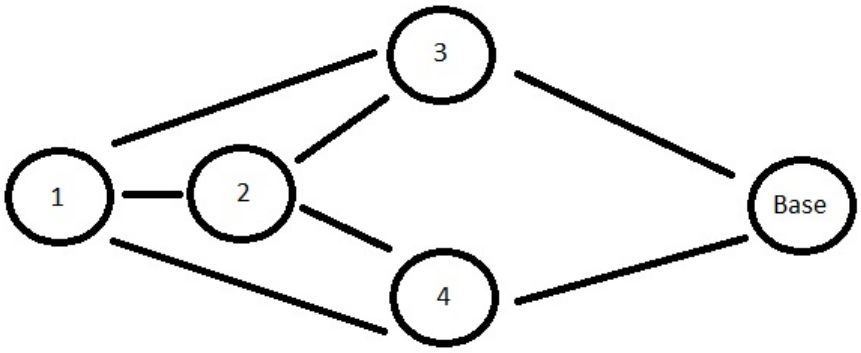

Figure 2: Basic network setup

transmitted will be broken into several segments. One will be from the temperature sensor, one will be solar energy collected, one will be battery life remaining, and the final segment is one number representing where the message has come from, and where it currently is. By the time the message arrives at the base station, this final segment will have the complete route the message has followed to get there.

Figure 3 illustrates the route a message takes, as well as the components of each message. The message starts with an address tag telling which node the message is destined for. The second number is the temperature in the area of the originating node, the third number is the remaining battery charge of the originating node and the final number is the solar information of the originating node. As can be seen, each node relaying the message only changes the address tag to allow for re-transmission. Node 4 does not change the address tag. Nodes 3 and 4 are not able to send messages to any other node for re-transmission. They are able to

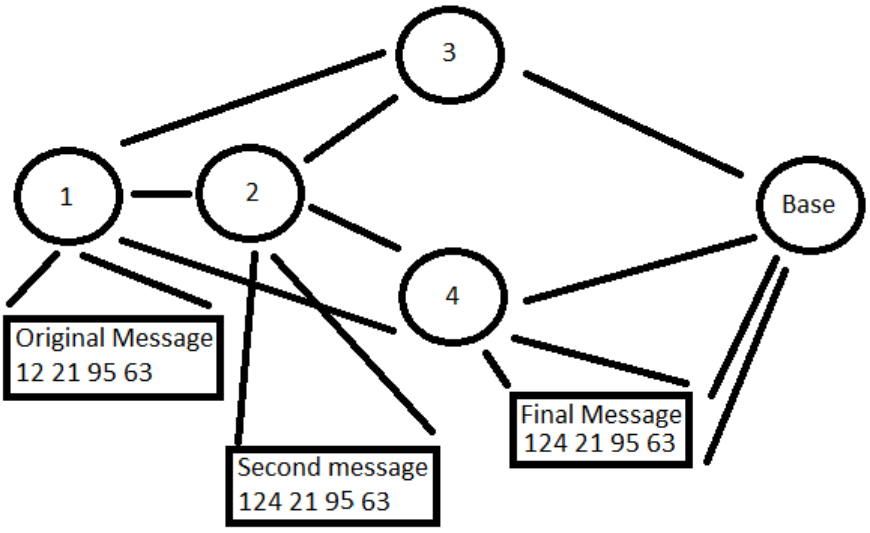

Figure 3: Message Route

broadcast information to other nodes, but other nodes will not relay any messages from them.

4.2 Problems With Implementation

As with any larger project there are problems that arise during the implementation of the planned network. The first problem that came about is the issue of connecting a solar cell that supplied 6.8V into a system that also uses a 9V battery, while the circuitry of the system itself runs off of 5V. The problem with so many different voltages is that the voltages determine which way current flows. If current flows the wrong way across a solar cell, it will not only ruin the solar cell, but also curtail the battery life of the system. With so many different voltages,

there is a surprisingly simple solution.

With a solar cell, energy flowing the wrong way will cause damage, so it is very important to make sure that current is flowing in the right direction (see Figure 4). Since the solar cell is part of the power supply, it needs to be connected in parallel with the 9V battery. Doing this, however, will result in current flowing backwards across the solar cell and causing damage. A simple diode correctly placed should fix or at least minimize this problem (see Figure 5). The power demands of each node are fixed, and any current that can be supplied by the solar cell will reduce the power drawn from the battery, thus increasing the network lifetime.

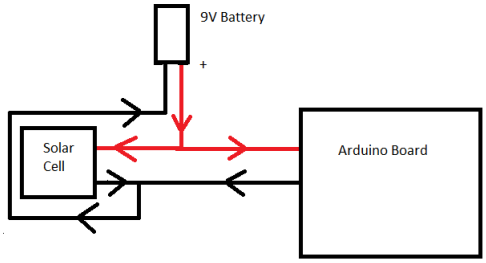

Figure 4: Poor Current Flow

In order to supply the correct voltage to the circuitry of the node, each node is equipped with a 5V voltage regulator. So as long as a voltage of at least 5V is supplied to the node, the regulator will take care of the rest.

The second problem that we have to deal with is how to get the values relating to how much battery life is left as well as how much solar energy is being received.

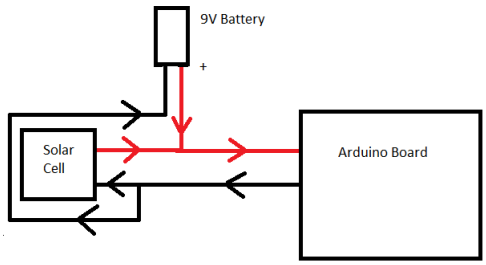

Figure 5: Correct Current Flow

Without complicated circuitry, the amount of power that is being supplied by the solar cell is very difficult to ascertain. The workaround used by Kansal et al. [18] is to simply use the voltage supplied by the solar cell as the power. In order to find the voltage, the reference voltage comes from the 9V battery [18]. This is an imperfect solution as the voltage in the battery will drop over time, skewing the solar voltage.

In the grand scheme of things, this is not a huge problem. A battery's voltage stays consistent over the majority of its lifetime, dropping off rapidly when the battery is becoming depleted. As the battery voltage drops, to the node it seems as though the solar cell is suddenly getting much more power. This will compound how quickly the node dies off as more signals are routed through it as a result of the "increased" power of the solar cell. The thing to note is that this will only happen at the very end of the battery's charge, and thus is not a major problem.

In terms of finding out how much battery life remains, there is no fixed reference

point to measure what voltage the battery is providing. This necessitates a model for the remaining battery power. Batteries remain fairly constant over the majority of their lifespan, but drop off quite quickly when they reach the end of their lives. This creates a problem when each node is in a dynamic power draw situation, and the amount of power drawn from the battery changes each time the network is used (in a real world situation).

In order to create a model for the battery, the number of messages that each node transmits over a lifespan was recorded several times, and an average of the total messages broadcast is used to create an inverse exponential model of the battery. Once the node has broadcast a certain number of messages, it crosses a threshold into battery degradation. This is not perfect data, as the threshold used is based on the best estimate of when the battery started to degrade. Table 1 describes the simple model for the battery. Each battery is different, and even under the closest to ideal environment possible, each node will return different data each time. Using an imperfect model of the battery may not be the ideal situation, but that model is still better than no model. This method for modelling the battery will for most applications be more than acceptable. If a very old rechargeable battery is being used that has had time to develop a memory (old rechargeable batteries will sometimes charge to a certain point and can lose their charge quite quickly) it has the potential to throw the model off by returning to

the battery's memory sooner than the model would predict.

Model Up To Threshold	$E_b = 1$
Model After Threshold	$E_b = 1 - \frac{1}{Threshold}$

Table 1: Battery Model

A third problem with implementation is creating a realistic test scenario with only four nodes. In order to create the maximum potential for nodes communicating with each other and therefore test how well the nodes were able to conserve their power, we devised the layout in Figure 2. Nodes 1 and 2 are able to communicate with all other nodes, but are unable to communicate with the base station. That job lies with nodes 3 and 4. This creates the most thorough test scenario, and if implemented in a larger scale would be in a similar layout except in a full circle around the base station, not just on one side.

In terms of network traffic, each node broadcasts its individual message once per second. This ensures a fairly real time picture of what is going on in the network. During the time a node is not transmitting, it will listen for a portion of that time to see if a message has been broadcast that contains its address. If it detects such a transmission it will re-transmit it to the next node or base station. The path that each message can take changes from day to day and situation to situation depending on the conditions of the network at that time. As the number of nodes increases or decreases the amount of time spent listening can change;

logically nodes that are closer to the base station will have a higher concentration of messages to re-transmit, and thus more time must be spent on listening than transmitting.

In order to diagnose problems with setting up the network, a method of keeping track of what route a message took to the base station needs to be devised, as well as ensuring that each node only received messages that were destined for it. If a message is being routed incorrectly, it would be inefficient to allow that to continue, and potentially disastrous to the network lifetime. The solution came in the form of a simple address tag. Each node is able to add a unique signature to a message that was being routed through it. The node generating the original message would append the first tag (its own unique ID times 10, plus the destination node), and each successive node multiplies the previous tag by 10, and add its identification number. For example, a message that was routed from node 1, to node 2, to node 4 would arrive at the base station with the tag of 124. Each node would be able to check and see if the message it received contained its address, ensuring reliable message delivery. This is a fast method of keeping track of the health of the network for a negligible cost.

Each of the problems presented in this section were dealt with and overcome to produce a cohesive network. The paper by Zeng et al. [39] is detailed enough in its description that the actual programming of the network is not a difficult

task, and the challenges surrounding physically building the network proved to be challenging but not insurmountable.

4.3 Software

The software used in this endeavour from one of the nodes has been included below. The language itself is Arduino, which is similar in style and syntax to C. Each node's code is approximately 120 lines long. In order to use a different protocol, the sendto() function would need to be modified to incorporate the new protocol.

In order to test the code, each node was incorporated one at a time in an incremental fashion. Node 1 was programmed, and checked to be sure that it was functioning properly. Then node 2 was programmed, made sure that it was able to communicate with node 1 as well able to transmit its own messages. Then nodes 3 and 4 were programmed and incorporated in a similar fashion. This method ensured that the network functioned correctly during the tests.

5 The Algorithm

5.1 The Basic Algorithm

The basic algorithm used is the GREES algorithm by Zeng et al. [39]. The algorithm used to determine where the information should be routed is calculated by placing a numerical value on how much battery life is left in a given node, with the lowest cost node receiving the message. The cost of sending the message from node N to node i over distance D ($C_M(N_i, D)$) is:

$$C_M(N_i, D) = \frac{(E_b(N_i) \cdot \eta^{\lambda_{N_i}})}{\log \eta \cdot (\mu_{N_i} + \epsilon) \cdot PRO(i, N_i, D)}, \tag{1}$$

where $PRO(i, N_i, D) = (Dist(i, D) - Dist(N_i, D)) \cdot FDR_{i,N_i} \cdot FDR_{N_i,i}$

This section will first look at the different terms that are contained in the algorithm, then discuss the different parts of the algorithm and their purpose within the scope of the algorithm.

5.2 Description of Terms

$E_b(N_i)$ - battery capacity of node N_i

$E_r(N_i)$ - battery charge remaining

η - an arbitrarily chosen constant.

λ_{N_i} - percentage of remaining battery life.

μ_{N_i} - most recently received expected energy harvesting rate on node N_i by node i.

ϵ - energy harvesting uncertainty.

$PRO(i, N_i, D)$ - progressive distance per data frame from i to N_i towards D.

FDR_{i,N_i} - Frame Delivery Ratio of messages sent from node i to node N_i

$FDR_{N_i,i}$ - Frame Delivery Ratio of messages sent from node N_i to node i

$Dist(i, D)$ - Distance node i is from the base station at D

$Dist(N_i, D)$ - Distance node N_i is from base station at D

The difference between a given node i and node N_i is N_i is the information from node N being received by node i. For example, 2_4 is information from node 2 being received by node 4.

5.3 Algorithm Elements Discussion

The algorithm outlined above has many different parts. This section examines in greater detail the different parameters, and will also provide a short description of what each parameter does in the algorithm. Not all elements that are listed above will be discussed, since some of the concepts are fairly simple, or not necessary to understand the algorithm and how it works. Some parts listed in the description of terms section have multiple instances, but they will be discussed only once in this section, as the discussion applies to all instances.

5.3.1 μ - Energy Harvesting Rate

Typically, this parameter has the largest effect over the lifetime of a network. If a node is receiving lots of energy, the effect on its battery of transmitting a message will be much smaller than if the node is receiving little or no energy.

This value is continuously changing throughout the lifetime of the network, and each node updates other nodes in the network about how much energy that specific node is extracting from its environment. With this information a node can more effectively route information by transmitting it to a node that will have less impact on its battery consumption.

5.3.2 ϵ - Energy Harvesting Uncertainty

The energy harvesting uncertainty term is very important with regards to the algorithm's performance. This tells the algorithm how accurate the value of μ is that is being received. The digital to analog converter (DAC) is receiving data from the solar cell that is attached to the node. The smaller the value of ϵ the more accurate the value of μ. This value has to be calibrated in order to find out just how accurate the reading is from another node, but once calibrated can remain fairly constant. If the uncertainty value is incorrect, a different node could potentially be more efficient to send data to, leading to a longer network lifetime.

This value can either be positive or negative. In most of our experiments

we treated this value as positive, which makes the algorithm read values more optimistically than is actually true. The effect of this is that the algorithm treats all solar values as a best case scenario. If the number were negative, the algorithm would treat $\mu + \epsilon$ as a worst case scenario. Overall, the amount that the algorithm is off by (in an ideal case) would be symmetric in terms of the network lifetime.

5.3.3 FDR - Frame Delivery Ratio

The FDR is how often a message that is sent gets received by the node it was intended for. The more reliable the networking protocol, the higher the FDR and therefore the performance of the network is improved.

This parameter attempts to balance how likely a node is to receive a message versus how much closer to the intended destination the intermediate node is. If a node is able to transmit a message farther towards the base station, but it is not likely to be received, it will balance that with a shorter transmission distance but increased likelihood of reception. A lost transmission is nothing but wasted energy, so its important to make sure as many messages as possible are reaching their intended recipients.

The FDR is a value that we started of by picking arbitrarily. In our case, we started off with a value of 0.8 and tried different values both above and below to try and see which value gave optimal performance. Ideally, this value would be calculated and not determined through trial and error, but that is beyond the

scope of our experiment. Experimentally we determined a value of 0.999 gave the best results, by holding the rest of the algorithm constant and only changing the FDR to determine its sensitivity and effects on the lifetime of the network.

5.3.4 η - Scaling Constant

The only way that η is defined anywhere in [39] is as an arbitrary constant. As λ decreases, this will bring down the value of η^{λ} in the numerator of the algorithm. As the value changes, it is balanced by the η in the denominator. In terms of the cost of transmission, as the battery level decreases, the cost of transmission to that node increases. In our case, we took each value of η to be the same, however for future research different values for different nodes could be looked at.

5.3.5 λ - Percentage of Remaining Battery Life

This is the parameter that measures how much charge remains in a battery. It is a simple calculation:

$$\lambda_{N_i} = \frac{E_b(N_i) - E_r(N_i)}{E_b(N_i)} \tag{2}$$

The percentage is how much charge remains in the battery subtracted from the full battery charge divided by a full battery charge. This fraction is then used in conjunction with the scaling constant η (discussed above) to give the network information about which nodes have more remaining battery life and which nodes have batteries that are running low on power.

This value should be chosen with some thought beforehand. If the number is less than one, the cost function could be negative, which is not going to give good results. The other problem is if the value of η is too high, then in essence this will amplify the effects of λ and distort the algorithm by focussing too much on λ. An appropriate value of η would be in the range of 10,000.

6 Results and Interpretation

This section will present a brief overview of what each experiment will be looking at and the corresponding results. Afterwards there will be an interpretation section, looking at why the algorithm performed the way it did. The sections that will be discussed in the parts below are: percentage of remaining battery life, the energy harvesting uncertainty ratio, the energy harvesting rate, the Frame Delivery Ratio, and the scaling constant.

Just as a brief summary of the previous section, the algorithm being examined in this section is called the Geographic Routing with Environmental Energy Supply (GREES) protocol and was developed by Zeng et al. [39]. The algorithm breaks down the cost of transmission, and will transmit a message to the lowest cost node. The cost of transmission is determined by:

$$C_M(N_i, D) = \frac{(E_b(N_i) \cdot \eta^{\lambda_{N_i}})}{\log \eta \cdot (\mu_{N_i} + \epsilon) \cdot PRO(i, N_i, D)} \tag{3}$$

where $PRO(i, N_i, D) = (Dist(i, D) - Dist(N_i, D)) \cdot FDR_{i,N_i} \cdot FDR_{N_i,i}$

6.1 λ - Percentage of Remaining Battery Life

6.1.1 Results

The value of λ is calculated using the formula:

$$\frac{E_b(N_i) - E_r(N_i)}{E_b(N_i)} \tag{4}$$

where $E_b(N_i)$ is the battery voltage as if the battery were just removed from a charger, and $E_r(N_i)$ is the current voltage provided by the battery. This is a calculated value that is based on monitored values by the sensor. It was a relatively easy value to manipluate within the program, with some interesting results. A graph of the results can be seen in Figure 6, and as you can see the network lasts longer by forcing the node to read the battery as empty.

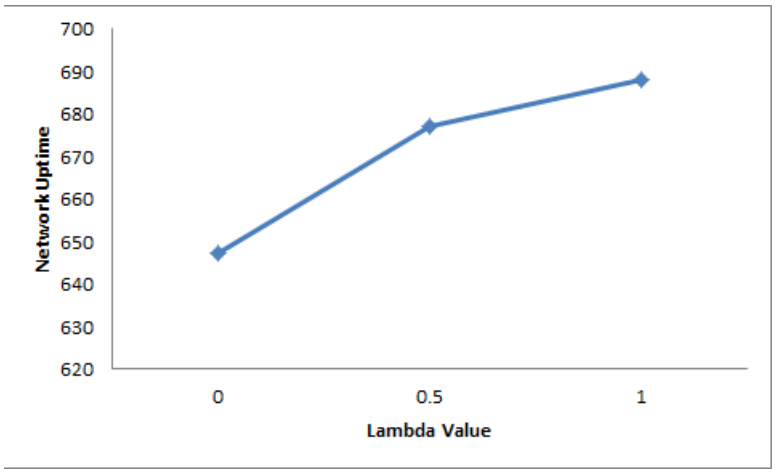

Figure 6: λ Values vs Network Uptime

6.1.2 Interpretation

In (5), by keeping the value of λ fixed, this allows the effectiveness of λ as a parameter to be tested. It should also be stated that λ as a parameter is only being changed in the code of the algorithm, the physical energy source starts with a full charge, and discharges at a normal rate. Assigning a value to λ instead of monitoring it allowed us to look at how the algorithm performed without λ, or how λ made the algorithm behave. By assigning a value to λ it essentially holds the numerator of (5) constant, forcing the cost to be completely determined by the denominator. For this test, the overhead lighting was consistent, so the value of μ was held constant, forcing the algorithm to choose its route based solely on $PRO(i, N_i, D)$.

A more detailed description of why λ makes the network last longer the closer the battery is to empty is as follows. The value of λ has less effect over η the closer λ is to 1. This brings the value of the top half of the equation closer to zero, making the bottom half of the equation much more important. The farther away from a node a receiving node or base station is, the higher the $PRO(i, N_i, D)$ value is. With the $PRO(i, N_i, D)$ being higher, it costs less to send a message to a more distant node.

If λ is closer to 0 than 1, the numerator of the equation has more weight in determining where a message gets sent, since η^{λ} is evaluated to be significantly

closer to 1 instead of being significantly less than 1. This forces the node to calculate a higher cost to send a message, and the algorithm sends the information on shorter hops between nodes to conserve energy. Logically, the shorter the network uptime, the more intermediate hops information is taking in order to reach its destination.

This parameter illustrates the balancing act this algorithm is trying to do, balancing the energy of sending a message farther away and taking fewer steps, versus taking more steps and using less energy per step. As the results indicate, it is more efficient to send a message a longer distance and use fewer steps than to broadcast shorter distances and relaying over multiple hops to conserve energy.

6.2 μ - Energy Harvesting Rate

6.2.1 Results

The energy harvesting rate, represented by the symbol μ, measures how much energy is being harvested by a node in a given period of time. This value is transmitted to all of the other nodes so that they can determine whether to route a message through that node or not.

The control experiments were carried out under fluorescent lighting which is not an optimal condition for solar cells. The fluorescent lighting was used in order to provide a consistent value for μ so that other parameters of the algorithm could

be measured and monitored.

When sunlight is provided to the solar cells, the network performance increases drastically. The first test result is not as good as the others, but that is because it was clouding over that day and the solar cell kept coming under shadow. A solar cell requires bright light to function at peak efficiency. If a node comes under shadow (from a cloud for example) it does not receive very much energy. When this is the case, a solar cell may be able to extract more energy out of an overhead fluorescent light. In direct sunlight, there is a marked extension to the lifetime of the network.

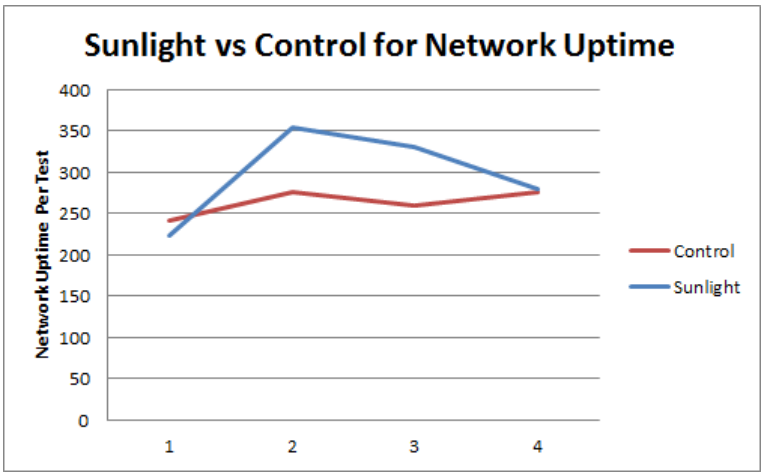

Figure 7: Sunlight vs. Control Graph

This graph provides a good representation of the performance of the network when dealing with sunlight or diffuse sunlight. Since this is the real world, the solar cells will not always have direct sunlight shining on them. This graph shows

the effect of a cloudy day compared to fluorescent lighting, as well as the significant network uptime increases when the cells have access to direct sunlight.

If one were to compare the results of this algorithm to just a straight transmission protocol that did not account for energy consumption or harvesting, the algorithm that accounts for energy would perform better. The other algorithm may be able to squeeze more throughput or speed from its lifetime, but that would depend on the algorithm chosen and the purposes of the network. The energy aware algorithm would perform better simply due to the fact that it is able to balance the power demands of each node with the required performance of the network. This allows the nodes to perform for the longest period of time as a cohesive whole, rather than dying out in a piecemeal fashion.

6.2.2 Interpretation

The parameter μ is a measured value that reflects how much solar energy a node is harvesting at any point over the course of a network's lifetime. The value of μ has a greater effect over the network uptime than any other parameter of the algorithm. This makes sense, considering that the algorithm is designed to be used on energy harvesting networks. If a network is not harvesting very much energy, the improvement in network uptime will not be very good. If a network is harvesting lots of energy, the network uptime gains should reflect that.

The value of μ in theory could be assigned a value and over the course of several

experiments its effect could be determined on the network. However, that was the reasoning behind using fluorescent lights as the control. Instead of programming in a hard number for the value of μ, the fluorescent light provides a consistent standpoint so that the measured value of μ would always be very close, if not the same, over many tests.

It should also be mentioned that the measured value of μ will seem to spike at the very end of the lifetime of the network. This is simply because the reference value of the voltage for the solar cell is provided by the battery. At the very end of a battery's life the voltage takes a sudden and sharp drop until the battery is dead. This will have the corresponding effect of spiking the solar cell voltage:

$$MeasuredSignal = \frac{SolarCellVoltage}{BatteryReferenceVoltage} \tag{5}$$

From this, as the battery voltage drops to zero the solar cell voltage will spike quite high. It is bounded above, as the signal converter has a maximum value.

The network sensitivity to the value of μ is relative. If μ is consistently wrong across all nodes, the network would continue to function as if the value of μ were correct. If each node were plus or minus a certain consistent value on what the sensor is reading, the network would continue to function at near optimal efficiency. If each node is not consistent in the error coming in, then the network sensitivity depends on the scale of the error. A small error will have little effect, and the

network will continue to function near peak efficiency. If the error is large, then a node will get either too much or too little traffic, causing problems with network traffic flow and battery lifetime issues. This is the very reason why ϵ is incorporated into the algorithm.

6.3 ϵ - Energy Harvesting Uncertainty Ratio

6.3.1 Results

The energy harvesting uncertainty ratio, represented by ϵ, is a calibrated error term for μ. The value of μ is constantly being sent throughout the network, in order to provide the nodes with the most up to date solar information. The information is used to calculate which node is the cheapest in terms of energy cost to transmit a message.

The uncertainty is a representation of how accurate the energy harvesting rate is. Ideally, this number would be calculated based on the uncertainty of the energy harvesting unit, and any other applicable factors of the node and network. As the solar cells we have do not supply this information, a trial and error method must be used. Several different numbers are tried for this value (see Figure 7), and the number that kept the network running for the longest is closest to the proper value for ϵ. If the node is broadcasting to the network that it is receiving a lot of solar energy but in reality the node is in the shade or it is cloudy, the uncertainty factor

attempts to correct this inconsistency. It should be mentioned that this parameter was determined experimentally by trying different values for ϵ, which leaves room for the fact that there could be another value of ϵ which performs better than the one we determined, and both of these could be different than the theoretical value of ϵ. Perhaps in the future, a method will be developed to check and resolve this issue.

Ideally, this value would be calculated and quantified before being coded into the algorithm. This would involve determining the error of the energy harvesting source. Afterwards, the value of ϵ could be played with as either larger or smaller to see if a better value could be determined through experimentation. In the ideal case, the calculated value of ϵ would be the correct value.

In order to test out the effect of the energy harvesting uncertainty ratio different values of ϵ are used. In the demonstrated network, the optimal value is found to be around 1, and the lifetime of the network falling away in an arc on either side of that.

6.3.2 Interpretation

All of the tests on the network are performed in a consistent, controlled environment. The solar panels only receive lighting from overhead flourescent lights, with negligible other light sources. This forces any values relating to the solar panels to be consistent, allowing the value of ϵ to be calibrated accurately.

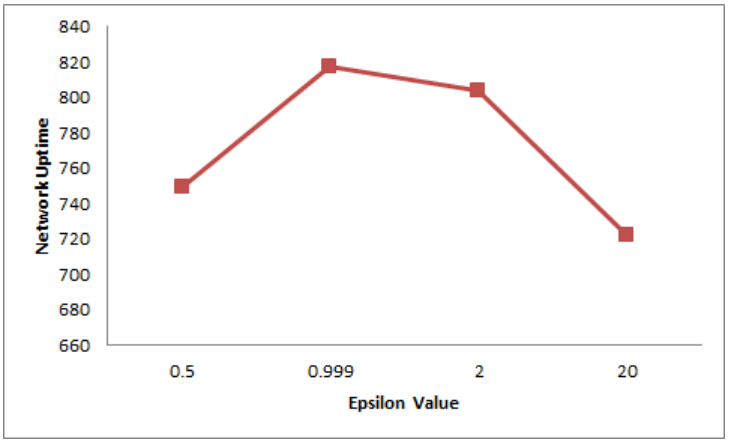

Figure 8: ϵ vs Network Uptime

The value of ϵ depends on the type and linearity of the error of the energy harvesting unit. Some energy harvesting aparatuses may have less error or more error as the energy harvesting intensity increases or may have less error or more as the energy harvesting intensity decreases. The value of ϵ depends on the method of energy harvesting as well as how much energy it is harvesting in a given period of time.

Our value for ϵ is determined through trial and error to be 0.999, which for our purposes can be treated as one. If the value of ϵ were too low or too high, the cost of transmission will be distorted. In an ideal situation the cost would affect each node in the same manner and ϵ would become redundant. In a real world application, ϵ has different values in different nodes, since each node is a little bit different.

If ϵ were too low, then the cost of transmission to one node would be higher,

and therefore that node would get less use. If all nodes have a value of ϵ that is too low, then the cost of transmission will all be adjusted by a common amount, and which node gets the transmission depends on other factors.

In a similar fashion, if the ϵ value is too high on one node, that node would be bombarded with messages since it has a lower cost of transmission. If all nodes have the same value of ϵ then again the amount of traffic a node receives will depend on other factors.

For the purposes of this test, all nodes in this network have the same value of ϵ. Perhaps for future research, a method could devised for calibrating and testing individual values of ϵ for each node. No two nodes are identical, so this will introduce different amounts of uncertainty to the calculations of μ. If, for example, a node occasionally comes under shadow, this requires a different value of ϵ than a node that sits at the highest point of the tallest tower in a city. Perhaps a starting point to determine individual values of ϵ would be to establish how much energy one source puts out relative to another. The more accuracy in this measurement, the better the calibration of ϵ.

6.4 FDR - Frame Delivery Ratio

6.4.1 Results

The Frame Delivery Ratio (FDR) is the ratio of frames (packets of information) that arrive at their desired destination. This number is always between zero and one, since more packets are not able to arrive at the destination than were sent to begin with.

This number consists of three parts per node. It is set up in the table below with the format x, y, z with each x, y, and z being the ratio of packets that were delivered to the other nodes. Taking node 1 for example, x would be the ratio between node 1 and node 2, y would be the ratio between node 1 and node 3, and z would be the ratio between node 1 and node 4. There are only 3 ratios because a node does not deliver messages to itself.

FDR Combination	FDR Result(Minutes of Uptime)
0.001, 0.001, 0.001	788
0.8, 0.001, 0.001	755
0.001, 0.8, 0.001	724
0.001, 0.001, 0.8	682
0.8, 0.8, 0.001	695
0.8, 0.001, 0.8	696
0.001, 0.8, 0.8	731
0.8, 0.8, 0.8(Control)	658

Table 2: Frame Delivery Ratio Results (Part 1)

6.4.2 Interpretation

The main algorithm introduced in [39] does not explicitly say anything about the FDR. The FDR is included in the $PRO(i, N_i, D)$ section. That term is defined by

$$PRO(i, N_i, D) = (Dist(i, D) - Dist(N_i, D)) \cdot FDR_{i,N_i} \cdot FDR_{N_i,i} \tag{6}$$

$PRO(i, N_i, D)$ is the progressive distance towards node N from node i over some distance D. $Dist(i, D)$ is the distance of node i from the base station at distance D, while $Dist(N_i, D)$ is the distance of node N from the base station. These distances are multiplied by the Frame Delivery Ratio of transmitting a message from node i to node N and then multiplied by the FDR of a message

0.8, 0.8, 0.8(Control)	658
0.999, 0.8, 0.8	774
0.8, 0.999, 0.8	767
0.8, 0.8, 0.999	739
0.999, 0.999, 0.8	772
0.999, 0.8, 0.999	801
0.8, 0.999, 0.999	775
0.999, 0.999, 0.999	830

Table 3: Frame Delivery Ratio Results (Part 2)

from node N to node i. What this accomplishes is a weighted scheme to balance how likely a message is to be received with how much closer to the base station the message gets upon a received transmission. It should be noted that in the equation there are two separate FDRs, FDR_{i,N_i} and $FDR_{N_i,i}$. For the tests, these ratios are given the same values, as it is unlikely that a message is more likely to be received in one direction than in the other direction.

In order to truly see how the FDR affects how long a network stays up and operational, it is important to look at the trends presented in the tables above. It is fairly easy to observe that the closer the FDR is to one, the longer the network lasts. The conclusion that comes out of that is the FDR of this network is quite good. The value of FDR was never assigned to one as it is impossible for all

messages to arrive, there will always be some dropped messages, however small that number may be. The network is tested in an area without much interference to the network, so most transmitted messages got through. Only a very small percentage of messages would be dropped, so 0.999 is a close approximation of the real value. An important distinction to note is that the values of the FDR were only changed in the code of the program, the FDR in the physical realm remained untouched.

If a method for determining the FDR with greater accuracy were devised, this would be advantageous to the network lifetime. That may be for future research, but is not done here. If the time is not taken to determine a value of the FDR, the assumption that almost all messages will be received is valid in most cases (barring areas with excessive network interference).

An FDR that is too small will generate a cost function that is slightly higher than one with an FDR that is too high. A high FDR node will be receiving a higher volume of traffic and will burn out faster. If a node has a lower FDR in comparison to the rest of the nodes, it will receive less traffic and will increase network uptime.

As one may have noticed from Tables 2 and 3, the sensitivity of the algorithm to the values of the FDR is not very significant. By taking values on opposite ends of the potential spectrum, the FDR does not exert a great amount of influence

over the performance of the algorithm. It is easy to observe that the closer to one the FDR is, the better the performance of the network, but the performance gains between 0.001 and 0.999 are not large enough to conclude that the FDR has a meaningful effect on the performance of the algorithm.

Another reason supporting the removal of the FDR from the algorithm would be that since the FDR is so difficult to measure to a high degree of accuracy this raises the potential for more harm than good to the network lifetime. One reason to keep the FDR in the algorithm would be if the FDR were drastically different across the network, but in a small scale environment with little electromagnetic interference, the FDR is not a particularly important part of the algorithm.

6.5 η - Scaling Constant

6.5.1 Results

The constant η is present in both the numerator and denominator part of the equation and in essence is to provide and accentuate the differences between how much charge is left in the battery of each node. As a battery runs down, η^{λ} reduces the value of the numerator and the value of the denominator remains constant, making the cost of transmission to a given node quite high.

In [39] the suggested value for this constant is 10000. A scattering of values for this constant was selected, some between zero and one, others above one. The

result was that 10000 was by far the best number to use for this constant as is shown in Figure 9.

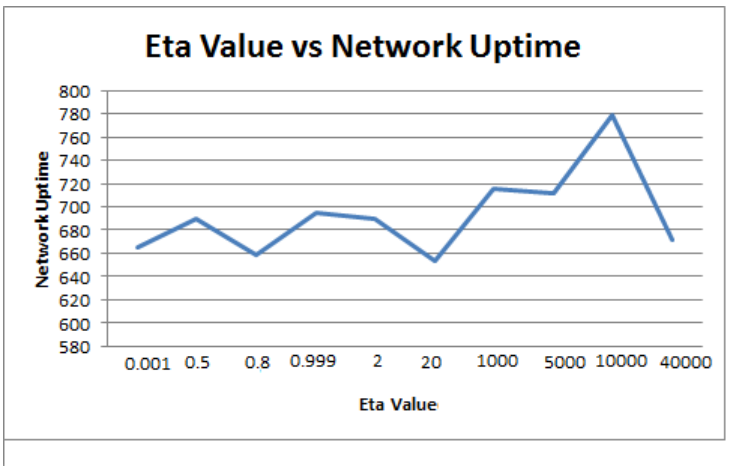

Figure 9: η Value vs. Network Uptime

6.5.2 Interpretation

This subsection will look at interpreting the data provided by changing the value of η. The purpose of η is to provide a scale for λ, something to measure and compare against. As the battery runs down, the scale for λ can not run down to zero though as the battery runs down. Using η and λ as an exponential gives the result of running the value of η^{λ} down to one. With the logarithm of η always being a constant value, as λ runs down, the cost of transmitting to a node goes up and up.

The values of η between zero and one provide an interesting bit of data. They show that the algorithm is still able to function, although not at peak efficiency.

The problem with a value between zero and one is that as a battery loses power, the cost of transmitting to that battery will decrease. This will result in a battery being depleted much faster than if the network were to be transmitting to a node with more battery charge.

It is worth noting that there is a local maximum as the value of η approaches one. With the algorithm in the current form, it would not be able to function if η were given the value of one. This would make the denominator zero and thus every time the equation is used it would return a value of infinity. The value of η can approach one though, and it produces the result of minimizing the effect of battery charge and relying more on the amount of solar energy being harvested. This goes to show that the algorithm is able to function in a reasonably efficient manner without needing to account for battery charge.

When the value of η is assigned a value in an appropriate range, it is then able to use the solar energy harvesting rate as well as the battery charge to operate at a higher level of efficiency. When the value of η is down at a lower level like two or 20, the effects it is able to exert over the algorithm are minimized. The higher the value of η the effects of λ become much more noticeable. This comes with a caveat though, the value of η needs to be able to be balanced with the other parts of the algorithm. If η gets to be too high, the only part of the algorithm that has any effect would be η and λ. A value in the tens of thousands gives an appropriate

balance.

7 Sensitivity - Analytic Approach

This section will look at the sensitivity of the algorithm to errors that are introduced and make recommendations for future use. To look at the errors present in the algorithm, Taylor series are used to expand the cost equation, with δ as the error term. The general form of the Taylor series used is $\tilde{C}_M(N_i, D) = C_M(N_i, D) + \sum_{i=1}^{\infty} f_i(\delta^i)$ where $\tilde{C}_M(N_i, D)$ is the cost when there is an error in a given parameter of δ. The format will always be truncated to the original cost equation, plus the first two terms in the sum (coefficients of δ and δ^2).

7.1 μ - Energy Harvesting Rate

For the first case we examine, we do not use Taylor series, but look at the error directly. There, the error of δ in μ is an error of:

$$\frac{(E_b(N_i)\cdot \eta^{\lambda_{N_i}})}{\log \eta\cdot (\mu_{N_i} + \delta + \epsilon)\cdot PRO(i, N_i, D)} - \frac{(E_b(N_i)\cdot \eta^{\lambda_{N_i}})}{\log \eta\cdot (\mu_{N_i} + \epsilon)\cdot PRO(i, N_i, D)}, \tag{7}$$

This equation simplifies to

$$\frac{-(E_b(N_i)\cdot \eta^{\lambda_{N_i}})\delta}{(\mu + \delta + \epsilon)(\mu + \epsilon)}. \tag{8}$$

If the value of δ is small, then we have

$$f(\delta) \approx \frac{-(E_b(N_i)\cdot \eta^{\lambda_{N_i}})\delta}{(\mu + \epsilon)^2}. \tag{9}$$

The linear dependence on δ makes it easy to see that as the value of δ increases the actual cost of transmission will be increasingly skewed in a negative fashion. This means that as δ increases, the cost of transmitting to nodes is less than to other nodes with a more accurate representation. If the value of δ is consistent across all nodes, the error will be relatively unimportant. If it is a single node, this node becomes more likely to be selected and as a result, network lifetime may be (significantly) reduced.

A second method for examining the error in the algorithm is done by using a Taylor series expansion. The first two terms of the Taylor series of μ are given below:

$$\begin{aligned} C_M(N_i, D) + f_1(\delta) + f_2(\delta^2) &= \frac{(E_b(N_i) * \eta^{\lambda_{N_i}})}{\log \eta * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} \\ &\quad - \frac{E_b(N_i) * \eta^{\lambda_{N_i}} * \delta}{\log \eta * (\mu_{N_i} + \epsilon)^2 * PRO(i, N_i, D)} \\ &\quad + \frac{E_b(N_i) * \eta^{\lambda_{N_i}} * \delta^2}{\log \eta * (\mu_{N_i} + \epsilon)^3 * PRO(i, N_i, D)}. \end{aligned} \tag{10}$$

For our experiments, the values of μ were between 0 and 1024. We took δ values between -500 and 500 and subtracted the value of $C_M(N_i, D)$ when δ=0 to see how much error would be generated by an inaccurate value of μ. The results are presented in Figure 10.

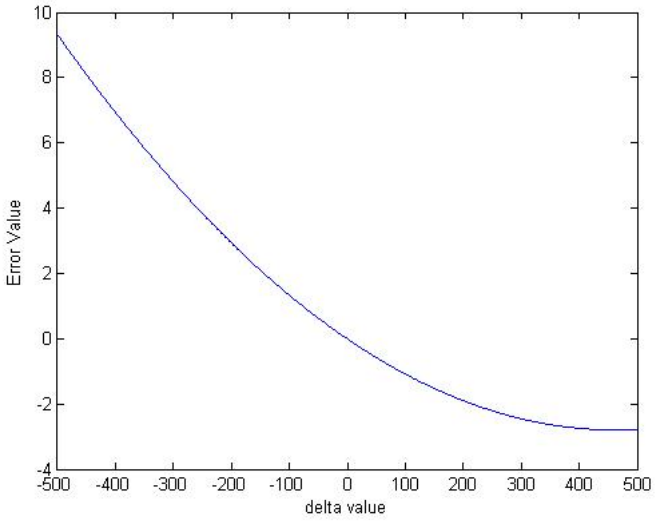

Figure 10: Error values of μ

As one can see, the cost error has a parabolic shape. The slope for a δ value of -1 is -0.121 and +1 is -0.121. This means that if the value of μ is off by a small amount, the effect of over and under estimation on the algorithm will be roughly the same. However, from observing the graph, the slope of δ starts to flatten out to the right of 0 and gets steeper the left. This leads to the conclusion that if μ is not perfect and δ is a larger value, it is better to overestimate μ than underestimate, as this will skew the data by a smaller amount.

Due to the structure of the cost equation, the sensitivity to ϵ is identical to that of μ. However, errors in ϵ are much easier to relate back to our experiments. Using different values of ϵ in our experiments resulted in an inverted parabola. If the ideal case were a flat line, and the cost of the errors (presented in Figure 10)

were subtracted from it, the resulting parabola would look something like Figure 8.

7.2 λ - Percentage of Remaining Battery Life

The Taylor series for λ is presented below:

$$\begin{aligned} C_M(N_i, D) + f_1(\delta) + f_2(\delta^2) &= \frac{(E_b(N_i) * \eta^{\lambda_{N_i}})}{\log \eta * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} \\ &+ \frac{E_b(N_i) * \delta * \eta^{\lambda_{N_i}}}{(\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} \\ &+ \frac{(E_b(N_i) * \delta^2 * \eta^{\lambda_{N_i}}) * \log \eta}{2 * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)}. \end{aligned} \tag{11}$$

The graph of the error function is presented in Figure 11.

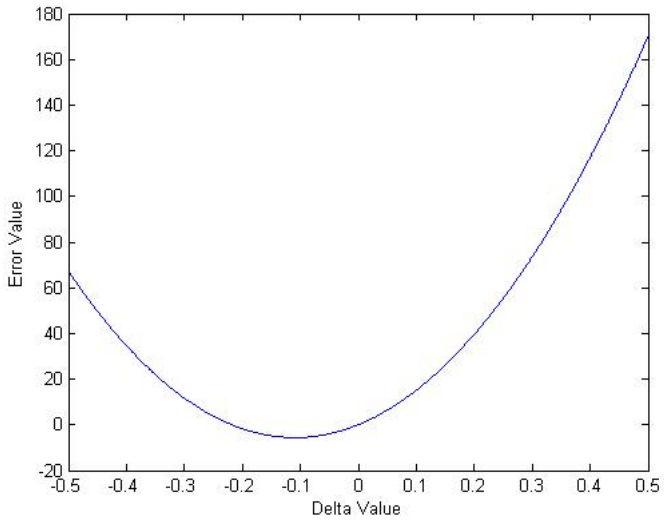

Figure 11: λ error function

As one can see in the scale of the error of λ it is very important to get the value of λ as close to the actual value as possible. The sensitivity of the algorithm to this value is quite high. Taking the derivative of the above graph tells us how fast the error is changing. For a δ value of -0.001, the error is changing at 0.1026 and a δ value of 0.001 is changing at 0.1035 and going beyond that increasing. This leads to the conclusion that it is better to underestimate the value of λ than to overestimate. Underestimating the value of λ leads to smaller values of $f(\delta)$ for small values of δ.

Based on the graph in Figure 6, our experimental results indicate that our conclusion in the previous chapter is not the case. Our experimental results show that it is better to overestimate the value of λ than underestimate it. Currently we have no explanation for this, and this should be explored more.

7.3 η - Scaling Constant

It should be noted that η is a chosen constant, not a measured or calculated value, so it does not have an error function. The Taylor series of η will look at the sensitivity of the GREES algorithm to the effects of η. The Taylor series of η is

very complicated, due to it appearing in two places in the cost function:

$$C_M(N_i, D) + f_1(\delta) + f_2(\delta^2) = \frac{(E_b(N_i) * \eta^{\lambda_{N_i}})}{\log \eta * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} - \frac{E_b(N_i) * \delta * \ln 10 * (\frac{e^{\lambda_{N_i} * \ln \eta}}{\eta * \ln \eta^2} - \frac{\lambda_{N_i} * e^{\lambda_{N_i} * \ln \eta}}{\eta * \ln \eta})}{(\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} - \frac{E_b(N_i) * \delta^2 * \ln 10 * \left(\frac{e^{\lambda_{N_i} * \ln \eta} * (\frac{\lambda_{N_i}}{2 * \eta^2} - \frac{\lambda_{N_i}^2}{2 * \eta^2})}{\ln \eta - \frac{(e^{\lambda_{N_i} * \ln \eta} * (\frac{1}{2 * \eta^2 * \ln \eta} + \frac{1}{\eta^2 * \ln \eta^2}))}{\ln \eta} + \frac{\lambda_{N_i} * e^{\lambda_{N_i} * \ln \eta}}{\eta^2 * \ln \eta^2}} \right)}{(\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} \quad (12)$$

Figure 12 shows the behaviour of the equation above.

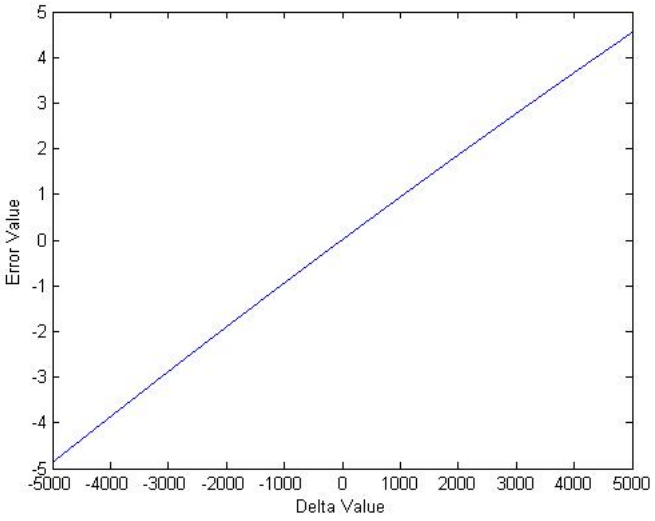

Figure 12: η Error Function

Although this graph appears to be almost linear, it is actually a small part of a parabolic arc. From this graph, one could conclude that the algorithm is not sensitive to different values of η. On a small scale, missing the actual value of η returns a very small value of $f(\delta)$. To the left of 0 gives a value of $9.4164 * 10^{-4}$

and to the right gives a value of $9.4163 * 10^{-4}$. These values are so small as to be negligible. Going to a larger scale, missing the value of η by 1000 returns an $f(\delta)$ of under 1. From this it is easy to conclude that the algorithm is not sensitive to errors in η.

Looking at η from our experiments, it is easy to see the effect on the lifetime of the network looking at different values. Choosing an η value that is very low and slowly moving to greater values mirrors the graph to an extent. The graph also very slowly moves in either direction as the difference between the ideal value and chosen values changes.

7.4 $PRO(i, N_i, D)$ - Progressive Distance Towards Base Station

The Taylor series of $PRO(i, N_i, D)$ is presented below:

$$\begin{aligned} C_M(N_i, D) + f_1(\delta) + f_2(\delta^2) &= \frac{(E_b(N_i) * \eta^{\lambda_{N_i}})}{\log \eta * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)} \\ &- \frac{E_b(N_i) * \eta^{\lambda_{N_i}} * \delta}{\log \eta * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)^2} \\ &+ \frac{E_b(N_i) * \eta^{\lambda_{N_i}} * \delta^2}{\log \eta * (\mu_{N_i} + \epsilon) * PRO(i, N_i, D)^3} \end{aligned} \quad (13)$$

The value of $PRO(i, N_i, D)$ presented a unique problem. The value of $PRO(i, N_i, D)$ can be either above OR below one. As a result, values both above and below one had to be taken into account, with different results for each. A $PRO(i, N_i, D)$ of below one is presented in Figure 13. If a message is to be transmitted over a

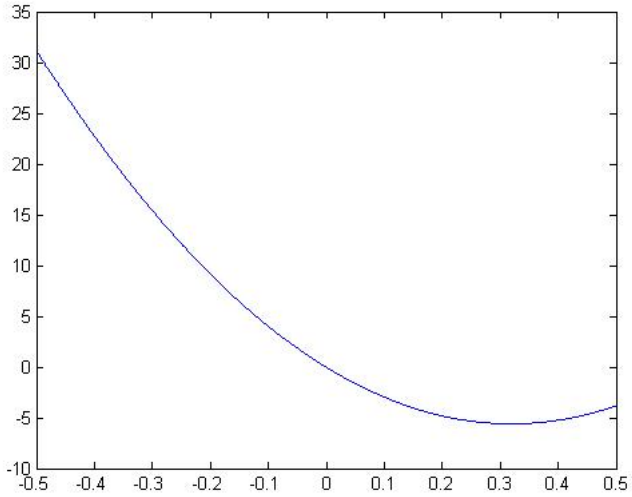

Figure 13: $PRO(i, N_i, D)$ of Less Than One

short distance, $PRO(i, N_i, D)$ is most likely going to be less than one. The error of $PRO(i, N_i, D)$ behaves slightly differently if $PRO(i, N_i, D)$ is above or below one. It makes sense that a node would have different error based on how far the message has to be transmitted, due to the fact that the benefits of transmitting over a short distance and using energy from more nodes needs to be weighed very carefully against transmitting longer distances, using more energy per transmission, and arriving at the base station in fewer steps.

When $PRO(i, N_i, D)$ is less than one, the error has a much greater effect than when $PRO(i, N_i, D)$ is above one. If the δ value of $PRO(i, N_i, D)$ is -0.001, the slope of the error is -0.0351 and if the δ value is 0.001 the slope is -0.0349. These error values may not seem large, but they are approximately 35 times larger than

the δ value. Based off of the graph, it is possible to conclude that the algorithm is sensitive to errors if $PRO(i, N_i, D)$ is less than one. If a value of $PRO(i, N_i, D)$ is not possible to determine entirely accurately, it would be wise to overestimate, but by as little as possible.

The values of $f(\delta)$ are less affected when the value of $PRO(i, N_i, D)$ is greater than one. The graph of $f(\delta)$ is presented in Figure 14: If a message has farther

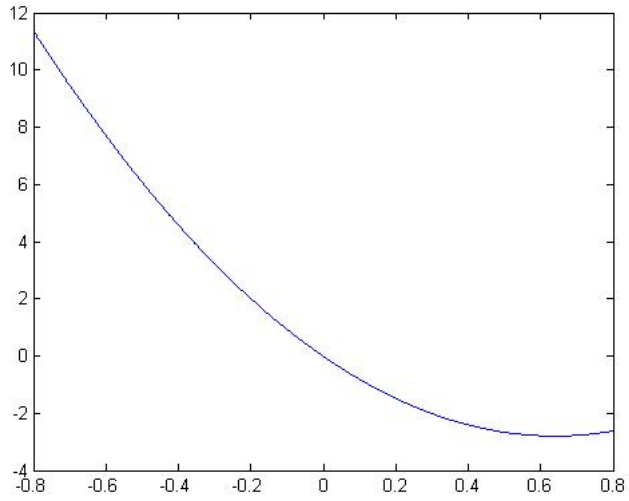

Figure 14: $PRO(i, N_i, D)$ of Greater Than One

to go, it is less important to get an accurate value of $PRO(i, N_i, D)$. When the δ value is just less than 0 the $f(\delta)$ value is -0.0088, and when δ is just over 0 the $f(\delta)$ value is -0.0087. The scale of the error when $PRO(i, N_i, D)$ is greater than one is much smaller than when $PRO(i, N_i, D)$ is less than one. From the graph above, if the value of $PRO(i, N_i, D)$ is greater than one, it is better to overestimate, but

inaccuracies have less effect than if $PRO(i, N_i, D)$ is less than one.

These graphs tell the story of the values of $PRO(i, N_i, D)$, if they were all relative to each other. If the absolute error were to be examined, they would be significantly larger, the difference between the relative errors would have to be taken into account. Relating $PRO(i, N_i, D)$ back to our experiments (comparing them to FDR since FDR is the only changing component of $PRO(i, N_i, D)$), the cross-over point between a $PRO(i, N_i, D)$ of less than one and greater than one would be at the control. For values of $PRO(i, N_i, D)$ greater than one the graph behaves predictably, as the error decreases the lifetime of the network increases (for reference see Table 3). We have no explanation for $PRO(i, N_i, D)$ less than one performance increasing as FDR values approach zero.

8 Conclusion and Further Research

8.1 Further Research

The work in this thesis could be extended in a few ways. Firstly, a method for developing individual values for ϵ would be useful in extending the network lifetime. This thesis used only one value of ϵ across all of the nodes in the network, but in a real world application, this should not be the case. There would be different values for ϵ based on various factors, including the position of the node, and anything in the area that might affect how much energy a given node is able to harvest (trees casting shadows over the node for example). In addition, not all energy harvesting sources put out consistent amounts of energy. A good starting point would be to identify the inconsistencies in the energy harvesting apparatus and to quantify them.

A second area of further research would be to develop the Frame Delivery Ratio (FDR) a bit more. Currently it is a number that is the same across all nodes. In a real world environment, some nodes would be more likely to receive a message than others due to various sources of noise, or distance between nodes, etc., and it would be good for an FDR to reflect that, as opposed to applying a blanket value for all nodes. Perhaps a starting point for this would be to compare the number of messages transmitted to a given node versus the number of messages received, and then repeating vice versa. This test would have to be repeated until a definite

value for the FDR has been developed for all of the network.

8.2 Conclusion

The purpose of this thesis is to look at the Geographic Routing with Environmental Energy Supply (GRESS) protocol for sensor networks developed by Zeng et al. [39]. We looked at the various parameters of the algorithm and the sensitivity of the algorithm to each of the parameters. We have shown that the most important aspect of the protocol is the energy harvesting rate.

As we have shown through our experiments, the algorithm is sensitive to some parameters and not sensitive to others. The algorithm is sensitive to energy harvesting and responds quite favourably to it. The algorithm is also quite sensitive to errors in $PRO(i, N_i, D)$ if $PRO(i, N_i, D)$ is less than one. The algorithm is also sensitive to values of λ, which has the potential to skew the lifetime of the network.

The network is not very sensitive to different values of η, picking a value that is over 1000 away from optimal results in a cost increase of less than one. The network is also not very sensitive to errors of $PRO(i, N_i, D)$ if $PRO(i, N_i, D)$ is greater than one.

References

[1] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–6. IEEE, 2007.

[2] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537 – 568, 2009.

[3] Arduino. Arduino XBee Shield. `http://www.arduino.cc/en/Main/ArduinoXbeeShield`, May 2011.

[4] K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA-based MAC for sensor networks. *System-level power optimization for wireless multimedia communication*, pages 21–40, 2002.

[5] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor network topologies. *Energy*, 5:6, 2004.

[6] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8:481–494, 2002.

[7] D. Chu, A. Deshpande, J.M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *ICDE'06. Proceedings of the 22nd International Conference on Data Engineering*, pages 48–48. IEEE, 2006.

[8] Marco Conti, Andrea Passarella, and Luciana Pelusi. Mobile-relay forwarding in opportunistic networks. In *Chapter in Adaptive Techniques in Wireless Networks (M. Ibnkahla, Editor)*. CRC Press, 2008.

[9] C. S. Raghavendra G. Lu, B. Krishnamachari. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *18th International Parallel and Distributed Processing Symposium, 2004.*, page 224. IEEE Press, 2004.

[10] B. Gedik, L. Liu, and P.S. Yu. ASAP: an adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(12):1766–1783, 2007.

[11] Arvind Giridhar and P. R. Kumar. Maximizing the functional lifetime of sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[12] P. Brighten Godfrey and David Ratajczak. Naps: scalable, robust topology management in wireless ad hoc networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, pages 443–451, New York, NY, USA, 2004. ACM.

[13] Himanshu Gupta, Samir R. Das, and Quinyi Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, *MobiHoc '03*, pages 189–200, New York, NY, USA, 2003. ACM.

[14] Jun H., , Ammar M., and Zegura E. Power management in delay tolerant networks: A framework and knowledge-based mechanisms. In *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*. Citeseer, 2005.

[15] J.C. Haartsen. The bluetooth radio system. *Personal Communications, IEEE*, 7(1):28–36, 2000.

[16] A. Jain and E.Y. Chang. Adaptive sampling for sensor networks. In *Proceeedings of the 1st International Workshop on Data management for Sensor Networks: in conjunction with VLDB 2004*, pages 10–16. ACM, 2004.

[17] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. *IEEE 24th International Conference on Data*

Engineering, 2008.

[18] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, 6:32–es, September 2007.

[19] T. Kijewski-Correa, M. Haenggi, and P. Antsaklis. Wireless sensor networks for structural health monitoring: a multi-scale approach. In *ASCE Structures 2006 Congress*. Citeseer, 2006.

[20] Y.A. Le Borgne, S. Santini, and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing*, 87(12):3010–3020, 2007.

[21] Libelium. Squidbee main page. `http://www.libelium.com/squidbee/index.php?title=Main_Page`, May 2011.

[22] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[23] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[24] P. Padhy, R.K. Dash, K. Martinez, and N.R. Jennings. A utility-based sensing and communication model for a glacial sensor network. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1353–1360. ACM, 2006.

[25] V. Paruchuri, S. Basavaraju, A. Durresi, R. Kannan, and S.S. Iyengar. Random asynchronous wakeup protocol for sensor networks. *First International Conference on Broadband Networks*, 2004.

[26] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[27] Han Qi, Sharad Mehrotra, and Nalini Venkatasubramanian. Energy efficient data collection in distributed sensor environments. *International Conference on Distributed Computing Systems*, 0:590–597, 2004.

[28] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, pages 70–80, 2002.

[29] Y.C. Tseng, C.S. Hsu, and T.Y. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *INFOCOM 2002. Twenty-First*

Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 1, pages 200–209. IEEE, 2002.

[30] Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh. iMouse: an integrated mobile surveillance and wireless sensor system. *Computer*, 40(6):60–66, 2007.

[31] D. Tulone and S. Madden. An energy-efficient querying framework in sensor networks for detecting node similarities. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 191–300. ACM, 2006.

[32] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, *SenSys '03*, pages 171–180, New York, NY, USA, 2003. ACM.

[33] M.C. Vuran and I.F. Akyildiz. Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 14(2):316–329, 2006.

[34] Guoliang Xing, Xiaorui Wang, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks*, 1:36–72, August 2005.

[35] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 70–84. ACM, 2001.

[36] X. Yang and N.H. Vaidya. A wakeup scheme for sensor networks: achieving balance between energy saving and end-to-end delay. In *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, pages 19–26. IEEE, 2004.

[37] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2004.

[38] J. Heidemann Yuan Li, Wei Ye. Energy and latency control in low duty cycle MAC protocols. In *Wireless Communications and Networking Conference*, pages 676 – 682. IEEE Press, 2005.

[39] Kai Zeng, Kui Ren, Wenjing Lou, and Patrick J. Moran. Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply. *Wireless Networks*, 15:39–51, January 2009.

A Software Code

```
//initialize vars

int id = 1;

int solar[4] = {0, 0, 0, 0,};

int batt[4] = {0, 0, 0, 0,};

int msgs[4] = {id, 0, 0, 0};

int msgr[4] = {0, 0, 0, 0};

int delayms = 20;

int count = 0;

int tmp, j = 0;

float FDRiN[4] = {0.001, 0.001, 0.001};

float FDRNi[4] = {0.001, 0.001, 0.001};

float gnu = 40000, eta = 20;

void setup()

{

  // initialize serial communication:

  Serial.begin(19200);

}


//command to send message
```

```
void sendmsg () {
  int n;
  for (n=0; n<4; n++) {
    Serial.write(msgs[n]);
    delay(25);
  }
  count++;
  Serial.println(msgs[0]);
}

//determine who to send message to
void sendto() {
  //initialize local vars
  int i;
  float lamda[3], PRO[3], cost [3];

  //find lamda
  lamda [0] = (8.4-batt[1])/8.4;
  lamda [1] = (8.4-batt[2])/8.4;
  lamda [2] = (8.4-batt[3])/8.4;
```

```
  //calculate who to send message to by
  //calculating cost of transmission
  for (i = 1; i < 4; i++) {
    PRO[i-1] = (4-i)*FDRiN[i-1]*FDRNi[i-1];
  }
  for (i = 1; i < 4; i++) {
    cost[i-1] = 8.4*pow(gnu, lamda[i-1])/(log10(gnu)*(solar[i]+eta)*PRO[
  }
  if (cost[0] < cost[1] && cost[0] < cost [2]){
    msgs[0] = id*10+2;
  }
  else if (cost[1] < cost[0] && cost[1] < cost[2]) {
    msgs[0] = id*10 + 3;
  }
  else {
    msgs[0] = id*10 + 4;
  }
}
```

```
//receive message, determine who it came from
void msgrec () {
  int n, temp = 5;

  for (n=0; n<=3; n++){
    if (Serial.available() >0){
      msgr[n] = Serial.read();
      temp = msgr[0];
    }
  }

  while (temp >= 10) {
    while (temp % 10 != 0){
      temp--;
    }
    temp = temp/10;
  }

  switch(temp){
    case 2:
```

```
        solar[1] = msgr[2];
        batt[1] = msgr[3];
      break;
      case 3:
        solar[2] = msgr[2];
        batt[2] = msgr[3];
      break;
      case 4:
        solar[3] = msgr[2];
        batt[3] = msgr[3];
      break;
      default:
      break;
    }
}

void loop()
{
  //read analog values
  if (count<3000)
```

```
    batt[0]=8.4;
  else
    batt[0] = (1/2^(count/100))*8.4;

  solar[0] = analogRead(1);
  delay(10);
  tmp = analogRead(2);
  delay(10);

  //construct message to send
  msgs[1] = tmp;
  msgs[2] = solar[0];
  msgs[3] = batt[0];

  //ensure board listens 4x more than it sends
  j++;
  if (j%4 == 0) {

    //find who to send to & send message
    sendto();
```

```
      sendmsg();
      delay(1000);
      j = 0;
    }
    //update values
    msgrec();
}
```

B Specifications

//SquidBee

SquidBee, the first open mote in the world.

Developed by Libelium

Notice:

- This document is just a **summary**, complete information, sources, schematics and examples can be downloaded from the URL's bellow.

URLs:

- SquidBee Project Wiki: http://www.squidbee.org
- Getting SquidBee: http://squidbee.libelium.com

Features:

- Arduino + XBee based module.
- Open source mote
- 9V battery power
- Easy programming (Arduino)
- 12 digital pin I/O
- 6 analog input pin
- 5 PWM analog output pin
- USB connection to PC (windows, linux and mac compatible)
- Wireless communication, XBee module based (ZigBee)
- Sensors
  - Temperature
  - Humidity
  - Lightness
  - Possibility to add news
- Networking topology Peer-to-peer, point-to-point, point-to-multipoint and mesh
- Addressing up to 65000 motes
- Module size 70 x 60 x 30 mm
- Enclosure size 120 x 65 x 40 mm

Technical specifications

Control module – Arduino

Processor speed	16 MHz
Flash (program) size	ATmega8 - 8kb ATmega168 – 16 kb
EEPROM	512 bytes
SRAM	1kb
Digital I / O	12
Analog I/O	6
Operating voltages	5 V - 12V

- More information www.arduino.cc

Wireless communication module

Power output	1 mW (up to 100 m) XBee 100 mW (up to 1000 m) XBee pro
RF data rate	250 kbps
Operating frequency	2.4 GHz
Operating voltages	5 V - 12V
Networking topology	Peer-to-peer, point-to-point, point-to-multipoint and mesh
Channel capacity	16 Direct Sequence Channels (software selectable)
Addressing	65,000 network addresses available for each channel

- More information www.maxstream.net

Sensors

- Temperature
- Humidity
- Light

Other sensors

Basic connection of analog sensor.

Resistive sensor → A resistor that changes its value with the magnitude to sense.

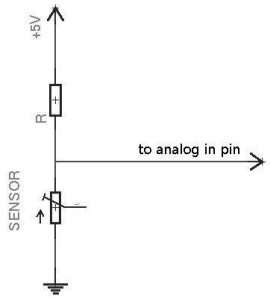

Fig 1. Connection for resistive sensor.

We call R pull-up resistor and it usually is 1 kΩ or 10 kΩ.

Analog sensor → Sensor witch output is an analog value, what means that it can get values from 0V to 5 V

We usually don't need pull-up resistor for this one and connect it directly to an analog in pin.

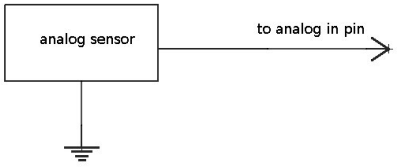

Fig 2. Connection for analog sensor without pull-up resistor.

Basic connection of digital sensor.

Digital sensor → Sensor witch output is a digital value 0 (0 V) or 1 (5 V).

If the sensor needs pull-up or pull-down resistor we connect a 1 kΩ or 10 kΩ resistor between 5 V (for pull-up) or 0 V (for pull-down).

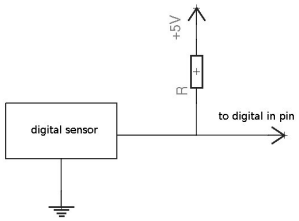

Fig 3. Connection for digital sensor with pull-up resistor.

If the sensor doesn't need pull-up or pull-down resistor we connect it directly to a digital in pin.

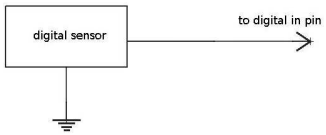

Fig 4. Connection for digital sensor.

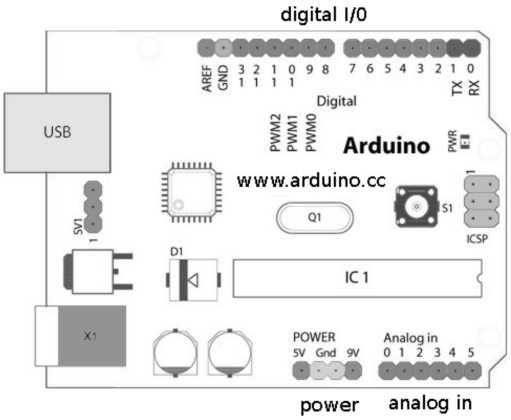

Fig 5. Arduino board.

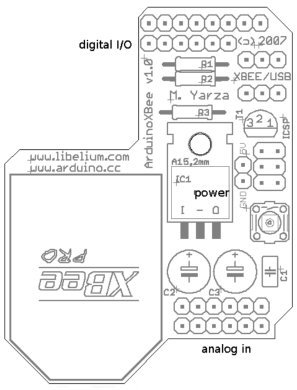

Fig 6. XBee shield.

To get the circuit schematics go to the Dowload section in: http://www.squidbee.org