LOAD BALANCED AND ENERGY AWARE ROUTING FOR LARGE SCALE WIRELESS SENSOR NETWORKS

by

Siddharth H Kamath

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering.

Charlotte

2011

Approved by:

_____
Dr. Asis Nasipuri

_____
Dr. James Conrad

_____
Dr. Robert Cox

ABSTRACT

SIDDHARTH HARISH KAMATH. Load balanced and energy aware routing in large scale wireless sensor networks. (Under direction of DR.ASIS NASIPURI)

Wireless Sensor Networks (WSNs) are battery powered, and minimizing energy consumption without compromising the data quality is imperative for maximizing its lifetime. Traditional routing protocols are seen to suffer from a load imbalance problem that is caused by concentration of data traffic only on a few critical nodes, which is solely decided by link quality considerations. As a result, some critical nodes may deplete their batteries faster and stop working much earlier than desired, which ultimately compromises the network lifetime. This work involves design of a load balanced and energy aware routing algorithm which tries to distribute data traffic over all the critical nodes using real time traffic data as well as voltage based run-time energy monitoring to modify the routing metric. This will ensure that sensor nodes will choose routes with higher energy and lower data traffic which would balance the energy consumption in the nodes and improve the network lifetime. Laboratory results and simulations indicate significant increase in network lifetime in comparison to the traditional tree routing protocol.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Asis Nasipuri for his invaluable guidance and relentless support and motivation throughout my thesis. His insight and expertise along with continued encouragement were major factors without which this research work would not have been possible. It has indeed been a privilege to be able to work under his guidance and supervision.

I would also like to thank Dr. Robert Cox and Dr. James Conrad for being guiding members of my thesis committee.

I would like to thank my housemates Yogendra, Syed, Daniel and Amar for their invaluable support and encouragement. I would also like to extend a special thanks to all my colleagues at the coffee shop for covering my shifts when I was busy with my research.

Most importantly, I want to thank my parents, grandparents and my brother for their unconditional love and blessings which made all this possible. I am deeply indebted and it is to them I dedicate my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

A Wireless Sensor Network (WSN) consists of a large number of low cost, low power embedded devices with sensing, computing and wireless communication capabilities linked together to perform distributed tasks. These wireless devices (nodes) are equipped with an integrated low power processor, memory and a radio and usually operate on batteries. Wireless sensor networks are becoming increasingly popular than their wired counterparts due to their capabilities of programmability, ease of deployment and distributed processing. Sensor nodes support various sensing platforms such as on-board temperature, pressure, ambient light and vibration sensors. Sensor nodes can be programmed to work with different networking topologies like peer-to-peer, star and mesh topology. Nodes in a WSN can be programmed to configure themselves into a mesh network and communicate with the data sink using multi-hop transmissions over other sensor nodes. WSNs can provide monitoring variables from areas which are intractable to humans or which require continuous monitoring.

There are many applications of wireless sensor networks ranging from military applications which involve control, communications, computing, surveillance and targeting system; health care applications which involve deploying nodes to monitor patients and assist disabled patients; and also commercial applications which include managing inventory, monitoring product quality and monitoring disaster areas like volcanoes [26,27,28].

Since sensor nodes operate on batteries, energy consumption is a key concern for their operation. A significant amount of power is consumed by the radio for wireless communication and by the sensor boards for sensing physical quantities. Further, sensor nodes are expected to have a lifetime on the order of months to years, since battery replacement is not an option for networks with a large number of physically embedded nodes. Energy optimization in case of sensor networks is much more complex since it involves not only reducing energy consumption of a single sensor node but also maximizing lifetime of an entire network. A lot of research is involved towards developing low-power routing techniques and energy efficient hardware architectures [16, 17, 18].

In this work we focus on an application that was designed to monitor the health of high-voltage equipment in an electric power substation using a wireless mesh sensor network. Design and development of this project was undertaken as a joint collaboration between the Electric Power Research Institute (EPRI) and researchers at the University of North Carolina at Charlotte (UNCC). This collaboration led to the development of a 122 node WSN that was deployed in Tennessee Valley Authority's (TVA) Paradise Substation in Kentucky (Figure 1.1). The project involved design of applications for a sensor mesh network and their deployment, to monitor substation equipment like transformers, circuit breakers and transformer bushings. This thesis work involves a study of the design issues encountered by traditional mesh networking protocols, which are energy consumption and load imbalance usually seen in such applications and focuses on improving the network lifetimes by designing a load balanced and energy aware routing protocol.

1.1 Overview of the Paradise Substation Monitoring Project

An electric power substation deals with generation, transmission and distribution of power, which consists of high voltage electrical equipment ranging from transformers, circuit breakers along with switching, protection and control equipment. These electrical equipments are often faced with an ageing infrastructure, risk of blackouts or brownouts and other potential problems that can result in outages for large service areas and can cost millions of dollars in lost revenues and damages. To ensure system reliability, continuous monitoring of the health of substation equipment is a critical for its maintenance.

The ability to foresee a problem, or at least identify the existence of a condition that may result in system failure is highly desirable [24]. Traditionally, planned maintenance philosophy has been followed without any regard to actual in-service duty or the condition and performance of the equipment. However, there has been a growing tendency towards predictive maintenance techniques. The collaborative project involving the University of North Carolina at Charlotte and EPRI was targeted to meet this objective by developing a wireless mesh sensor network (WMSN) for early detection of equipment failures.

1.1.1 Substation Monitoring Tasks and Objectives

A substation consists of heavy electrical equipment like transformers, circuit breakers, transfer switches etc. A substation electrical equipment is observed to have some physical component (temperature, pressure etc.) associated with it that can indicate its normal operation and health. Some important equipment and their physical components that require regular monitoring are described as follows:

- Transformers: Transformers in an electric power substation, step-up the generator

voltage to a higher level before transmission to utility customers. Transformers at the Paradise substation are oil-filled where the core and coil assembly are placed in a tank filled with high-dielectric cooling oil. These transformers may fail due to high mechanical and thermal stresses induced by high voltages. Abnormal rise in temperature is a good indicator of potential transformer failures. Continuously monitoring transformer surface temperatures can be used to detect anomalous conditions. These temperatures can be recorded using resistance temperature detector (RTD) sensors.

- Circuit breaker: A circuit breaker is used as a protection device to interrupt fault currents and switch loads. Oil-filled circuit breaker surface temperatures are continuously monitored, and the relative tank temperature differences are used to indicate fault conditions. Circuit breakers filled with SF6 gas for cooling also need to be monitored for gas leaks and so, periodic readings of SF6 gas density at circuit breakers must be made available.
- Transformer bushings: Transformer bushing is an insulator to allow connection of transformers to external circuit. Transformer bushings degrade over time. An indication of bushing degradation can be obtained by detecting phase differences between voltages obtained at bushing taps.

Figure 1.2 is a satellite view of TVA's Paradise substation that also shows deployment of sensor nodes on the substation equipment.

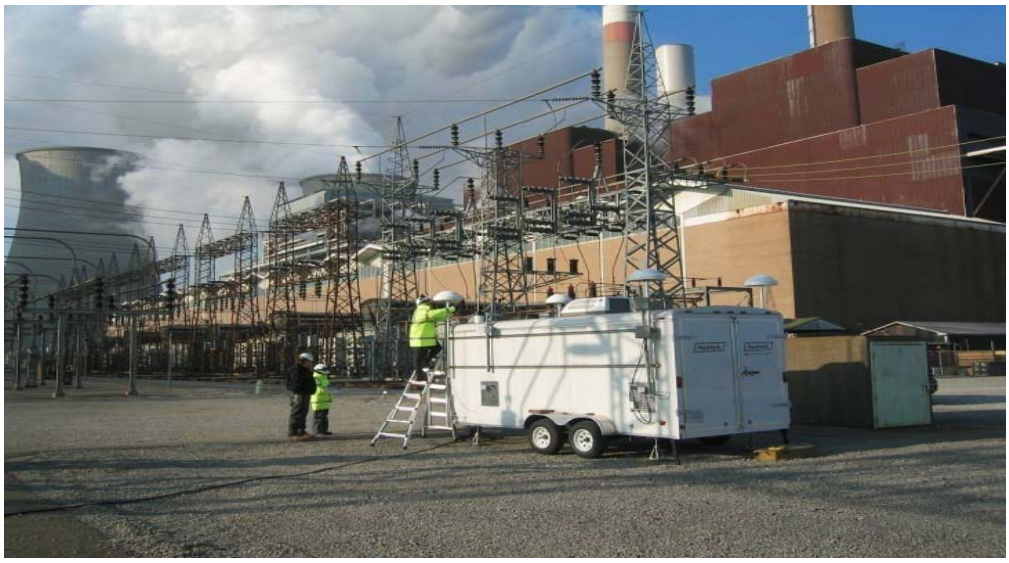


Figure 1.1: TVA's Paradise Substation at Kentucky

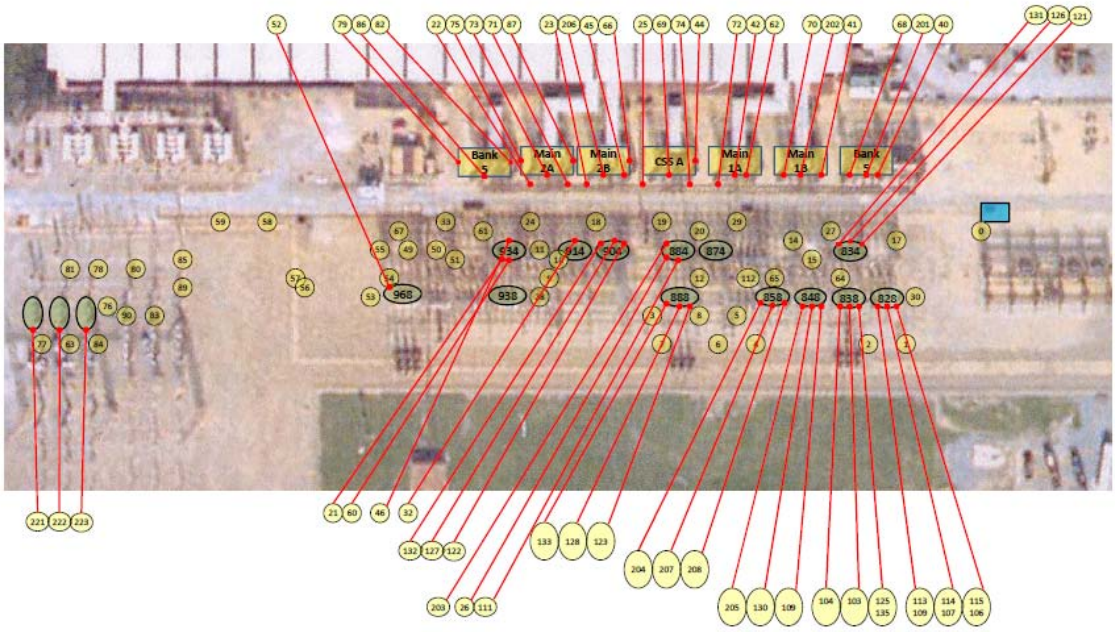


Figure 1.2: Satellite view of TVA's Paradise Substation showing deployment location of sensor nodes. The blue rectangle on the right is the base station.

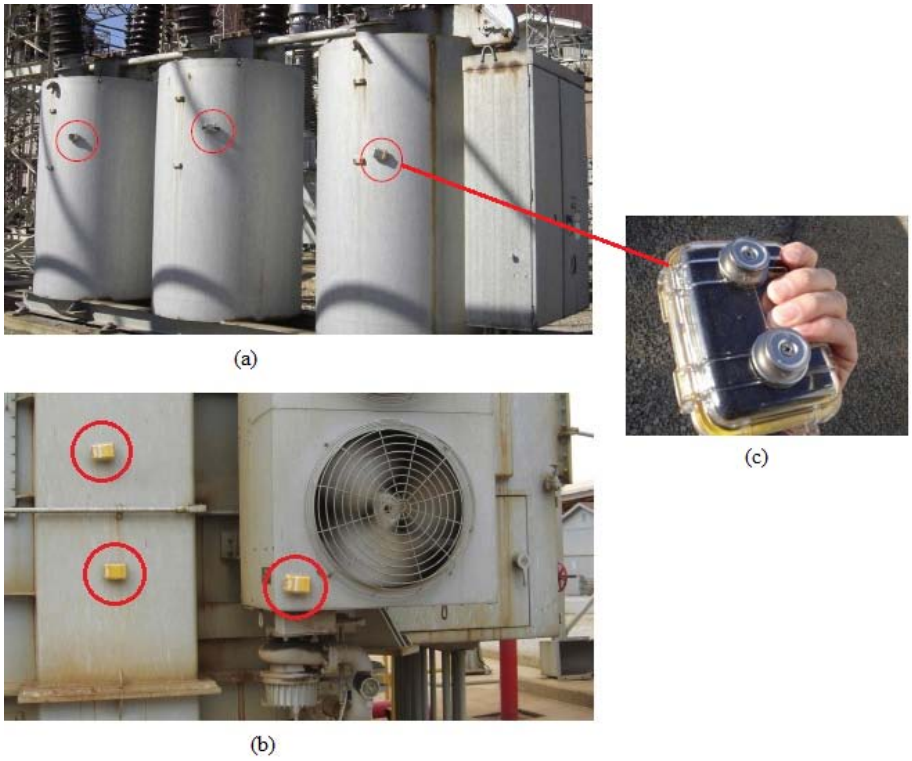


Figure 1.3: Sensor Nodes deployed at TVA's Paradise Substation on electrical equipment (a) Transformers and (b) Circuit-breakers (c) Enlarged view of the node

Battery operated sensor nodes programmed to perform periodic sensing tasks like surface temperature monitoring, vibration sensing, gas density monitoring and ambient light sensing according to the monitoring tasks are deployed on the electrical equipment. Node deployment was done in phases towards an eventual target of 150 nodes. Presently, the Paradise network consists of 122 nodes. All sensor nodes are expected to form a mesh network autonomously and transmit measured physical quantities periodically to a central base station (blue rectangle shown in Figure 1.2). All collected data is visualized to help monitor the health of all electrical equipment in substation.

The primary objectives of the TVA Paradise network are as follows:

- RF Connectivity: TVA's Paradise substation has electrical components set up over a wide area. This implies that the monitoring network should be capable of

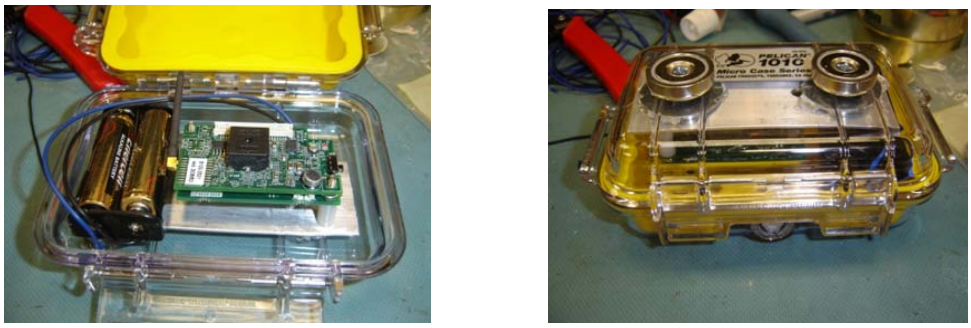
covering this area by establishing connectivity through the wireless sensor nodes deployed throughout the substation. The metallic structures in the substation environment also make the low power transmissions from sensor nodes unpredictable. Hence, ensuring connectivity over a wide area throughout the substation environment is a primary challenge.

- Application development: Monitoring techniques used in the substation environment involve use of sensors and techniques, some of which are not designed for standard wireless sensor networks. For example, gas density and transformer bushing monitoring involve sampling of physical quantities that do not use the available standard on-board sensing devices. Therefore, it is very important to determine a set of prognostic and diagnostic applications that are useful when monitoring physical quantities in a substation environment.
- Battery life: Sensor nodes are equipped with batteries with finite energy. It is very important to use this judiciously as frequent battery replacement in a large-scale sensor network is not feasible. The main challenge here is to minimize battery consumption by using efficient routing techniques and exploring innovative ways like solar-energy harvesting.

1.1.2 System Operation

Wireless sensor nodes used for monitoring electrical equipment in the Paradise substation were configured using Crossbow's MICAz motes, which are processor-radio boards for wireless communication. Crossbow's sensor and data acquisition cards, MTS310 and MDA320 mate directly into the MICAz motes for sensing of physical quantities. The software application was designed using Crossbow's *XMesh* protocol

which is a full featured multi-hop, ad-hoc mesh networking protocol working on the TinyOS platform.



(a) (b)

Figure 1.4: (a) MICAz sensor node with the sensor board, connected to AA size alkaline batteries. (b) Hardware components enclosed in a weatherproof casing fitted with magnets to attach it-self to metal structures.

Sensor nodes used for monitoring surface temperatures of transformers and circuit breakers use resistor temperature detectors (RTD's) mounted on the MDA320 data acquisition board. Each node is housed in environmentally protected enclosures equipped with two 1.2V NiMH or alkaline batteries (Figure 1.4). For vibration and *sound sensing*, a 2-axis accelerometer and the microphone on Crossbow's MTS310 sensor boards were used. Similarly, for ambient temperature sensing was performed using the onboard temperature sensor on the MTS300 sensor board while gas density sensing was performed using the Trafag 8774 [25] sensors.

The nodes perform periodic sampling of surface temperatures, vibration and sound intensity with a data interval of T_D= 900*s* depending on the application. Each node sends a route update every T_{RUI} =7200*s* to update network routing information. All physical data transmitted by the nodes is collected at the base station and logged on to a PC through a serial port via Crossbow's MIB510 programming board or to a Stargate

gateway. All data is made available on-line through a secure FTP server which can be accessed remotely. The data is stored as *.dat* flat files for further processing and pattern study.

Figure 1.5 shows sample signals obtained from the sensing nodes. Data from different sensing nodes is compared to determine faults in oil-cooled circuit breakers. A faulty circuit breaker will tend to be warmer than others in the same ambient conditions. Similarly, data collected from vibration and sound-monitoring sensors attached to transformers can also be used to detect abnormal running conditions depending upon the deviation of data from the average.

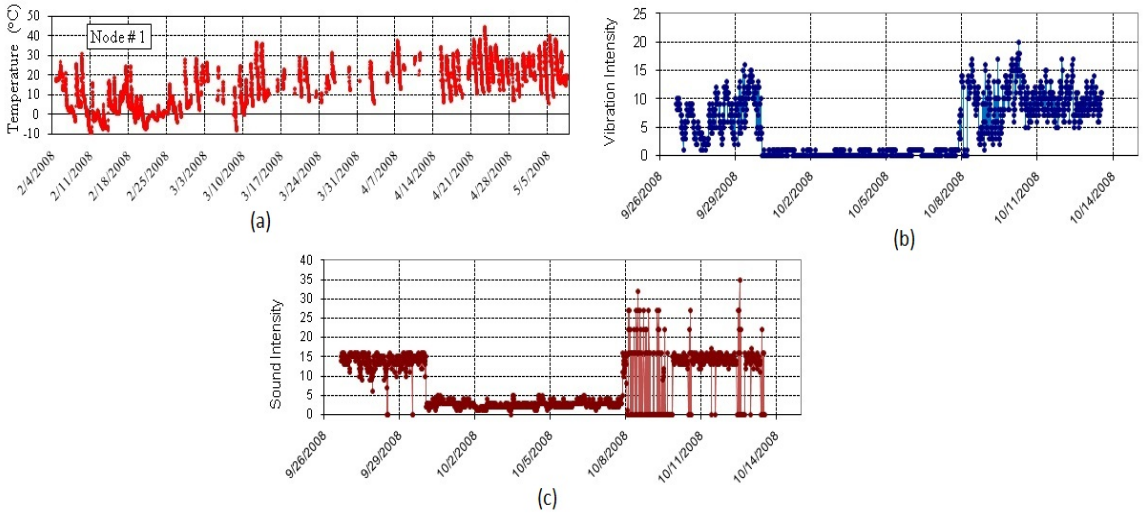


Figure 1.5: Measurements obtained from sensor nodes – (a) Surface temperature, (b) Vibration Intensity, (c) sound intensity

1.2 Motivation and Objectives of this thesis

The *XMesh* protocol used in the Paradise network use link quality estimates to decide the next hop (parent) to forward a data packet to the base station. All nodes in the network are programmed to choose a parent with the least cost (best link quality). Because of this practice, a typical scenario was observed in the Paradise network, where a

few nodes with better link quality were involved in forwarding a major share of the data traffic. Transmission and reception of each data packet involves energy, and a very high rate of forwarding packets would result in rapid draining of battery capacity of the sensor nodes. This load imbalance in the network would result in high data traffic forwarding nodes (critical nodes) to drain their batteries and stop operating earlier than other nodes in the network. Therefore, the substation equipment being monitored by these nodes would, become vulnerable, as they would no longer be monitored. Untimely death of some nodes may also lead to undesirable effects like network partitioning.

It is also essential that sensor nodes with a low battery capacity should not be permitted to forward a high number of data packets. A high traffic load at low battery capacities would result in a node battery voltages dropping below the cut-off level.

Hence, there was a need to develop a routing mechanism that would take into consideration the problem of load imbalance and battery depletion before deciding the routes. Traditional routing protocols do not consider data traffic and energy consumption while routing packets. We try to address this problem by designing a routing algorithm;

- That would take real-time data traffic of a node into consideration while deciding the routes, viz. making the routes *load balanced.*
- That would consider the real-time energy of a node before forwarding data packets, viz. making the nodes *energy aware.*

Lastly, we integrate concepts of load balancing and energy awareness to design a hybrid protocol that would bear advantages of both.

1.3 Organization of thesis

This thesis report is organized into five chapters. Chapter 1 provided an introduction

to the electric power substation monitoring and presented a detailed explanation its system operation. Chapter 2 describes the system components of the Paradise network. It gives a detailed explanation of the hardware and software platforms used and discuss the performance of the Paradise mesh network. Chapter 3 discusses load imbalance routing issue and presents the *load-balancing algorithm* along with the laboratory experiments and calculations performed to prove improvements in network lifetime. Chapter 4 presents the *energy aware algorithm* along with its advantages and simulation results. Chapter 5 explains and demonstrates the integration of the load balancing and energy aware routing algorithms. Chapter 6 concludes the thesis work and suggests future work to improve network performance.

CHAPTER 2: SYSTEM DESCRIPTION OF THE PARADISE NETWORK

A wireless sensor network typically consists of hardware components featuring sensor nodes equipped with processors, memory and radio capable of wireless communication; sensor boards which feature temperature, light, acceleration and acoustic sensors and software components which work on top of the hardware controlling sensing, processing and communication activities.

There are number of hardware platforms and operating systems that have been specifically designed to address the needs of wireless sensor networks. For the substation monitoring at Paradise, *MICAz* wireless sensor nodes manufactured by Crossbow Technology were used as the hardware platform and *XMesh* routing protocol, which operates on the TinyOS platform, was used for creating a wireless mesh network. This section presents the details of the hardware and software platforms.

2.1 Hardware Platform for the Paradise Network

The hardware platform for the WMSN consists of Crossbow's MICAz motes which are *processor-radio* boards for wireless communication, sensor and data acquisition cards (MTS300 and MDA320) which mate directly into the motes and Mote Interface Boards (MIB510) which allow developers to interface motes to PC's, PDA's or the internet.

- Crossbow's *MICAz* motes (Figure 2.1) are one of the most commonly used platforms for sensor networks [19]. It features Atmel's ATMega128L processor running at 8MHz, 128KB program memory, 512KB measurement flash and 4KB EEPROM.

The ATMega128L has 8 channels of 10-bit Analog to Digital Converter (ADC) facilitating usage of a variety of sensors being connected to the mote. For RF communication, it uses the Chipcon CC2420 chipset operating at 2.4GHz ISM band and uses Direct Sequence Spread Spectrum (DSSS) that is resistant to RF interferences. The nodes operate on AA size batteries with a voltage range of 2.7V to 3.3V. An important feature of MICAz motes is that it draws only 1µA current in its sleep mode where processor spends most of its time. MICAz motes support the TinyOS operating system that is open-source and flexible for application development. Apart from all the above advantages, another reason for choosing MICAz as a preferred platform was support of additional sensors and data acquisition boards which can be mounted on the 51-pin connector. MICAz platform also supports analog inputs, digital I/O, I2C, SPI and UART interfaces.

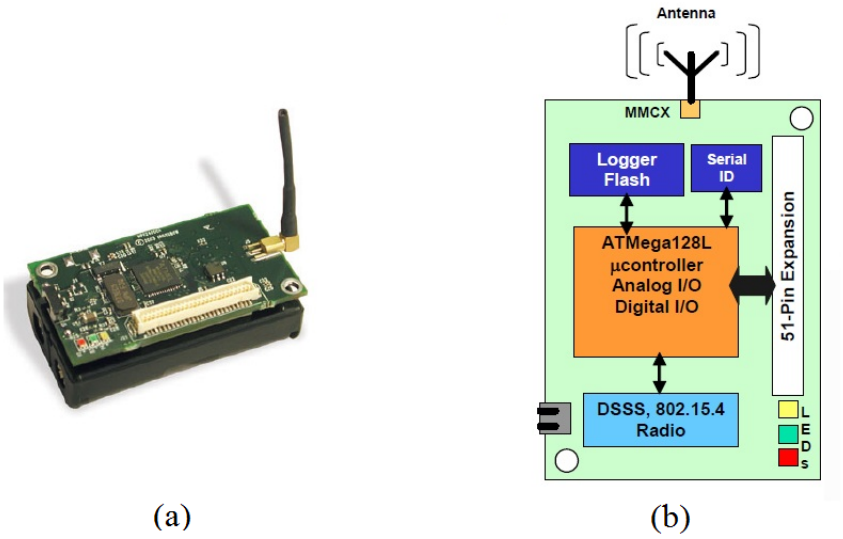


Figure 2.1: (a) MICAz mote by Crossbow Technology Inc. (b) Architecture of MICAz motes (Source: http://www.xbow.com)

- *MTS300CB* [21] (Figure 2.2 (a)) is the basic sensor board that plugs into the 51-pin I/O connector of the MICAz mote. It features an on-board ambient light sensor,

temperature sensor, microphone and a sounder. More advanced sensor boards like MTS310 feature a dual-axis accelerometer and magnetometer. The sensors on MTS300 are connected to different ADC channels of the microprocessor on the MICAz through the connector.

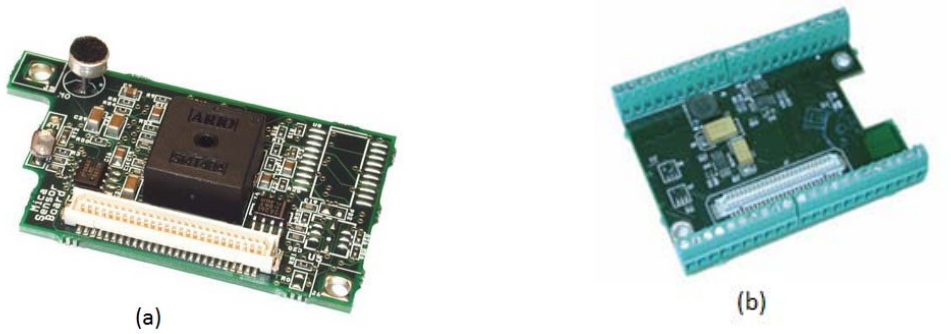


Figure 2.2: (a) *MTS300CB* sensor board, (b) *MDA300CA* Data Acquisition Board (Source: http://www.xbow.com)

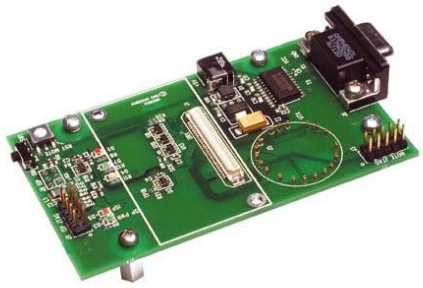


Figure 2.3: MIB510 Serial Gateway. (Source: http://www.memsic.com)

- The *MDA300CA* Data Acquisition Board (Figure 2.2(b)) includes an on-board temperature and humidity sensor, sensor excitation and a high-speed counter. With its multifunction direct user interface, the MDA300CA offers a convenient and flexible solution to those sensor modalities commonly found areas such as environmental and habitat monitoring as well as many other custom sensing applications. It supports up

to 11 channels of 12-bit analog input including 7 single-ended or 3 differential ADC channels.

- The *MIB510CA* [22] (Figure 2.3) allows a PC to collect data from each of the sensors in the network. Any MICAz node can function as a base station when connected to the MIB510CA serial interface board. In addition to data transfer, the MIB510CA also provides an RS-232 serial programming interface. The MIB520 has the same features, but it replaces the RS-232 serial port with a USB interface.

2.2 Software Components of the Paradise Network

2.2.1 The TinyOS Operating System

TinyOS [3] is an open-source operating system designed for wireless embedded sensor networks, which was released under the BSD (Berkeley Software Distribution) license. It features a component-based architecture that enables simple implementation of applications while minimizing code size as required by memory constraints inherent in sensor networks [4]. TinyOS differs from most other operating systems in that its design focuses on ultra low power operation for the low-power microcontrollers in sensor nodes. It provides a set of important abstractions and services that facilitate sensing, communication and storage. It defines a concurrent execution model, so that developers can build applications out of reusable services and components without having to worry about unforeseen interactions. TinyOS application and systems, as well as the operating system itself is written in the *nesC* language which is a dialect of C with features to reduce RAM and code size, enable significant optimizations and help prevent low-level bugs like race conditions [6]. A typical nesC program is a collection of *components* with

a one-to-one mapping between component and its source file name. For example, the file *LedsC.nc* contains the nesC code for component *LedsC.*

There are two kinds of components: *modules* and *configurations.* Module code declares variables and functions, calls functions and compiles to assembly code whereas configuration sections consist of nesC *wiring* code. *Wiring* is a mechanism to link components together via *interfaces.* Interfaces describe a functional relationship between two or more components. *Provided* interfaces are intended to represent the functionality that the component provides to its user, where as the *used* interfaces represent the functionality the component needs to perform its job. Interfaces have two specific functions: *command* and *events.* Users can "call" commands and providers can "signal" events. Conversely, users must implement events and providers must implement commands [6].

One of the important features of TinyOS architecture is that it is non-blocking; viz. it has only one stack [4]. Every long running operation is *split phase*; when a program calls a long-running operation, the call returns immediately, and the called abstraction issues a callback when it completes. The advantages of split-phase operations are:

- Split-phase calls do not tie up stack memory while they are executing.
- It keeps the system responsive as the call returns immediately.
- Split-phase interfaces enable a TinyOS component to start several operations at once and have them execute in parallel.

Apart from these advantages, it supports many different hardware platforms like Crossbow's MICAz, MICA2, MICA2DOT, TelOS, Eyes (TU Berlin) and iMote (Intel)

[7]. TinyOS also supports Texas Instrument's MSP430 microcontroller and any mote that uses this processor can be easily ported [4].

2.2.2 *XMesh* Routing Protocol

As stated earlier, the network at Paradise substation was designed using Crossbow's *XMesh* routing protocol and the adaption of the applications suites that were part of the Moteworks software package. *XMesh* is a full-featured multi-hop, ad-hoc, mesh networking protocol developed by Crossbow Technology Inc. Like a typical mesh network, it is a routing protocol programmable on Crossbow motes capable of hopping radio messages to a base station where they are passed to a PC or client. Each node promiscuously listens to the radio traffic in its neighborhood, and selects a parent that would be least costly in terms of energy to reach the base station. *XMesh* has many features and benefits to support its choice as a popular mesh networking protocol.

- *XMesh* has the ability to autonomously choose routes for delivering packets after random deployments. Because of the periodic route updates transmitted, the network is self healing and capable of forming new routes if some nodes in the network go offline.
- *XMesh* always tries to maintain bi-directional communication between the base station and sensor nodes scattered in the network. It supports both *upstream* and *downstream* communication. Upstream communication refers to transmission of data packets from sensor nodes to the base station whereas downstream communication implies end-to-end acknowledgement sent by the base station to the motes to indicate packet delivery. End-to-end acknowledgements are often disabled to minimize

communication costs and save battery power where guaranteed delivery of all packets is not a priority.

- *XMesh* provides multiple power modes for operation: *high power, low power and the extended low power modes* respectively in order of the power utilized for communication. Depending on the application and layout, custom power modes can be chosen to maximize battery life of motes.
- *XMesh* support transmission of health packets that are sent periodically to the base station by every node in the network to indicate health and operating capacity of the nodes. This information may include battery voltage, radio signal strength indicator and network traffic. Transmission of health packets is again optional and can be disabled to save power.
- *Time Synchronization* is also an important feature of in XMesh to reduce the size of the packet *preamble. XMesh* protocol programs the mote radio of wake up every 125 msec to sniff the radio channel for data and route packets. For any mote to transmit a data or route packet, it has to indicate the onset of a valid data packet by sending an initial *preamble packet* to indicate its channel occupancy. Preambles in the low-power mode are 140msec in size and have a significant current draw during each packet transmission. XMesh provides a way to minimize the preamble size to 40msec by enabling time synchronization by sending periodic control packets to synchronize timers in all nodes in the network.
- *XMesh* also supports Over the Air Programming (OTAP) that allows users to reprogram all motes in the network with a new code. This application uses the

downstream communication feature where the base station transmits the new code to all nodes over multiple hops.

One of the main advantages of XMesh is its power configuration, each of which is a trade-off between power and data rates. They are:

- *XMesh-HP*: This is a high power-consuming mode typically suited for nodes with continuous power supply. The radio is continually powered and the nodes can transmit and receive messages at any time. Since they are continually powered, route update messages can be sent at a faster rate, which improves agility of the network.
- *XMesh-LP*: This is a low power mode where the radio is programmed to wake up once every 125 msec to listen for transmission by neighboring nodes. Neighboring nodes, also configured for XMesh-LP, send a long preamble, which is typically longer than 125 msec before sending a message so that the destination node could sniff the preamble, and keep its radio on for message reception. Time synchronization can also be included to achieve further optimization by reducing preamble size. Because of low-power communications, the *XMesh-LP* mode was used in the Paradise network.
- *XMesh-ELP*: This power mode consumes the least amount of power but only works for nodes which are located on the edge of the network and which do not participate in data forwarding for other nodes. These devices normally remain in the sleep state for very long times and wake up only to sense and transmit its own data to the parent.

Because of random deployment of sensor nodes, mesh networking protocols should be capable of determining efficient routes autonomously. Route determination in *XMesh* involves two main processes:

- *Link Estimation:* Link estimation is an operation to gauge the link quality of neighboring sensor nodes by promiscuously listening to their data traffic and route updates. The estimates are stored in an internally defined neighbor table which is a data structure used to store selected link information from the data and route packets. *XMesh* usually defines a neighbor table size of 16; if a node can hear packets from more than 16 nodes; it chooses to populate the neighbor table with nodes having better link qualities. Since the base station is not energy or memory constrained, it can be programmed to have a larger table size. *XMesh* uses a pre-defined *route update interval* (RUI) to send route update packets. Route update packets are used to compute bi-directional link quality for parent selection. It uses the metrics *receive estimate (RE)* and *send estimate (SE)* to calculate link qualities;

  Receive estimate is the radio receive quality of the link from a specific neighbor, computed by applying the Exponentially Weighted Moving Average (EWMA) algorithm to the percentage of received packets. It is calculated as;

$$NewEstimate = 255 * packets_{received}/(packets_{received} + packets_{missed})$$

$$ReceiveEstimate = (1 - \alpha)ReceiveEstimate + \alpha * NewEstimate \quad (2.1)$$

  Where α is the EWMA factor between 0 and 1 that defines the weight of the new estimated value. A higher value of α would give a greater weight to the *NewEstimate* and vice-versa.

  Send estimate is the radio transmits quality of the link to a specific neighbor obtained from the route update messages. All route update broadcasts contain the *receive estimate* of its neighbors so that bi-directional *link cost (LC)* can be calculated as;

$$LC = (1 \ll 18)/(SE * RE) \quad (2.2)$$

Route update messages from a neighboring nodes contains a value, *neighbor cost* (NC) which is the cost estimated by the neighbor to send its packets to the base station. The final metric calculated by each node is the *Optimal Cost (OC)* calculated by adding *LC* and *NC.*

- *Parent Selection:* Each node scans its neighbor table for a parent with the least cost as per the above computation in Equations (2.1) and (2.2). A neighboring node is considered as a parent if:
  - It has already joined the network
  - It is not a descendant node for the last three route update intervals so that there is no loop formation, and
  - It is not an *Extended Low Power* node.

2.3 Performance Issues of *XMesh* in the Paradise Network

Although XMesh is seen to perform fairly well in terms of data reliability, connectivity, scalability and ease of operation, it was observed to have some problems with routing performance and irregularities in power consumption.

2.3.1 Energy Consumption

As shown in Figure 2.4, for evaluation purposes, the network at TVA's Paradise Substation was hypothetically divided into four zones according to their proximity from the base station. Density of nodes was highest in zones C and D and least in zone A. Each node in the network periodically sends its internal battery voltage reading with every data packet. These readings were used to determine the actual energy consumption in the Paradise network.

Even though battery voltage cannot be assumed to be an accurate indicator of battery capacity, the fact that all sensor nodes operate on the same voltage with approximately the same rate of power consumption makes battery voltage an important variable to measure energy consumption.

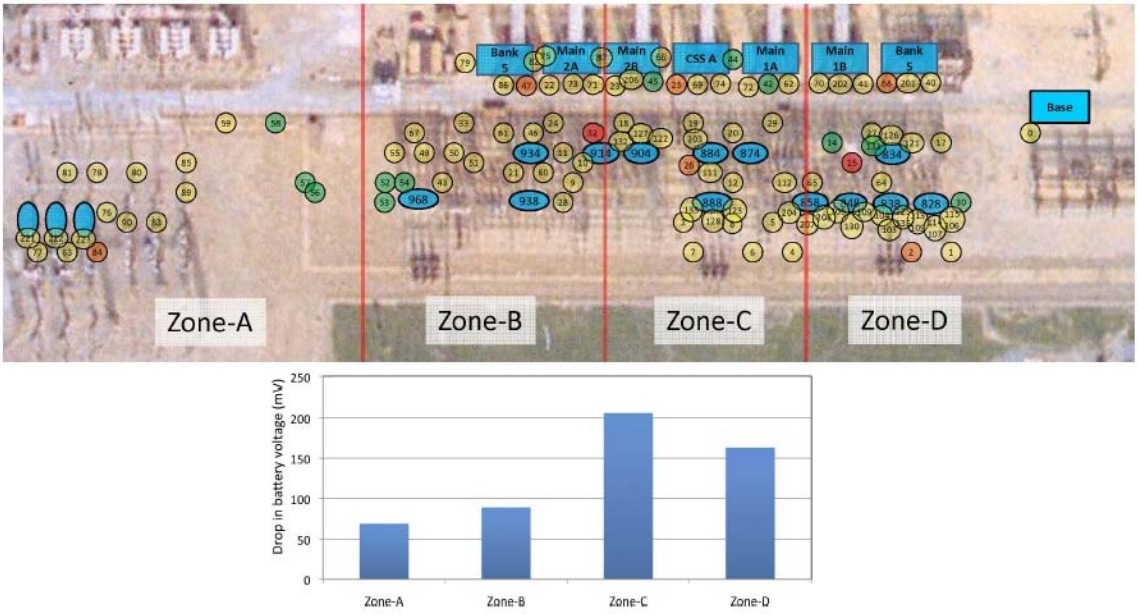


Figure 2.4: Drop in battery voltages in the Paradise network by zone

This can also be attributed to the fact that nodes with identical energy characteristics will have similar voltage drops related to battery capacity usage. Figure 2.4 also shows a graph that indicates drop in battery voltages for nodes in each zone between January and May 2010. It is observed that nodes farthest from the base station (Zone A) have experienced the least voltage drop, while Zone C, which is near the base station, is seen to have the highest drop in battery voltage. This is basically due to the fact that sensor nodes in Zone C are responsible for delivering most of the packets from Zone A and Zone B and also experiences higher amount of overhearing traffic. This type of operation

will ultimately result in, nodes in Zone C dying earlier than the ones in other zones which will collectively result in decreasing the lifetime of the network.

2.3.2 Routing Performance

To visualize the routing performance, a C program was developed to read all log files and generate a routing table with all routing and data information such as the number of descendants, node parent ID and the number of packets that were transmitted on a daily basis.

Pooling the results of the program showed an imbalance in the number of descendants for many nodes. For example, nodes 15 and 32 were seen to carry majority of the data forwarding traffic from more than 20 nodes. Apart from this, six other nodes were involved in forwarding traffic of 10-20 nodes while a large majority of nodes were involved in forwarding very little data traffic. These nodes are marked in red, orange and green, respectively, in Figure 2.4. This performance clearly demonstrates the network imbalance issue involved in mesh networks. Even though the load imbalance did not have a large impact on energy drainage of these critical nodes in the Paradise network, it may cause early death for nodes that carry heavy forwarding traffic.

To improve routing performance and increase network lifetime, it is essential for each sensor node in the network to decide routes based on dynamic load and residual energy of its neighbors, such that, the data traffic load is balanced in all part of the network and nodes with lower residual energy are involved in forwarding very little data traffic.

CHAPTER 3: LOAD BALANCED ROUTING

In order to achieve a balanced traffic load in the network, the routing protocol must take real time data traffic into consideration while computing the routes. The routing protocol should favor only those routes that use nodes with less traffic load. The key problem in implementation of this concept is that it can cause oscillation in routes leading to instability and inefficiency. Hence, load balanced routing must have appropriate damping considerations. Imbalance in traffic load, also known as the *hot-spot* problem is discussed in section 3.1. Load balanced routing is expected to distribute data traffic over a larger area to facilitate uniform energy usage over the network.

3.1 The *Hot-Spot* Problem

A *large scale mesh network* consists of sensor nodes scattered densely over a wide area. Nodes are expected to autonomously decide routes to the base station. Most of the routing protocols are designed in a way where nodes choose the best routes according to the routing metric broadcasted by the neighboring nodes. In a real world scenario where nodes are subject to different physical or environmental conditions, each node has a different link quality than the other even if they are at the same distance from the base station. External factors include placement or location of nodes that also involves the height at which nodes are placed from the ground level, external interference or reflections [14]. The Paradise Substation environment is also characterized by surrounding metallic structures that cause unpredictable transmission link quality

characteristics. Because of these factors, link qualities of nodes are highly variable; which gives rise to the “hot-spot” problem where many nodes in the network choose only one particular node with the highest link quality as a parent. This leads to a surge in data traffic at this critical node resulting in high energy consumption and battery capacity drainage. The hot-spot problem that occurred in the Paradise network is illustrated in Figure 3.1 which shows a snapshot of some routes that hop over the same node (node 32).

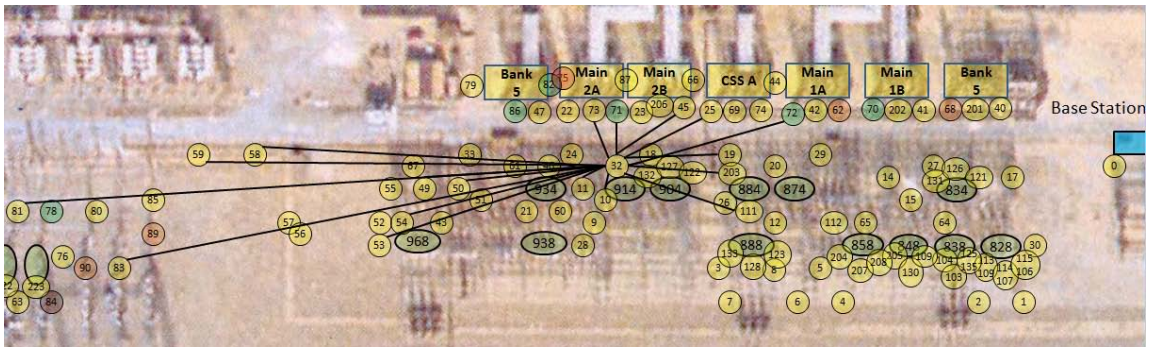


Figure 3.1: Hot-spot problem in the Paradise network. The figure shows node 32 as the preferred parent for many nodes.

Node 32 shown is one of the critical nodes in the network. *Critical nodes* are the ones which are usually *single-hop* from the base station, having a higher link quality than other sensor nodes as a result of which it is chosen as a preferred next-hop (parent node) by other neighboring sensor nodes. The result of the hot-spot problem is traffic congestion at node 32 which causes significant energy drainage at that node.

For wireless sensor network applications, the energy used by a node consists of energy consumed by receiving, transmitting, listening for messages on the radio channel, sampling data and the sleep mode.

$$E = E_{rx} + E_{tx} + E_{Listen} + E_d + E_{sleep} \quad (3.1)$$

$E_{d,}$ the energy involved in sampling is a constant factor which is predetermined as it

depends on the sampling rate required by the application. E_{sleep} is at least an order of magnitude smaller than the other terms in the equation, so it has negligible effect on total energy *E*. Most of the energy is spent on transmitting, receiving and listening of radio packets sent over the radio. In case of the scenario described in Figure 3.1, because of the high amount of traffic at node 32, a lot of energy is drained while receiving the packets over the radio from all of its descendants and then transmitting each packet towards the base station. Theoretically, the energy levels of the critical nodes will drop significantly as compared to other nodes in the network which may result in early death for the node leading to network partitioning and other undesired effects [2]. The electric equipment such as a transformer or a circuit breaker on which the sensor node was attached will become vulnerable as it will no longer be monitored and any potential problems will go undetected. This would defeat the very purpose of having a monitoring network.

Because of energy and routing performance issues, it is very important to design a routing protocol that maintains a balanced load over the network and is also energy aware, which would result in graceful degradation of the energy resources in the nodes. Since the *XMesh* networking protocol is a Crossbow proprietary, its source code is not available for development purposes. Hence it is very difficult to address any networking issues or to customize it. However, TinyOS-2.1.1 features the Collection Tree Protocol (CTP) that is also a mesh networking protocol on the lines of *XMesh* with similar features and networking capabilities. We will be using CTP for all further research as it is open-source. An advantage of CTP is that it is open-source, and hence, can be easily adapted for incorporating new features such as load and energy balanced routing. In the following

sections, we will discuss the features of CTP and our proposed approach for adapting it for our own work.

3.2 Collection Tree Protocol (CTP)

CTP is a tree based routing protocol wherein some sensor nodes in a network advertise themselves as tree roots (data sink) whereas all other nodes in the network are expected to transmit data packets to these tree roots. CTP is an address-free protocol in that a node does not send packets to a particular root; instead, it only chooses the next hop parent depending on a routing gradient. Nodes generate routes to the data sink using a routing gradient [8, 9].

The CTP protocol assumes that the data link layer provides four variables:

1. Provides an efficient local broadcast address.
2. Provides synchronous acknowledgments for unicast packets.
3. Provides a protocol dispatch field to support multiple higher-level protocols.
4. Has a single-hop source and destination field.

CTP has a neighbor link table wherein it stores link quality estimates of some of its nearby neighbors. The table size can be modified, however a large size can lead to slow and sluggish response because of the memory required to store the details and to compare all links serially. CTP is designed for relatively low traffic rates and is well suited for an application like sub-station monitoring where the traffic rate is very low.

CTP uses Expected Transmissions (ETX) as its routing gradient. The data sink or base station always broadcasts a metric of 0. The ETX of the node is the ETX of its parent plus the ETX of its link to the parent. This measure assumes that nodes use link-level retransmissions. Given a choice of valid routes, the CTP algorithm should choose the

route with the lowest ETX value. CTP represents ETX value as 16-bit fixed point real numbers with precision of hundredths.

3.2.1 The CTP Routing Engine

The CTP routing engine is responsible for computing route for the next hop. The Collection Tree is proactively maintained by periodic beacons sent by each node. The beacons are jittered in time to prevent synchronizations in the network. All nodes maintain the same beacon sending rate.

- Neighbor table: Every time a node receives an update from a neighbor, it records the information if that node is a part of the neighbor table. The neighbor table keeps the nodes with the best path metric as candidates for parent. As mentioned above, there are two main uses of maintaining the neighbor table; the first one being the selection of the parent node which is evidently the neighbor with the best path metric and the second is to actually choose the next hop towards the root which may or may not be the current parent. The operation of routing engine has two main tasks and one main event:
    - The *updateRouteTask* is called periodically and chooses a new parent.
    - The *sendBeaconTask* broadcasts the current route information to the neighbors.
    - The *BeaconReceive* event occurs upon receiving beacons from neighboring nodes which updates the neighbor table.
- Parent Selection: All sensor nodes in the network choose only one node as their parent and next hop depending on the cost. The cost of a route is an abstract measure of distance; it may be number of hops to the sink, expected number of transmissions

or some other estimate. The routing engine accesses the neighbor table to choose a parent from a list of potential parent candidates. When a node is first switched on, it broadcasts beacons and requests its neighboring sensor nodes to send route update packets in return so that they would populate their neighbor table for parent selections. A neighbor is selected as a parent if its bi-directional cost is sufficiently smaller in cost than other neighboring nodes.

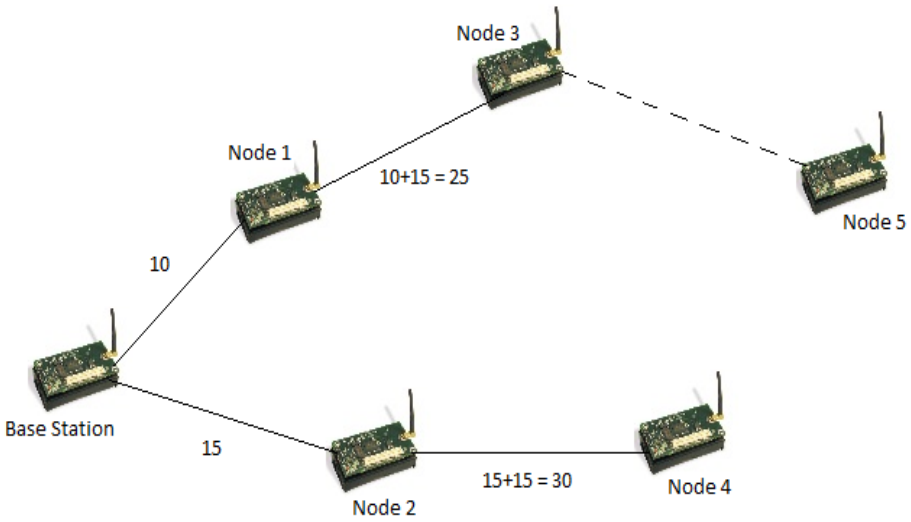


Figure 3.2: Parent selection and *link cost* broadcasts. The numbers shown on the links indicate *link cost.*

- It may also switch to a new parent if the link quality to the current parent drops below a preset threshold, if the data sink in unreachable through the current parent, or is a routing loop is detected. Selection of parent is more clearly explained in Figure 3.2.

Each node broadcasts its cost:

Node cost = parent's cost + link cost to parent

As shown in the Figure 3.2, Node 1 and Node 2 broadcast a link cost of 10 and 15 respectively. Base station always broadcasts a metric of zero to encourage neighboring

nodes for a one-hop communication. The difference in the link costs of two nodes may be simply because one has a better channel quality than the other. Node 3, which is a descendant of Node 1, broadcasts a metric of 25 which is a sum of metric of Node 1 and its own path metric leading to Node 1. Similarly, Node 4, a descendant of Node 2 broadcasts a routing metric of 30. The routing engine of Node 5 records the metric of both, Node 3 and Node 4 and chooses the former to be its parent because of a lower link cost.

3.2.2 The CTP Forwarding Engine

The CTP Forwarding Engine is responsible for queuing and scheduling of outgoing packets. It has the following responsibilities:

- Transmitting packets to the next hop, retransmitting when necessary and passing acknowledgment based information to the link estimator.
- Deciding when to transmit packets to the next hop.
- Detecting routing inconsistencies and informing the routing engine.
- Maintaining a queue of packets to transmit, which are a mix of locally generated and forwarded packets.
- Detecting single-hop transmission duplicates caused by lost acknowledgments.

- Message Queuing: The forwarding engine maintains two levels of queues. The top level queue is known as the “client queue”. The client queue is one level deep and keep track of whether a client has an outstanding packet. The lower level queue is used to store both, locally generated traffic and traffic to be forwarded to the base station. When the forwarding engine receives a packet, it first checks if the packet is a

duplicate. If the packet is not a duplicate, it puts the received packet on the send queue.

- Duplicate Packet Elimination: To avoid duplicate packets, the forwarding engine at the originator node appends a sender ID, a sequence number to each outgoing packet and the THL (Time Has Lived). To avoid sending out duplicate packets, each node maintains a cache of most recent sequence numbers for each of its child nodes. If a parent finds another incoming packet with the sender ID and the sequence number as same, it will not insert the packet in the send queue for transmission. For maximum efficiency, the transmit cache should be large; but under experimental conditions, it has been found that having a cache size of four slots is enough to suppress most duplicates on a test-bed [8, 9].
- Prevention of routing loops: Routing in CTP is in terms of cost gradient, where the base station has a cost of zero and a node's cost is the cost of its next hop plus the cost of the link to that next hop. If there are no loops, the gradient value decreases monotonically along a route. The forwarding engine en-queues the local gradient value in each packet header. If a node receives a packet whose gradient value is less than its own, it would mean that the gradient is not monotonically decreasing as expected. This would mean a possible routing loop. The forwarding engine informs the routing engine to advertise its own routing metric through beacon broadcasts so that this would inform the neighboring nodes to broadcast their routing metric as well. This process will help the routing engine update its neighbor table with the latest values and assist it in choosing a new parent in case of a routing loop.

3.2.3 The Four Bit Link Estimator

CTP takes input from the link estimation layer [10] to broadcast its own routing gradient. The routing gradient is a function of inputs taken from the *physical layer, link layer* and the *network layer* of the routing protocol. Figure 3.3 shows representation of the link estimator. Outgoing arrows represent information that the estimator requests on packets it receives. Incoming arrows represent information the layers actively provide. Each layer from the protocol stack can supply valuable information about the link dynamics.

- *Physical layer* provides information on the channel quality during packet transmissions. This information is provided by the Link Quality Indicator (LQI) of the underlying CC2420 radio layer through the *LinkPacketMetadata* interface. It calculates channel quality based on the number of bit errors in a data packet. A packet with few bit errors is more likely to have a better channel quality as compared to a packet with many bit errors. This information is distilled down to the *white bit*, which denotes whether the channel quality was high.
- *Link layer* provides information about the delivery and acknowledgement of packets. The link estimator uses periodic broadcast probes to measure incoming packet reception rates. They calculate the bi-directional link quality which is a probability of a packet being transmitted and its acknowledgement received. The best path is the one that minimizes the total number of transmissions in delivering a packet over multiple hops to the destination. This metric is called the Expected Transmission Cost (ETX). ETX is calculated as;

$$ETX = \frac{1}{Forward\ delivery\ ratio} \times \frac{1}{Reverse\ delivery\ ratio} \tag{2.3}$$

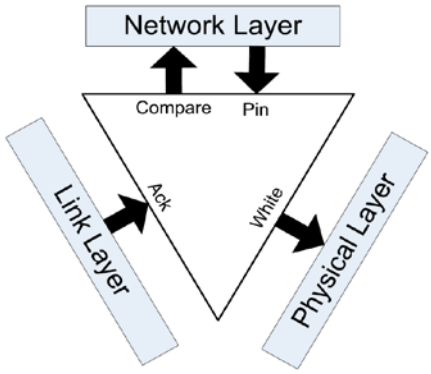


Figure 3.3: Representation of the *Link Estimator* [10]

- *Network layer* uses information from the physical and link layers to form a neighbor table where it stores network related information of neighboring nodes recorded by listening to route packet broadcasts. Because of memory constraints in sensor networks, the neighbor table is of a limited size; the network layer decides which links are valuable for routing and which are not. The network layer provides two bits of information, namely, the *pin bit* and the *compare bit*. The pin bit is applied to entries in the neighbor table which are considered to be mature links. A link becomes *mature* after a certain number of predetermined packets are received and an estimate is computed. The compare bit indicates whether the route provided by the sender of the packet is better than the route provided by one or more entries in the link table.

3.2.4 Adaptive Beaconing in CTP

Maintaining the network operational for a long time is of utmost importance for any routing protocol. All nodes need to maintain link information of its neighboring nodes in order to choose a parent, transmit data packets to the next hop and avoid formation of routing loops. This routing information is periodically broadcasted by all nodes in the

network at fixed time intervals. A single beacon updates many nearby nodes. It is essential that all nodes should have the latest routing information in order make a proper selection of the next hop.

Protocols like *XMesh* typically broadcast control beacons at fixed intervals [12, 13]. A smaller beacon interval leads to a more agile network but a higher cost whereas a larger interval uses less bandwidth and energy but can let topological problems persist for a longer time due to stale routing information. Adaptive beaconing extends the Trickle algorithm [11] to break this tradeoff to achieve both agility and low cost. The mechanism it follows is to transmit a version number for each node using a randomized timer. If a neighboring node hears a same version number from some other node, it suppresses its own transmission. When a timer interval expires, it doubles its interval. This interval continuously doubles itself until it reaches a preset maximum value beyond which it remains stable. This would represent a stable network. The timer interval resets itself to a minimum value again on three events:

- If it is asked to forward a data packet from a node whose routing metric (ETX) is not higher than its own. This would otherwise indicate a possible routing loop.
- Its routing cost decreases significantly. A timer reset is done to inform neighboring nodes of the significant decrease in cost which would mean a good link quality. This would encourage neighboring nodes to choose this node as a parent or next hop.
- If it receives a packet with a “P bit” set. According to the CTP routing frame, a set P (Pull) bit indicates that a node that broadcasted this frame wishes to hear beacons from its neighbors. The causes may be because a node has just joined the

network and needs to build its routing table or if a node does not have a valid next hop.

Adaptive beaconing thus reduces its beacon interval only when required in the above mentioned conditions while otherwise maintaining a longer interval to lower costs.

The study of Collection Tree Protocol (CTP) shows important advantages over the *XMesh* Routing protocol:

- CTP uses the four-bit link estimator which is seen to have better estimation capabilities than other estimators like the *MultihopLQI.* [8].
- CTP supports adaptive beaconing which is very essential to maintain network agility in multi-hop networks.
- We can design our own custom routing protocol by modifying the routing metric by adding variables calculated from the load and energy state.

The problem of load imbalance is seen in the Paradise network which uses the *XMesh* protocol. To visualize the effects of load imbalance in CTP, an experiment was performed to re-create the problem on a MICAz test-bed which is discussed in detail in section 3.3.

3.3 Experimental Verification of the *Hot-Spot* Problem

An experiment was performed to simulate the conditions of a real world network in the laboratory with Crossbow's MICAz motes, running the CTP routing protocol.

- A total of 18 *MICAz motes* (which includes the base station) were chosen to for the experiment to form a mesh network. All the motes were programmed with the *MViz* multi-hop application (*tinyos-2.1.1/apps/MViz).* The *MViz* application was modified to be similar to the Paradise network for experimentation purposes. The *base-station*

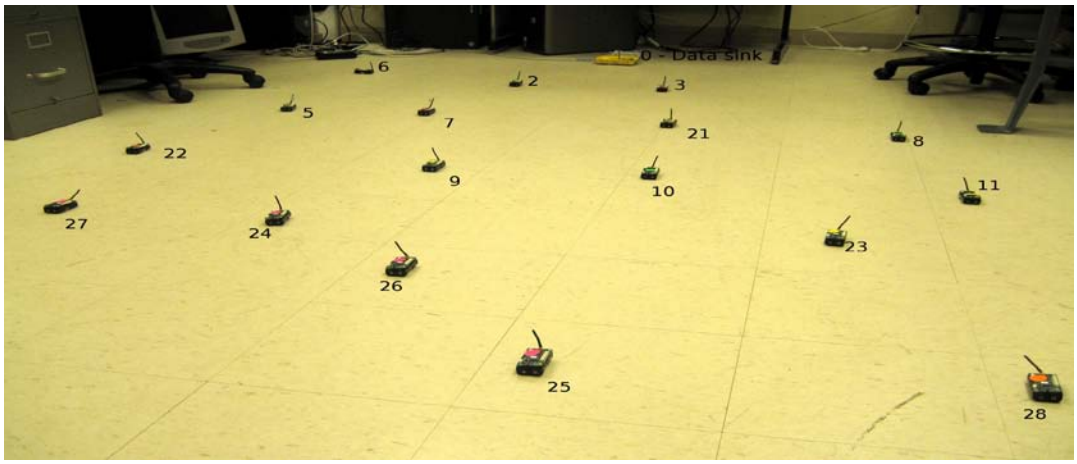
mote programmed with a node ID of 0, serves as the *data sink*. All nodes were arranged in *random layouts* for different iterations at a distance of 2 to 3 feet from each other. All nodes were programmed with an extremely low radio transmit power level of '3' or -25 dBm for indoor experimentation purpose.

- The base station was connected to a laptop running TinyOS version 2.1.1 on a Linux box (Ubuntu 10.10) using a MIB520 programming board through a USB port.
- A *serial listen* program developed in C included in the TinyOS repository was modified for the *MViz* application for data logging in text files. The *serial listen* executable listens on the serial interface when executed. A *real time log monitor,* also developed in C, continuously reads the logs files being updated in real time and displays the network data in a tabular form for human interpretation. The application variables for the experiment are enlisted in Table 3.1.

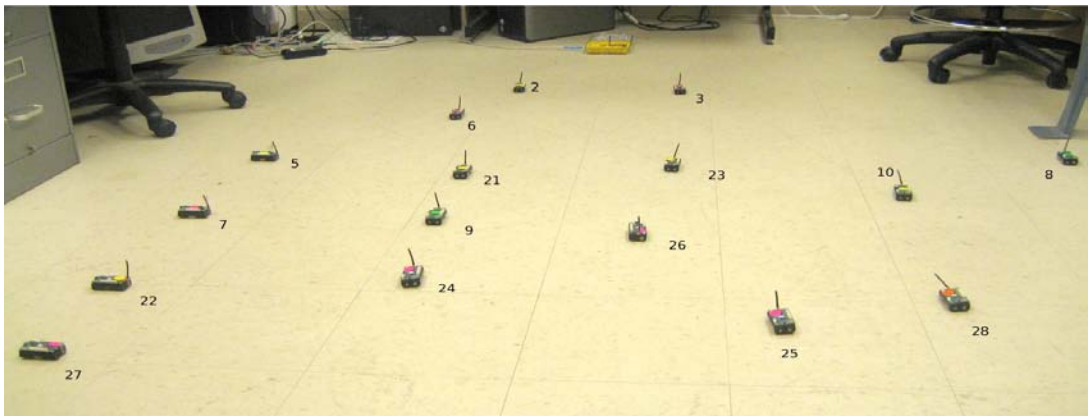
Table 3.1: Application variables for load balancing experiment

Total number of nodes in the network	17
Data transmit interval	15 seconds
Route update interval	60 seconds
Output signal power level	-25dBm ,level 3
Low power listening interval	125 milliseconds

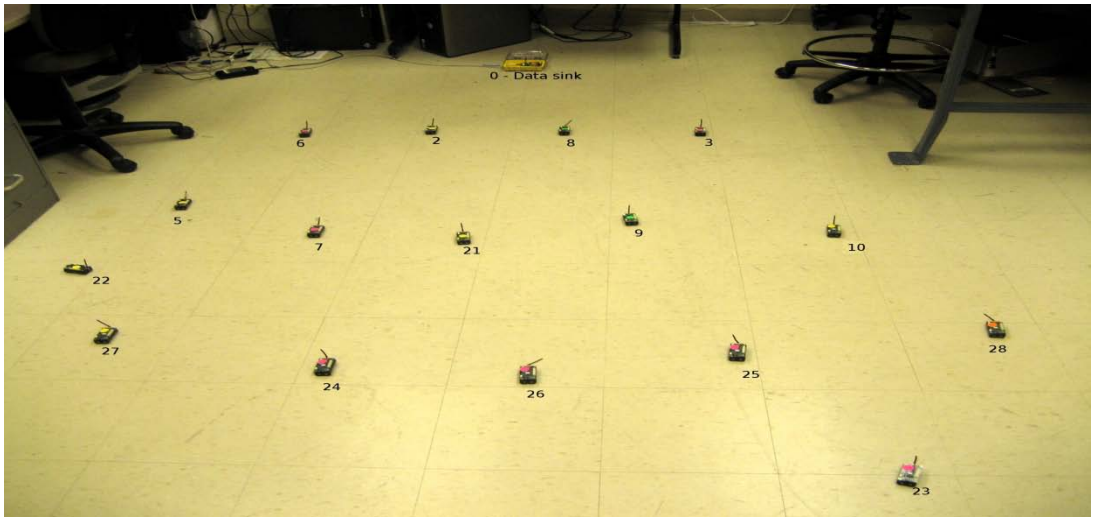
The experiment was run several times for different layouts. For each run, the ETX metric and the amount of forwarded traffic for the critical nodes were recorded at two different times. Table 3.2 shows the experimental results. In the table, a run number *x.y.z* indicates a layout (or deployment scenario) *x*, experiment number *y*, and record number *z*. The three different layouts considered in this experiment are shown in Figure 3.4.



(a)



(b)



(c)

Figure 3.4: Random layouts for the experiment, (a) Layout 1, (b) Layout 2, (c) Layout 3

For each layout, the experiment was repeated twice (z = 1, 2). Record number 1 was recorded when the nodes had completed at least 20 data sampling intervals whereas record number 2 recorded when the nodes had completed at least 50 data sampling intervals.

This method gives a more comprehensive overview of the operation of the network as nodes in the network are given sufficient time to discover stable routes. The *data rate* in Table 3.2 is the rate per unit time at which data packets were forwarded by the critical nodes to the base station. Since the sampling interval in this experiment is 15 seconds, the data rate is calculated *per minute*. It is evident from the recorded data rates that there exists a formidable level of load imbalance in the network. For example, in case of *run* 1.1.1, node 7 which carries most of the forwarding traffic is the *hot spot*, whereas other critical nodes 3 and 2 forward negligible amounts of data traffic. In the same run, even when the link quality of node 7 decreases (*run #* 1.1.2), it continues to forward a large number of data packets. This is because each node is programmed to change a route only when it finds a link that is better than a value of its present link by *parent switch threshold* [9]. This threshold value is usually between 10 and 20. The *parent switch threshold* however, is important to prevent oscillation of nodes from one parent to another. In most of the runs, there is a significant variance in the data forwarding rates of the critical nodes.

3.4 Effects of the *Hot-Spot* Problem

The hot-spot problem can have a significant effect on the overall network lifetime. *Network lifetime* in the Paradise network is calculated as,

$$I_{Node} = \frac{I_{tx}}{T_d} + \frac{I_{tx}}{T_{RUI}} + N * I_{rx}\left(\frac{1}{T_d} + \frac{1}{T_{RUI}}\right) + D * \frac{I_{tx}}{T_d} + \frac{I_{sense}}{T_d} + 8 * I_p/T_p \qquad (3.2)$$

Table 3.2: Data traffic and Network Lifetime without Load Balancing

Run #	Critical Node Id	ETX metric	Data Rate	Network Lifetime	Variance (Data Rate)
1.1.1	3	17	12	42	152
	7	10	30		
	2	14	0		
1.1.2	3	13	11	40	213.55
	7	29	35		
	2	17	0		
1.2.1	2	10	13	38	172.66
	3	10	36		
	5	15	5		
1.2.2	2	10	12	37	196
	3	10	40		
2.1.1	2	16	0	44	101.18
	3	13	2		
	7	13	25		
	8	20	4		
2.1.2	2	11	4	40	194.88
	3	10	32		
	8	17	1		
2.2.1	3	10	9	37	309.55
	8	16	41		
	10	19	0		
2.2.2	3	10	40	37	284.66
	8	15	0		
	10	16	11		
3.1.1	3	10	37	38	247.68
	5	24	0		
	8	15	1		
	22	28	1		
3.1.2	3	10	40	37	285.68
	5	22	2		
	8	14	1		
	22	25	0		
3.2.1	2	10	18	42	42.25
	6	10	31		

where, I_{tx} and I_{rx} are *receive* and *transmit* currents for MICAz respectively. T_d and T_{RUI} are the data and route update intervals which are predetermined according to application. I_{sense} is the current consumed for sensing physical quantities by the data acquisition boards whereas I_p is the processing current. A factor of N is multiplied to the receive currents to include the worst case scenario where every node in the network overhears data and route packets of every other node. In case of load balanced routing, D is a factor which represents the number of descendants according to the packet forwarding rate. It is calculated as;

$$D = Packet\ Rate/(60/T\_d\ ) \tag{3.3}$$

if the data rate is in seconds. As evident from equation 3.2, the node current I_{Node}, keeps varying with the number of descendants in direct proportion. A node with a high packet forwarding rate will consequently have a higher current consumption than other nodes in the network which decreases the lifetime of the network. According to battery chemistry analysis, a higher discharge rate leads to a lower battery capacity [15].

Table 3.3: Experimentally obtained currents and durations of events in a MICAz mote

Event	Current (mA)	Duration (msec)
Route Update Tx	24	140
Route Update Rx	24	140
Data Tx	24	140
Data Rx	24	140
Sensing	25	20
Processing	11	3

We use equation 3.2 to estimate the lifetime of each node and the network lifetime. The current measurements were performed by recording the voltage drop across a 10Ω resistor that was inserted in series with the mote battery. Table 3.4 gives experimentally

obtained currents and duration of events in a MICAz mote for the temperature sensing application.

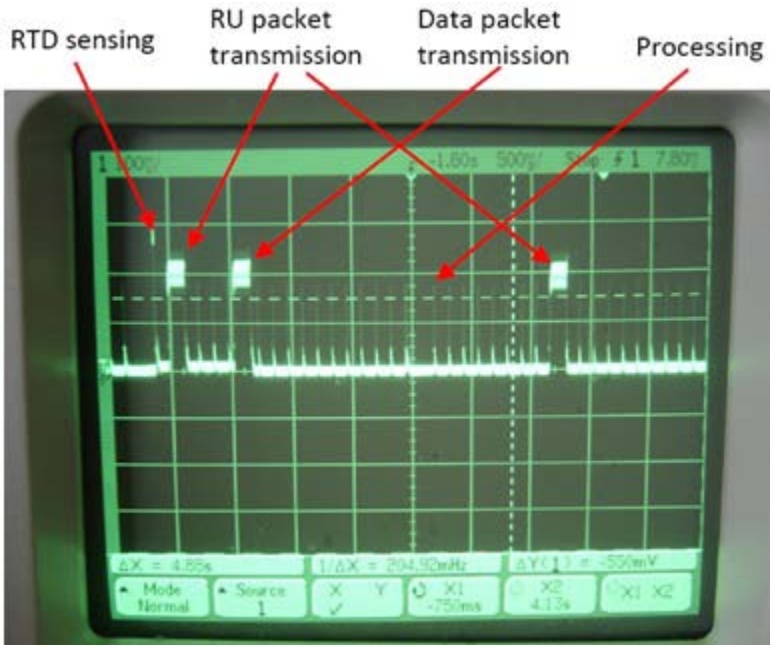


Figure 3.5: Current consumption of a MICAz mote in temperature sensing application

A C program was developed that takes inputs like the number of nodes in the network and the data rate of each critical node to determine the depletion of charge at each node. Each node was assigned a battery capacity of 5000mAH as its initial capacity. The program was executed for different runs and data forwarding rates. Total current consumption for each node is calculated using equation (3.2). Using the total current consumption, the lifetime of each node was calculated as:

$$Lifetime = BatteryCapacity / I_{Node} \quad (3.4)$$

Network Lifetime is defined as the time taken by the first node in the network to stop operating upon reaching cut-off voltage. We need this program to compare the network lifetime without load balancing with that obtained using the proposed load balanced routing algorithm described in section 3.2.

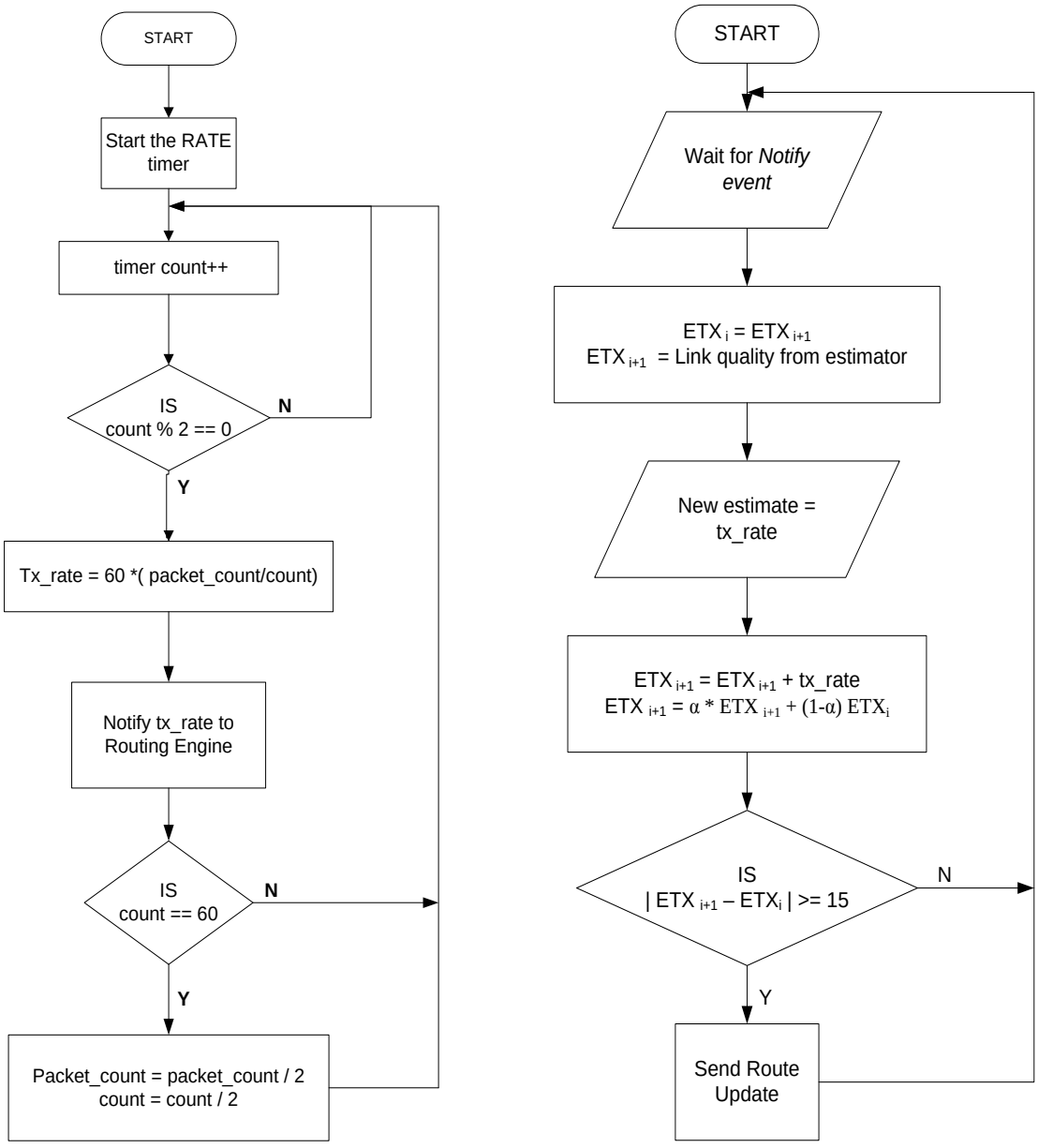
3.5 Load Balanced Routing

Table 3.2 shows significant variance in the data forwarding rates of the critical nodes. This effect leads to a decrease in network lifetime. A method to counter this effect is to include real time data traffic experienced by each node while designing the routing metric; this is known as *load balanced routing*.

3.5.1 The Load Balancing Algorithm

The network layer in CTP consists of the *CTPRoutingEngine* and the *CTPForwardingEngine* which are involved in routing decision implementation and packet forwarding respectively. To implement load balanced routing, it is essential to create a communication path between them to share real time traffic information. The *forwarding engine* forwards data packets received from its descendants and is capable of keeping count of the number of packets forwarded per unit time. The *routing engine* needs this information to make routing decisions based on network traffic. An interface, *Notify.nc* was created which would share the traffic data with the routing engine. A generic timer interface, *RateTimer* is defined to calculate the rate at which the packets are forwarded. As shown in the flowchart in Figure 3.6, for a data rate of 15 seconds, the packet forwarding rate is shared with the routing engine frequently so that the routing engine will update its routing metric accordingly. The routing metric (ETX) is directly proportional to the data traffic. A higher traffic level will lead to a higher value of ETX which will discourage at least some node to change their routes to a less congested node.

It is expected that, a rapid change in data traffic will also lead to similar change in the routing metric (ETX). Such change will cause the nodes to change their parent very frequently which will lead to unstable links and oscillation of routes from one parent to



(a) (b)

Figure 3.6: Flowcharts for load balanced routing, (a) CTP Forwarding Engine and (b) CTP Routing Engine

another. To avoid such situations, the routing metric is smoothed out using the Exponentially Weighted Moving Average (EWMA) algorithm which is represented as;

$$ETX_{i+1} = \propto * \ ETX_{i+1} + \ (1-\propto) * \ ETX_i \qquad (3.4)$$

Where, α is the *tuning parameter* or weight assigned. Depending on the weight it reacts to incoming values, to which it assigns more weight if $\alpha > 0.5$, and less weight is $\alpha < 0.5$. It does not drop older value from its time window, but slowly squeezes them out with the passage of time. Implementation of EWMA algorithm gives time for the links achieve stability.

3.5.2 Experiment Setup

A layout exactly similar to the one from section 3.1.1 with similar parameters was used in this case to show the benefits of load balancing in similar environments. All sensor nodes in the network were programmed with the code to incorporate load balancing in their routing strategies. The experiment with load balancing was run several times for same sets of layouts as described before. Table 3.4 shows the experiment results.

An observation of data forwarding rate, routing metric (ETX) and the variance in forwarding rates gives a clear idea of the balance in traffic shared between the critical nodes in the network. Another look at equation (3.2) proves that as the data forwarding rate is distributed among the critical nodes, the average number of descendants per node *D*, drops significantly which results in a lower average current consumption per node.

Figure 3.7 shows a graph of comparison of network lifetimes calculated using equation 3.2. For a network of 17 nodes and a sampling interval of 15 seconds, the average network lifetime showed an increase of,

7.94 days = 190.56 hours

A graph of variances of packet forwarding rates with and without load balancing was also plotted as shown in Figure 3.8. The variance of packet forwarding rates without load balancing is seen to very high compared to the runs with the load balanced algorithm. A

higher variance of packet forwarding rate indicates greater imbalance in the network. Load balanced routing is seen to distribute traffic among all nodes which may not increase the lifetime of individual nodes but increases the overall lifetime of a network.

Table 3.4: Data Traffic and Network Lifetime with Load Balancing

Run #	Critical node Ids	ETX metric	Data Rate	Lifetime (Number of days)	Network Lifetime	Variance (Data rate)
1.1.1	5	22	16	49	47	92.6
	6	11	0	65		
	7	30	23	47		
1.1.2	5	26	18	49	49	68.22
	6	10	0	65		
	7	22	17	60		
1.2.1	2	18	10	56	49	6
	3	24	16	49		
	21	31	13	53		
1.2.2	2	14	5	60	49	30.88
	3	22	15	53		
	21	22	18	49		
2.1.1	2	26	22	47	47	0
	3	30	22	47		
2.1.2	2	30	24	44	44	0.25
	3	30	23	47		
2.2.1	8	38	28	42	42	0.25
	9	37	27	44		
2.2.2	5	27	12	53	49	9.55
	8	22	13	53		
	25	32	19	49		
3.1.1	2	10	2	65	44	94.88
	5	26	24	44		
	6	38	21	47		
3.1.2	2	10	3	65	44	76.22
	5	23	17	49		
	6	30	24	44		
3.2.1	3	14	10	56	53	3.5
	8	18	13	53		
	21	27	8	56		
	22	22	9	56		

Table 3.4 (continued)						
3.2.2	3	10	4	60	49	20
	8	18	12	53		
	21	22	16	49		
	22	14	8	56		

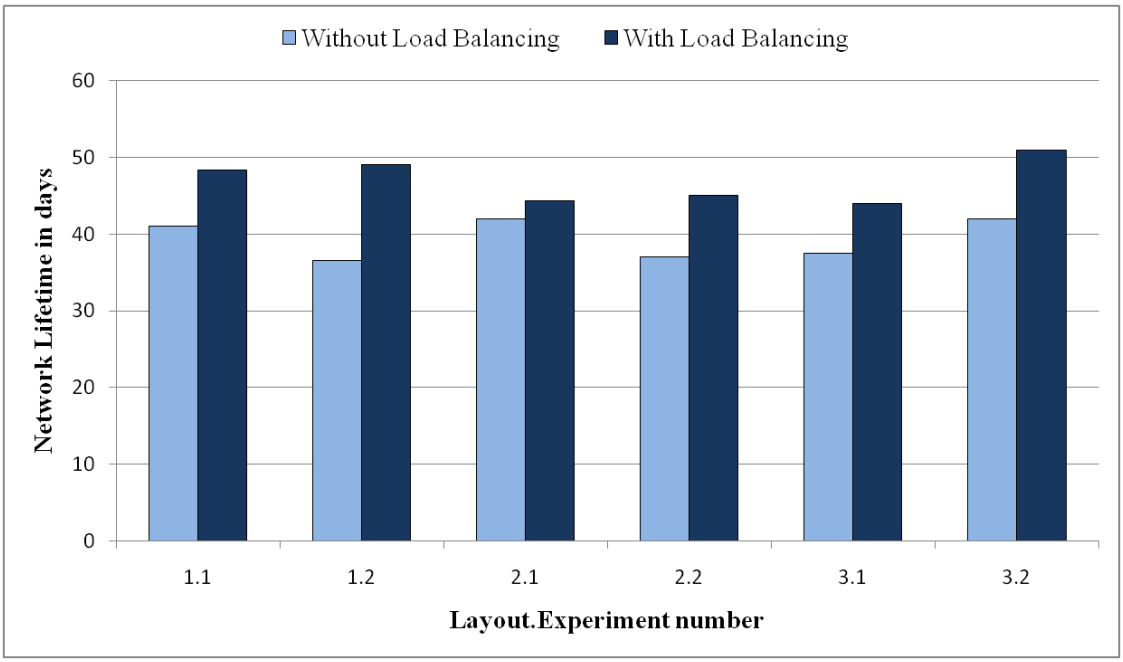


Figure 3.7: A comparison of average network lifetimes with and without load balancing

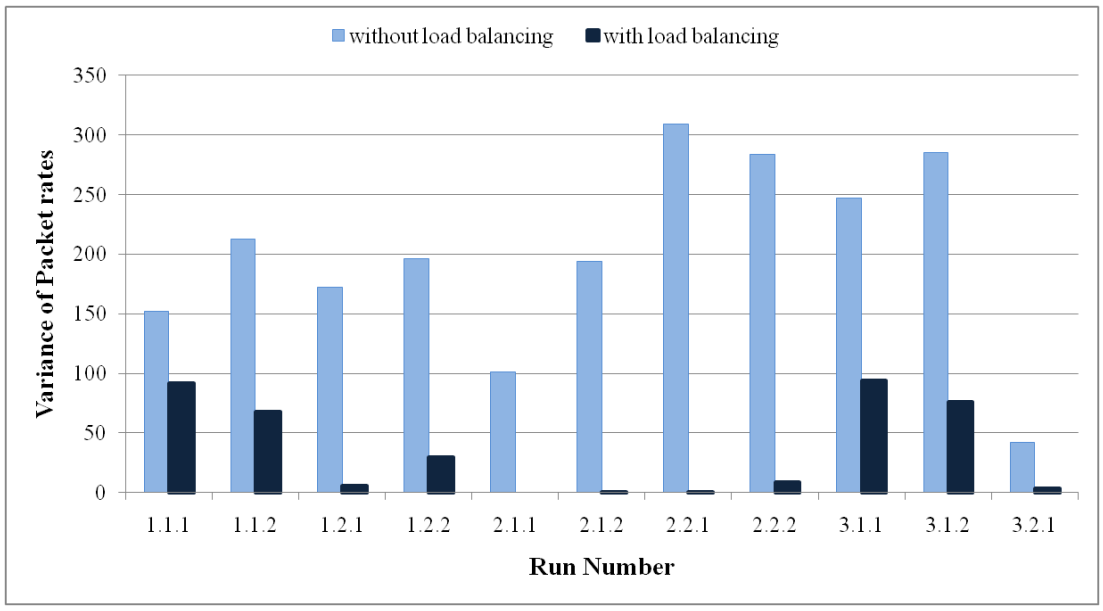


Figure 3.8: A graph showing variance in packet forwarding rates for each run

CHAPTER 4: ENERGY AWARE ROUTING

Using a low energy path to forward data packets frequently leads to energy depletion of nodes along that path and may lead to undesirable effects like network partitioning. Such problems are generally addressed by routing protocols that consider available energy resources in the nodes along the path referred to as *energy aware routing* [16, 17, 18]. The basic principle behind energy aware routing is to modify the routing metric according to real time energy of sensor nodes so that routing is discouraged over nodes with lower energy. This would ensure that the optimal path does not get depleted and network degrades gracefully rather than being partitioned.

The source of energy for sensor nodes is its battery. Most modern day sensor nodes operate on alkaline, lithium ion or nickel-metal hydride (NiMH) batteries having capacities ranging from 1500mAh to 5000mAh. Some batteries are connected in series to provide an optimum voltage for operations. Apart from mote radio communications and processors, the battery capacity depends on other factors such as

- Battery discharge rate
- Battery chemistry which determines its state of charge (SOC)
- Ambient operating temperature

Since the battery capacity depends mainly on the discharge rate, it is very essential to consider its effects in detail.

4.1 Discharge Profile of a Typical Alkaline Battery

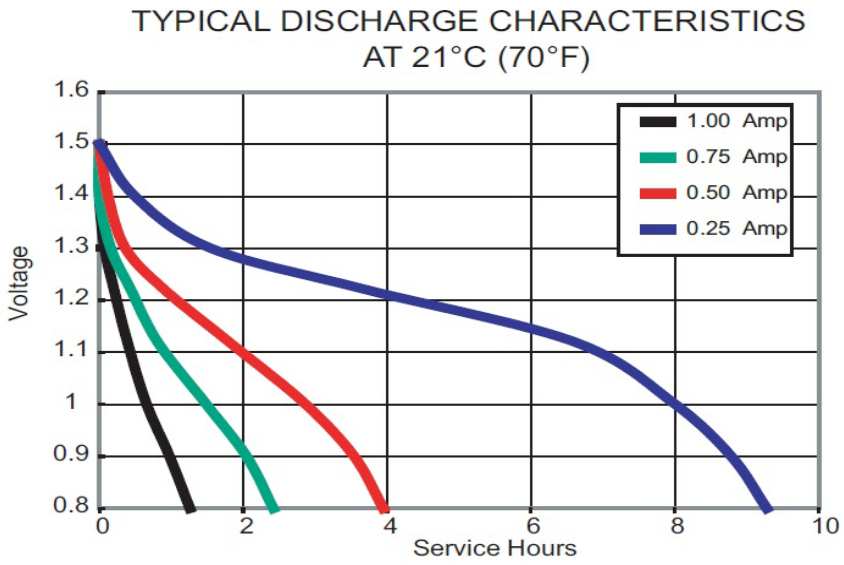


Figure 4.1: Battery discharge characteristics which show rate dependent capacity. (Source: Duracell MN1500 Alkaline battery Datasheet)

Theoretical capacity of a battery is the maximum amount of charge that can be extracted from a battery based on the amount of active material it contains [15]. A typical battery discharge behavior is sensitive to numerous factor which including the discharge rate, temperature, and number of charge-recharge cycles. Consequently, battery discharge behavior deviates significantly from the behavior of an ideal energy source.

Battery capacity decreases as discharge rate increases. As shown in Figure 4.1, as the discharge current increases, the battery voltage reaches its cut-off point at a time earlier. This mainly results due to the electrochemical processes inside a battery; a higher discharge rate accelerates the diffusion process inside the battery creating a lag between reaction and diffusion rates.

MICAz sensor nodes operate in a voltage range of 2.7V to 3.3V [22] with 2.7V being the cut-off voltage. A very high packet forwarding rate will lead to higher discharge rates and rapid depletion in battery capacity resulting in a drop in battery voltages rather sooner than expected. Since critical nodes near the base station are subject to high traffic

rates, such effects result in the death of these critical nodes. All data traffic from this node will then be shifted to another node according to mesh network operations and this would again result in capacity depletion of other critical nodes in the network. This effect is less pronounced in sensor nodes because of low discharge currents; however, it is important to balance network traffic according to relative battery capacities of neighboring nodes.

4.2 Determining Battery State of Charge (SOC)

State of Charge (SOC) is defined as the available capacity expressed as a percentage of some reference, sometimes its rated capacity. The preferred SOC reference should be the rated capacity of a new cell rather than the current capacity of the cell because the cell capacity gradually reduces as the cell ages [23].

There are several methods of estimating the state of charge of a battery. Some are specific to particular cell chemistries. Most depend on measuring some convenient parameter which varies with the state of charge.

- *Coulomb counting* is a method of *current* based SOC estimation. The energy contained in an electric charge is measured in *Coulombs* and is equal to the integral over time of the current which delivered the charge. The remaining capacity in a cell can be calculated by measuring the current charging or discharging the cells and integrating this over time. In other words the charge transferred in or out of the cell is obtained by accumulating the current drain over time. The calibration reference point is a fully charged cell, and the SOC is obtained by subtracting the net charge flow from the charge in a fully charged cell. This method, known as *Coulomb counting*, and provides higher accuracy than most other SOC measurements since it measures the charge flow directly.

Three current sensing methods may be used;

- *Current shunt* - The simplest method of determining the current is by measuring the voltage drop across a low resistance value, high precision, series, sense resistor between the battery and the load known as a current shunt. This method of measuring current causes a slight power loss in the current path, heats up the battery and is inaccurate for low currents and hence this method is not suitable for capacity determination in MICAz motes.
- *Hall Effect transducers* avoid the above problem, but they are more expensive. They also cannot tolerate high currents and are susceptible to noise.
- *GMR magneto-resistive sensors* have higher sensitivity and provide a higher signal level. They also have better high temperature stability than Hall Effect devices. But these sensors are even more expensive and difficult to accommodate on sensor nodes.

- *Impedance measurements*: During the cell charge-discharge cycles the composition of the active chemicals in the cell changes as the chemicals are converted between the charged and discharged states and this will be reflected in changes to the cell impedance. However impedance is difficult to measure while the cell is active as well difficult to interpret since impedance is temperature dependent.
- *Charge estimation algorithms* such as Fuzzy logic, Kalman filter and Neural networks are also used to determine SOC; but incorporating these complex algorithms in the already memory constrained sensor nodes is difficult and hence not practically implemented.
- *Voltage based SOC estimation*: Measuring state-of-charge by voltage is the simplest

method, but it can be inaccurate. Cell types have dissimilar chemical compositions that deliver varied voltage profiles. Temperature also plays a role. Higher temperature lowers the open-circuit voltage, a lower temperature raises it, and this phenomenon applies to all chemistries in varying degrees.

MICAz motes used in the Paradise network have an internal voltage reference that can be used to measure the terminal voltage using the *VoltageM* module provided by TinyOS [22]. All motes in the network are exposed to similar physical environmental conditions and have fairly the same current draw under standard conditions. If a relation can be established between the battery voltage and its capacity, it can be applied to all nodes in the network.

4.3 Relation between Battery Voltage and Capacity

4.3.1 MICAz battery voltage monitor

The eight-channel ADC on the ATMega128L processor can be used along with the internal voltage reference to measure battery voltage (V_{batt}) by wiring the TinyOS component *VoltageM.nc* to the application [22]. In order to track the battery voltage, the precision voltage reference is monitored to determine the ADC full-scale voltage span which corresponds to V_{batt}. The battery voltage (V_{batt}) can be computed from the ADC reading by:

$$V_{batt} = V_{ref} \times \frac{ADC\_FS}{ADC\_Count} \tag{4.1}$$

where:

ADC_FS = 1024

V_{ref} = Internal voltage reference = 1.223 volts

ADC_Count = Data from the ADC measurement of internal voltage reference

4.3.2 Battery voltage and Percent Capacity

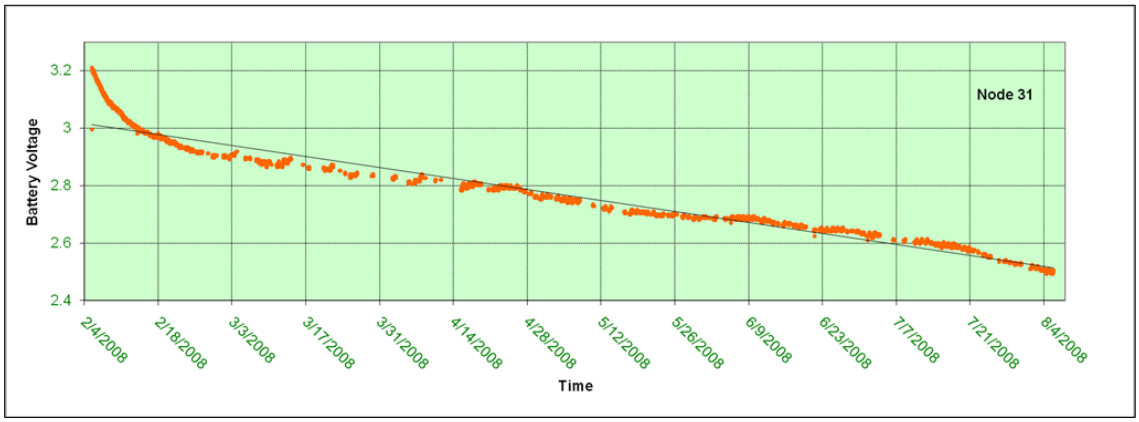


Figure 4.2: Battery discharge characteristic of a node in Paradise network

Figure 4.2 shows a battery discharge curve for a node in the Paradise network. All nodes in the network are exposed to similar physical conditions. Nodes deployed for one particular application, for example temperature sensing, have the same data rate, route update intervals and hence, have similar current consumption characteristics. From this fact it can be concluded that battery capacity will be fairly same for nodes with identical drop in battery voltages. Estimating *residual capacity* using battery voltage can be inaccurate. Hence, we coin the term *Percent Capacity* which is a better indicator of residual capacity corresponding to battery voltages.

For battery voltage V_{batt} = 3V, the *percent capacity is* assigned to be 100% and at a voltage slightly lower than the cut-off voltage of V_{batt} = 2.6V, the percent capacity is 0%. Even though the *actual capacity* of the battery at 2.6V is not 0mAh, it is assumed to be 0% as it is unusable if the battery voltage drops below 2.7V. Using the data from the graph of Figure 4.2 as a reference, another graph of *percent capacity* Vs *battery voltage* using ADC readings was plotted.

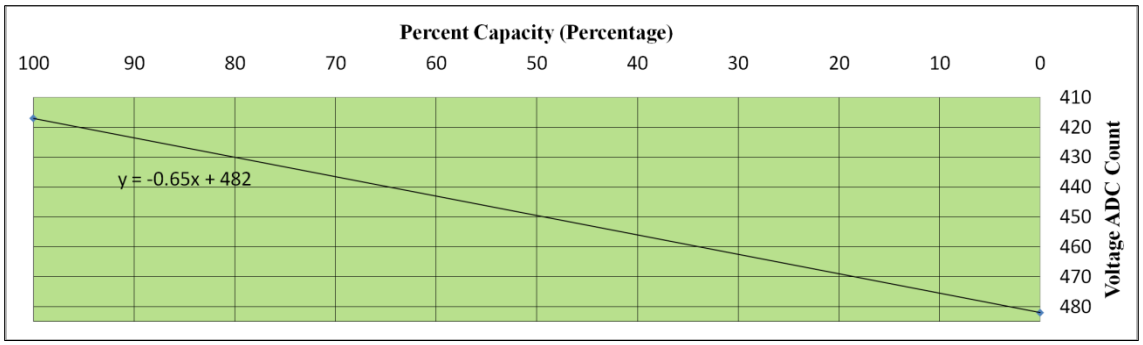


Figure 4.3: A graph of Percent Capacity Vs Voltage ADC count

From the ADC count, the battery voltage corresponding to that count was calculated using equation (4.1) as shown in Table 4.1.

Table 4.1: ADC readings and its corresponding battery voltage

ADC count	Battery Voltage (mV)
417	3000
482	2600

A linear equation was fitted for the graph shown in Figure 4.3 as:

$$PercentCapacity = \frac{(482 - VoltageADC)}{0.65} \tag{4.2}$$

4.4 Energy Aware Routing Algorithm

We propose to use the relation between battery voltage and *percent capacity* as the basis for energy aware routing in sensor networks. To implement this, we modify the routing metric, ETX inversely to the percent capacity. To maintain simplicity of protocol and also minimize mathematical operations and save processor cycles, a simple constant of proportionality is chosen.

$$ETX \propto 1/percentcapacity$$

$$ETX_{Mod} = ETX_{init} + 1000/percentcapacity \tag{4.3}$$

Where, ETX_{Mod} and ETX_{init} are the value of modified ETX value and the initial ETX respectively. E_{init} is the value calculated by the underlying four bit link estimator [10].

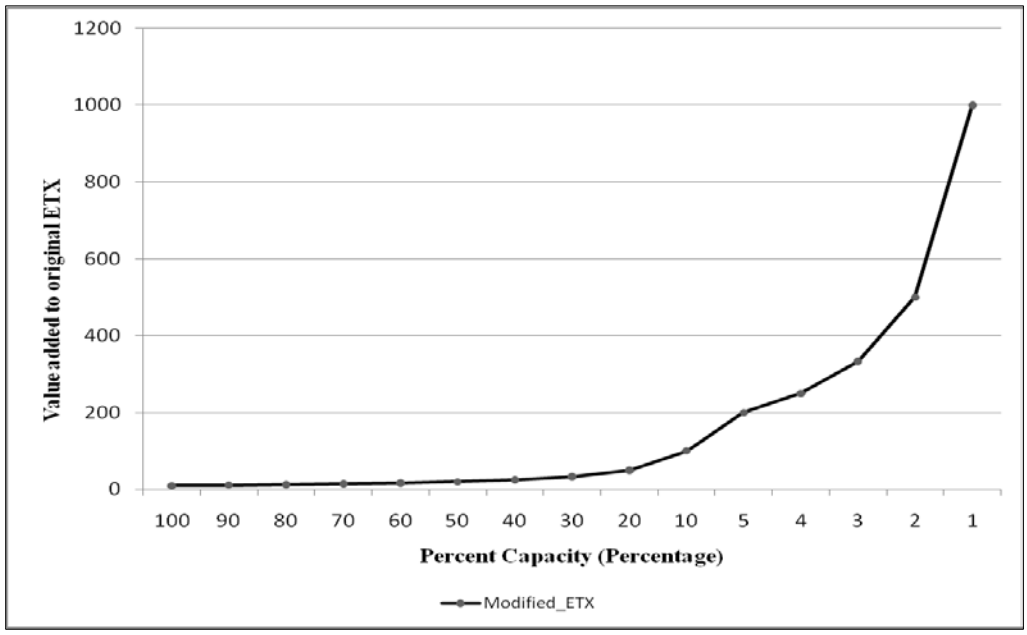


Figure 4.4: ETX values modified according to percent capacity

The energy aware routing pseudo code is shown below.

```
begin
voltage_adc = Get the voltage value from ADC
if voltage_adc >= 482
        voltage_adc = 481            /* If voltage drops below 2600mV */
endif
percent_capacity = (482 – voltage_adc) / 0.65      /* Calculate the percent capacity */
modified_etx = modified_etx + (1000 / percent_capacity)
end
```

4.5 Experiment and Simulation Results

4.5.1 Experimental Demonstration of Energy Aware Routing

An experiment was carried out to practically demonstrate routing in sensor nodes according to changes in battery voltage. Two sensor nodes were chosen to operate as

critical nodes, which would be responsible to forward data traffic to the base station. The two critical nodes were connected to a *variable power supply* to vary the voltage levels to simulate practical operating scenarios. The critical nodes are initially maintained at a voltage level of 3000 mV. Sensor nodes were randomly deployed around the critical nodes for them to autonomously form a mesh network as shown in Figure 4.5 (a). The figure shows deployment of nodes at uniform distances only for diagrammatic representation purpose. The supply voltage of C1 was then progressively decreased in steps of 100 mV with the following observations:

- The voltage of critical node C1 is decreased to 2900mV.

  V_{batt} = 2900mV = 431 ADC counts

  Percent Capacity = 79 %

  From the *percent capacity* a value of *13* is added to the original ETX value as per the energy aware algorithm. This causes a cumulative increase in the ETX metric resulting in at least 2 nodes changing their parent to C2 which is maintained at 3000mV.

- A decrease of 100mV results in a further increase of the ETX value of node C1 and another two descendants changing their parent, as a comparison of the ETX values of C1 and C2 now gives a better ETX value of C2 than C1.

- If the voltage is decreased to 2650mV which is slightly below the cut-off voltage for MICAz motes, the ETX increases exponentially as shown in Figure 4.4 which discourages any node from hopping over C1, which implies that energy aware routing works in a way to keep all sensor nodes operational for longer amounts of time by choosing parents with sufficient energy.

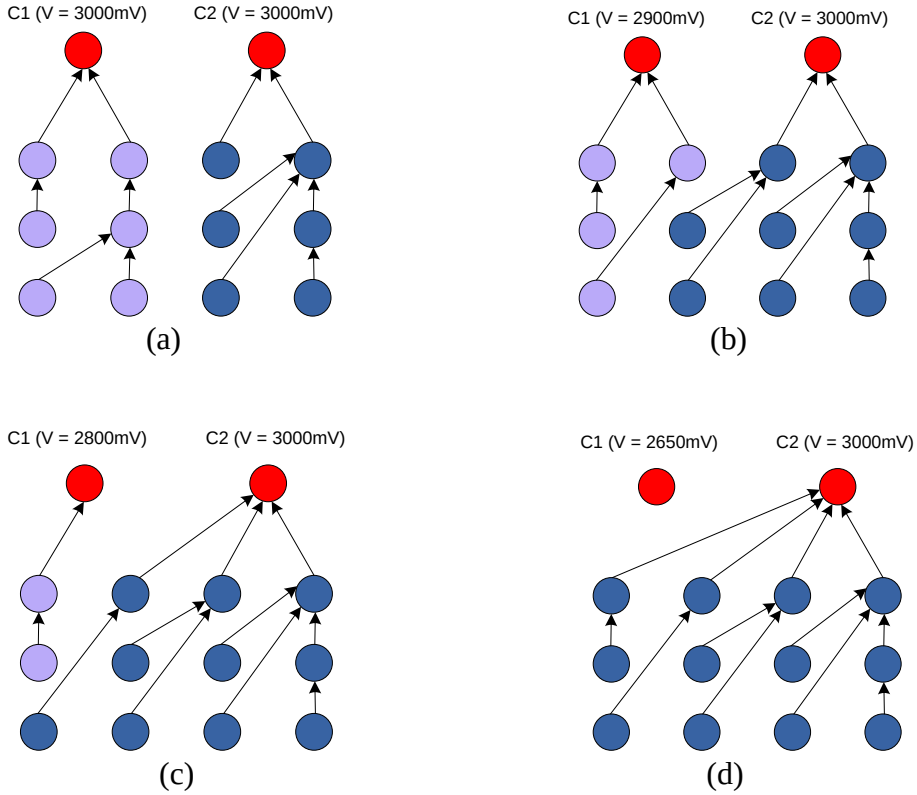


Figure 4.5: Energy Aware Routing: Change in routes according to changes in battery voltage. Nodes represented in red are critical nodes. (a) At V_{C1} = 3000mV (b) At V_{C1} = 2900mV (c) At V_{C1} = 2800mV (d) At V_{C1} = 2650 mV

4.5.2 Simulation Results for Energy Aware Routing

To visualize the advantages of energy aware routing, a simulator was designed in C which would route packets to the base station depending on the *percent capacities* of the sensor nodes. Two nodes were chosen as *critical nodes* and assigned initial capacities of 5000 mAh. For simulation purpose, it is assumed that all nodes will completely utilize their battery capacity before they stop operating.

- The network size which is the number of nodes in the network is a user input value. Each node is assigned a metric between 10 and 100 using a random number generator. The metric is bi-directional, that is, one sensor node will be assigned routing metrics to indicate link qualities to both critical nodes. Depending on this

assignment, each node in the network will choose a parent with the lowest bi-directional metric according to protocol routing. The sampling interval was set to 15 seconds and the route update interval was set to 60 seconds. Since the network layout is random, all nodes will be distributed to both critical nodes in an uneven manner with one critical node having more descendants than the other. To calculate current consumption, the standard current equation (3.2) for sensor nodes is used.

- As the simulation is run, current consumption of each node will depend also on the factor *D,* which is the number of descendants for packet forwarding. If the simulation is run without energy aware routing, the *critical node* having more number of descendants will run out of battery capacity earlier than the other. The time for the first critical node to stop operating is the *network lifetime*.
- In case of energy aware routing, each critical node was assigned a routing metric which changes according to equation (4.3). As the battery capacity decreases, the ETX value increases in inverse proportion. When the ETX value crosses the *parent switch threshold*, the nodes would switch routes to the critical node with higher battery capacity. This cycle continues until the capacities of both batteries reach zero at the same time. This approach balances network traffic according battery capacities and thus increases network lifetime.

The parameters used for the simulation and the results are shown in Table 4.2. An increase in network lifetime using energy aware routing can be clearly seen from the graph in Figure 4.7. It can also be observed that the difference between network lifetimes with and without energy aware routing decreases as the network size increases. This is because as network size increases, network traffic overheard by each sensor node

increases which increases the overall current consumption.

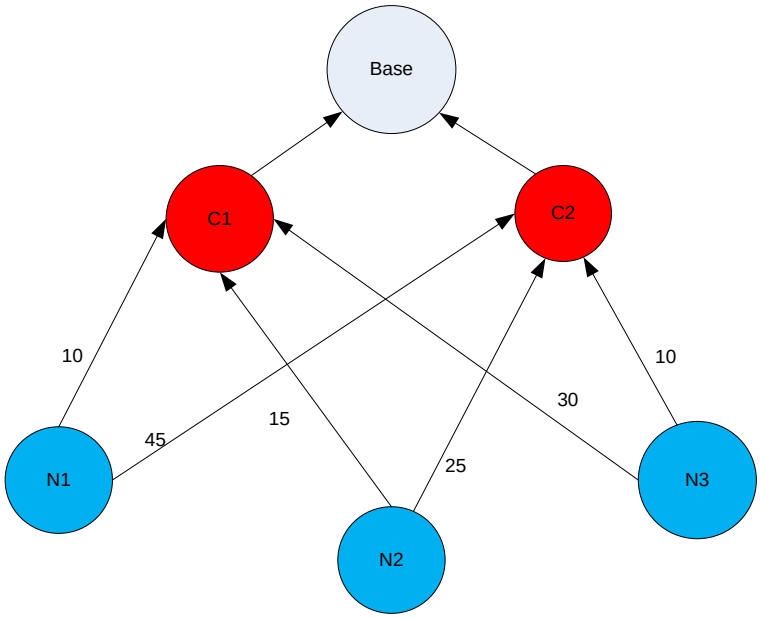


Figure: 4.6: Random assignment of ETX value in the energy aware routing simulator

Table 4.2: Network lifetime with and without energy aware routing

Number of nodes	Network Lifetime – CTP (*hours)*	Unused battery capacity – CTP (mAh)	Network Lifetime - Energy Aware Routing (*hours)*
10	372	1302.9	409
15	271	1186.11	299
20	189	1657.44	213
25	164	1289.61	178
30	124	1957.16	143
35	110	1837.06	126
40	99	1557.01	111
45	90	1495.24	101
50	79	1535.85	90

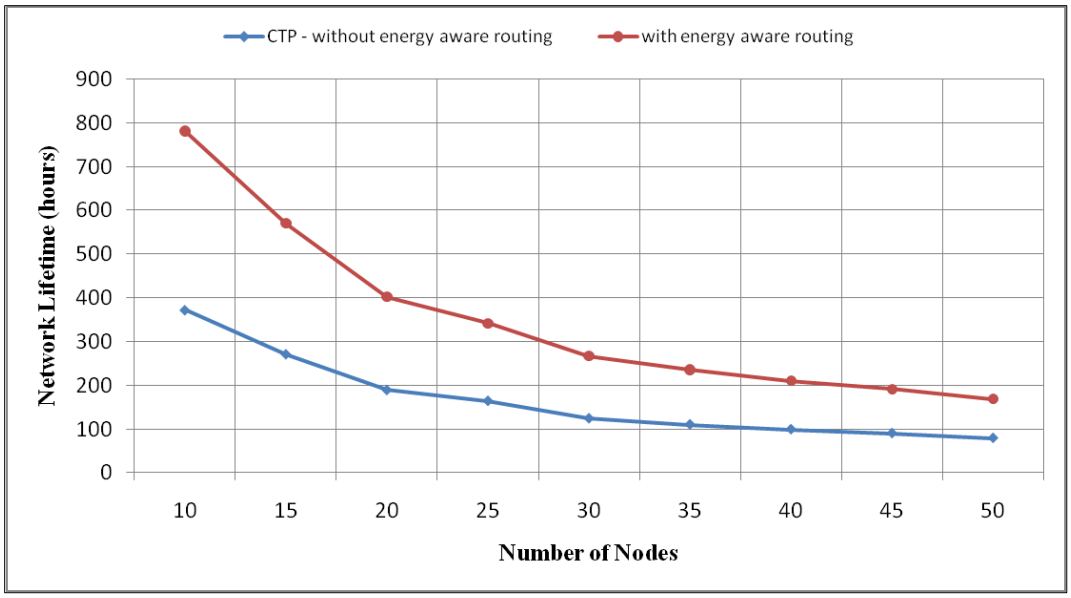


Figure 4.7: A graph of network lifetimes – with and without energy aware routing

CHAPTER 5: DESIGN OF A HYBRID ROUTING PROTOCOL

The load balanced routing protocol is seen to increase the lifetime of a network by balancing the data traffic among all nodes in the network. Simulations of energy aware routing protocol also show an increase in network lifetime when compared to the traditional Collection Tree protocol. We now present a hybrid routing protocol that advocates an integration of the concepts of *traffic load balancing* and *energy awareness* in each sensor node while deciding routes. The need for a hybrid protocol is further explained in section 5.1.

5.1 The Need to Integrate Load Balancing and Energy Awareness

The proposed load balanced routing protocol which was discussed in Chapter 3 employs real time data forwarding traffic to modify the routing metric but does not take into account the energy of each node in the network. In the same way, the energy aware routing protocol proposed in Chapter 4 involves monitoring the battery voltage to determine routes, but may distribute the data traffic in an uneven manner. This motivates the integration of load balancing and energy awareness in a routing framework.

An experiment was carried out to show the traffic imbalance in the network if only the battery voltage was considered. A mesh network was deployed in the lab where three sensor nodes were chosen to behave as *critical nodes* in the network. Two nodes were connected to a variable power supply to vary the battery voltage in order to observer a change in the routing metrics as shown in Figure 5.1.

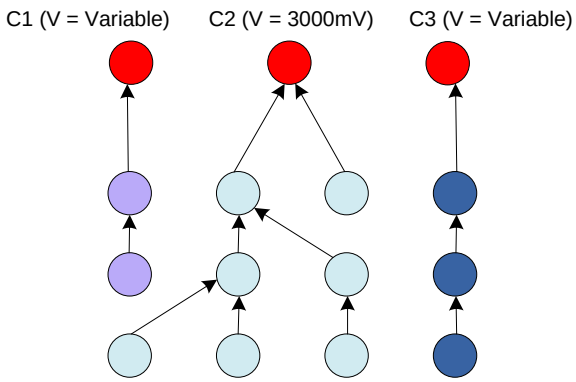


Figure 5.1: Experiment setup for the energy aware routing protocol. Critical nodes C1 and C3 are connected to a variable power supply.

All critical nodes were initially maintained at a voltage level of 3000mV and the effect of load imbalance while using energy aware routing was observed by varying the supply voltage in two of the critical nodes. Without load balanced routing, all sensor nodes autonomously discovered routes to form a mesh network which was unbalanced in terms of data traffic. When the voltage level of node C1 was lowered from 3000mV to 2700mV in steps of 100mV all descendants of C1 changed their parent to either C2 or C3. But this change was highly uneven as the choice of parent was based only in terms of link quality. For subsequent runs, supply voltage for node C3 was reduced keeping voltages of C1 and C2 constant to gather results. The results are tabulated in Table 5.1.

It can be clearly seen from the results in Table 5.1, that though the network reacts well to adaptively change routes according to changes in supply voltage, the change observed in forwarding traffic is not *evenly* distributed as opposed to load balanced routing. This form of network operation, though energy aware, will still result in creation of *hot-spots* in the network resulting in increased energy usage in some sensor nodes. Hence, it is very important to incorporate routing strategies of both, *energy aware* as well

as load balanced routing to counter problems that might arise due to application of the protocols separately.

Table 5.1: Experimental results of Energy Aware routing with three critical nodes

Run #	Critical node voltages (mV)	ETX metric	Packet forwarding rate (packets per minute)
1	C1 = 3000 C2 = 3000 C3 = 3000	45 40 43	8 26 10
2	C1 = 2900 C2 = 3000 C3 = 3000	50 40 41	6 29 9
3	C1 = 2800 C2 = 3000 C3 = 3000	53 40 41	4 30 10
4	C1 = 2700 C2 = 3000 C3 = 3000	98 40 44	1 35 9
5	C1 = 3000 C2 = 3000 C3 = 2900	42 41 49	4 37 5
6	C1 = 3000 C2 = 3000 C3 = 2800	51 43 57	12 30 3
7	C1 = 3000 C2 = 3000 C3 = 2700	51 40 107	15 12 0
8	C1 = 3000 C2 = 3000 C3 = 3000	42 43 54	20 8 10

5.2 The Hybrid Routing Algorithm

The hybrid routing algorithm integrates routing strategies of *load balancing* as well *energy awareness* to exploit the advantages of both. The hybrid routing algorithm will result in graceful degradation of the network as the sensor nodes are programmed to choose routes with lower data traffic and a path which has higher energy that the other.

The pseudo code for the hybrid routing algorithm is as follows:

begin
previous_etx = modified_etx /* store the old ETX value */
actual_etx = Value of ETX from the Link Estimator
modified_etx = actual_etx + tx_rate /* Modify ETX according to traffic */
voltage_adc = Get the real-time voltage
if *voltage_adc > 482*
voltage_adc = 481 /* If voltage drops below 2600 mV */
endif
percent_capacity = (482 – voltage_adc) / 0.65
modified_etx = previous_etx + (1000 / percent_capacity) /* Modify ETX according to battery voltage */
*modified_etx = α * modified_etx + (1-α) * previous_etx* /* EWMA Algorithm to smooth out the estimate */
end

As shown in the pseudo code, the *actual_etx* value, which is derived from the underlying link estimator, is modified by adding the *tx_rate* which represents the traffic load on the sensor node. This *modified_etx* value is further modified by adding the effects of depletion of battery voltage derived from the internal voltage interface. The final *modified_etx* is further smoothened out by applying the EWMA algorithm to prevent oscillations.

5.3 Results

To observe the viability of the hybrid routing concept, an experiment was carried out which involved a setup similar to the one used in section 5.1. The only difference was that, the hybrid routing protocol was used in this case. The experiment was carried out in three phases. In the table *x.y* indicates a phase *x* and varying voltage readings *y*. The different phases of the experiment are explained as follows:

Phase 1: All the critical nodes are initially maintained at a voltage level of 3000 mV. The load balancing algorithm works in a way to balance the data traffic among all critical nodes in the network (Run # 1.1).

Phase 2: This phase involves decreasing the voltage of node C1 from 3000 mV in steps of 100 mV to 2700 mV. As the voltage level of node C1 decreases, we observe that application of *energy aware routing* algorithm increases the ETX value of C1 which discourages other nodes in the network from choosing C1 as their parent and the *load balancing* algorithm works in a way to balance this difference in data traffic among nodes C2 and C3 (Run # 2.1, 2.2 and 2.3).

Phase 3: Voltage of nodes C1 and C3 were decreased to a minimum of 2700 mV. This phase shows the working of *energy aware routing* as all nodes hop over node C2 since it has the maximum voltage and consequently higher energy (Run # 3.1, 3.2).

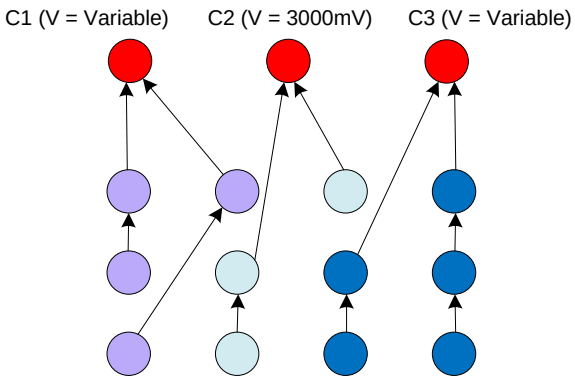


Figure 5.2: Experiment setup for demonstrating the hybrid routing protocol. Nodes C1 and C3 are connected to a variable power supply.

Phase 4: This phase involved keeping the voltage level of node C3 constant at 2700 mV while increasing the voltage of node C1 to 3000 mV which is equal to voltage of node C2. This phase again demonstrated *load balancing* as it equally distributes data traffic among nodes C1 and C2.

Phase 5: This phase involves bringing the voltages of all nodes to 3000 mV again, which proves that when the energy levels of nodes are equal, *load balancing* is more proactive as it helps maintain the balance in data traffic across all neighboring nodes whereas in case of unequal energy levels in nodes, *energy aware algorithm* works in a way to balance them by directing more traffic to the nodes with higher energy.

Table 5.2: Experimental results of the hybrid routing protocol

Run #	Critical node voltages (mV)	ETX metric	Packet forwarding rate (packets per minute)
1.1	C1 = 3000 C2 = 3000 C3 = 3000	51 53 55	17 14 20
2.1	C1 = 2900 C2 = 3000 C3 = 3000	51 50 51	9 16 20
2.2	C1 = 2800 C2 = 3000 C3 = 3000	53 54 55	3 16 22
2.3	C1 = 2700 C2 = 3000 C3 = 3000	100 54 55	1 18 23
3.1	C1 = 2700 C2 = 3000 C3 = 2800	100 62 69	0 26 7
3.2	C1 = 2700 C2 = 3000 C3 = 2700	100 74 93	0 39 0
4.1	C1 = 2800 C2 = 3000 C3 = 2700	59 54 106	10 20 0
4.2	C1 = 3000 C2 = 3000 C3 = 2700	49 46 107	15 12 0
5.1	C1 = 3000 C2 = 3000 C3 = 3000	42 43 54	12 12 14

As seen from the results recorded in Table 5.2, all sensor nodes in the network would choose their routes to balance data forwarding traffic as well as battery capacity among neighboring nodes. The application of the combined strategies of load balancing and energy routing would not only avoid formation of *hot-spots* in the network but would also work in a way which would result in graceful degradation of the network.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The main challenge in sensor networks is to provide services in the face of severe resource constraints, most importantly that of energy. The Paradise substation monitoring project was a unique application designed to continuously monitor the health of high voltage equipments found in electric substations. Though the network, designed with the *XMesh* routing protocol gave fair results, certain problems of imbalance in data traffic load and battery usage was found in different part of the network as explained in Chapter 2; which formed the motivation for this thesis.

The potential problems due to load imbalance and uneven battery usage were proved by experiments carried out in the laboratory. A solution to design a load balanced and energy aware routing protocol was proposed to increase the longevity of the network. Collection Tree protocol (CTP) which works on the TinyOS-2.1.1 platform was chosen as the mesh networking protocol because of its features like an efficient link estimator, adaptive beaconing, low-power listening and its open-source nature. Two separate algorithms; one which considers real time data traffic load on each sensor node and the other which considers remaining battery capacity based on voltage were designed and implemented. Laboratory experiments and simulation results showed significant increase in the overall network lifetime in comparison to the traditional tree routing protocols. The

final phase involved designing a hybrid routing algorithm which would have advantages of both; the load balanced and energy aware routing algorithms.

6.2 Future Work

The inclusion of real time data traffic and battery voltage status to calculate the routing metric proves to be effective to facilitate graceful degradation of the network. However, simulations showed that a significant amount of power was being consumed in overhearing data and routing packets of neighboring nodes. Since all MICAz motes in the Paradise network are programmed to operate at maximum radio output power, one node can overhear packets from many other nodes in the network. This significantly increases current consumption and reduces battery lifetime in a large scale network.

A solution to counter this problem could be, dynamically controlling the radio power for transmitting packets. The principle behind this method is that, any two nodes which are at a very short distance from each other can communicate using minimum radio power. Thus lowering the radio power for all communications can reduce current consumption significantly. The radio power control abstraction on top of the load balanced and energy aware algorithm can achieve a considerable increase in network lifetime.

REFERENCES

[1] A. Nasipuri, R. Cox, H. Alasti, L V. Zel, B. Rodriguez, R. McKosky, and J. Graziano, "Wireless Sensor Network for Substation Monitoring: Design and Deployment," Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys'08), pages 365-366, November 5-7, 2008.

[2] A. Nasipuri, R. Cox, J. M. Conrad, L. Van der Zel, B. Rodriguez, and R. McKosky, "Design considerations for a large-scale wireless sensor network for substation monitoring," Proceedings of IEEE SenseApp'10, pages 882-889, October, 2010.

[3] TinyOS, An open source operating system for sensor networks, http://www.tinyos.net

[4] TinyOS documentation wiki, http://docs.tinyos.net/

[5] Crossbow Technology, Manufacturer of MICAz motes, sensor and data acquisition boards, mote interface boards, http://www.xbow.com/.

[6] Philip Levis, David Gay, TinyOS Programming, Cambridge University press, 2009.

[7] Intel Research Mote, Intel Corporation Research, http://webs.cs.berkeley.edu

[8] O. Gnawali, R. Fonesca, K. Jamieson, D. Moss, P. Levis, "Collection Tree Protocol," Proceedings of the 7th ACM Conference on Embedded Networked Sensors, 2009.

[9] Tinyos 2 tep 123: The collection tree protocol. http://www.tinyos.net/tinyos-2.1.0/doc/html/tep123.html

[10] R. Fonseca, O. Gnawali, K. Jamieson, P. Levis, "Four Bit Wireless Link Estimation," In Hotnets-VI, Atlanta, Nov. 2007.

[11] P. Levis, N. Patel, D. Culler, S. Shenker, "Trickle – A self-regulating algorithm for code maintenance and propagation in sensor networks," 2004.

[12] XMesh Users Manual, 2005-2007 Crossbow Technology, Inc.

[13] MultiHopLQI. http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI, 2004.

[14] R. Baumann, S. Heimlicher, M. Strasser, "A Survey on Routing Metrics," Andreas Weibel Computer Engineering and Networks Laboratory ETH-Zentrum, 2007.

[15] R. Rao, S. Vrudhula, D N.Rakhmatov, "Battery Modeling for Energy-Aware System Design," IEEE Comput., vol 36, pp. 77-87, 2003.

[16] R C. Shah and J. Rabaey, “Energy Aware Routing for Low Energy Ad Hoc Sensor Networks,” IEEE WCNC, pp. 17-21, 2002.

[17] L. Lin, N B. Shroff, R. Srikant, “Energy-aware Routing in Sensor Network: A Large System Approach,” Science Direct, 2007.

[18] T. Yang, Y K. Toh, “Run-time Monitoring of Energy Consumption in Wireless Sensor Networks,” IEEE Conference on Control and Automation, 2007.

[19] MICAz Datasheet. MICAz – A Wireless sensor mote manufactured by Crossbow Technology, Inc. Source: http://www.memsic.com.

[20] MDA300: Data Acquisition Board. http://www.xbow.com/Products/

[21] MTS300/310: Sensor Module. http://www.xbow.com/Products/

[22] Crossbow MPR/MIB User Manual, Revision B, June 2006.

[23] Battery State of Charge Determination. http://www.mpoweruk.com/soc.htm

[24] A A. Al-Al-Shaik, I M. El-Amin, “Substation Maintenance Practices in a Saudi Electric Utility,” 6th Saudi Engineering Conference, 2002.

[25] Trafag, Gas Density Controls. http://www.granzow.com/gasdensitycontrols/8774/.

[26] Xioyan Hong, M. Gerla, R. Bagrodia, “The Mars Sensor Network: Efficient and Power Aware Communications,” IEEE MILCOM 2001, Oct 2001.

[27] G.Werner-Allen, J. Johnson, “Monitoring volcanic eruptions with a wireless sensor network,” in Proceedings of the second European workshop on WSNs, 2005.

[28] A. Mainwaring, J. Polastre, D. Culler, “Wireless sensor networks for habitat monitoring,” First ACM workshop on WSNs, 2002.