ENERGY CONSUMPTION IN WIRELESS SENSOR NETWORKS USING GSP

by

María Gabriela Calle Torres

Electronics Engineer, Universidad Pontificia Bolivariana, Medellín, Colombia, 1995

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Master of Science in Telecommunications

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This thesis was presented

by María Gabriela Calle Torres

It was defended on

April, 2006

and approved by

Richard Thompson, Telecommunications Program Director

Prashant Krishnamurthy, Associate Professor, Telecommunications Department

Thesis Director: Joseph Kabara, Assistant Professor, Telecommunications Department

ENERGY CONSUMPTION IN WIRELESS SENSOR NETWORKS USING GSP

María Gabriela Calle Torres, M.S.

University of Pittsburgh, 2006

The energy consumption rate for sensors in a wireless sensor network varies greatly based on the protocols the sensors use for communications. The Gossip-Based Sleep Protocol (GSP) implements routing and some MAC functions in an energy conserving manner. The effectiveness of GSP has already been demonstrated via simulation. However, no prototype system has been previously developed. GSP was implemented on the Mica2 platform and measurements were conducted to determine the improvement in network lifetime. Results for energy consumption, transmitted and received power, minimum voltage supply required for operation, effect of transmission power on energy consumption, and different methods for measuring lifetime of a sensor node are presented. The behaviour of sensor nodes when they are close to their end of lifetime is described and analyzed. A comparison with other models for energy consumption is made and suggestions for future work are presented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank first and foremost to God, who has been my strength during all my life. Also, I would like to thank my advisor, Dr. Joseph Kabara, for all his guidance and help during this project. I want to say thank you to Dr. Richard Thompson who, jointly with Fulbright, Laspau and Universidad del Norte, made possible for me to study in the School of Information Sciences at University of Pittsburgh.

1.0 INTRODUCTION

1.1 ENERGY CONSUMPTION IN WIRELESS SENSOR NETWORKS

Advances in wireless communication technology are enabling the deployment of networks of small sensors. These sensor networks have applications in military monitoring, health, industrial control, weather monitoring, commodity tracking, home control , etc [3], [32]. As promising as this technology seems, many design issues must yet be resolved before Wireless Sensor Networks become fully functional [3].

A critical constraint on sensors networks is that sensor nodes employ batteries. A second constraint is that sensors will be deployed unattended and in large numbers, so that it will be difficult to change or recharge batteries in the sensors. Therefore, all systems, processes and communication protocols for sensors and sensor networks must minimize power consumption. The existing research on energy consumption of sensors is usually based on either theoretical models or computer simulations. One widely cited model of energy consumption by Heinzelman *et. al* has been used extensively as a guide for simulations and the design of low power consumption communication protocols [24]. Section 2.2 discusses more of these models, however, few studies exist which have measured the energy consumption of sensors in a sensor network. A study by Anastasi *et. al* measured energy consumption of a sensor node by measuring the average current consumption with a voltmeter [17]. Another study measured the power consumption of sensors, using an oscilloscope to determine power consumption in each of several states, however tests were conducted over short time intervals and with no statistical validation [26].

1.2 CONTRIBUTION OF THIS THESIS

This thesis presents the performance analysis of a sensor network implemented using the Crossbow Mica2 Motes using the GSP routing protocol[1]. The network employed six nodes and one sink. Tests were conducted over four months to provide long-term insights into the behavior of this network. The main contribution of this thesis is a detailed study of the power and energy consumption of the nodes in a sensor network while they are working, and what happens when they are about to die. Although the definition of when a sensor node is alive and when it is dead is clearly described in theory, the results of this research show that the definition is less clear in practice. Two major definitions exist for the death of a sensor network. First is when all nodes die, presented in [1], [24], [28], among other publications. Second is when the first node dies, used in [28], [29] and others. Both definitions depend on the definition of the death of an individual node. The definition of the death of a node is generally related to energy depletion. That is, one node is considered alive while its battery has enough energy to keep it working, according to approaches presented in [27], [30] and others.

Nevertheless, in this thesis it was found that this definition of death for a sensor node may not be accurate enough to define the lifetime of an actual physical implementation of a sensor node. The components and their tolerances give rise to behavior where there are difficulties in determining when a node was dead. Even when according to all specifications a sensor should not be working, most nodes continue functioning but in an unpredictable manner. The conclusions of this thesis deal with the real behavior of electronic communication devices. These characteristics are often ignored in the analysis of sensor networks today and they can play very important roles in the life and death of a sensor network.

2.0 BACKGROUND

2.1 WHAT IS A SENSOR?

A general definition of a *sensor* is “a device that produces measurable response to a change in a physical or chemical condition” [4]. More specifically, a sensor is "a device that responds to a stimulus, such as heat, light, or pressure, and generates a signal that can be measured or interpreted" [31]. The Sensor Network community often (but not always) defines a sensor node as a small, wireless device, capable of responding to one or several stimuli, processing the data and transmitting the information over a short distance using a radio link. Sensor nodes employ electronic circuits that minimize power consumption [3]. Typically sensors are thought of as measuring light, sound and temperature. However, sensors can measure other variables, such as electromagnetic fields or vibrations [2]. Sensor transmit values wirelessly to one or several sinks [3].

A *Sensor Network* is a wireless, ad hoc network, made of a large number (hundreds or thousands) of nodes, whose positions occur randomly. The OSI model and the classic layered view of communication networks may or may not apply directly to sensor networks. Nonetheless, the terminology is used throughout this document to provide the reader with a frame of reference. Other models of sensor network communications include a protocol stack model that includes physical, medium access control, network, transport and application layers as well as power management, mobility management and task management planes [3]. However, no model is used universally.

2.2 ENERGY CONSUMPTION MODELS

2.2.1 The classical energy consumption model

Heinzelman *et. al* proposed an energy consumption model for sensors based on the observation that the energy consumption would likely be dominated by the data communications subsystem [24]. Table 2.1 reproduces their model.

Table 2.1 Radio Characteristics, Classical model

Radio mode	Energy Consumption
Transmitter Electronics ($E_{Tx-elec}$) Receiver Electronics ($E_{Rx-elec}$) ($E_{Tx-elec} = E_{Rx-elec} = E_{elec}$)	$50nJ / bit$
Transmit Amplifier (ε_{amp})	$100\,pJ / bit / m^2$
Idle (E_{idle})	$40nJ / bit$
Sleep	0

The model considers a low power consumption radio that was slightly better than some standard definitions, like Bluetooth [24]. The model provides a commonly used starting point, however, the model has not been verified against the behavior of a physical radio in a wireless sensor network. When computing node energy consumption, the CPU and the sensors are consumers that may or may not be neglected, depending on the nature of the application. So, the radio model must be used jointly with some figure of the energy consumption of those elements, because in the end, power supply must feed all the system and not just the radio.

2.2.2 µAMPS Specific Model

Shih *et. al* presented a model developed for a specific platform, the µAMPS Wireless Sensor Node. The platform has a StrongARM SA 1110 microprocessor with a clock speed from 59 Mhz to 206 Mhz. The model takes into consideration the energy consumed by the microcontroller, energy lost due to leakage and the average consumption of the radio [33]. Table 2.2 summarizes the model characteristics.

Table 2.2 Sensor states for µAMPS Model

State	SA-1110	Sensor,A/D	Radio	Pk (mW)
Active	active	sense	tx/rx	1040
Ready	Idle	sense	rx	400
Monitor	sleep	sense	rx	270
Observe	sleep	sense	off	200
Deep Sleep	sleep	off	off	10

The µAMPS model doesn't specify the power consumed in transmitting or receiving one bit. Nonetheless, the platform uses transmission rate of 1 Mbps, so one can calculate the energy required for transmitting one bit, following a method based in the approach presented by Hill *et. al* in [23]. The energy used in transmitting or receiving one bit and is found by using the power value.

Time to send or receive one bit = 1 / 1 Mbps = 1 µsec

$$Energy = Power * Time \qquad (2.1)$$

where Power is in Watts and Time is in seconds

$$Energy_{Txonebit} = 1040 * 1 * 10^{-3} \text{ W} * 1 * 10^{-6} \text{ sec}$$

$$Energy_{Txonebit} = 1.04 \text{ µJ/bit} \qquad (2.2)$$

$$\text{Energy}_{\text{RxonebitReadystate}} = 0.4\ \mu\text{J/bit} \quad (2.3)$$

$$\text{Energy}_{\text{RxonebitMonitorstate}} = 0.27\ \mu\text{J/bit} \quad (2.4)$$

The difference between μAMPS model and the classical model presented in [24] is on the order of two orders of magnitude for the transmission case and one order of magnitude for the receiving case

2.2.3 Mica2 Specific Model

Polastre et. al proposed a model that presents the total energy consumption for Mica2 as the summation of energy transmitting, receiving, listening, sampling data and sleeping [27]. Values are calculated using the expected consumption of the CPU and the radio, which can be found in specific datasheets [27]. Table 2.3 presents a summary of current consumption.

Table 2.3 Current consumption for Mica2 Model

Operation	**Time (s)**		**I (mA)**	
Initialize radio (b)	350E-6	t_{rinit}	6	c_{rinit}
Turn on radio (c)	1.5E-3	t_{ron}	1	c_{ron}
Switch to RX/TX (d)	250E-6	$t_{rx/tx}$	15	$c_{rx/tx}$
Time to sample radio (e)	350E-6	t_{sr}	15	c_{sr}
Evaluate radio sample (f)	100E-6	t_{ev}	6	c_{ev}
Receive 1 byte	416E-6	t_{rxb}	15	c_{rxb}
Transmit 1 byte	416E-6	t_{txb}	20	c_{txb}
Sample sensors	1.1	t_{data}	20	c_{data}

As the authors present current consumption and time, and assuming that Mica2 is powered by a 3V source [2], one can calculate energy in transmitting and receiving one bit, as:

$$Energy = Current * Voltage * Time \quad (2.5)$$

Where current is in Amperes, Voltage is in Volts and Time is in seconds.

$$\text{Energy}_{Tx} = 20 * 10^{-3} \text{ A} * 3 \text{ Volts} * 416 * 10^{-6} \text{ sec} / 8 \text{ bits} = 3.12 \text{ μJ/bit} \quad (2.6)$$

$$\text{Energy}_{Rx} = 15 * 10^{-3} \text{ A} * 3 \text{ Volts} * 416 * 10^{-6} \text{ sec} / 8 \text{ bits} = 2.34 \text{ μJ/bit} \quad (2.7)$$

The difference with the Heinzelman model is two orders of magnitude [24]. With the μAMPS model, energy for transmission is comparable, while energy for reception is one order of magnitude bigger in the Mica2 case.

2.2.4 Mica2 Specific Model with actual measurements

Shnayder *et.al* presented a current consumption model based on measurements on the Mica2 platform [26]. A summary of the model is shown in Table 2.4.

Table 2.4 Current consumption with actual measurements

Mode	Current	Mode	Current
CPU		Radio	
Active	8.0 mA	Rx	7.0 mA
Idle	3.2 mA	Tx (-20 dBm)	3.7 mA
ADC Noise Reduce	1.0 mA	Tx (-19 dBm)	5.2 mA
Power-down	103 μA	Tx (-15 dBm)	5.4 mA
Power-save	110 μA	Tx (-dBm)	6.5 mA
Standby	216 μA	Tx (-dBm)	7.1 mA
Extended Standby	223 μA	Tx (dBm)	8.5 mA
Internal Oscillator	0.93 mA	Tx (+dBm)	11.6 mA
LEDs	2.2 mA	Tx (+dBm)	13.8 mA
Sensor board	0.7 mA	Tx (+dBm)	17.4 mA
EEPROM access		Tx (+10 dBm)	21.5 mA
Read	6.2 mA		
Read Time	565 μs		
Write	18.4 mA		
Write Time	12.9 ms		

Values presented in the table are calculated independently. The total current is found by summing the consumption for each active components. As an example, in calculating energy per bit transmitted and received, one may include only the CPU in the active state and presume the

radio is transmitting with a power of +10dBm (worst case). The authors don't specify the bit rate used, so, assuming the same time used in the previous model, and using expression (2.5), the energy cost per bit transmitted is:

$$\text{Energy}_{Tx} = (8+21.5) * 10^{-3} \text{ A} * 3 \text{ Volts} * 416 * 10^{-6} \text{ sec} / 8 \text{ bits} = 4.602 \text{ μJ/bit} \quad (2.8)$$

$$\text{Energy}_{Rx} = (8+7) * 10^{-3} \text{ A} * 3 \text{ Volts} * 416 * 10^{-6} \text{ sec} / 8 \text{ bits} = 2.34 \text{ μJ/bit} \quad (2.9)$$

Values obtained for this model are very similar to the values obtained for the model presented in [27]. The difference in transmitting one bit may be due to the fact that Polastre *et. al* didn't specify the transmission power they were using and this level may be different than the one used in [26].

2.3 PLATFORM DESCRIPTION

Experiments were conducted using Crossbow motes. The following section describes, the hardware platform, along with the rechargeable batteries and TinyOS, the operating system used in the motes.

2.3.1 Crossbow Motes

Crossbow Mica2 motes were employed as sensor nodes in the experimental testbed. The Mica2 mote, or more properly, MPR400, is powered by 2 AA batteries. According to Crossbow, the module should be powered with DC voltage between 2.7 and 3.3 Volts [8]. The main components are one Atmel Atmega 128L microcontroller and the radio [7]. Figure 2.1 illustrates the module.

Figure 2.1 MPR400 (Mica2)

The 128L is an 8-bit RISC microcontroller with 128 Kbytes of programmable Flash memory, 4 Kbytes EEPROM, 4Kbytes internal SRAM and can manage up to 64 Kbytes of external memory, as an optional feature. The microcontroller has an 8-channel 10bit ADC, an 8-Mhz crystal [8] and is programmable via UART or via JTAG interfaces. In the experimental testbed, the Mica2 were programmed using the UART port, through the MIB510, a base module provided by Crossbow [9]. Figure 2.3, from [9], presents the microcontroller block diagram.

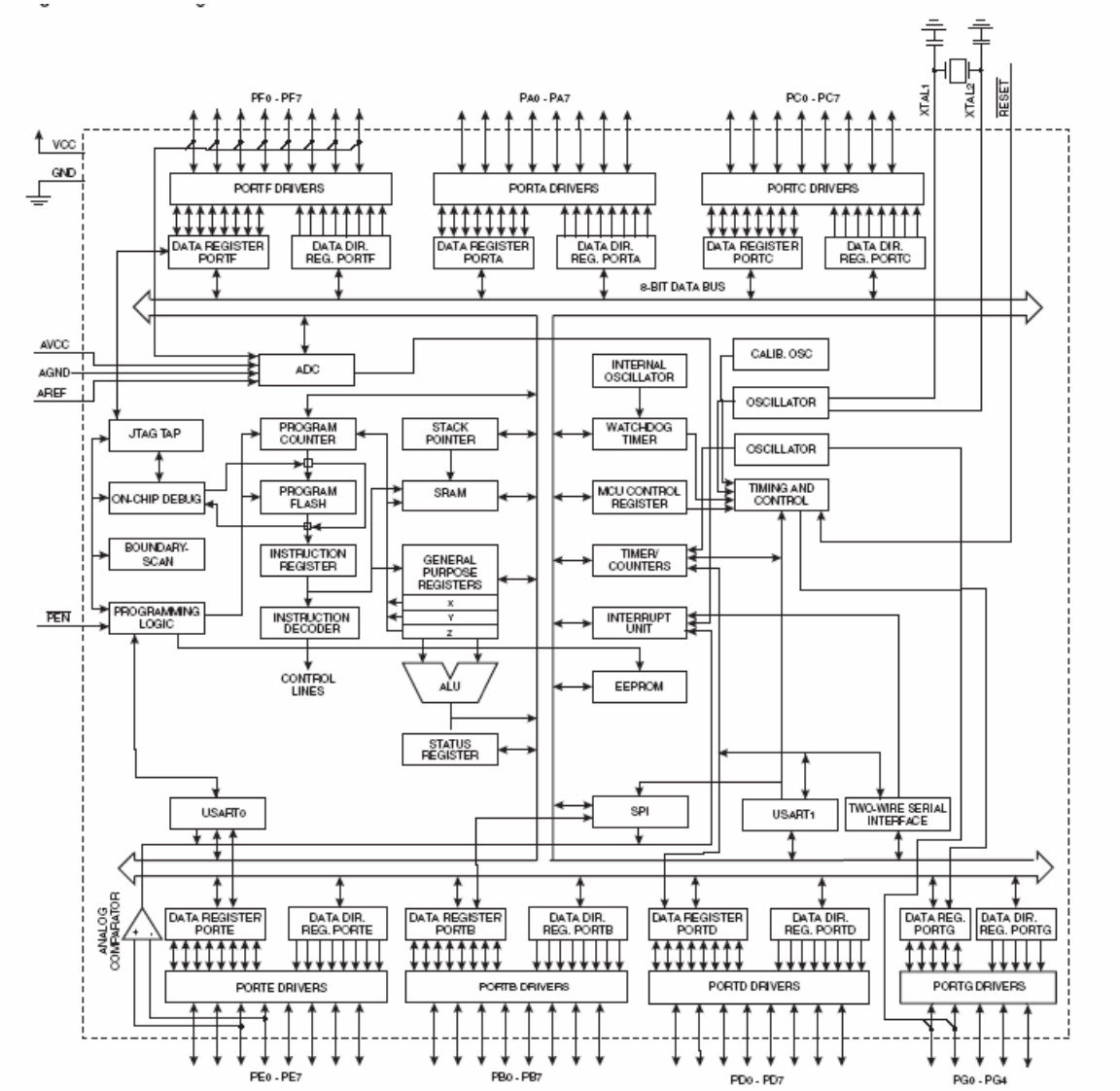

Figure 2.2 ATMEL ATMEGA 128L Microcontroller Block Diagram

- 6 bidirectional 8-bit ports and one bidirectional 5-bit port.
- Two USART (Universal Synchronous/Asynchronous Receiver/Transmitter) ports, for serial communication.
- 32 general purpose registers.
- One Real Time Counter and four flexible Timer/Counters
- One byte oriented Two-wire Serial Interface,
- One 8-channel, 10-bit ADC
- Programmable Watchdog Timer with Internal Oscillator,

- One SPI serial port, IEEE std. 1149.1 compliant JTAG test interface

The processor and radio turn on and off constantly in order to save energy. Although this strategy extends battery lifetime, it reduces battery capacity due to current surges [8]. More on this subject will be explained in the following section.

The main part of the radio is implemented using an integrated circuit Chipcon CC1000. In the MPR400, the CC1000 operates in the band of 902-928 MHz. Software controlling the CC1000 can select one of up to 54 channels with a separation of 500 kHz. The modulation operation of the CC1000 is fixed, it uses binary FSK with optional Manchester or NRZ encoding [5]. The transmission power is software controllable, to a maximum of 5dBm. Figure 2.3 shows the CC1000 block diagram [5].

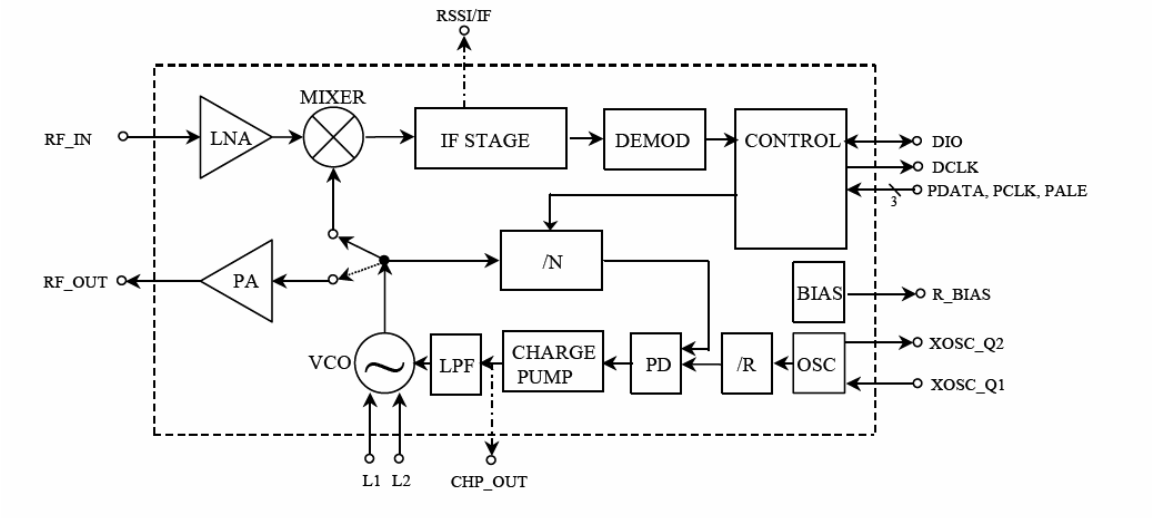

Figure 2.3 CC1000 Block diagram

All blocks presented in the figure are implemented in hardware. Signals received through the antenna pass to a Low Noise Amplifier (LNA) and are transformed to Intermediate Frequency using the Mixer. In the IF Stage, signals are amplified and filtered. After that, signals are sent to the demodulator (DEMOD) and the demodulated data is sent to the DIO pin, which

should be connected to the CPU. To send a signal, bits coming from the DIO pin modulate the RF output using Frequency Shift Keying. The Voltage Controlled Oscillator (VCO) output is connected to the Power Amplifier (PA). There is a frequency synthesizer for generating the local oscillator signal needed for the Mixer and the Power Amplifier. The synthesizer is built with the Crystal Oscillator (XOSC), a Phase Detector (PD), the CHARGE PUMP, the VCO and two frequency dividers (/R and /N)[5].

2.3.2 Rechargeable batteries

Batteries used for these experiments were Nickel Metal Hydride rechargeable batteries. It was imperative to use rechargeable batteries, because the experimental plan called for each mote to run continuously until the batteries were depleted. Economics dictated using rechargeable batteries. Nevertheless, for testing mote behavior, there were some tests run with regular AA batteries which provided the motes with 2.75 V (in average). Behavior with rechargeable or non-rechargeable batteries was similar. All batteries have a nominal voltage and a nominal charge capacity, (C), usually specified as how many Amperes a battery can deliver during one hour. Consider, for example, a battery with C = 1200 mAh. The battery is capable of delivering the equivalent of 1.2 Amperes (1200 mA) for one hour. The equivalent number of Joules is:

$$\textit{Energy (Joules)} = \textit{Current} * \textit{1hour} * \textit{3600 sec / 1 hour} * V \qquad (2.10)$$

$$\text{Energy} = 1200 \text{ mA} * 1 \text{ hour} * 3600 \text{ sec} / 1 \text{ hour} * 1.2 \text{ V} = 5184 \text{ Joules}$$

One characteristic from NiMH rechargeable batteries makes them different than the ones recommended by Crossbow. These batteries have a nominal voltage of 1.2 Volts [5]. If the two batteries were recharged at this nominal voltage, they will supply the mote with 2.4 Volts (very low compared to Crossbow recommendation). However, during the experiments it was found that batteries, when well charged, get 1.3 or even 1.4 Volts, providing a voltage close to the original specification.

Another important feature of every battery is the Discharge curve. There is no technical information available from the exact type of batteries used in the experiments, but Figure 2.4 presents an example for a similar battery of the same manufacturer [12].

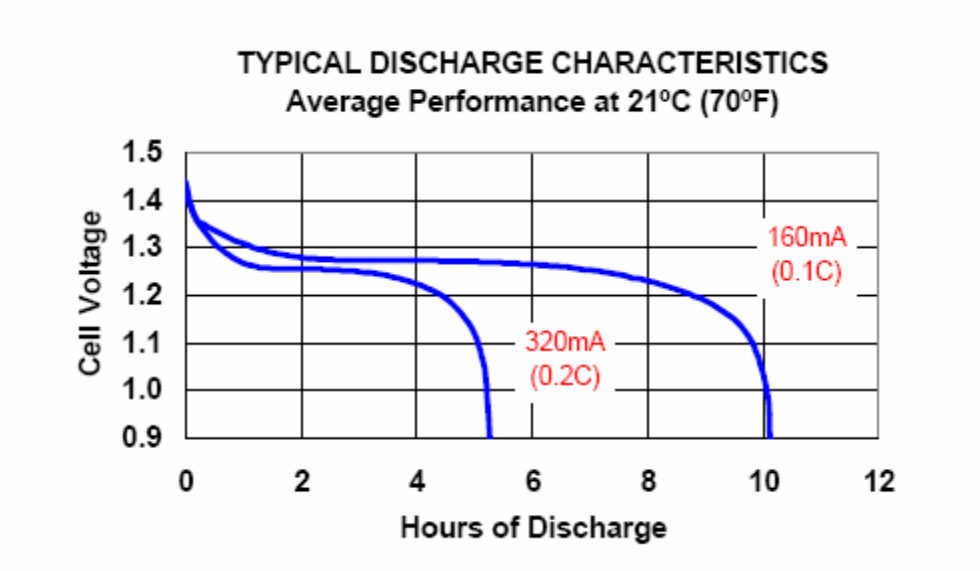

Figure 2.4 Energizer NH15 Discharge Curve.

The curve belongs to one AA NiMH battery with 1600 mAh, 1.2 Volts of nominal voltage. The ones used in the experimental testbed were 1200 mAh, and later on the tests, 2500 mAh batteries. The figure shows that at the beginning of the curve, when the batteries have the voltage recommended by Crossbow, they have a steep slope in voltage decrease. The linear part of the curve, when the batteries typically operate, is approximately 1.28 Volts, under the recommended voltage for the motes.

2.3.2.1 Voltage Depression

When batteries are not correctly charged, they are affected by a phenomenon commonly known as “Memory Effect”, and technically known as Voltage Depression. The phenomenon is a drop in voltage and a loss in capacity of the battery that occurs after several cycles of charging

the battery without allowing it to fully discharge. Although NiCd batteries are more frequently affected by Voltage Depression, NiMH batteries can be affected too. Most batteries can recover from the phenomenon with some cycles of full discharge and charge. Motes lifetime is affected by voltage depression in two ways: first, the voltage provided to the node may be below the required level and the second is that the battery has less energy stored, so mote lifetime becomes shorter. Figure 2.5, illustrates voltage depression (from [13]):

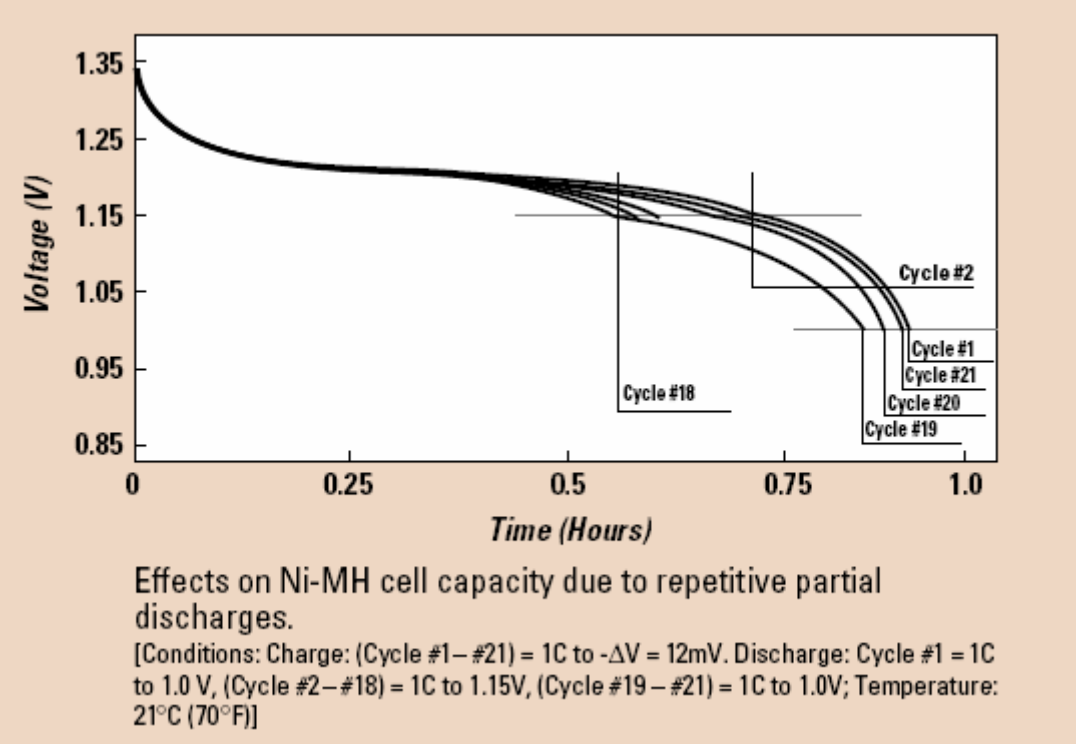

Figure 2.5 Voltage Depression in a Ni-MH battery.

2.3.3 TinyOS

Crossbow motes are an open platform that can be programmable using TinyOS, an operating system specially designed for devices with high constraints in memory, processing capacity, etc, as is the case of wireless sensor networks [7].

TinyOS differs from a regular operating system in several ways:

- TinyOS allows only one executable file to be loaded and therefore one process to be running. An application will consist of a scheduler and additional components that will be compiled to form a single executable.
- TinyOS has an event-driven architecture and a concurrency model based on tasks and hardware event handlers [34]. Tasks are non-preemptive and they run until completion. Hardware event handlers occur in response to hardware interrupts and can preempt the execution of a task or another hardware event handler [7].
- TinyOS doesn't have a different space for the kernel and for the user [16]
- TinyOS uses a single shared stack [16]

All operations of TinyOS are implemented in a language called nesC, an extension of C [15]. TinyOS has a library of components (pieces of nesC code) that can be used or changed to meet the requirements of a specific application. All radio and processing components needed for basic operations with the motes are included in the distribution that Crossbow supplies to their customers. All applications in TinyOS are made of one or more components. There are two types of components: modules and configurations. The configurations describe how the components are connected together, so, this is called “wiring”. All applications must have at least one configuration. The actual code for describing the implementation of the functions can be found in the modules. In a module, the programmer can also implement one or more interfaces. An application doesn’t need to have modules. Interfaces provide an abstract definition of the interaction of two or more components. The interface does not have any code and it is similar to a function prototype in C [7]. Figure 2.6 illustrates the structure of an application in TinyOS.

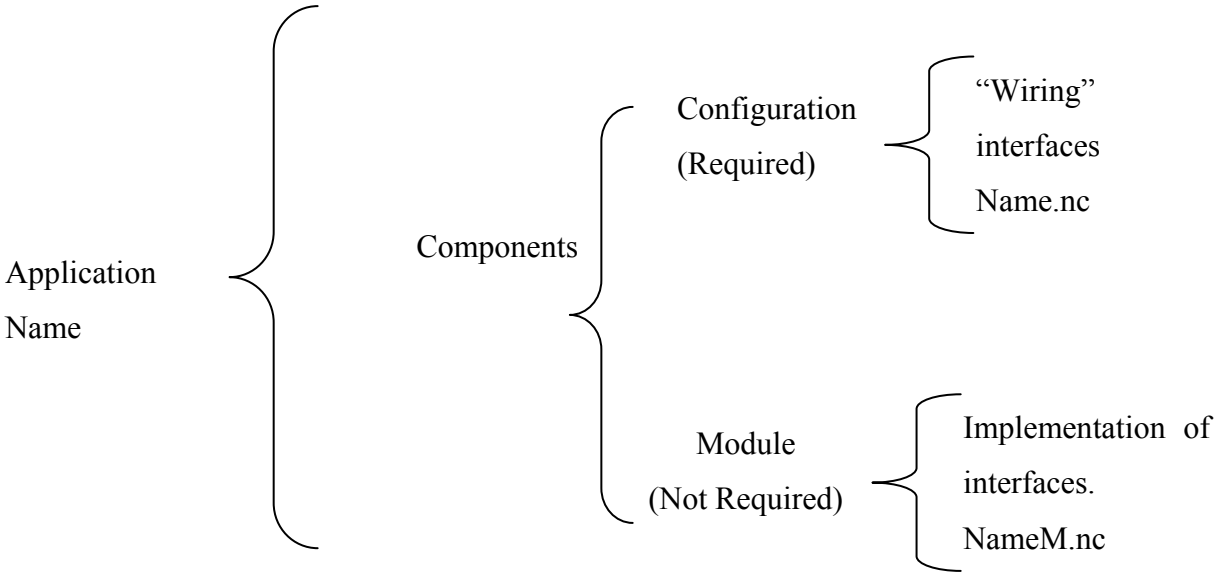

Figure 2.6 Structure of application in TinyOS.

In figure 2.6, the application is called simply Name. The application Name has one configuration, which is mandatory, and is called Name.nc, and one module, called NameM.nc. Components can have any name in TinyOS, but for convention, a configuration should have only the name and the extension nc, while a module should end with the M capital letter [7]. Please see Appendix B for a typical example application.

2.4 MAC PROTOCOLS FOR SENSOR NETWORKS

Many protocols have been proposed for sensor network Medium Access Control. The following sections present some of the most widely known protocols, including the one used in these experiments. Each of the MAC protocols has advantages and disadvantages, and there is no general agreement on which one is the best for Sensor Networks. Some protocols appear to perform better in some applications and other protocols may be well suited in other situations. The protocol descriptions in this section provide examples of the general concepts around which

most MAC protocols are designed, so the reader can understand the interaction of the MAC and routing protocols used in these experiments.

2.4.1 CSMA-Based Medium Access:

In CSMA the transmitting node listens to the medium before sending information [20]. Different versions of this protocol are used in wired and wireless networks [22]. The scheme proposed by Woo *et. al* includes constant listen periods and random delays to reduce collision probability (collision avoidance) [22]. CSMA protocols typically exhibit short delay and good performance under low traffic load conditions. However, as traffic increases, collision probability also increases and CSMA protocols efficiency decreases[20]. CSMA/CA is the MAC protocol used in the experimental testbed.

2.4.2 Self-Organizing Medium Access Control for Sensor Networks (SMACS):

The SMACS protocol, where nodes make a neighbor discovery process and simultaneously they assign a schedule for transmitting and receiving [18]. When communicating with a neighbor, the node chooses two time slots in random but fixed frequencies. The whole network does not need to be synchronized, but there must be synchronization between neighbors. Synchronization is achieved after exchanging a minimum of 6 messages between neighbor nodes. The protocol assumes a relatively large number of frequency bands and all nodes must keep track of their schedules with their neighbors [18]. According to [3], one advantage of SMACS is that there is no need for one master node or for network synchronization. The disadvantage is overhead that nodes need in order to form the links between them, not only in messages but in memory and processing.

2.4.3 Hybrid TDMA/FDMA-Based:

The scheme assumes that all nodes are close to a high powered base station (less than 10 meters apart). The idea is to minimize the total power consumption of the network, finding an optimum

number of channels that depends upon the relation between the power consumption of the transmitter and the power consumption of the receiver. If the transmitter has bigger power consumption, TDMA is preferred. Otherwise, FDMA will be used [19]. According to [20], one advantage of this kind of schemes is that there are no collisions, because all time slots or bandwidth are assigned separately to each node. Some disadvantages are limited coverage area (only 10 meters), and the waste of resources when the node does not have anything to send.

2.5 ROUTING PROTOCOLS FOR SENSOR NETWORKS

There are several routing protocols proposed for sensor networks. GSP, the routing protocol used in this experiment, is based on the Flooding concept.

2.5.1 Flooding:

Flooding is a method where every packet received is retransmitted to all the nodes in the network [20]. Variations include only retransmitting the packet if it has not reached a maximum number of hops or if the destination node is the node itself [3]. In order to know this, some kind of addressing scheme must be used. Flooding is a simple algorithm, but it has several disadvantages when used in sensor networks [21]:

- Implosion: duplicated messages are sent to the same nodes.
- Overlap: If two nodes are in the same region, they may sense the same signal at the same time and transmit the same information twice.
- Resource blindness: The Flooding method does not depend on whether energy resources are scarce or not. The method works the same in any of the two situations.

Figure 2.7 illustrates an example network. Nodes B and C can listen to A and vice versa. Node D is in range of C and B only. When node A sends a packet, nodes B and D receive it and they retransmit it. As A is in the same range, it will hear again the same packet that it sent, and it

will retransmit the same packet. The packet eventually will propagate through the whole network, but there will be a big amount of duplicate packets, if no improvements are applied to the algorithm.

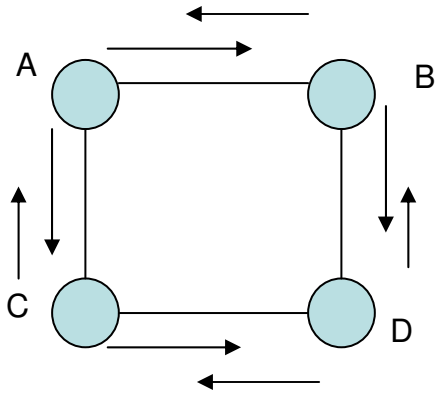

Figure 2.7 Flooding Algorithm example

2.5.2 Gossiping

The method tries to improve the flooding algorithm by the following procedure: the nodes have a probability p of broadcasting the packet they receive. With probability 1-p, the received packet is discarded [25]. Gossiping avoids the implosion problem, but the time it takes for the packet to get to the destination is long, according to [3]. There are no synchronization requirements [3]. As an example, a similar network as before is presented in Figure 2.8. In this case, node A transmits a packet. With probability p, B retransmits the packet and with probability 1-p C drops the packet. When the packet gets to D, the coin is tossed again and the packet in this figure is sent again. Notice that C spent energy receiving the packet that would be dropped, according to 1-p.

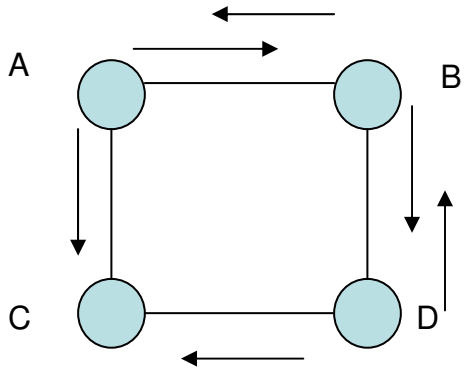

Figure 2.8 Gossiping example

2.5.3 GSP: Gossip-based Sleep Protocol for Energy Efficient Routing in Wireless Sensor Networks

GSP uses a duty cycle for the transmission. In one part of the duty cycle, the radio is on, so the node can transmit and receive. With a probability p, the radio will be off in the next part of the duty cycle, so the node will not be able to transmit or receive any packet. When a node receives a packet, it must retransmit it [1]. Figure 2.9 illustrate an example network employing GSP. In this case, A sends a packet. Assuming that B has its radio on and C its radio off, B will receive the packet and will retransmit the packet. C did not spend energy in receiving the packet. D will receive the packet only if its radio is on. In that case, it will retransmit the packet.

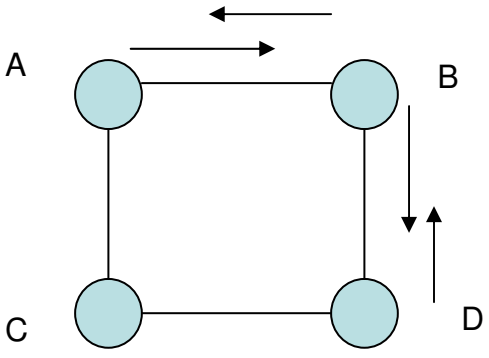

Figure 2.9 GSP algorithm example

3.0 EXPERIMENTAL DESIGN

The goal of the experiments was to measure network lifetime, defined as the time it takes for the first node to die. Although the classic layered model of communications system doesn't apply in exactly to sensor networks because of the cross-layered tasks, it is helpful to reference a known model when showing the different functions of this experiment.

3.1 EQUIPMENT

The network implemented in this work consisted of six nodes and one sink. Each node was a Crossbow Mica2 motes, using rechargeable AA batteries. Initial tests employed Energizer Accu Rechargeable, 1200 mAh at 1.2 V NiMH batteries. Later experiments used Merkury Rechargeable NiMH batteries, 1.2 V and 2500 mAh. Each run employed only one type of battery.

The sensor board in Mica2 consumes 0.7 mA while CPU in active state consumes 8.0 mA and the radio can consume between 3.7 and 21.5 mA [26]. Since these experiments were designed to measure power consumption of communications protocols and since power consumption by the sensing electronics is low, motes in this thesis did not use the sensor board.

In order to make a consistent description of the behavior of the motes, it was necessary to use devices such as Tektronix PS280 DC Power supplies, prototyping boards, and measurement equipment such as Fluke 8050A Digital Multimeters, Hewlett Packard 54600A oscilloscopes and Agilent 89600 Vector Signal Analyzer and Spectrum Analyzer.

3.2 PHYSICAL LOCATION

The location of the nodes was selected in such a way that the topology of the network is built with three nodes directly connected to the sink and other three nodes that can not communicate directly with the sink. In this way, GSP performance could be analyzed and message relaying through the network could be observed, since messages from all nodes were registered at the sink. Initially, there was a coverage area test using 22 possible locations. Figure 3.1 shows locations selected, motes used for the test and connectivity pattern.

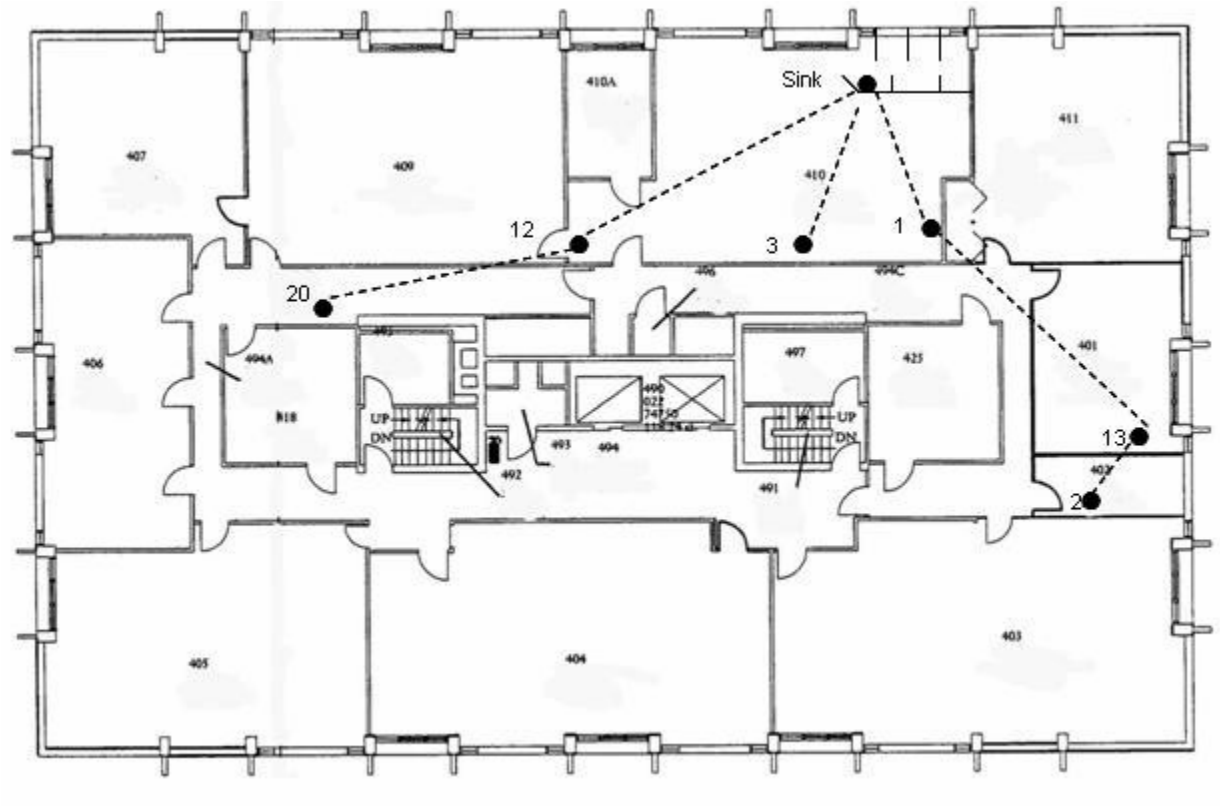

Figure 3.1 Node location and Connectivity

Nodes 0 (sink), 1, 3 and 12 are located in the Wireless Laboratory. The rest are located as follows: node 2 in room 402, Graduate student office; node 13 in room 401, PhD Student

Lounge; node 20 in locker number 24. Motes 1, 3 and 12 communicate directly with the sink. Motes 2, 13 and 20 should reach the sink through some of the other motes, using GSP. Mote number 20 was located inside a metallic locker closed all the time, but communication was successful to mote number 12 and from there to the sink. Table 3.1 lists distances from every mote to the sink.

Table 3.1. Distances from the nodes to the sink

Node	d (m)
1	5.32
2	15.22
3	4.19
20	16.31
12	8.57
13	13.44

The sink receives all messages and sends them to a computer, using a serial port. All packets received are registered with the date and time they were received. Table 3.2 shows an example listing of the registered packets.

Table 3.2 Log file example

Dest Add		Type	Group	Length	Pad	Pad	Pad	Src Add	pad	counter		pad	pad	pad	date
7E	00	04	7D	0A	B2	67	7C	02	00	8B	75	34	29	40	Sat Sep 10 11:29:29
7E	00	04	7D	0A	B2	67	7C	0C	00	8B	7E	34	29	40	Sat Sep 10 11:29:33
7E	00	04	7D	0A	B2	67	7C	0C	00	8B	7F	34	29	40	Sat Sep 10 11:29:33
7E	00	04	7D	0A	4D	4D	4D	0C	1C	8B	81	A9	95	C7	Sat Sep 10 11:29:33

The fields shown as "pad" and the date and time are created by the PC after the sink node transmits the data to the PC. Figure 3.3 illustrates the frame transmitted over the radio link.

3.3 PHYSICAL LAYER CHARACTERISTICS

The motes were programmed to transmit in the 903 MHz band at the maximum power, 5dBm, using Binary FSK with Manchester coding at 19.2 kbps.

3.4 MAC LAYER PROTOCOL

The experiments employed the default MAC protocol (CSMA/CA) provided by Crossbow. The CSMA/CA variant begins with the mote listening to the medium. If the medium is idle, the protocol waits during a backoff time. If the medium is still free after the backoff time, the node transmits. If the medium is not free, the mote waits during a congestion backoff time to sense the medium again. There are no collision detections, no acknowledgements and no sequence numbers. The MAC protocol will not retry to send a frame after a fail in transmission. Failures occur when there is another message that is still being sent. In this case, the new message will be dropped, because the buffer can not be modified [34]. Process result (success sending a frame or failure in transmission) is signaled to an upper layer, which will then decide what to do.

Remember that Crossbow Motes use TinyOS, so it is possible to modify the code and refine this protocol as much as the experiment requires. There was no need for that in GSP implementation, so the basic version was used. Figure 3.2 shows the link layer frame structure. Note that because GSP imposes no additional overhead on top of the link layer, the frame format and the packet (network layer entity) format are identical.

Bytes	8	2	2	1	1	1	2	2	2
	Pream	Sync	DA	Type	Group	Length	Counter	SA	CRC

Figure 3.2 Frame Structure

- Pream are 8 bytes that make the preamble. The 8 bytes are repetitions of 0xAA (10101010) or 0x55 (01010101). Here, the 0xAA series was used, according to [6]
- Synch are 2 bytes for synchronization. These are 0x33CC.
- DA is the destination address. In these experiments only the Broadcast address (FFFF) was used in this field.
- Type is the type of frame that is being sent and it is similar to a TCP/UDP port, in the sense that it says what application the frame must be delivered to.
- Group is the identifier for the particular group of motes. Using Crossbow motes, there can be several groups of nodes and this is the address for each group. In the experimental testbed, only one group was used: group 0x7D or 125 in decimal.
- Length is the length of the message being sent. The length used in the experiment was 4 bytes, distributed as follows: First two bytes are the counter. The next two bytes are the Source Address, that is, the address of the mote that originally sent the message. These four bytes are sent in Little Endian format.
- Last 2 bytes are CRC checking, which help the mote to know if the frame was received correctly or not. The MAC layer doesn't drop packets with bad CRC. Upper layers will decide if they are discarded or not.

Total frame size is 21 bytes.

The mote switches into transmission mode after doing backoff. Switching takes 250 microseconds. After transmitting the packet, the mote switches back into reception mode. That takes another 250 microseconds. So, the "Time frame" presented in Figure 3.3 should be considered.

Time (msec)	6	0.25	9	0.25
	Backoff timer	SwtoTX	Message	SwtoRx

Figure 3.3 "Time frame"

The number showed in the backoff timer is an estimate because the real number corresponds to the time it takes to transmit between 1 and 32 bytes [6]. The times are distributed in the interval [0.416 , 13.312] msec. If the node senses a busy medium after doing the original backoff, another timer called Congestion Backoff starts. The timer picks a random number between 1 and 16 bytes, so the times are distributed in the interval [0.416 , 6.656] msec [6]. The time frame shown in Figure 3.3 applies only when one node transmits.

3.5 NETWORK LAYER PROTOCOL

The Classical GSP protocol used was the most basic version, using a first extreme case. The radio is on during 2.5 seconds. With probability p = 1, the radio will be off during the next 2.5 seconds. When the mote receives a packet, the mote will retransmit it. So, there is a duty cycle of 50% which can be seen in Figure 3.4.

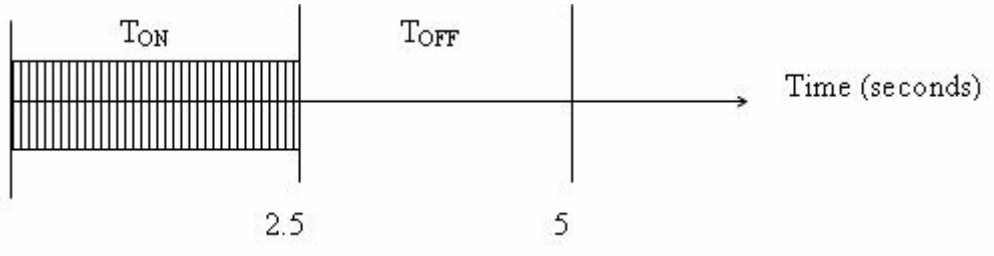

Figure 3.4 Duty Cycle

3.6 APPLICATION LAYER

A simple application was created, each node has one counter that is incremented and sent to the network every 80 miliseconds. If there was a transmission error reported by the MAC layer, the application will resend the same counter and it will only try to send the next value of the counter when it receives a success signal from the MAC layer. 52 μsec are needed for transmitting one bit at 19.2 kbps. As the mote microcontroller can execute up to 8 MIPS with the 8Mhz crystal, in the time needed to transmit one bit, the mote can execute up to 416 instructions [9]. The application and routing algorithm were programmed using TinyOS, adapting nesC application code to the needs of the experimental setting. The adapted application is called CntToLedsAndRfm. Code can be seen in Appendix A.

3.7 MEASUREMENTS

This section describes the measurement system and protocol.

3.7.1 Direct Network lifetime

All nodes were powered with recently recharged batteries for measuring network lifetime in a direct way. The initial voltage of each of the nodes was measured using a voltmeter, and then all nodes were started at the same time and located in their positions. Nodes started to send the counter through the network using GSP. Packets finally reached the sink, which is another node attached to the MIB510 module, AC powered and directly connected to a PC using a serial port. The sink sent all packets received to the PC and they were stored in plain text files, along with the date and time when they were received. The sink was not running GSP and hence could only receive packets. Its radio was On all the time, so it only received packets, showed the counter last

three digits in its LEDS and sent the messages to the serial port. The network was left running this way until the sink stopped receiving packets from some of the nodes. That was the visible signal of the death of those nodes. Then, information of the files was analyzed, to know exactly when the first and last packets from each node were received. Subtracting the date and time, it was possible to calculate the time each node was transmitting (alive). The moment when the first node was dead was considered the lifetime of the network.

3.7.2 Average voltage

Average DC voltage was measured using the Fluke Multimeter. At the beginning of every run, the voltage in each battery and then at the terminals of the node was measured. The voltage value was steady as long as the node was off. When the node was turned on, the voltage started to fluctuate around the original value. At the end of every run, the voltage in each node was measured again. The voltage dropped when the radio was on and the value increased when the radio was off. But these values were not constant, so they were very difficult to measure and they were not used for the experimental testbed.

3.7.3 Instantaneous voltage and current

In this case, the network was implemented using one node and one sink. Values found were used to indirectly measure energy consumption of one node. One node was connected in series to a resistor of 10.03 ohms and to a DC Power supply. Using 54600A oscilloscope, voltage drop over the resistor was measured. Current was calculated using values given by the oscilloscope. A small resistance value was chosen in order to minimize additional voltage drop. The setting is shown in figure 3.5.

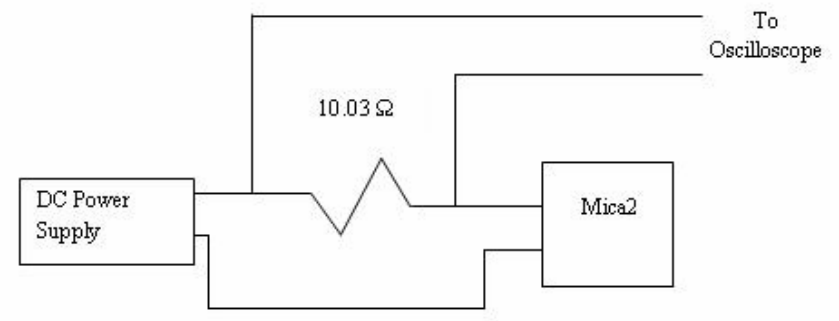

Figure 3.5 Connection for Voltage measurements

When the node was receiving, the current was measured in the node while another node running GSP was transmitting. The second node started transmitting and the original node received the frames and retransmitted them accordingly.

3.7.4 Antenna Calibration

Antennas used for measuring transmitted power in the motes are Maxrad MFB8133. The specified bandwidth goes from 806 to 866 Mhz [11] and factory tuning is 813 Mhz. Since the smallest frequency that motes can use is 903 Mhz, power transmission measurements needed a correction factor. Figure 3.6 illustrates the setting for antenna calibration. A frequency sweep was done using the signal generator to find out the antenna and cable response in the nodes frequency range.

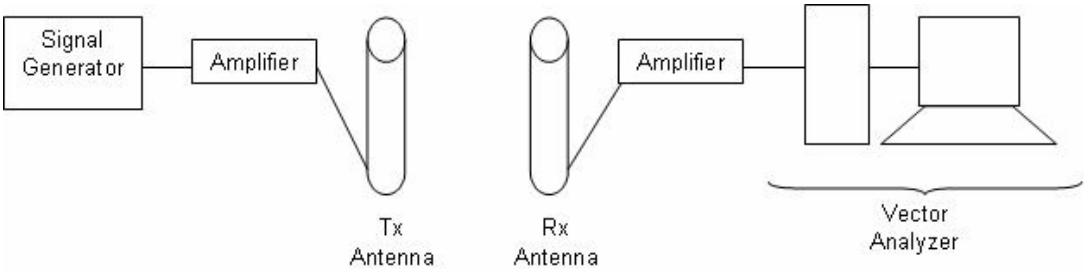

Figure 3.6 Antenna Calibration Equipment

3.7.5 Transmitted and Received power

Agilent 89600 Vector Signal Analyzer was used for monitoring output transmit power, placing a receiving antenna close to the transmitting node. Figure 3.7 shows the setting.

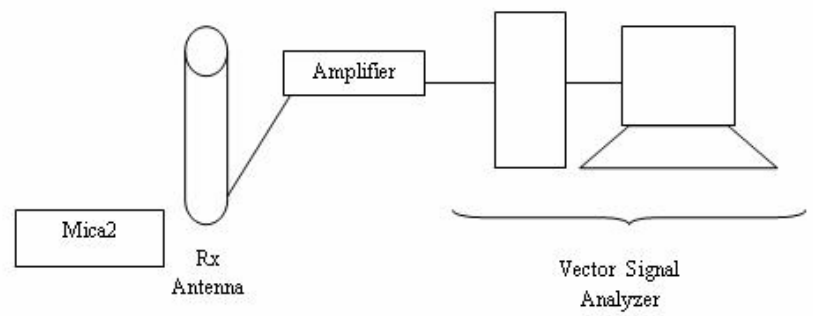

Figure 3.7 Equipment setting for measuring transmitted power

The same instruments were used for verifying received power, but this time the receiving antenna was located close to the sink. Vector Signal Analyzer shows via software the waveforms received by the antenna in time and frequency domains. The same setting was used to measure the noise when none of the nodes was transmitting.

3.7.6 Transmitted and received frames

Using the same setting showed in section 3.7.5, Vector Signal Analyzer Demodulation function was used to verify the frame structure and its contents. The transmitted packet was decoded locating the receiving antenna close to the transmitting node. After checking the frame contents, that frame was located in the corresponding file generated by the computer attached to the sink, confirming that the frame was transmitted and received correctly.

4.0 EXPERIMENTAL RESULTS AND ANALYSIS

This chapter presents several approaches for measuring network lifetime. Multiple tests were necessary to build an accurate model. Results measuring the direct energy consumption in each node are presented and used to derive an energy consumption model. Finally, the procedure to calculate expected lifetime of the mote is explained in detail using the derived model with GSP and CSMA.

4.1 FIRST APPROACH FOR MEASURING LIFETIME OF THE NETWORK

Recall Chapter 2 defined the network lifetime as the time from the beginning of the experiment (turning all sensors on) until the moment when the first one died. Although other definitions exist this one reflects the worst case. Figure 4.1 illustrates one example where after the first node dies the network is useless even if the remaining nodes are alive.

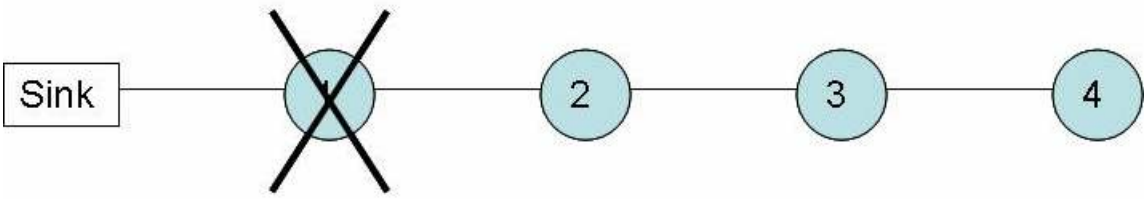

Figure 4.1 Network lifetime defined by the first node to die

In the initial experiments, mote number 11 always died within one minute of starting the experiment. The cause of this situation turned out to be a hardware problem, and it was corrected by replacing mote 11 with mote 20. Results from the series one experiments were not consistent, either in the time it took for the first mote to die or in the voltage found when the mote was dead. In series two experiments, after detecting a dead mote, that mote was restarted (that is, a hard reboot of the mote). 8 out of 10 times, the mote started transmitting again, showing that it was not dead yet. The death of the mote was redefined based on restarting the mote three times, at least 10 minutes apart from each other. If no packet was received from the mote located in its original position during the 30 minute restart period, the mote was defined to be dead. Mote lifetime was measured by adding together the times when the motes had messages registered in the log files.

The definition for the lifetime of a sensor node usually depends upon battery lifetime [27, 30]. If the mote were to stop transmitting and never transmits again, the definition would be helpful. But, as the experiments show a new definition is required to account for motes that stop transmitting, but could potentially begin transmitting at a later time. Table 4.1 lists results from series 2 experiments. Figure 4.2 shows the network topology with the first node to die in the in a square. The most frequent first mote to die first was number 13. Using that figure, there is no clear correlation between location and death of a node.

Table 4.1 Network lifetime restarting nodes

Run #	First dead Mote	Lifetime (minutes)	Vinitial (Volts)	Vfinal (Volts)
1	11	1.75	2.712	2.502
2	11	1.9	2.724	2.563
4	11	1.25	2.782	2.537
5	13	153.00	2.605	2.505
7	11	85.15	2.682	2.557
8	13	95.70	2.727	2.489
9	3	458.97	2.602	1.272
10	13	37.82	n/a	2.467
11	3	183.45	2.74	1.82
12	12	192.57	2.616	1.813
13	13	1051.00	2.727	2.547
14	13	205.02	2.734	2.619
15	20	19.12	2.605	1.25
16	1	1717.77	2.756	2.547
17	13	2983.17	2.828	2.441
18	20	492.00	2.789	2.633
19	1	452.83	2.75	2.496

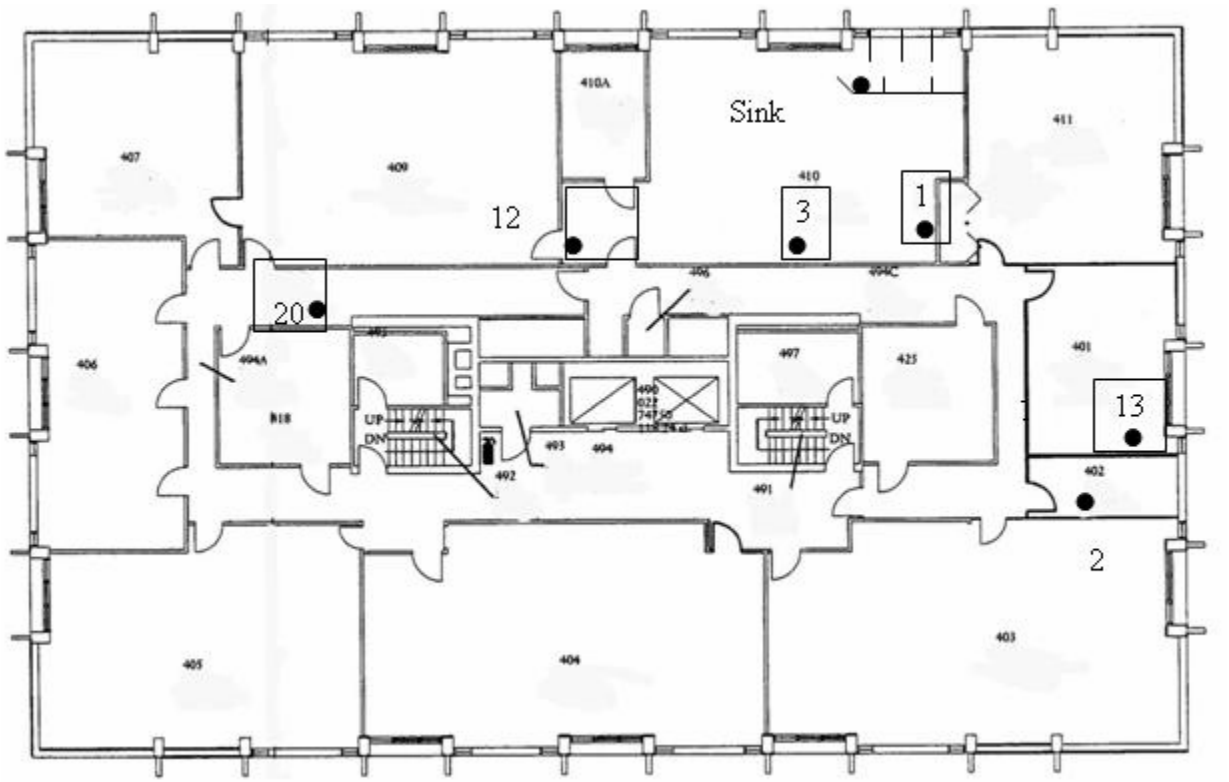

Figure 4.2 Location of first nodes to die

Figure 4.3 illustrates the distribution of lifetimes for the first dead node. Discarding results shorter than 5 minutes, the average is 580.54 minutes ± 366.76 (C. I. 90%). The range makes it difficult to draw any useful conclusions.

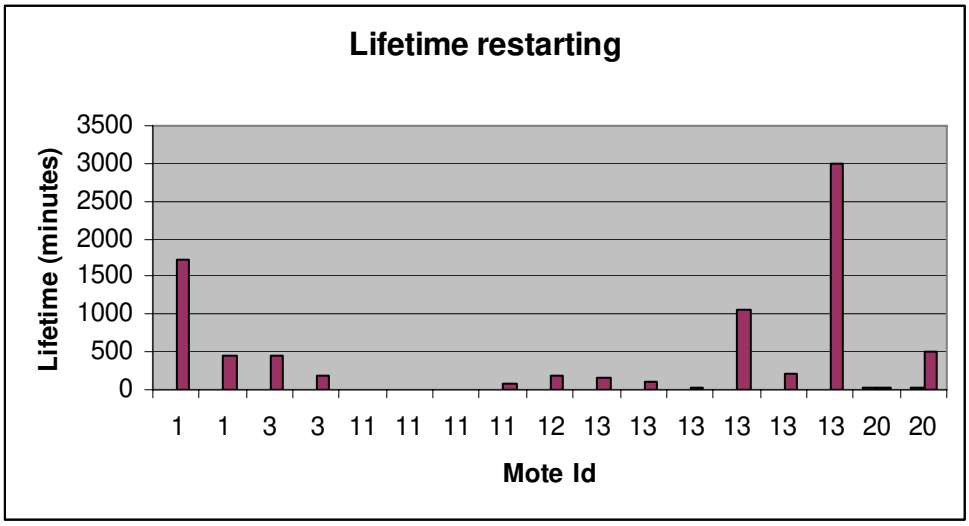

Figure 4.3 Lifetime restarting the nodes

4.2 SECOND APPROACH

A second approach measured network lifetime as the time until data from a mote no longer reached the sink, without restarting the motes. Data from the second series of experiments was re-analyzed, Table 4.2 shows the results. In Figure 4.4 the first motes to die are shown again with squares. In this analysis every mote was the first to die at least once, but the most frequent mote to die first place was mote number 13.

Table 4.2 Lifetimes without restarting

Run #	First dead Mote	Lifetime (minutes)	Vinitial (Volts)	Vfinal (Volts)
1	11	1.75	2.712	2.502
2	11	1.90	2.724	2.563
4	11	1.25	2.782	2.537
5	13	153.00	2.605	2.505
7	11	85.15	2.682	2.557
8	13	95.70	2.727	2.489
9	2	45.00	2.602	n/a
10	13	37.82	2.687	2.467
11	13	12.78	2.672	n/a
12	20	86.75	2.67	n/a
13	3	646.37	2.652	2.564
14	20	201.02	2.773	2.611
15	13	1.57	2.694	2.415
16	1	257.47	2.756	2.547
17	12	28.63	2.698	2.404
18	2	220.55	2.801	2.646
19	13	1.80	2.738	2.718

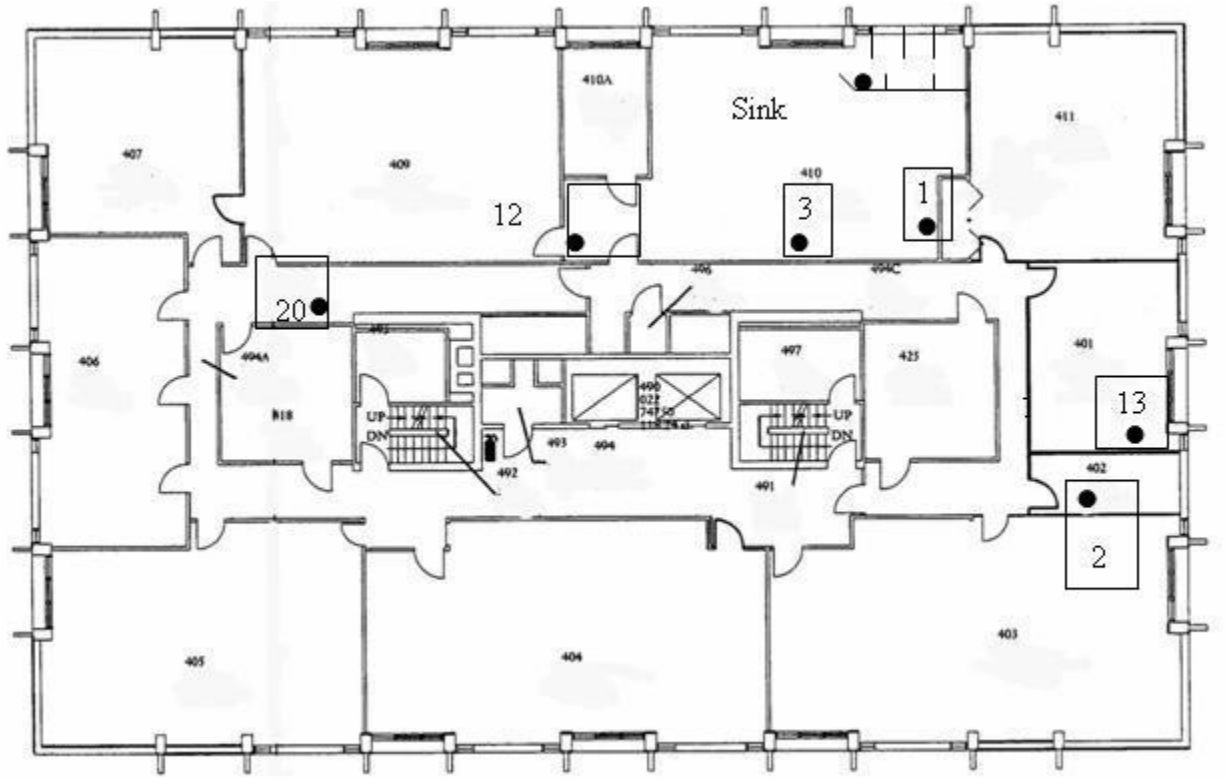

Figure 4.4 Location of first node to die, second approach

Figure 4.5 shows the lifetime distribution for the first nodes to die. Again, discarding all results lower than 5 minutes, average lifetime was 154.32 minutes ± 83.29 (C. I. 90%). These results are not acceptable either.

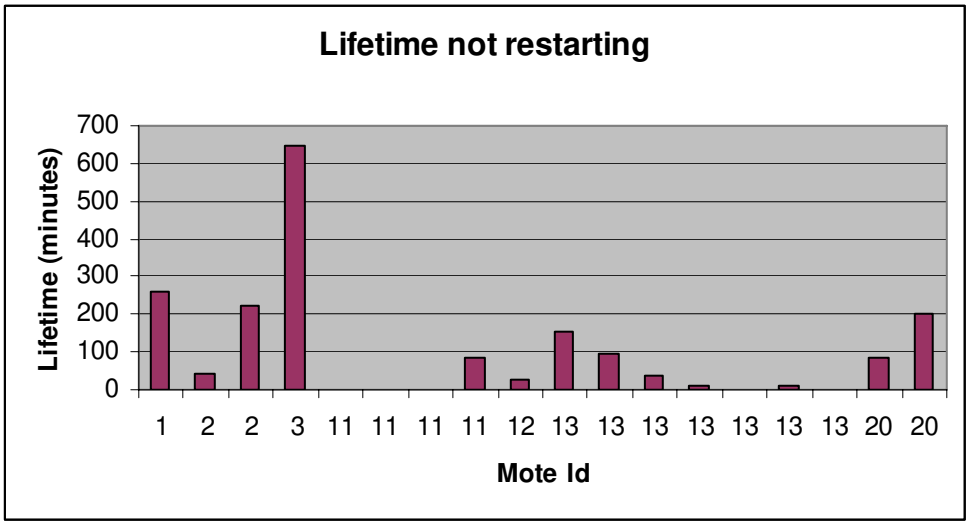

Figure 4.5 Lifetime without restarting the motes

One important fact not considered in series 1 and series 2 tests is the Voltage Depression phenomenon, which could affect the measurements by diminishing the nominal capacity of the batteries, because they were not fully discharged before they were recharged. To avoid this problem, experiments in run number 5 and later, batteries were discharged as much as possible using them in electronic devices such as a digital camera. After run number 12, batteries were discharged completely using flashlights. However, although with optimal charging conditions the average lifetime from run number 13 increased (discarding results lower than 5 minutes), the results remained inconsistent. In a third series of experiments, designed to isolate the battery factor, one mote was connected to the MIB510 and left running. The MIB510 is AC powered and supplies the mote with a constant DC voltage, however after only a few hours, the mote died anyway.

4.3 DIFFERENT ALTERNATIVES

Several factors were affecting the node and network lifetimes. So, a fourth series of experiments were designed to test the lifetime for only one mote so as to avoid interference from the rest of the network and to determine to limiting factor for network lifetime.

4.3.1 Signal received

The first alternative considered was to verify the behavior of the signal when packets were received and when they were not. The first experiments measured signal power received by the sink during a run and comparing peak values when the packets were received and when they were not. Using the procedure described in 3.7.4, two antennas were used in order to find the loss introduced by the measurement system. The calculation of the additional loss is:

:

$$P_{tx} = -5 \text{ dBm}$$

$$P_{rx\ (average)} = -38.852 \text{ dBm}$$

$$\text{Losses} = P_{tx} - P_{rx\ (average)}$$

$$\text{Losses} = -5\text{dBm} - (-38.852 \text{ dBm})$$

$$\text{Losses} = 33.852 \text{ dBm}$$

We can approximate the loss introduced by antennas and cables as follows:

$$L_{Ant} = \text{Losses} / 2$$

$$L_{Ant} = 16.93 \text{ dBm}$$

So, signals effectively measured are 16.93 dBm higher.

Signal received level was measured using the Agilent 89600 Spectrum Analyzer. Figure 4.6 shows one example of the received signal. Although the signal level showed is -78.715 dBm, using L_{Ant} the real received level was -61.79 dBm.

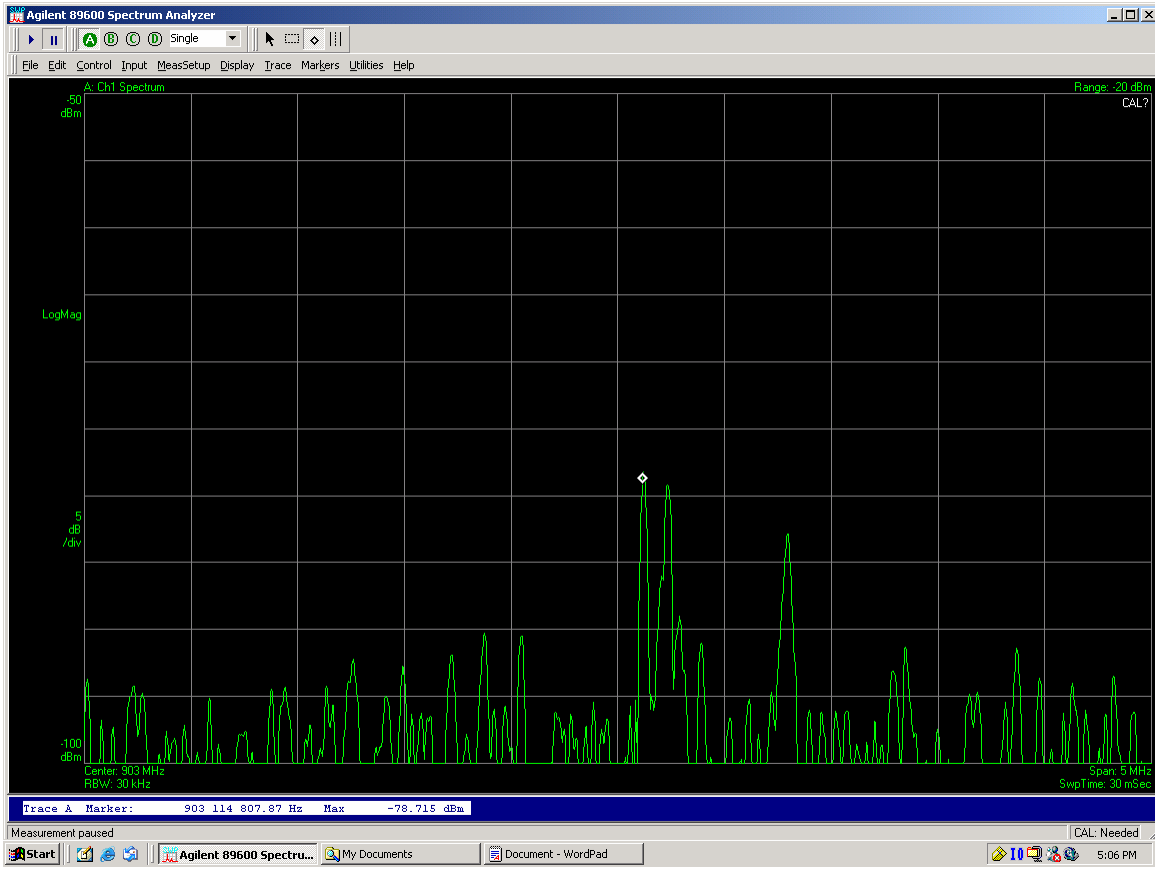

Figure 4.6 Signal level in the receiver, Mote 1

Signals were recorded over the course of an experiment, and figure 4.7 shows a summary of the signal level when the sink was receiving and not. Packet reception quality is -100 when it is receiving perfectly. It is -50 when it's receiving packets but not as frequently as it should be. And it is -10 when the sink is not receiving at all. Signal levels can be seen in the graph. There is

perfect reception for signal values of -63.47dBm and above. But there are also -62.474 dBm signals where sink is not receiving. The received signal didn't show a significantly different level when the signal was received or not, working with the same mote.

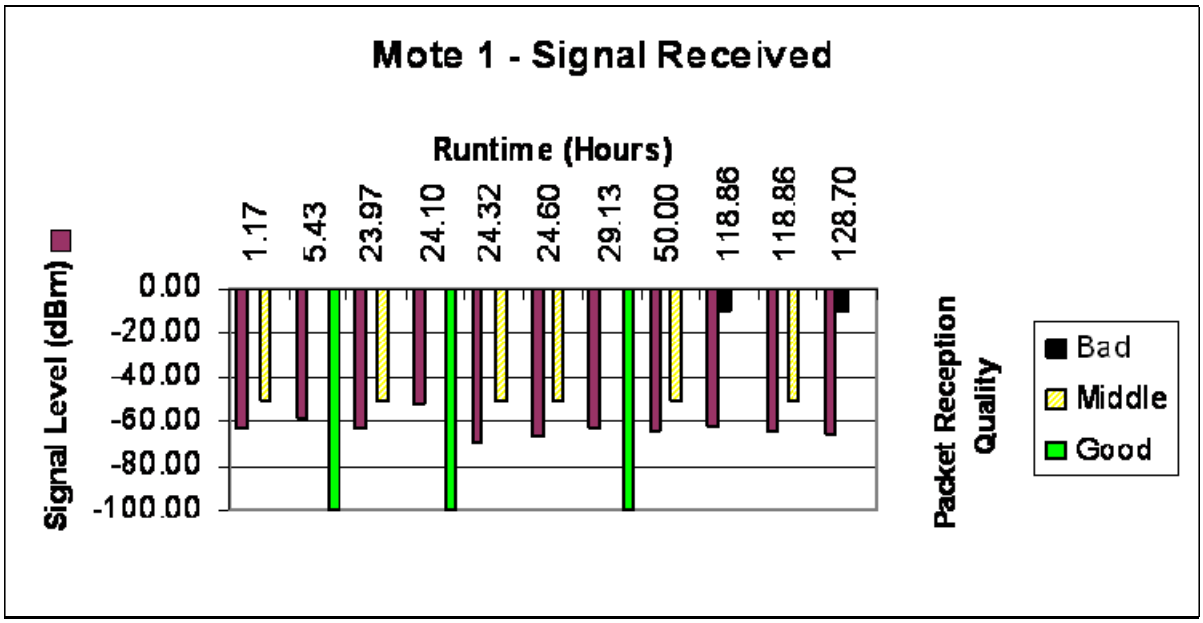

Figure 4.7 Signal received in the sink for mote number one

Signal reception values are very different from the receiver theoretical sensitivity, (-98 dBm) [5]. In summary, reception of data does not appear to stop because of a large change in the signal, or because the transmitter stops transmitting. Additionally, motes which were considered dead could be located closer to the sink, and the sink will begin receiving the packets. Sometimes the nodes needed to be restarted and sometimes this was not necessary. Signal strength tests did not give a lot of information on what was the problem in the behavior of the motes, so other tests were performed.

4.3.2 Noise

Noise was measured according to section 3.7.5. The goal was to look for signals that could be affecting the ones from the motes. Figure 4.8 shows one example of a noise signal with a peak of -89.461 dBm (-72.53 with L_{Ant}), in 902.561 Mhz. The signal is neither constant, nor periodic. Also, noise was measured close to the sink and result summary is shown in table 4.3. There was no correlation found between noise level and the status of the node (dead, alive).

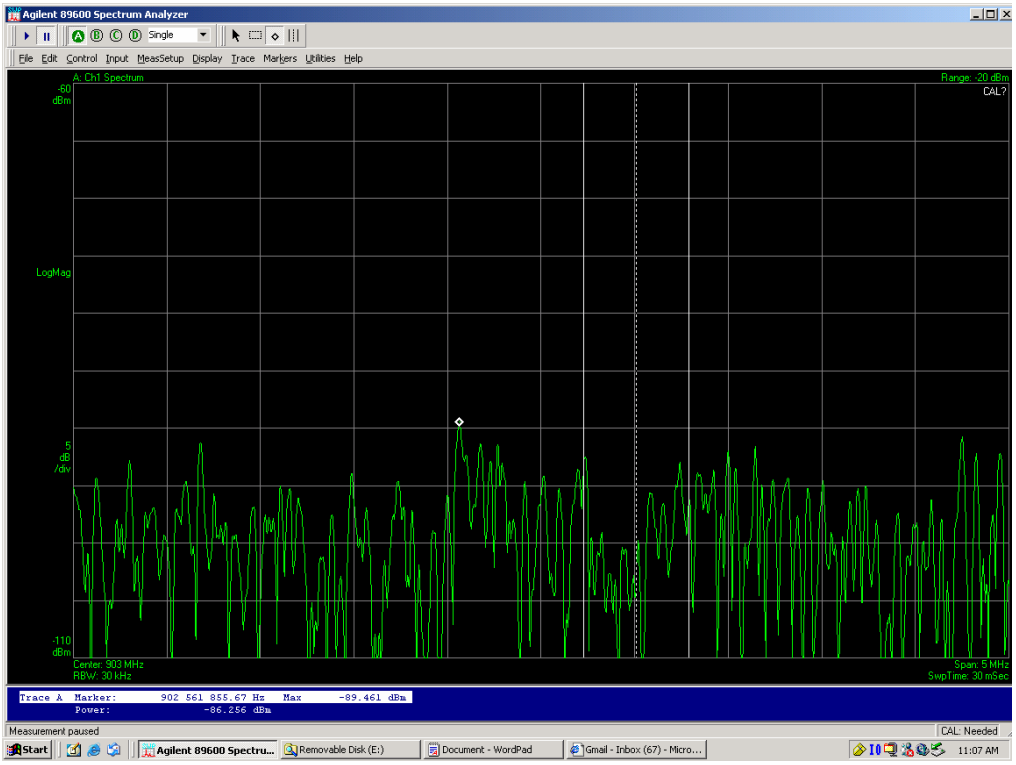

Figure 4.8 Noise close to the transmitting mote

Table 4.3 Noise measured close to transmitting mote and to the sink

	Peak Noise in Transmission (dBm)	Peak Noise in Reception (dBm)
Average	-72.384	-69.534
CI 95%	0.973	0.916

4.3.3 Signal strength measurements indoors

Received signal strength was characterized as a function of separation distance between motes. Figure 4.9 shows received signal level for 5 dBm transmission power. Measurements were done inside the Wireless Laboratory in the SIS Building according to the procedures in 3.7.5. Received signal strength values are in negative dB. The more negative the value, the smaller the received signal level. Path loss can be as high as 50dB. This may be due to multipath propagation into the test area.

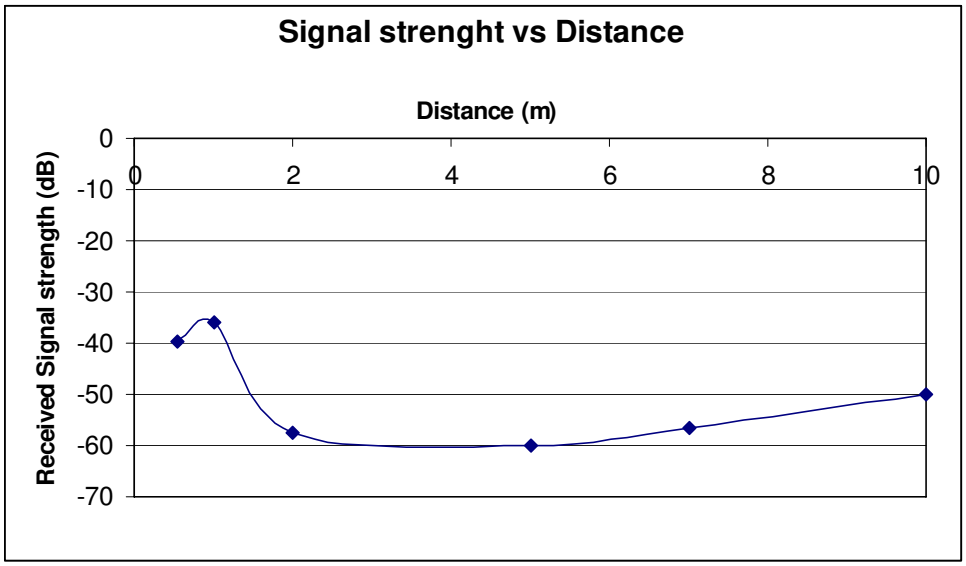

Figure 4.9 Received Signal Strength with distance

4.3.4 Modulated signal analysis

As power levels didn't explain why packets were not received, the next experiments analyzed the modulated signal when the sink could receive information and when it could not. Figure 4.10

presents the received signal waveform when the sink could receive a frame and the mote was close to the sink. The waveform is a clear example of an FSK modulation, where two frequencies can be seen. The first frequency is 111.731 kHz and the second one is 58.514 kHz. In this particular measurement, the peak power is around -50dBm (-33.07 dBm with L_{Ant})

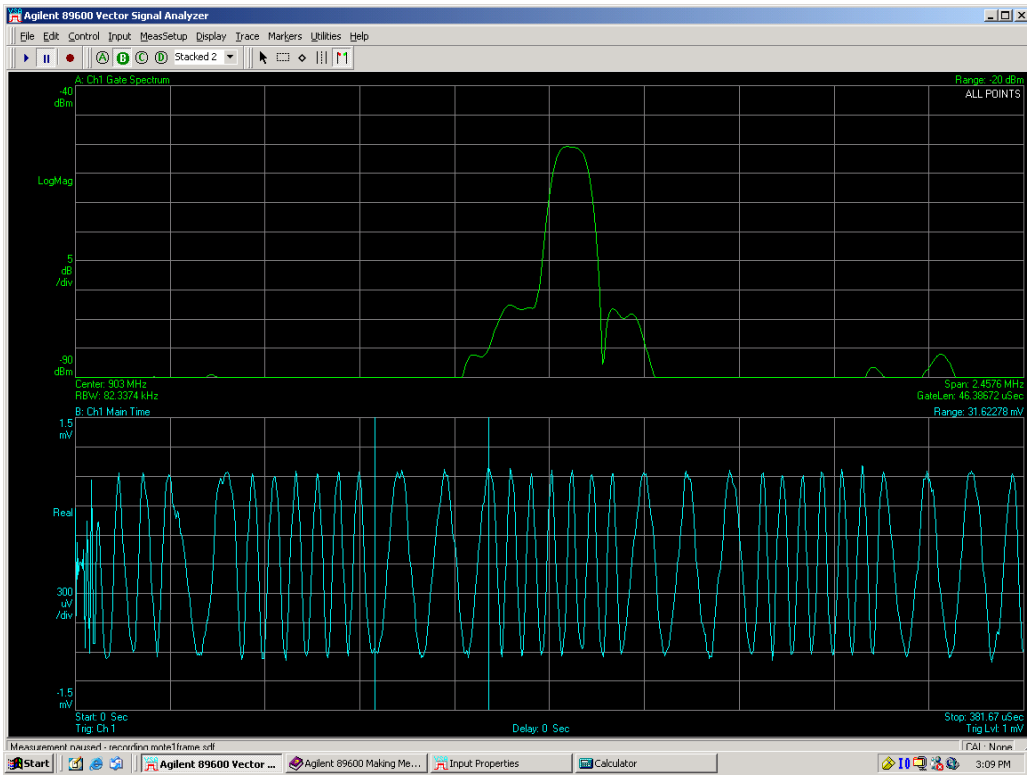

Figure 4.10 Received frame when node is "alive"

When motes were considered dead, they continued transmitting but the sink did not receive the signal until the mote was restarted. Figure 4.11 displays the waveform found using the Vector Signal Analyzer with the transmitting mote close to the sink. The transmitted signal arrives at the receiver at -41.56 dBm with L_{Ant} strong enough to be received, but it is distorted, so the sink can not demodulate it.

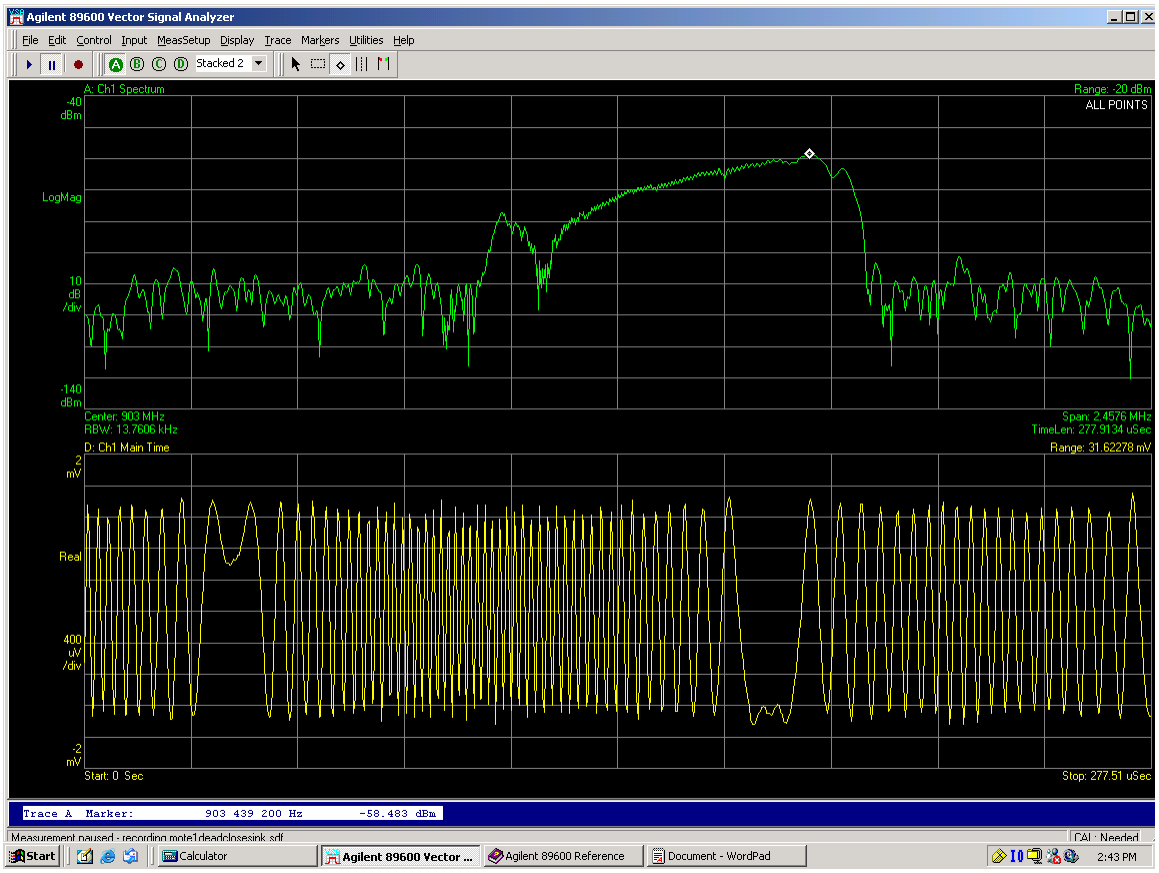

Figure 4.11 Received signal from Mote 1 when packets are not received

Motes keep transmitting distorted signals until a hard reboot is performed on them. After this restart, motes begin sending proper FSK signals again and the sink can receive the frames. Figure 4.12 shows received signal after restarting the mote.

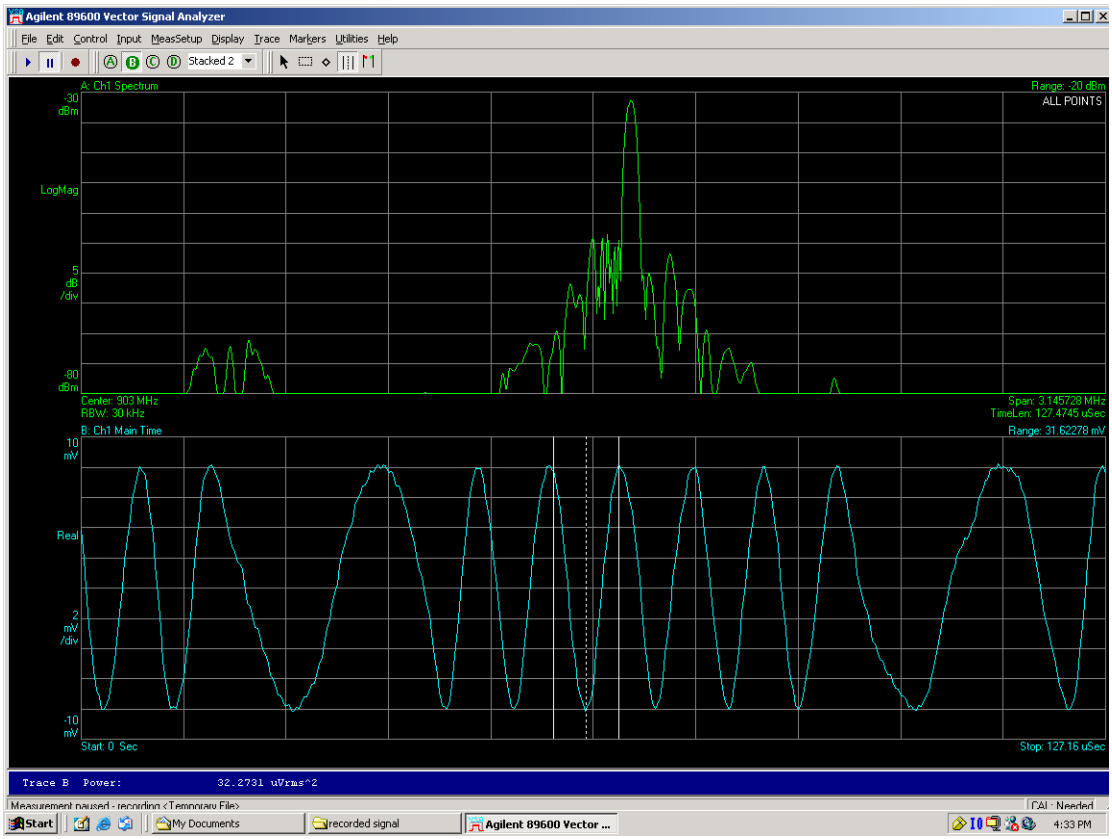

Figure 4.12 Received signal after restarting the mote

4.3.5 Frame analysis

The frame structure was analyzed to determine that it complied with the documentation of nesC programs. Figure 4.13 presents one example of results given by the Demodulation function of Vector Signal Analyzer.

.

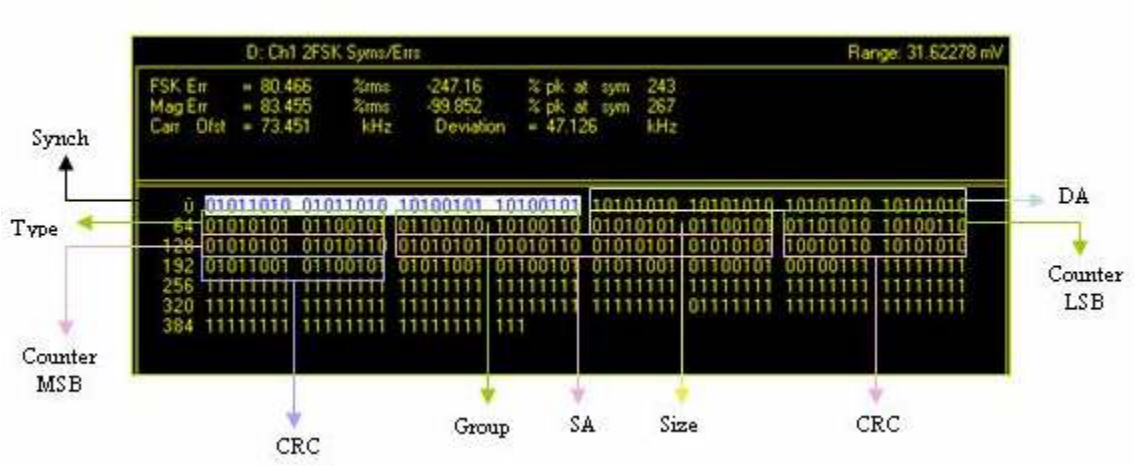

Figure 4.13 Decoded Frame

The frame was found in Manchester Format, where a 0 corresponds to 01 sequence and a 1 corresponds to 10 sequence, according to [5]. The exact information is as follows:

Synch = 01011010010110101010010110100101 = 0011001111001100 = 0x33CC

DA = 10101010101010101010101010101010 = 1111111111111111 = 0xFFFF

Type = 0101010101100101 = 00000100 = 0x04

Group = 0110101010100110 = 01111101 = 0x7D

Size = 0101010101100101 = 00000100 = 0x04

Counter LSB = 0110101010100110 = 01111101 = 0x7D

Counter MSB = 0101010101010110 = 00000001 = 0x01

SA = 0101010101010110 0101010101010101 = 0001 0000 = 0x01 in Little Endian

CRC = 10010110101010100101100101100101 = 1001111100100100 = 0x9F24

Checking with the correspondent file the frame was found, so the frame structure is correct. When the mote is considered dead, the Synch field can not be found, so the frame can not be correctly decoded.

4.3.6 Minimum Supply Voltage

Motes were also tested to determine the lowest supply voltage possible for correct operation. Using equipment showed in 3.7.3 without the resistor, voltage in the mote was directly changed in the power supply to see what was the minimum required. The test was performed for mote number one only, using three levels of transmission power. Figures 4.14, 4.15 and 4.16 illustrate test results. Mote states are: Active when mote is working according to the application programmed in it, NCB (Non-Consistent Behavior) when mote is transmitting but its behavior is not consistent with the application and NT (Non-Transmitting) means the mote is not transmitting and is considered dead.

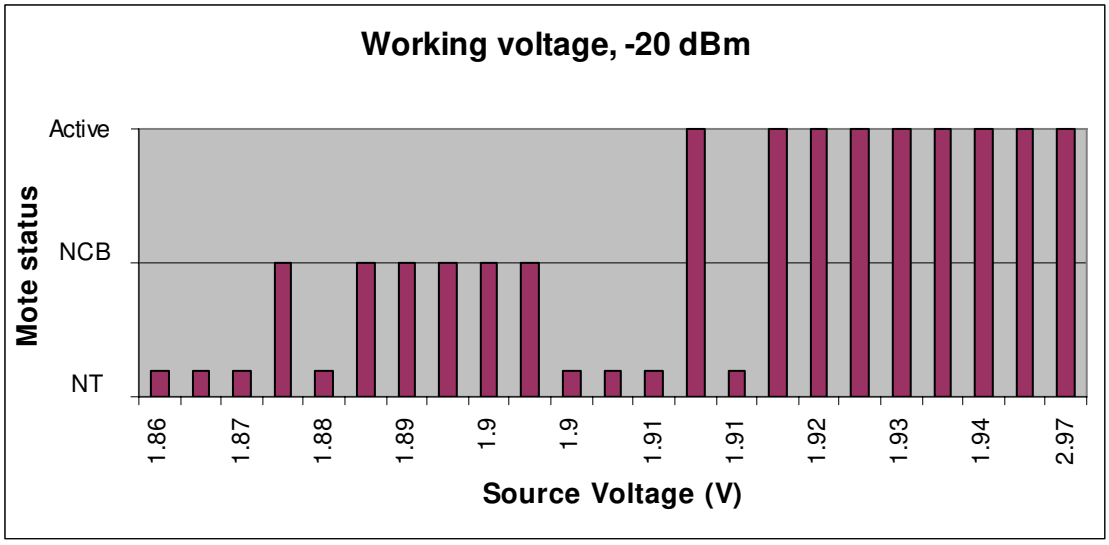

Figure 4.14 Minimum Voltage Supply, -20 dBm

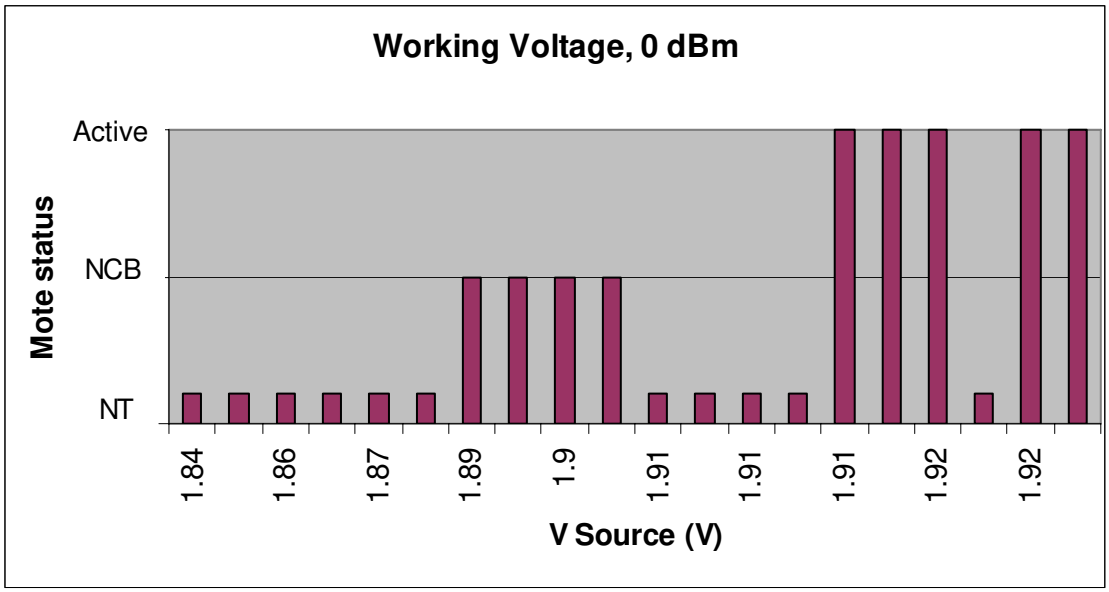

Figure 4.15 Minimum Voltage Supply, 0 dBm

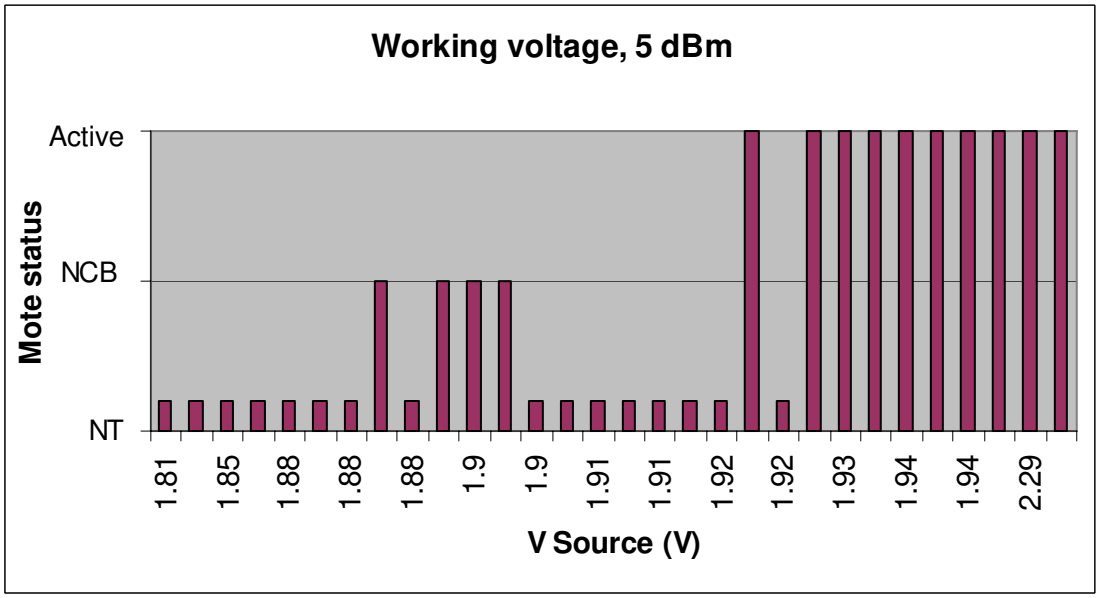

Figure 4.16 Minimum Voltage Supply, 5 dBm

In all cases of transmission power levels, the Active zone voltage supply is above 1.922 V; Non-Consistent zone (where mote is transmitting but its behavior is not consistent with the application) voltage supply goes from 1.898 V to 1.871 V and Non Transmitting voltage supply goes below 1.871 V. Mote transmission reached the same distance as when the mote has a voltage supply according to Crossbow specifications, even if the mote was showing non-consistent behavior. So, transmission power does not depend upon the voltage supply, as long as the voltage is enough for the microcontroller to work and the radio to transmit.

4.4 ENERGY CONSUMPTION

Series 1-4 experiments better defined the death of the motes. Motes appear to hang after some hours and they must be restarted to keep working. One mote was connected to the MIB510 (which is AC powered) in order to avoid battery problems. The mote hung anyway. Searches were done in Internet to find bugs in the applications. There were two commands that should be changed [10] in CC1000RadioIntM.nc, a component that plays a key role in mote radio communications. The bug reported was supposed to be the cause of the radio stack to hang. In the experimental testbed the bug was corrected according to the recommendation in [10], but the problem still remains. So, it seems that lifetime does not depend exclusively on energy. Probable causes are hardware or software problems. The most significant cause may be using object oriented programming which may not be suitable for the particular hardware platform used. nesC uses a lot of existing libraries and they were not validated, because that was outside the scope of this thesis. So, another alternative to estimate network lifetime using GSP and other protocols is to study energy consumption of the mote individually in several states: transmitting, listening, receiving and with the microcontroller on but the radio off. In this order of ideas, the procedure described in section 3.7.3 was used and Figure 4.17 presents one example of the results.

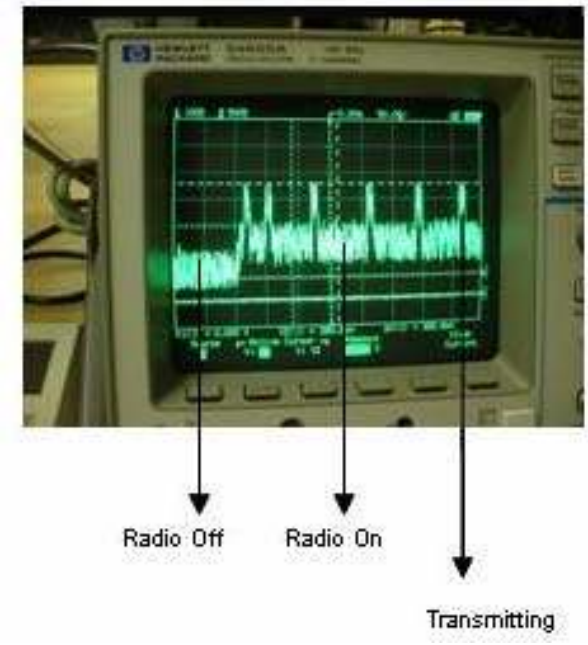

Figure 4.17 Voltage in the resistor, 903 Mhz, 5dBm

Using these waveforms it was possible to identify the periods when a frame was transmitted, when radio was on (but only listening) and off. During frame transmission time, which is 9 msec, there are peaks of current consumption. Figure 4.18 shows the transition from Radio Off to Radio On Transmitting in more detail. The transient seems to have an exponential behavior.

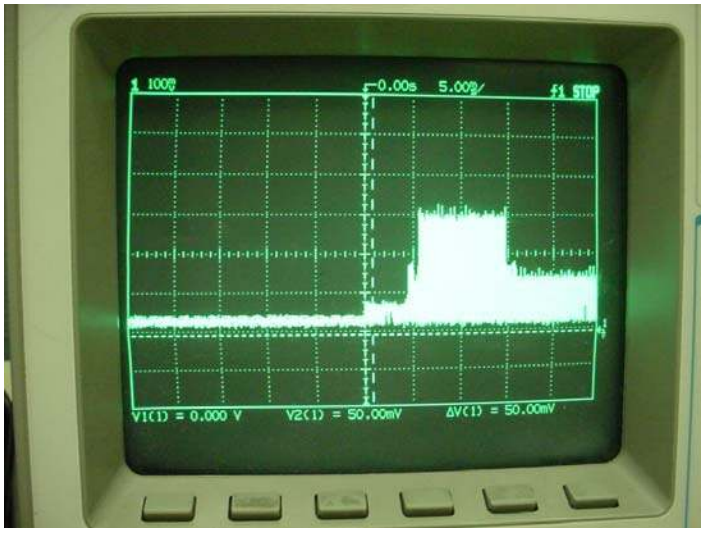

Figure 4.18 Resistor voltage Switching state from Radio Off to Radio On in transmission

Figure 4.19 presents the voltage signal when a frame is sent. Transition from transmitting to listening seems to have also an exponential behavior.

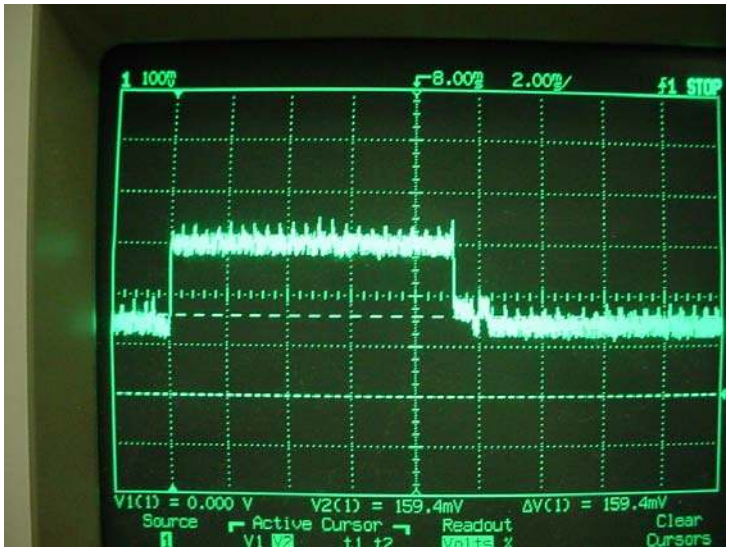

Figure 4.19 Resistor voltage, Frame Transmission

Figure 4.20 shows voltage waveform measured during the receiving state for one mote programmed for just doing the reception. Since the voltage is stable, the mote energy consumption is stable, whether it is receiving or just listening when no frame is sent. The situation was confirmed by waveforms measured in the mote programmed with GSP. So, energy consumption during these two radio states is the same.

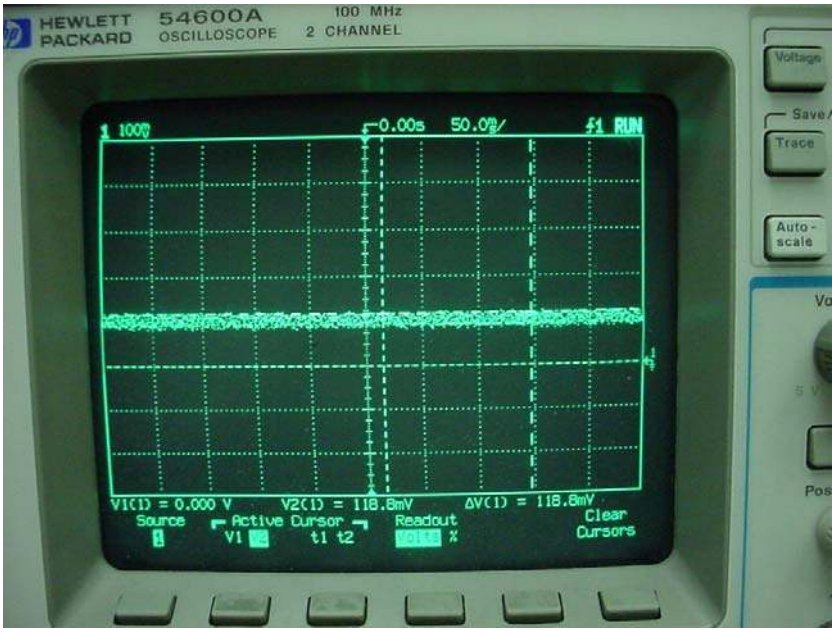

Figure 4.20 Voltage in the resistor, only receiving

Because the signal contained a significant noise component with the oscilloscope measurements, a spectrum analysis was performed in order to see if there was any other significant component in the signal. Figure 4.21 displays the waveform found using the FFT function with HP54600A oscilloscope. Main components are located in 12.45 Hz and multiples of this frequency. That is the frequency of 1 pulse every 80 msec. These are the expected spectral components of a square wave, and there are no other significant components.

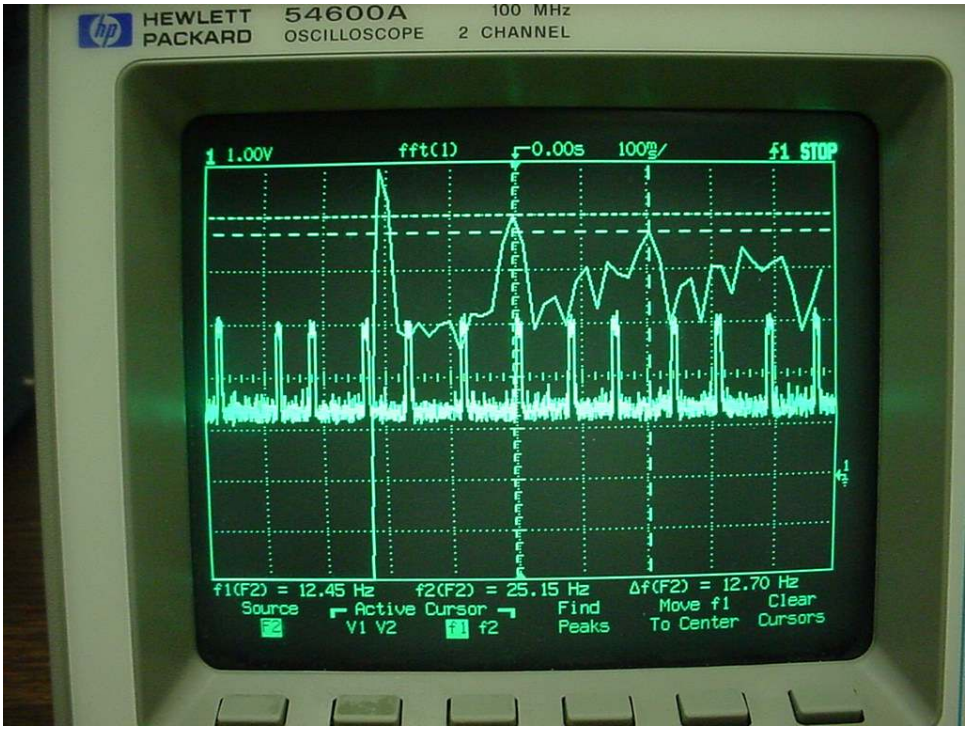

Figure 4.21 Transmitted Signal FFT

Energy consumption measurements were performed for six different motes and the summary can be seen in tables 4.4 to 4.6.

Table 4.4 Voltage in the Resistor

	Radio off (mV)	Radio on (mV)	Transmitting (mV)
Average	59.22	161.35	309.54
CI 95%	1.82	1.80	1.99

Table 4.5 Current through the circuit

	Radio off (mA)	Radio on (mA)	Transmitting (mA)
Average	5.92	16.14	30.95
CI 95%	0.18	0.18	0.20

Table 4.6 Voltage in the transmitting node

	Radio off (V)	Radio on (V)	Transmiting (V)
Average	2.91	2.81	2.66

Figures 4.22 to 4.24 show confidence intervals of current consumption for all motes studied. In all states there are overlapping confidence intervals, so it can not be said that current consumption is statistically independent for all motes in each state.

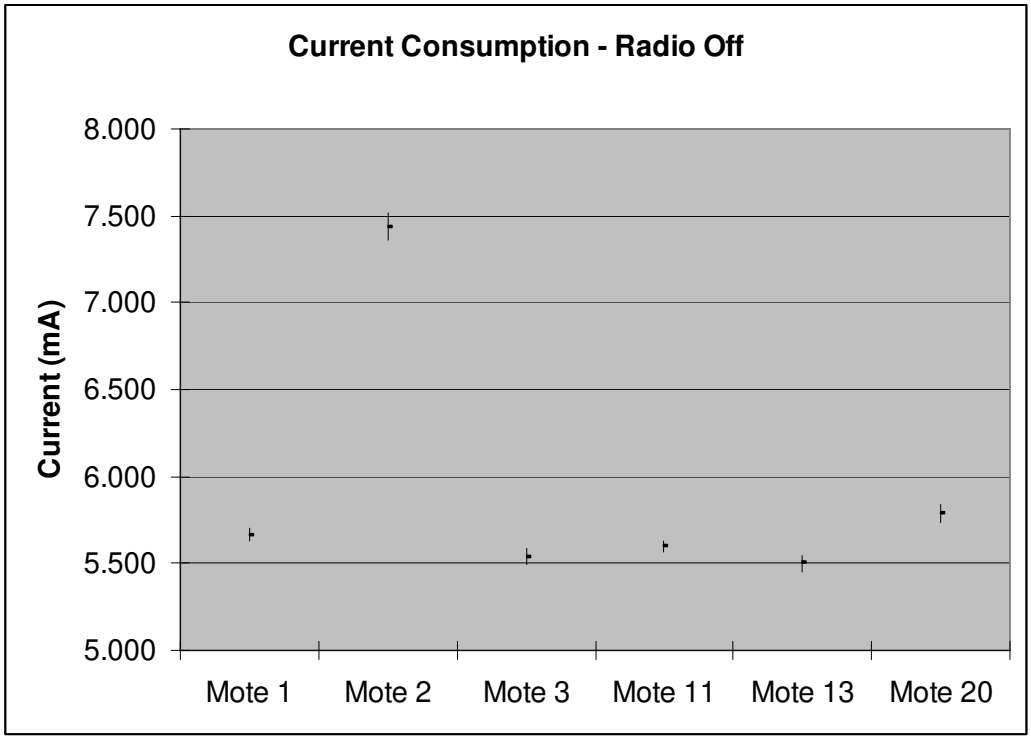

Figure 4.22 Consumption comparison, Radio Off

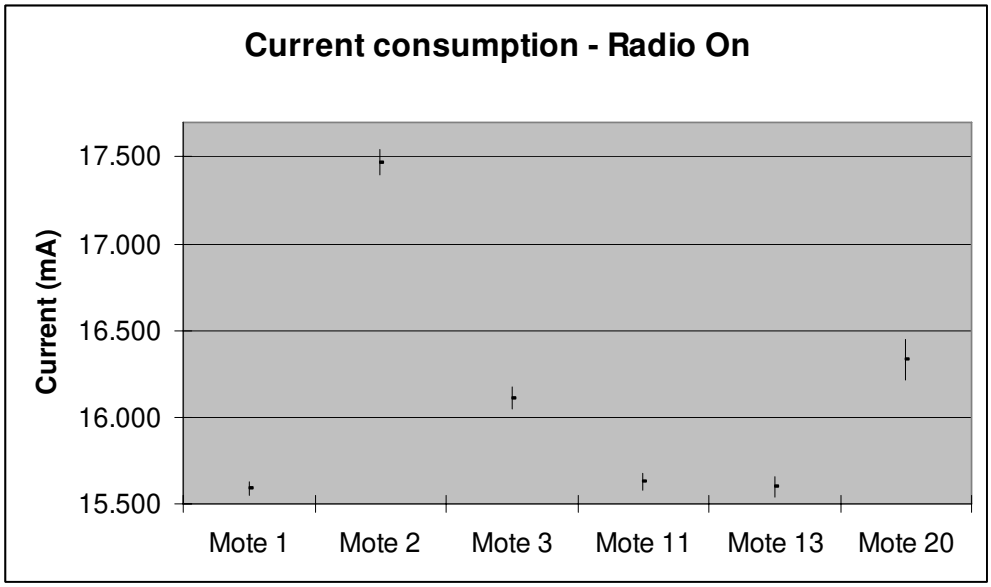

Figure 4.23 Consumption comparison, Radio On

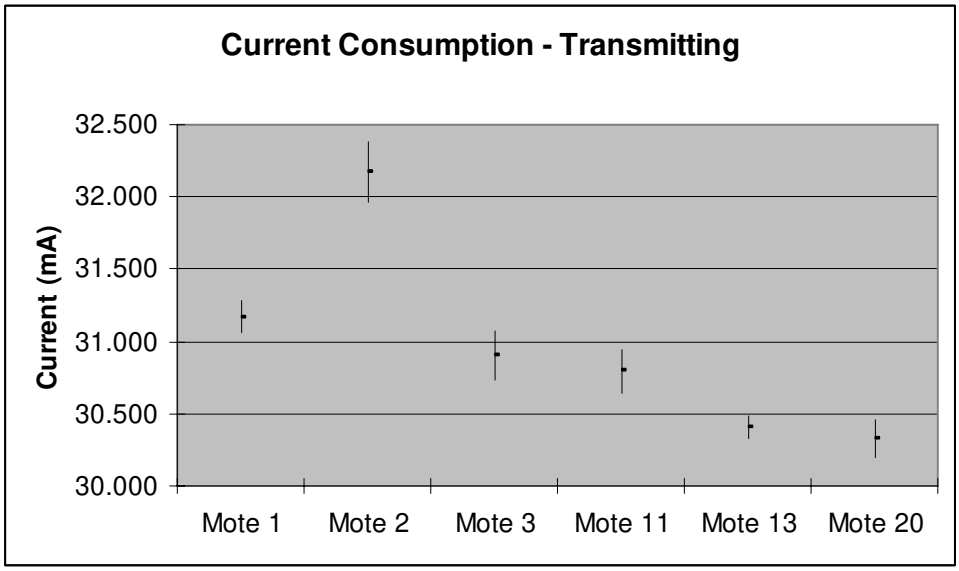

Figure 4.24 Consumption comparison, Transmitting

Figure 4.25 presents power consumption for all motes in all radio states.

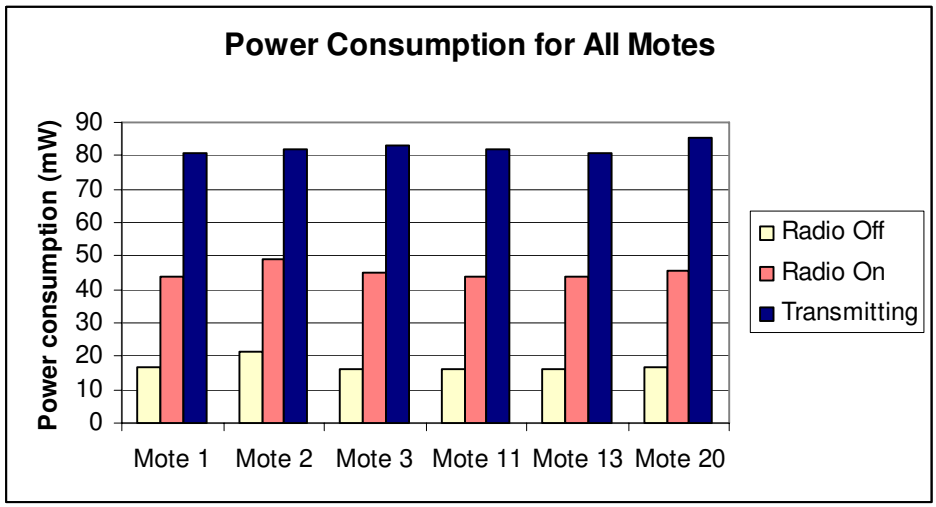

Figure 4.25 Power Consumption for all motes

Average results showed in tables 4.4 to 4.6 were used to find energy consumed by one mote, using the following procedure:

$$Energy = W * t \qquad (4.1)$$

Or

$$Energy = V * I * t \qquad (4.2)$$

In the Radio Off state, energy consumption is:

$Energy_{Radio\ Off}$ = 2.91 V * 5.92 mA * t

$$Energy_{Radio\ Off} = 17.23 * t \ \ (mjoules) \qquad (4.3)$$

Where t is the time in seconds when the radio was off. The protocol used in the experiments had the radio off half of the time, so in one hour this part of the system worked 1800 seconds and consumed 31.01 joules or equivalently, 2.96 mAh, using expression (2.10) and mote voltage equal to 2.91 V. In the Radio On state, without transmitting, a similar procedure can be followed, using equation (4.2):

$Energy_{Radio\ On}$ = 2.81 V * 16.14 mA * t

$$Energy_{Radio\ On} = 45.35 * t\ (mjoules) \qquad (4.4)$$

Where t is the time during which the radio was on but no transmitting. The node was sending one message every 80 msec and the message duration is 9.0 msec, so the radio is on without transmitting during 71 msec in one cycle. During the 2.5 seconds for the radio being on, there are 31.25 cycles of transmit one frame and listening.

31.25 cycles * 71 msec = 2.2 sec.

So, proportion of the radio on but just listening is

2.22 sec / 2.5 sec = 88.75 %

According to this, in one hour there are 1597.5 seconds (88.75% of half an hour, because of the 50% duty cycle) where the radio is on but not transmitting. So, energy spent listening in one hour is 72.45 joules using expression (4.4) or 7.16 mAh. We can also find energy consumed in receiving a bit knowing the bit duration, which is 52 μsec and using the approach followed by Hill *et. al* in [23]. Using expression (4.4), the result is:

$Energy_{Receiving} = 45.35 * 52 * 10^{-6}$ mjoules per bit

$Energy_{Receiving}$ = 2.36 μjoules per bit

Following similar procedures for transmission, the following expression is obtained

$$Energy_{Tx} = 82.33 * t \text{ (mjoules)} \qquad (4.5)$$

Where t is time in seconds when the radio was transmitting a frame. In this implementation, the time is 9.0 msec, so energy consumed during the transmission is 740.97 μjoules for each frame. In 2.5 seconds, the mote is sending 31.25 frames. That is 0.28 seconds of transmitting in each 2.5 second cycle. That is 11.25% of the time spent in transmission. In one hour that is 202.5 seconds, which means the mote consumes 16.67 joules (because of the duty cycle) or 1.74 mAh. Energy for transmitting one bit can be found using expression (4.5):

$Energy_{Tx}$ per bit = $[82.33 * 52 * 10^{-6}]$ mjoules per bit

$Energy_{Tx}$ per bit = 4.28 μjoules per bit

Table 4.7 Summary of Energy consumption

State	Radio Off (mjoules)	Radio On (mjoules)	Transmiting (mjoules)	Energy per bit Tx (µjoules)	Energy per bit Rx (µjoules)
Expression	17.23 * τ_1	45.35 * τ_2	82.33 * τ_3	4.28	2.36

Values for Energy per bit are based in the time it takes to send a bit, so they depend upon the data rate. These values are only valid for 19.2 kbps. Figure 4.26 presents energy consumption for each mote individually running GSP.

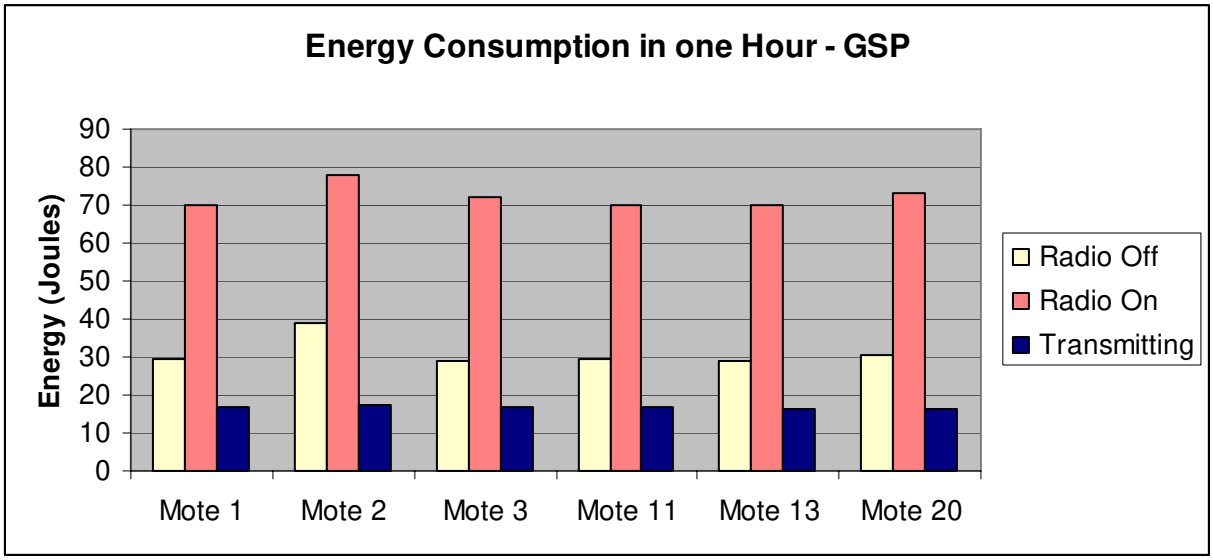

Figure 4.26 GSP Energy consumption in one hour

Total Energy consumption of one mote can be calculated using the expression:

$$E = \sum_{i=1}^{3} P_i \tau_i \qquad (4.6)$$

Where Pi is the expression for power consumption in each state of the mote and τi is the time in seconds spent by the mote in each state. 1 is for Radio Off, 2 for Radio On in the listening or receiving mode and 3 for Transmitting. So, energy consumption for GSP and this particular application in one hour, with each mote running individually, can be found adding the values previously calculated:

Energy GSP in one hour = 31.00 joules + 72.45 joules + 16.67 joules
Energy GSP in one hour = 120.12 joules

Energy consumption can be compared in similar applications. As an example, Shnayder *et. al* in [26] measure and simulate energy consumption of several applications. One of them is similar to the one used here, and is called CntToRfm. The application sends a counter through the radio and it consumes 1.985 Joules in 60 seconds. Authors don't specify packet rate, packet size, transmission power or transmission rate, so the comparison may not be exact. Nevertheless, it can be found that in one hour, that application would consume 119.1 Joules, almost the same as the GSP application run here. CntToRfm does not use any routing protocol at all because it is only intended to transmit the counter from one source to one destination, in one hop. The current consumption found in [26] is also similar to the one found here. Table 4.8 presents a summary.

Table 4.8 Current consumption comparison with [26]

	Radio Off (mA)	Radio On (mA)	Transmitting (mA)
GSP project	5.92	16.14	30.95
Results in [26] *	8.0	15.0	21.8

* Note: In [26] the results are not shown for the exact same situation as in this experimental testbed. That can be seen in Table 2.5. So, the Radio Off value is the one showed in Table 2.5 as the active CPU value. The Radio On value is the one showed for Reception plus active CPU in Table 2.5. The closest transmission power value showed in Table 2.5 is 6 dBm, so,

the current consumption showed in this table is the one for 6 dBm and it also includes active CPU and radio transmission values.

It is also possible to compute energy spent by the microcontroller executing one instruction, considering the completion of roughly one instruction every clock cycle. With an 8 Mhz clock, that means one instruction takes 125 ns. Using the value in S_1 (Radio Off, just the microcontroller working), 17.23 * 125 ns = 2.15 njoules.

4.5 DIFFERENT TRANSMISSION POWER LEVELS

All procedures followed in the previous section considered the worst case, when motes are transmitting with the maximum transmission level. Two other transmission power levels (0 dBm and -20dBm) were tested, to know if they had effects over energy consumption. These levels were chosen because they were the middle and smaller transmission power levels that can be programmed in one mote, according to [7].

4.5.1 Results using 0 dBm

Figure 4.27 presents voltage in the resistor for 0 dBm transmission power. Waveforms for 0 dBm are similar to the ones found for 5 dBm, but transmission voltage levels are smaller. Tables 4.9 to 4.12 show the measurement summary; energy consumption for 0 dBm is lower than the one calculated for 5 dBm.

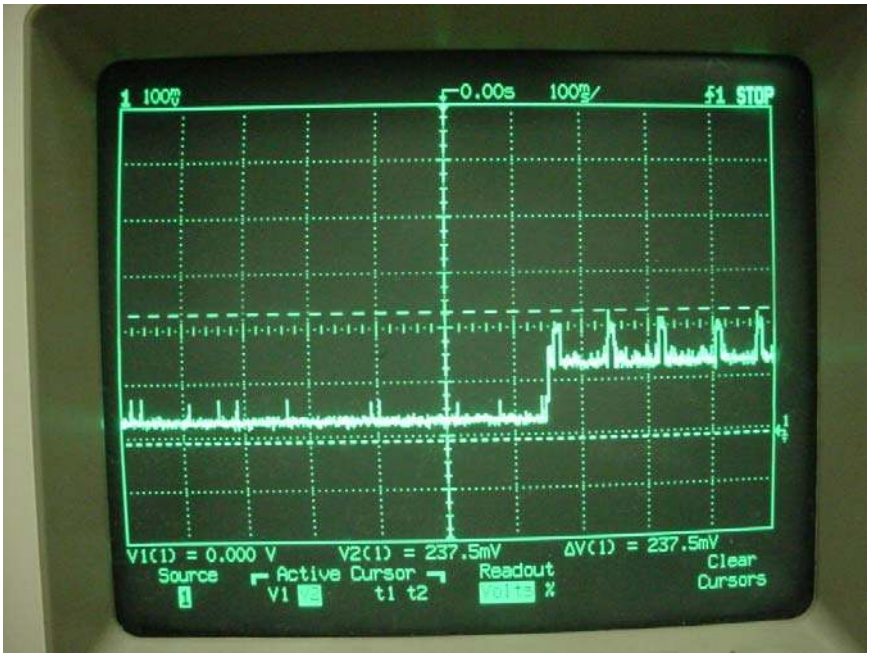

Figure 4.27 Voltage in the Resistor, 0 dBm

Table 4.9 Voltage in the resistor, 0dBm Transmission Power

	Radio off (mV)	Radio on (mV)	Transmitting (mV)
Average	42.97	149.85	213.37
CI 95%	0.27	0.38	0.58

Table 4.10 Current through the circuit, 0 dBm Transmission Power

	Radio off (mA)	Radio on (mA)	Transmitting (mA)
Average	4.30	14.99	21.34
CI 95%	0.03	0.04	0.06

Table 4.11 Voltage in the node, 0 dBm Transmission Power

	Radio off (V)	Radio on (V)	Transmiting (mV)
Average	2.93	2.82	2.76

Table 4.12 Node power consumption

	Radio off (mW)	Radio On (mW)	Transmitting (mW)
Min	16.54	42.20	58.73
Average	16.61	42.30	58.88
Max	16.69	42.41	59.03

4.5.2 Levels using -20dBm

Figure 4.28 presents results found using minimum transmission power, where the waveform is not as well defined as before. Energy consumption is clearly different when the radio is on and off, but there are no periodical peaks that can be identified without a doubt as frame transmissions. Within the limits of the measurements, transmission and reception consume energy at the same rate when -20 dBm transmission power is used.

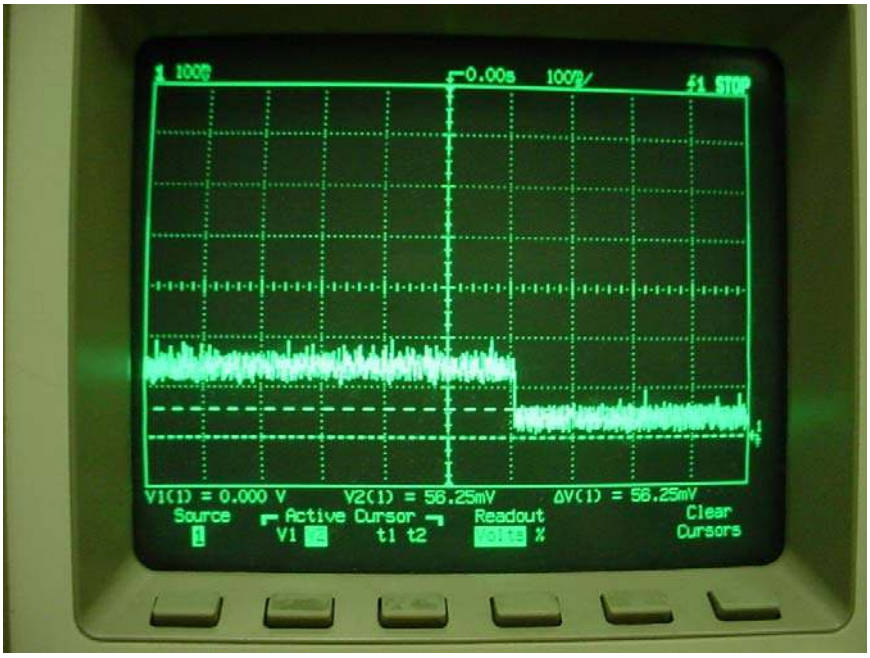

Figure 4.28 Voltage in the resistor, -20 dBm

Tables 4.13 to 4.16 summarize the measurements. Current levels are very similar to the first case (power transmission equal to 5dBm). But, as there are no peaks in transmission, with -20dBm energy consumption is smaller. Another important result found with these measurements is with this transmission power, the transmitter and the receiver should be approximately 25 cm apart from each other for assuring good reception.

Table 4.13 Voltage in the resistor, -20dBm

	Radio off (mV)	Radio on (mV)
Average	56.56	160.33
CI 95%	0.24	0.51

Table 4.14 Current through the circuit, -20 dBm

	Radio off (mA)	Radio on (mA)
Average	5.66	16.03
CI 95%	0.02	0.05

Table 4.15 Voltage in the node, -20 dBm

	Radio off (V)	Radio on (V)
Average	2.92	2.81

Table 4.16 Node power consumption, -20 dBm

	Radio off (mW)	Radio On (mW)	Transmitting (mW)
Min	16.43	44.96	44.96
Average	16.50	45.10	45.10
Max	16.56	45.23	45.23

Figures 4.29 and 4.30 illustrate the difference in power and energy consumption for the three radio states and the three transmission power levels. Figures show ranges found with 95% C.I. Total energy consumption in one hour is very similar for -20dBm and 0 dBm: 110.9 and 109.4 Joules respectively. For 5 dBm, total energy consumed is 125.5 Joules.

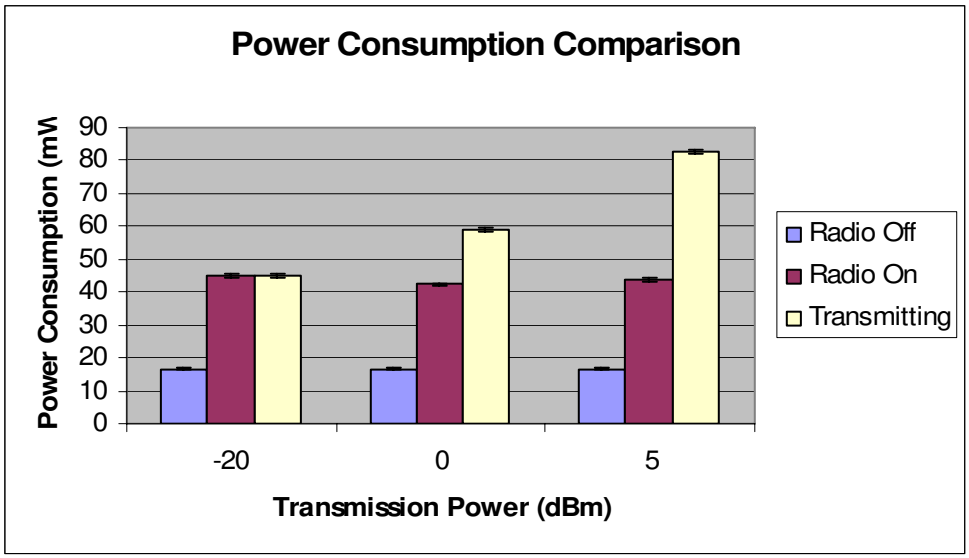

Figure 4.29 Power consumption comparison

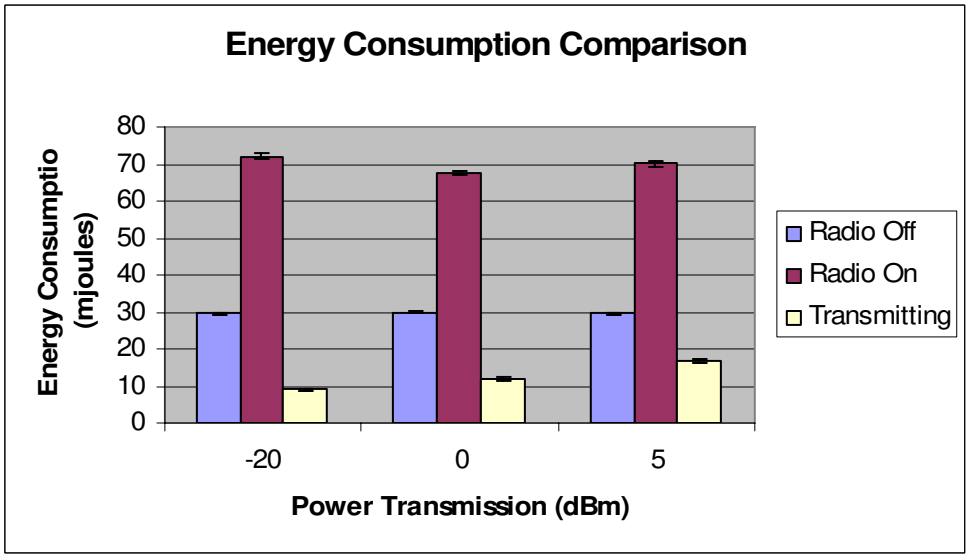

Figure 4.30 Energy Consumption in one hour for GSP, Radio On

4.6 CALCULATION OF EXPECTED LIFETIME

Using the energy consumption rate values, it is possible to estimate sensor network lifetime for different protocols, as long as times they spent in the different states are known. In the previous section calculations were shown for knowing the consumption of one node running GSP during one hour. When there are several nodes running GSP, calculations are different because of the flooding effect inherent to the protocol. Nevertheless, it is possible to find an upper limit for the energy consumption of the motes, and, together with the discharge curve of the battery, estimate the lifetime. In this section, a worst-case analysis is done for estimating mote lifetime in the specific application used for the experimental testbed.

Originally, the application will send a packet every 80 msec. Every packet will take 9.0 msec to be sent or to be received. When the node receives a packet, it will retransmit it immediately. As shown in Figure 3.3, there is a "Time frame", necessary to transmit the frame and enter the listening mode. The situation can be expressed as:

$$TF = B + SwTx + TxTime + SwRx \qquad (4.7)$$

Where

B = Backoff time

SwTx = Time to switch the radio to transmit mode. Typically 250 μsec according to [16]

TxTime = Time to transmit a frame

SwRx = Time to switch the radio to receive mode. Same value as SwTx

Considering that Flooding will make the mote receive a big amount of packets, the worst case of energy consumption can be calculated assuming that the mote will receive and retransmit as many packets as possible when the Radio is On. Figure 4.31 represents this situation.

B	SwTx	TxTime	SwRx	RxTime	B	SwTx	TxTime	SwRx

Figure 4.31 Communication for worst case energy consumption - GSP

In the worst case, there is a cycle that is repeated during the whole T_{ON} of the Duty Cycle. The cycle time T is calculated as follows:

T = B + SwTx + TxTime + SwRx + RxTime or, better:

$$T = B + 2SwTx + RxTime + TxTime \quad (4.8)$$

Where all times mean the same as before, and RxTime is the time it takes to receive a frame. The number of cycles in one T_{ON} is as follows:

$$N = T_{ON} / T \quad (4.9)$$

Knowing that B, SwTx and RxTime belong to the RadioOn state, the time for each state can be found this way:

$$Time\ for\ S_1 = T_{OFF} \quad (4.10)$$

$$Time\ for\ S_2 = N * (B + 2SwTx + RxTime) \quad (4.11)$$

$$Time\ for\ S_3 = N * TxTime \quad (4.12)$$

Expressions number (4.10), (4.11) and (4.12) together with (4.8) can be used to find the worst case of Energy consumption in GSP protocol in one duty cycle, as follows:

$$E(t)_{one\ duty\ cycle} = S_1(Toff) + S_2\ (N * (B + 2SwTx + RxTime)\ ) + S_3\ (N* TxTime) \quad (4.13)$$

Knowing how many duty cycles there are in one hour, and converting Energy to current using expression number (2.10):

$$E(t) \quad \frac{D}{3600} x \sum_{i\ 1}^{3} \frac{S_{idutycyle}}{V_i} \qquad (4.14)$$

Where D is the amount of duty cycles in one hour and V_i is the voltage for the node in each state. Equation (4.14) will give energy consumption in mAh.

Analyzing the battery discharge curve shown in figure 2.2, the curve remains linear around 1.28 volts, and when the battery has spent near 80% of its capacity (160 mA x 8 hours or 320 ma x 4 hours), the voltage drops. Applying the same criteria to any capacity C, measured in mAh, of batteries used, a general expression for lifetime is:

$$L = 0.8 \; x \; C \; / \; E(t) \qquad (4.15)$$

Using these expressions with all data gathered in the experiments, the expected lifetime of GSP for our particular set of parameters can be calculated. Assuming the smallest backoff value (0.416 msec), the worst case of lifetime would be as follows:

T = 0.416 + 2 * 0.25 + 9.0 + 9.0

T = 18.916 msec

N = 2.5 sec / 18.916 msec

N = 132.16

D = 3600/5

D =720

$$E(t) \quad \frac{720}{3600} \; \frac{17.23 * 2.5}{2.91} \; \frac{45.35 * 132.16 * 9.916 * 10^{-3}}{2.81} \; \frac{82.33 * 132.16 * 9.0 * 10^{-3}}{2.66}$$

E(t) = 14.55 mAh

If battery capacity is 1200 mAh, the expected lifetime of the node is:

L = 0.8 * 1200 mAh / 14.55 mAh

L = 65.96 hours

In the worst case, each node has an expected lifetime of 65.96 hours. Calculation can be closer to the real situation by using the actual time that mote spends in Radio On and Off states. Measurements show that average time spent in Radio Off state is 2.43 seconds, while time spent in Radio On state is 2.48 seconds. As difference between expected and actual values is very small, the difference in lifetime is also very small.

5.0 CONCLUSIONS AND FUTURE WORK

Experiments show that sensor node lifetime is not solely limited energy storage. Sensor lifetime can also be limited by software and other hardware components, however these experiments did not show which one is the prevalent cause. To estimate sensor lifetime, should it be limited by energy storage capacity, measurements of energy consumption rate were taken for Mica2 motes using GSP for routing and a CSMA. Results are limited to specific experiment parameters. Nevertheless, they provide guidance in predicting the energy consumption of a node and they can be used as a base for obtaining the total consumption of one node in a wireless sensor network.

Measured results were used to derive a general expression of energy consumption in GSP protocol, but they can be extended to other protocols, provided that all times the radio spends in different states are known. Comparing to the analytical model presented by Heinzelman *et. al* in [24], energy consumption found in this work is two orders of magnitude larger, including the consumption of the microcontroller. Contrary to the ideas in [24] measurements show that energy consumed in transmitting one bit is almost twice the energy consumed in receiving a bit. The difference between the measurement model presented here and the Heinzelman model is that these measurements use a fixed energy level for transmissions. The Heinzelman model implicitly assumes perfect power control, i.e., a perfect knowledge of distance between the sender and receiver, a continuously variable transmission power level and no channel fading.

In this platform, CPU energy consumption in active state is three orders of magnitude smaller than energy spent in transmitting or receiving a bit. So, in this case CPU can execute roughly one thousand instructions to spend the same energy as it is spent in one bit communication.

Energy consumption differs when different transmission power levels are used. In particular, energy consumed with protocols in the experimental testbed for the maximum power level is higher than medium and minimum levels, by around 15 joules in one hour. Energy consumption for intermediate and lower levels is very close (109.4 and 110.9 joules in one hour, respectively).

Experiment settings in this work allow a more precise analysis and estimation of lifetime than those found by Anastasi *et. al* in [17]. Oscilloscope is an adequate instrument for measuring short duration signals in different frequencies, as the ones expected when the message frequency and message duration are less than 16 msec. Most voltmeters are calibrated to measure signals only in DC or in AC, so the averages found in [17] may not be adequate.

Results in this work are comparable to the ones presented by Shnayder *et. al* in [26]. Energy consumption of similar applications was found to be very close, even though application in [26] does not use a routing protocol. Although the comparison may not be exact, the result is positive for GSP, and can be explained by the fact that GSP does not include any overhead either in packet format or in control packets.

The measurements in this study validate the theoretical studies for GSP. The measurements show that GSP can be implemented in a fully functional way in a commercial platform for a sensor network. This implementation is not difficult and does not require any tuning, setup phase or special operation after deployment, even after death of some of the nodes. The nodes do not require special hardware for executing GSP. Each of which is a desirable characteristic for a routing protocol in a sensor network.

Also, this study shows that under specific parameters, like the ones used in this testbed, synchronization for nodes in GSP is not required, since most packets arrived to the sink when the nodes were functioning properly. GSP avoids clustering and defining other specific roles for nodes, functions that can generate a considerable overhead for some routing protocols.

Future work may include measurements with different parameters, for frequency, transmission power levels and load in the microcontroller. Additionally, other platforms should be considered, to create a more general model. Also, a more accurate model for battery behavior should be included, since 0.8 * C is a starting point but does not reflect the available capacity under a varying current load.

Additional future work may include the study of MAC protocols for complementing GSP functionality. The experimental testbed used a generic CSMA/CA protocol and a more careful desgined MAC protocol may improve GSP communication performance and energy consumption. Also, using techniques such as source address checking, GSP can be improved so as to limit the flooding effect. In order to minimize the complexity of this scheme, one node can check if the packet it received is from the node itself, and decide not to forward it.

Although this tests were performed using only one sink, for GSP there is no constraint in the number of sinks that can be used in the network. Future work may include an implementation with several sinks and analysis of GSP performance for that situation.

APPENDIX A

NESC APPLICATION

Applications modified for implementing GSP:

\opt\tinyos-1.x\tos\platform\mica2\CC1000RadioIntM.nc

\opt\tinyos-1.x\tos\platform\mica2\CC1000RadioC.nc

\opt\tinyos-1.x\contrib\xbow\apps\CntToLedsAndRfm\ CntToLedsAndRfm.nc

\opt\tinyos-1.x\tos\lib\counters\ Counter.nc. Here, counter frequency is changed and source address and counter are stored in the frame data field.

\opt\tinyos-1.x\tos\lib\counters\ IntToRfmM.nc. The code was modified to place the original address in the frame

\opt\tinyos-1.x\tos\interfaces\IntOutput.nc

RfmToLeds was used for the sink, changing the following:

\opt\tinyos-1.x\tos\lib\counters\RfmToInt.nc

\opt\tinyos-1.x\tos\lib\counters\RfmToIntM.nc

C:\tinyos\cygwin\opt\tinyos-1.x\tools\java\net\tinyos\tools\Listen.java

Code wiring for GSP implementation:

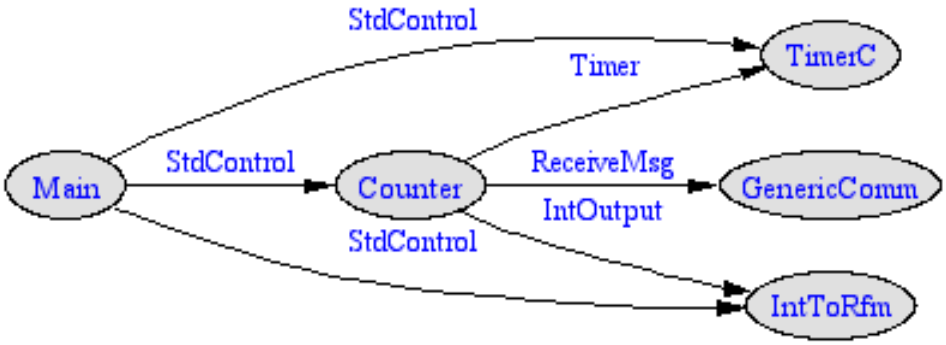

Figure A-1 CntToLedsAndRfm

Figure A-2 IntToRfm

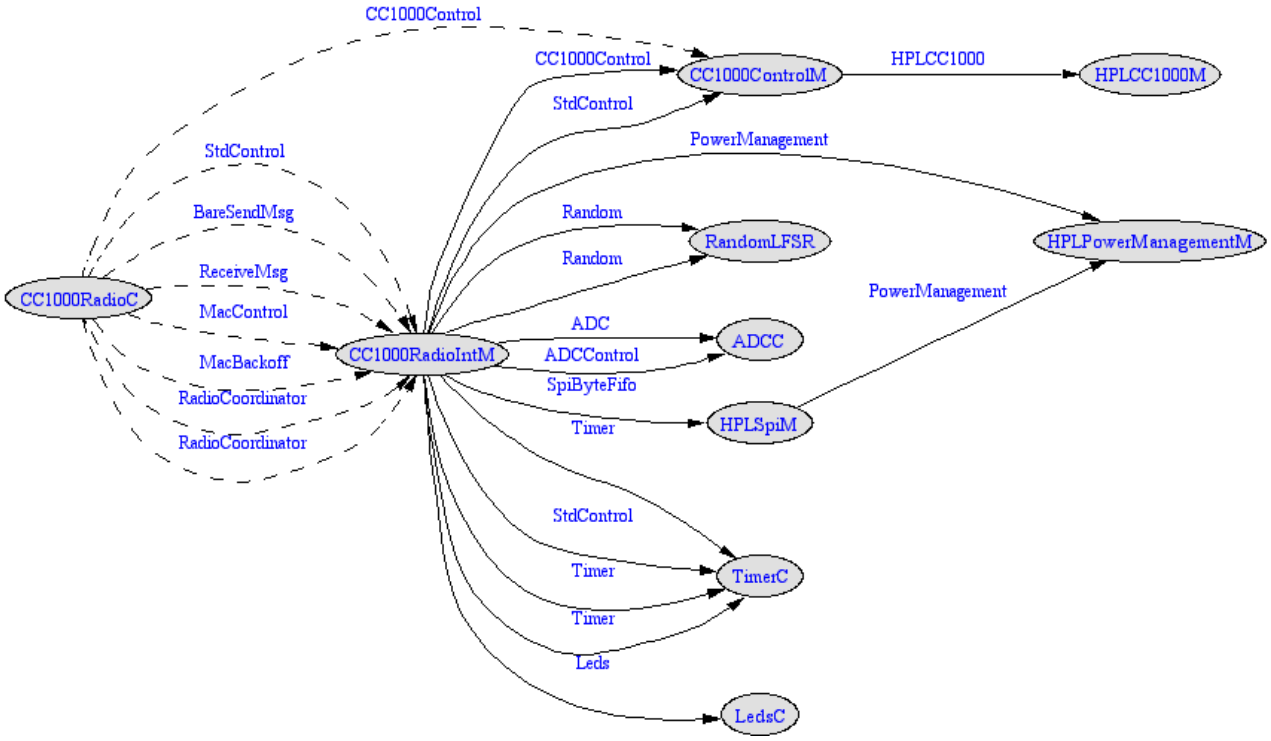

Figure A-3 CC1000RadioC

Sink:

Figure A-4 RfmToLeds

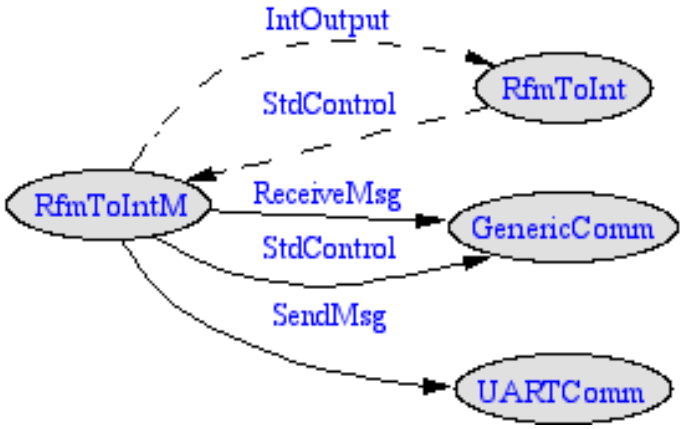

Figure A-5 RfmToInt

APPENDIX B

TYPICAL EXAMPLE APPLICATION IN NESC: Blink

Blink is an application to turn the red LED of the mote on and off at 1 Hz rate. Blink has two components: one configuration (called Blink.nc) and one module (called BlinkM.nc). The code for the configuration is as follows [34]:

```
configuration Blink {
}
implementation {
 components Main, BlinkM, SingleTimer, LedsC;
 Main.StdControl -> SingleTimer.StdControl;
 Main.StdControl -> BlinkM.StdControl;
 BlinkM.Timer -> SingleTimer.Timer;
 BlinkM.Leds -> LedsC;
}
```

Every configuration must have the word "configuration" at the beginning of the file. After that, the components that are used in the application are showed (Main, BlinkM, SingleTimer, LedsC). Every TinyOS application should have a Main. The Main.StdControl.init() command is the first command executed in a TinyOS application. A component can be initialized with the init() command, executed for the first time with the start() command and stopped (turned off, for example) with the stop() command. All components are connected or "wired" so the component at the left of the arrow uses an interface and the component at the right side of the arrow provides (implements) the interface [34].

The other component of Blink is the module. The code is as follows, according to [35]. Comments added explain the functionality of every code part.

```
module BlinkM {
 provides {
```

```
  interface StdControl;
 }
 uses {
  interface Timer;
  interface Leds;
 }
}
implementation {

 /**
  * Initialize the component.
  *
  * @return Always returns <code>SUCCESS</code>
  **/
 command result_t StdControl.init() {
  call Leds.init();
  return SUCCESS;
 }


 /**
  * Start things up. This just sets the rate for the clock component.
  *
  * @return Always returns <code>SUCCESS</code>
  **/
 command result_t StdControl.start() {
  // Start a repeating timer that fires every 1000ms
  return call Timer.start(TIMER_REPEAT, 1000);
 }

 /**
  * Halt execution of the application.
  * This just disables the clock component.
  *
  * @return Always returns <code>SUCCESS</code>
  **/
 command result_t StdControl.stop() {
  return call Timer.stop();
 }


 /**
  * Toggle the red LED in response to the <code>Timer.fired</code> event.
  *
  * @return Always returns <code>SUCCESS</code>
  **/
 event result_t Timer.fired()
 {
  call Leds.redToggle();
  return SUCCESS;
 }
}
```

REFERENCES

[1] X. Hou, D. Tipper, D. Yupho and J. Kabara, "GSP: Gossip-based Sleep Protocol for Energy Efficient Routing in Wireless Sensor Networks", The 16th International Conference on Wireless Communications, Calgary, Alberta, Canada, 2004.

[2] http://www.xbow.com/Support/Support_pdf_files/XBOW_Smart_Dust_ProductInfoGuide .pdf

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. "A Survey on Sensor Networks", IEEE Communications Magazine, August 2002.

[4] J. Kabara, Class notes from TELECOM 2721, Wireless Data Networks. Lecture number 5, june 2005.

[5] Chipcon AS, "Chipcon SmartRF CC1000 Datasheet, Revision 2.1.", April 19, 2002.

[6] CC1000RadioIntM.nc , program into the communication stack of Crossbow Motes. V1.29, april 26, 2004.

[7] Crossbow Technology Inc, "Wireless Sensor Networks, Getting Started Guide". Rev. B. Document: 7430-0022-05. August 2004.

[8] Crossbow Technology Inc, "MPR- Mote Processor Radio Board MIB-Mote Interface / Programming Board User's Manual". Rev A. Document 7430-0021-05. December 2003

[9] ATMEL, "ATmega128 ATmega128L Summary". Rev:2467LA-AVR-05/04.

[10] http://mail.millennium.berkeley.edu/pipermail/tinyos-commits/2004-August/004891.html

[11] Maxrad, "Base Station Antennas MFB800 Mhz Omnis, Datasheets", PCTel Antenna Products Group Inc.

[12] Eveready Battery Company, "Energizer Engineering Data for NH15 battery", Found on : http://data.energizer.com/PDFs/nh15-1600mAh.pdf

[13] Duracell Ni-MH Rechargeable batteries. Online Documentation. Ni-MH Technical Bulletin, Chapter 5. Found on http://www.duracell.com/oem/Pdf/others/nimh_5.pdf

[14] Digibattery, "NiMH Reachageable Batteries: Rechargeable Battery FAQs", Found in http://www.digibattery.co.ik/ni-mh_store1.html.

[15] D. Gay, P. Levis, D. Culler, E. Brewer, "nesC 1.1 Language Reference Manual". May 2003.

[16] Ch. Lu, "Berkeley Motes and TinyOS", CS851 Presentation, September 5, 2001

[17] G. Anastasi, M. Conti, A. Falchi, E. Gregori, A. Passarella., "Performance Measurements of Mote Sensor Networks" MSWiM'2004, October 4-6, 2004, Venezia, Italy.

[18] K. Sohrabi et al., "Protocols for Self-Organization of a Wireless Sensor Network," IEEE Pers. Commun., Oct. 2000, pp. 16–27.

[19] E. Shih et al., “Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks,” Proc. ACM MobiCom ’01, Rome, Italy, July 2001, pp. 272–86.

[20] A. Tanenbaum. “Computer Networks”, 4th Edition. Prentice Hall, Upper Saddle River, New Jersey, 2003.

[21] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks,” Proc. ACM MobiCom ’99, Seattle, WA, 1999, pp. 174–85

[22] A. Woo, and D. Culler, “A Transmission Control Scheme for Media Access in Sensor Networks,” Proc. ACM MobiCom ’01, Rome, Italy, July 2001, pp.221–35.

[23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister. "Architecture Directions for Networked Sensors", ASPLOSIX 11/00 Cambridge, MA, USA.

[24] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks” in IEEE Hawaii International Conference on Systems Sciences, 2000.

[25] Z. J. Haas, J. Y. Halpern, and L. Li, “Gossip-based ad hoc routing,” in Proceedings of IEEE INFOCOM, 2002.

[26] V. Shnayder, M. Hempstead, B. Chen, G. Werner Allen, and M. Welsh, "Simulating the Power Consumption of LargeScale Sensor Network Applications", SenSys’04, Baltimore, Maryland, USA, November 3–5, 2004.

[27] J. Polastre, J. Hill, D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks", SenSys’04, Baltimore, Maryland, USA, November 3–5, 2004.

[28] M. Bhardwaj, A .P. Chandrakasan, "Bounding the Lifetime of Sensor Networks Via Optimal Role Assignments", IEEE INFOCOM 2002.

[29] J. Chang, L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks", IEEE INFOCOM, Tel Aviv, Israel, 2000.

[30] V. Raghunathan, C. Schurgers, S. Park, M. B. Srivastava, "Energy-Aware Wireless Microsensor Networks". IEEE Signal Processing Magazine, march 2002, pp40-50.

[31] http://www.stjude.org/glossary?searchTerm=S

[32] J. Polastre, "Sensor Network Media Access Design", Computer Science Division. EECS Department, University of California, Berkeley. (c) 2003 Regents of the University of California.

[33] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks", ACM SIGMOBILE 7/01 Rome, Italy.

[34] TinyOS tutorial, found on www.tinyos.net/tinyos-1.x/doc/tutorial/

[35] Blink application, source code in TinyOS distribution by Crossbow. Version 1.1