6-1-2009

Lifetime Estimation of Wireless Body Area Sensor Network for Patient Health Monitoring

Frank Agyei-Ntim
University of Denver

LIFETIME ESTIMATION OF WIRELESS BODY AREA SENSOR NETWORK FOR PATIENT HEALTH MONITORING

A thesis

Presented to

the Faculty of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Frank Agyei-Ntim

June 2009

Advisor: Dr. Kimberly Newman

Author: Frank Agyei-Ntim
Title: Lifetime Estimation of Wireless Body Area Sensor Network for Patient Health Monitoring.
Advisor: Dr Kimberly Newman
Degree Date: June 2009

ABSTRACT

Wireless Body Area Sensor Networks (WBASN) is an emerging technology which utilizes wireless sensors to implement real-time wearable health monitoring of patients to enhance independent living. These sensors can be worn externally to monitor multiple bio-parameters (such as blood oxygen saturation (SpO2), blood pressure and heart activity) of multiple patients at a central location in the hospital.

In health monitoring, the loss of critical or emergency information is a serious issue so there is a concern for quality of service which needs to be addressed. It is important to have an estimate of the time the first node will fail in order to replace or recharge the battery. A common type of failure happens when a node runs out of energy and shuts down.

In this work, Monte Carlo simulation is used to determine the lifetime of WBASN. The lifetime of the WBASN is defined in this work as the duration of time until the first sensor failure due to battery depletion. A parametric model of the health care network is created with sets of random input distributions. Probabilistic analysis is used to determine the timing and distributions of nodes' failures in the health monitoring network.

ACKNOWLEDGMENTS

Writing this thesis required a great of effort that went beyond my initial expectations. I would first like to thank my mother Mrs. Christiana Boateng, without whose support and encouragement I wouldn’t have simply finish this work.

I would like to thank my advisor Dr. Kimberly Newman for all her help and guidance that she extended to me throughout my thesis work. Her feedback and suggestions have encouraged me to refine the thesis document and helped me complete this thesis in a timely manner.

I also want to extend my gratitude to Dr Roger Salters and Dr Mohammad Mahoor for agreeing to serve on my committee. I thank them for their feedback and valuable suggestions that have enabled me to improve the quality of my thesis work. Finally I am also grateful to Dr Cynthia McRae for serving as the chairperson of my committee.

TABLE OF CONTENTS

Abstract ……………………………………………………………………………..iii
Acknowledgments …………………………………………………………..…...iv

Chapter 1: Introduction 1
1.1 Background ...……………………………………………………......……..1
1.2 Problem Statement ………………………………………………………………3
1.3 Contributions ……………………………………………………………………4

Chapter 2: Network Architecture and Requirements ……………………………...7
2.1 Network Architecture …………………………………………………………7
2.1.1 Sensor Nodes ……………………………………………………………8
2.1.2 Base Stations ……………………………………………………………9
2.1.3 Wireless Technology Standard: ZigBee (802.15.4) …………………………..9

2.1.4 Simulation Environment………………………………………………..11
2.1.4.1 NS-2 …………………………………………………………...11
2.1.4.2 TOSSIM ………………………………………………………12
2.1.4.3 SensorSim ……………………………………………………..13
2.1.4.4 J-Sim ………………………………………………………….13

2.2 Network Requirements…………………………………………………………14
2.2.1 Monitored Signal: Electrocardiogram (ECG) ………………………………14
2.2.1.1 Probability distributions ………………………………………………17
2.2.2 Monitored Signal: Pulse Oxygen Saturation (SpO_2) ………………………..18
2.2.2.1 Probability distributions ………………………………………………19

2.2.3 Mobility Models ……………………………………………………………19
2.2.3.1 Random Waypoint Model ……………………………………………..20
2.2.3.2 Random Walk Model …………………………………………………21
2.2.3.3 Random Directional Model ……………………………………………23
2.2.3.4 Smooth Random Mobility Model ……………………………………..23

2.2.4 Routing Algorithms …………………………………………………………26
2.2.4.1 Single Hop ……………………………………………………………27
2.2.4.2 AODV …………………………………………………………………27
2.2.4.3 Directed Diffusion ……………………………………………………28
2.2.4.4 Multi-hop Medium Access Control MAC ……………………………29

Chapter 3: Monte Carlo Method ...32
3.1 Monte Carlo Simulation ...32
3.2 Steps in Monte Carlo Simulation ...32
3.3 Input Randomization and Distribution ...33

Chapter 4: Implementation of Monte Carlo Simulation ...37
4.1 Stage 1 Simulation ...37
4.2 Stage 2 Simulation ...37
4.3 Estimation of time for first node to fail ...39
4.4 Stochastic Properties ...40
4.5 Output Distribution ...40
4.6 Scripts ...41

Chapter 5: Results ...43
5.1 Simulation Scenario ...43
5.2 Discussions ...44
5.2.1 Power Flow Analysis ...45
5.2.2 Network Infrastructure Failure Analysis ...50
5.2.3 Nodes' Lifetime ...61

Chapter 6: Conclusion ...67

Bibliography ...70

Appendix A: Matlab Scripts Descriptions ...73
A.1 Stage 1 scripts ...73
A.2 Stage 2 scripts ...74

Appendix: B Python Scripts Descriptions ...79
B.1 Stage 1 scripts ...79
B.2 Stage 2 scripts ...79

Appendix: C Modifying J-SIM TCL Scripts ...81
C.1 Stage 1 scripts ...83
C.2 Stage 2 scripts ...85

LIST OF TABLES AND FIGURES

LIST OF TABLES

2-1 Comparison of ZigBee to other Wireless Standards ……………………………………9

3-1 Steps in Monte Carlo Simulation …………………………………………………33
4-1 Simulation process using power use approximation ………………………………..39
4-2 Estimating the time for the first node to fail ……………………………………...40

6-1 Comparison of the Probabilistic analysis with various routing protocols …………..67

LIST OF FIGURES

1-1 CodeBlue architecture for emergency response ……………………………………...5

2-1 Architecture layout of Wireless Body Area Network ………………………………..8
2-2 A cardiac cycle and its constituents ………………………………………………15
2-3 Example of node running in the Random Waypoint Model ………………………...21
2-4 Example of nodes running in the Random Walk Model ……………………………22
2-5 A simplified schematic of Directed Diffusion protocol ……………………………29
2-6 A Multi-hop MAC scheme …………………………………………………………30

5-1 CDFs of the average power use by nodes running the AODV ……………………..48
5-2 CDFs of the average power use by nodes running the Single-hop …………………48
5-3 CDFs of the average power use by nodes running the Multi-hop MAC ……………49
5-4 CDFs of the average power use by nodes running the Directed Diffusion …………49
5-5 Number of active nodes vs. time for AODV ……………………………………..54
5-6 Distributions of time to be left with a percentage of working nodes for AODV …...55
5-7 Number of active nodes vs. time for single-hop ……………………………………56
5-8 Distributions of time to be left with a percentage of working nodes for
Single-hop ………………………………………………………………………...57
5-9 Number of active nodes vs. time for Multi-hop MAC ……………………………...58
5-10 Distributions of time to be left with a percentage of working nodes for
Multi-hop MAC …………………………………………………………………..59
5-11 Distributions of time to be left with a percentage of working nodes for
Multi-hop MAC …………………………………………………………………..60
5-12 Mean time for 1st, 2nd, 3rd, 4th and 5th node to fail for AODV ……………………..63
5-13 Mean time for 1st, 2nd, 3rd, 4th and 5th node to fail for Single-hop …………………64
5-14 Mean time for 1st, 2nd, 3rd, 4th and 5th node to fail for Multi-hop …………………..65
5-15 Mean time for 1st, 2nd, 3rd, 4th and 5th node to fail for Directed Diffusion …………66

CHAPTER 1: INTRODUCTION

1.1 Background

Wireless Body Area Sensor Network (WBASN) is an emerging technology which utilizes wireless sensors to implement real-time wearable health monitoring of patients to enhance their independent living. By outfitting patients with wireless, wearable vital sign sensors, collecting detailed real-time data on physiological status is greatly simplified. These sensors monitor multiple bio-parameters (such as blood oxygen saturation (SpO2), blood pressure and heart activity) of multiple patients at a central location in a managed care facility. Hospitalization and nursing are invoked, whenever abnormalities are noticed by the health care network. This enables patients especially the elderly to lead normal life with the confidence that medical assistance is at hand whenever needed.

Several research projects focused on WBASN have been undertaken to address the needs of medical care such as node mobility, a wide range of data rates and high degrees of reliability and security: In [1] the hardware and software architecture of a Wireless Body Area Network (WBAN) for ambulatory health status monitoring is discussed. A prototype including two activity sensors and electrocardiogram (ECG or EKG) sensor, a Personal Server and a Network Coordinator was developed to integrate the WBAN into a broader multi-tier telemedicine system using ZigBee as wireless technology.

The MobiHealth project [2] developed a customizable vital signals monitoring system based on a Body Area Network (BAN) and a health service platform utilizing Universal Mobile Telecommunication Service (UMTS) and General Packet Radio Service (GPRS) networks. The BAN prototype was tested in clinical trials with different healthcare scenarios such as high-risk pregnancies, and cardiac arrhythmia. Wireless technologies such as Bluetooth and ZigBee were used for intra-BAN communication whereas GPRS and UMTS for external communication.

A scenario from the Ubimon project [3] was developed at the department of Computing, Imperial College London, aimed at investigating Healthcare delivery by combining wearable and implantable sensors. The project proposed monitoring patients under natural physiological state in their daily life. The described scenario monitored routine vital signs.

HealthGear [4], a wearable real-time health system for monitoring and analyzing physiological signs developed at Microsoft Research Department, consists of a set of physiological sensors connected via Bluetooth to a cell phone. It is used with an oximeter to constantly monitor and analyze the user's blood oxygen level SpO_2 and heart rate.

CodeBlue is a research project at Harvard University. It integrates sensor nodes and other wireless devices into a disaster response setting [5]. This project developed a pulse oximeter sensor, two-lead ECG and a specialized motion-analyzer sensor. It outfits patients in emergency and disaster environments with wearable wireless sensors and allows care-givers to continuously monitor the status of their patients. While these projects address the needs of medical care to some extent, the concern for quality of service in terms of battery life of the WBASN is still not addressed.

1.2 Problem Statement

WBASN runs on batteries which are rarely replaced or recharged. Since the failures due to the availability of energy being exhausted are unavoidable, it is very important to determine or have an estimate of the time the first node will fail so as to replace or recharge the battery in order not to lose vital signs. A common type of failure happens when a node runs out of energy and shuts down. The timing and distribution of such failures critically impact the ability of the WBASN to collect real-time data of physiological status of patients in a health care environment. A lifetime estimation of the WBASN for health monitoring is presented in this work. The lifetime of the WBASN is defined in this work as the duration of time until the first sensor failure due to battery depletion.

Current approaches depend on analytical or experimental methods with expensive hardware trials. For example in [20, 21], mathematical models and hardware measurements are used to determine the lifetime of wireless sensors. Similarly [22] models a single node lifetime in wireless sensor networks using Type-2 Fuzzy Membership function (MF), i.e. a Gaussian MF with uncertain standard deviation. However, because of many constraints imposed on sensor networks, such as energy limitation, decentralized collaboration and fault tolerance, algorithms for sensor networks tend to be quite complex and usually defy analytical methods that have been proved to be fairly effective for traditional networks. It appears that simulation is the only feasible approach to the quantitative analysis of sensor networks. Therefore simulation using probabilistic analysis is used in this work to determine the lifetime of WBASN for health monitoring of patients.

1.3 Contributions

The lifetime of the WBASN is estimated by performing probabilistic analysis on a simulation of the network. The probabilistic analysis is performed using the Monte Carlo method. Monte Carlo method involves generating random input vectors with known distributions, and then running the simulation with these vectors as input. The resulting output vectors provide the output distributions. Alternatives such as the mean value methods, generally assume a deterministic system with differentiable variables [32]. In most networked systems, the presence of discrete structures often results in discontinuities, and possibly non-monotonic responses which can result in large errors due to local minima. The use of randomized methods such as random back off in most multi-hop networks such as LEACH also results in non-deterministic behavior. Thus, the Monte Carlo method is the best option to estimate probability distributions of the node failures of the WBASN.

The random input parameters of the Monte Carlo method are based on power and radio characteristics of CodeBlue [5] project. CodeBlue architecture is chosen as a starting point since is it based on the Crossbow mica mote hardware that is part of several popular WSN simulation tools. Other equally good WBASN architectures include MobiHealth project, Ubimon project and HealthGear developed at Microsoft Research Development. CodeBlue integrates wearable sensor nodes and other wireless sensor devices into a disaster response setting. CodeBlue comprises of a suite of protocols and services that heterogeneous devices such as wireless sensors, location beacons, PDAs and laptops co-ordinate their activities as shown in Figure 1-1 below.

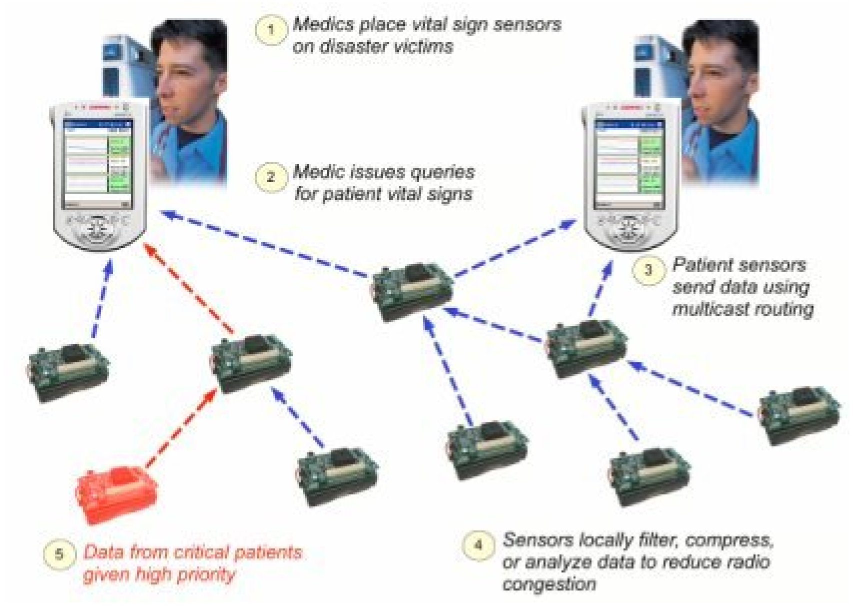

1 Medics place vital sign sensors on disaster victims
2 Medic issues queries for patient vital signs
3 Patient sensors send data using multicast routing
4 Sensors locally filter, compress, or analyze data to reduce radio congestion
5 Data from critical patients given high priority

Figure 1-1: CodeBlue architecture for emergency response [5]

Each patient is outfitted with electrocardiogram ECG or EKG (both are used interchangeably) and pulse oximeter sensors that monitor the patient's heart rate and oxygen saturation respectively. ECG is the continuous record of the voltage changes that reflect the cyclic electro-physiologic events in the myocardium (heart). The time varying motion of the cardiac vector produces the body surface ECG for one heart beat. Each ECG waveform consists of vital signs such as P complex, P waves, QRS complex, QT intervals and so on. The patterns the sensor must detect in the ECG waveform are the complexes, inter-wave segments and the cardiac intervals (for more details see Chapter 2). The detection of these parameters is modeled as a stochastic process with Gaussian distribution. Gaussian distribution is chosen because the parameters of the ECG waveform (P complex, QRS complex, T complex, R-R interval and so on) are detected independently. The parameters of the complex that need to be measured are the peak (amplitude) and duration. The sensors analyze the detected parameters and characterize them as normal or abnormal before transmitting the data to the base station.

Oxygen saturation is a measure of how much oxygen the blood is carrying as a percentage of the maximum it could carry. One hemoglobin molecule can carry a maximum of four molecules of oxygen; if for instance, a hemoglobin molecule is carrying three molecules of oxygen then it is carrying ¾ or 75% of the maximum oxygen it could carry. This is detected by the pulse oximeter. The detection of oxygen saturation is modeled as stochastic process with Exponential distribution. Exponential distribution is chosen because it represents a constant average rate. Each patient is allowed to walk freely in a 200m by 200m square area. The mobility model adopted for the nodes is Smooth Random Mobility (for the choice of mobility see Chapter 2).

When a node detects abnormal ECG or a pulse oxygen saturation data above some threshold, it attempts to report this event to a base station. Sources produce events of random magnitudes at random intervals. Mobility and stimulus related variables are treated as random variables. Stimuli (ECG and SpO_2) are randomly generated. Finally probabilistic analysis is performed using the Monte Carlo method. This results in a probability distribution of the average power consumption of each node before nodes start to fail, and distribution of the node loss as a function of time. The probability distributions of energy consumed by the nodes and network lifetime are obtained from multiple sample runs. In order to make the simulation time manageable, sequential approximation techniques are used to reduce the simulation time.

CHAPTER 2: NETWORK ARCHITECTURE AND REQUIREMENTS

The WBASN is simulated as a discrete event simulation using J-Sim network simulator. The architecture of the WBASN follows the J-Sim component model. J-Sim uses three top level components: the target node (which produces stimuli), the sensor node (that reacts to the stimuli), and the sink node (the ultimate destination for stimuli reporting). Sensor nodes are modeled as a combination of physical components (such as CPU, battery, radio etc.) and logical components representing the protocol stack. Different nodes in the network are linked using virtual channels that simulate a single independent radio channel. A similar method is used for sensor stimuli. The most commonly used network simulators and emulators are NS-2, J-Sim, SensorSim and TOSSIM. J-Sim is chosen in this work because of the component-based architecture that is easily customized to simulate specific applications, and there is support for the connection of real hardware sensors to the simulator. The architecture of the WBASN and its requirements including the input source, mobility models and routing protocols are presented below.

2.1 Network Architecture

The components of the network are mobile input sources (targets), sensor nodes, base station or sink. General energy consumption and performance models are used for these components and may be fine tuned through comparison to actually hardware. The candidate wireless technology chosen for our case study is based on the ZigBee wireless communication standard which is a subset of the IEEE 802.15.4. The architectural layout of the WBASN is shown in Figure 2-1 below.

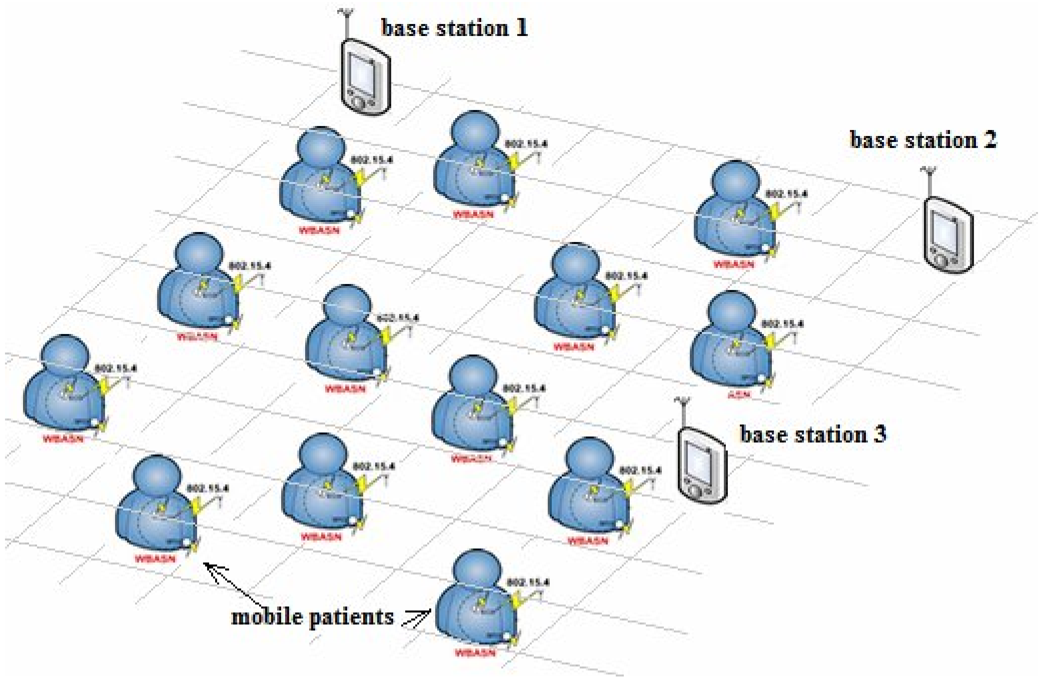


Figure 2-1: Architectural layout of wireless body area network

2.1.1 Sensor Nodes

The sensor nodes sense, process and transmit detected parameters of ECG and oxygen saturation data to the base station. The sensors consist of electrocardiogram (ECG) and pulse oximeter sensors. The ECG sensor monitors the heart electrical activity and a sudden change in heart rate can indicate a need for urgent intervention [4]. Pulse oximeter sensor monitors the oxygen saturation of the patient.

Radio transmission is assumed to follow the free space propagation model. The received power P_r from a transmitter with power P_t at a distance d to the receiver is given by Equation 2-1:

$$P_r = P_t \frac{G_t G_r \lambda^2}{L(4\pi d)^2} \tag{2-1}$$

The values Gt and Gr are the transmitter and receiver antenna gains, λ is the wavelength of the transmission and L is system loss.

The nodes also use a simplified battery model whose capacity is assumed to be always constant (i.e. not a function of the current). An alternative battery model is the Coin Cell model which specifies the battery capacity as the function of its current.

2.1.2 Base Station

The base station collects the abnormal signs of ECG and pulse oximeter signs from the sensor nodes. The base station is assumed to have infinite power because in a typical hospital the base is connected the main power supply.

2.1.3 Wireless Technology Standard: ZigBee (802.15.4)

There are many wireless options available to WBASN designers and which share the unlicensed 2.4GHz band. These are ZigBee (IEEE 802.15.4), Bluetooth (IEEE 802.15.1) and Wi-Fi (IEEE 802.11). The chart below compares the performance of the ZigBee compared to other standards.

Criteria	**ZigBee**	**Bluetooth**	**Wi-Fi**
MAC Standard	802.15.4	802.15.1	802.11
Maximum over-the-air data rate	250kbps	1Mbps	54Mbps
Transmit current	35mA	40mA	400+mA
Standby current	3uA	200uA	20mA
Memory requirements	32-60KB	100+KB	100+KB
Networking options	Mesh networking	Point to multipoint	Point to multipoint
Max. Transmission Range	100m	10m	30m

Table 2-1 Comparison of ZigBee to other standards [9]

Bluetooth is a popular standard applied to wire replacement applications. It is based on an IEEE Personal Area Network PAN standard, 802.15.1. Bluetooth operates with a 1 Mbps data rate. Bluetooth and ZigBee have similar transmit currents, but ZigBee has a significantly lower standby current. This is because devices in Bluetooth networks must frequently report into the network to maintain synchronization, so they cannot easily drop into a “sleep” mode.

Wi-Fi is a Wireless Local Area Network WLAN standard, so it requires almost continuous activity by devices in the network. The advantage of this standard is the tremendous amount of data that can be moved from point to multi-point. Wi-Fi hardware is designed to operate off a significant power source.

Of the three wireless standards, only ZigBee offers the flexibility of mesh networking. Another advantage of ZigBee is the reduced memory requirements of ZigBee. ZigBee applications are typically simple. ZigBee end devices can “sleep” while still maintaining network association [9].

From the chart above ZigBee and Bluetooth are the only wireless standards suited for WBASN which requires low power management. Both standards are in the personal area networking category. Both have similar radios, as evidenced by their transmit currents. The difference between the two standards is in their target applications.
Bluetooth targets medium data rate, continuous duty applications like file transfer and streaming telecom audio. ZigBee on the other hand, targets low data rate, low duty cycle applications. End point devices do not transmit or receive as frequently in these applications, resulting in exceptional battery life. Based on these performances, ZigBee

as defined by the underlying 802.15.4 specification is better suited to health care network than Bluetooth.

2.1.4 Simulation Environment [10]

The most commonly used network simulators and emulators are NS-2, J-Sim, SensorSim and TOSSIM. J-Sim is chosen in this work because of its component-based architecture which easily simulate specific applications, and there is support for the connection of real hardware sensors to the simulator. The various simulators and emulators and their weaknesses are discussed below.

2.1.4.1 NS-2

NS-2 [11 and 12] is the most popular simulation tool for sensor networks. NS-2 is an object-oriented discrete event simulator; its modular approach has effectively made it extensible. Simulations are based on a combination of C++ and OTcl. In general, C++ is used for implementing protocols and extending the NS-2 library. OTcl is used to create and control the simulation environment itself, including the selection of output data. NS-2 extensibility is perhaps what has made it so popular for sensor networks. In addition to the various extensions to the simulation model, the object-oriented design of NS-2 allows for straightforward creation and use of new protocols. Its status as the most used sensor network simulator has encouraged further popularity, as developers would prefer to compare their work to results from the same simulator.

However, NS-2 does not scale well for sensor networks. This is in part due to its object-oriented design. While this is beneficial in terms of extensibility and organization, it is a hindrance on performance in environments with large numbers of nodes. Every node is its own object and can interact with every other node in the simulation, creating a large number of dependencies to be checked at every simulation interval, leading to an n^2 relationship. Another limitation of NS-2 is the lack of customization available. Packet formats, energy models, MAC protocols, and the sensing hardware models all differ from those found in most sensors. One last drawback of NS-2 is the lack of an application model. In many network environments this is not a problem, but sensor networks often contain interactions between the application level and the network protocol level.

2.1.4.2 TOSSIM

TOSSIM [13], is actually an emulator which is different from a simulator in it runs actual application code. TOSSIM is designed specifically for TinyOS applications to be run on MICA Motes. The developers had four key concepts in mind when creating TOSSIM: scalability (the system should be able to handle thousands of nodes with different network configurations), completeness (as many system interactions as possible must be covered in order to accurately capture behavior), fidelity (subtle interactions must be captured if testing is to be accurate), and bridging (validating the implementation of algorithms). In order to achieve its goal of scalability, each node in the simulator is connected in a directed graph where each edge has a probabilistic bit error. For perfect transmission, the bit error is 0, and can be changed for different situations. Also in the name of efficiency, every node in TOSSIM runs the same application code; all nodes are identical. The goal of scalability is successfully achieved; results show that it is able to handle larger networks than almost any other simulator or emulator. Most application code is run unchanged. The only differences are in some places where the application interacts with hardware. TOSSIM's probabilistic bit error model leads to inaccuracies, and reduces the simulator's effectiveness in analyzing low level protocols. Accuracy loss also occurs during compilation, when fine grained timing and interrupt properties are lost, affecting interactions with other nodes in the network. Additionally, despite the fact that the sensor's data gathering hardware is simulated, the phenomena that trigger reactions are not.

2.1.4.3 SensorSim

SensorSim [14] uses NS-2 as a base, and extends it in three important ways. First, it includes an advanced power model. The model takes into account each of the hardware components that would need battery power in order to operate. Secondly, SensorSim includes a sensor channel. The third extension to NS-2: an interaction mechanism with external applications. The main purpose is to interact with actual sensor node networks. This allows for real sensed events to trigger reactions within the simulated environment. In order to accomplish this, each real node is given a stack in the simulation environment. The real node is then connected to the simulator via a proxy, which provides the necessary mechanism for interaction. One further extension to NS-2 is the use of a piece of middleware called SensorWare. This middleware makes it possible to dynamically manage nodes in simulation. This provides the user with the ability to provide the network with small application scripts than can be dynamically moved throughout the network. Like NS-2, SensorSim faces a scalability problem. Additionally, SensorSim is not being maintained and is not currently available to the public.

2.1.4.4 J-Sim

J-Sim [15] is a general purpose Java-based simulator modeled after NS-2. Unlike NS-2, however, J-Sim uses the concept of components, replacing the notion that each node should be represented as an object. J-Sim uses three top level components: the target node (which produces stimuli), the sensor node (that reacts to the stimuli), and the sink node (the ultimate destination for stimuli reporting). Each component is broken into different parts and modeled differently within the simulator. The breakdown of each component makes it easy to use different protocols in different simulation runs. J-Sim features several improvements on NS-2 and other simulators. Most importantly, its component based architecture scales better than the object oriented model used by NS-2 and other simulators. Furthermore, J-Sim features an improved energy model and the ability to simulate the use of sensors for phenomena detection. Like SensorSim, applications may be simulated, and there is support for the connection of real hardware sensors to the simulator. J-Sim is relatively complicated to use. While no more complicated than NS-2, the latter simulator is more popular and, thus, more people are willing to spend the time to learn how to use it. J-Sim, while more scalable than many other simulators, also faces its share of inefficiencies. Java, in general, is arguably less efficient than many other languages. There is also unnecessary overhead in the intercommunication model. J-Sim is chosen in this work because of its component-based architecture which easily simulate applications, and there is support for the connection of real hardware sensors to the simulator.

2.2 Network Requirements

2.2.1 Monitored Signal: Electrocardiogram (ECG) [16]

ECG is the continuous record of voltage changes that reflect the cyclic electro-physiologic events in the myocardium. ECG is measured using electrodes attached to the body surface and connected to an instrumentation amplifier. The time varying motion of the cardiac vector produces the body surface ECG for one heartbeat. With each heartbeat

ECG inscribes a series of deflections that are labeled as P, Q, R, S, T and U as shown in Figure 2-2 below.

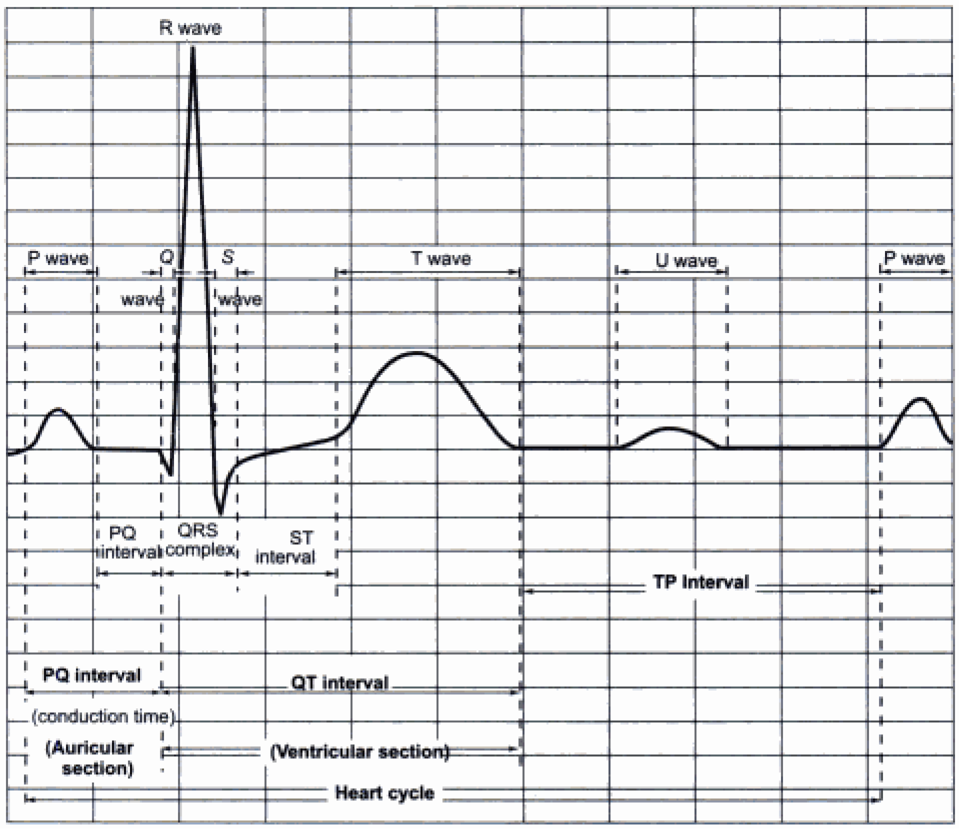


Figure 2-2 A cardiac cycle and its constituents
(From Trahanias P. and E. Skordalakis, Syntatic Pattern Recognition of the ECG. IEEE Trans. Pattern. Anal. Machine Intell. 1990-12).

The patterns that the sensors must recognize in the ECG waveforms are

- the complexes
- the inter-wave segments, and,
- the cardiac intervals.

These patterns are indicated in the Figure 2-3 above. The three complexes are the P complex, QRS complex and the T complex. The parameters of the complex that needs to be measured are their peak (amplitude) and duration. The same measurements are required in rest of the waves. However in the case of the inter-wave segments and the cardiac intervals, the duration parameter is of interest.

The first step in the identification of the ECG waves is the detection of the QRS complex. Once this is obtained additional signal processing techniques need to be used to identify the other waves and features of the ECG signal. The concept of Template Matching techniques [17] based on correlation is used in this work to detect the QRS complex in the ECG signal. Other methods which could have been used include Template Subtraction methods and Differentiation-Based techniques. Signals are said to be correlated if their wave shapes match or are similar. A measure of this is provided by the correlation coefficient which is a maximum when the signals are close. In this method a template of this QRS morphology is cross correlated with an incoming signal. A cross correlation function estimate $r_{xy}(m)$ of sequences $x(n)$ and $y(n)$ is defined by the equation (2-2) below assuming that both the sequences have been measure from $n = 0$ to $n = N - 1$.

$$r_{xy}(m) = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-m-1} x(n+m)y(n) \text{ for } 0 \le m \le N-1 \end{cases} \qquad (2\text{-}2)$$

Here the template can be thought of a window that moves over the incoming signal point by point. Once the QRS complex is detected, the algorithm developed for the detection of the R-wave is initiated. It looks for R peaks by first difference methods. The R wave spike detection looks for a change from positive to negative first difference and then checks whether the size of the change is larger than a set threshold. If both these conditions are satisfied, the point at which the change occurred is marked as an R peak.

Once the R peak is identified the sensor can go on to calculate the heart rate and the other features of the ECG waveform. The heart rate is calculated by knowing the period between the successive R peaks.

If say, the peaks are detected at intervals (normalized to the sampled interval) 50 and 135 in an ECG, sampled 100 times per second, then the heart rate in beats per minute is given by the Equation 2-3.

Heart rate in beats per minute (BPM) $\frac{100}{(135-50)} \times 60 = 71$ (2-3)

It is now obvious to characterize the ECG waveform as *normal* or *abnormal* considering the various shapes of the complexes, time durations and their amplitudes.

2.2.1.1 Probabilistic Distribution

The detection and analysis of the ECG signals can be modeled as a stochastic process with Binomial distribution. The Binomial distribution is chosen because the final analysis of the features or parameters of the ECG signal detected can be characterized as either normal or abnormal. The binomial function is given by

$$f(x) = \frac{n!\,p^x(1-p)^{n-x}}{x!(n-x)!} \tag{2-4}$$

Where x the number of times a QRS complex and R-R wave are detected by the sensors and n is the number of samples and p is the probability of the detection of the QRS complex or R-R wave

The mean and standard deviation (SD) of the distribution are given by

$$Mean = np \tag{2-5}$$

$$SD = \sqrt{np(1-p)} \tag{2-6}$$

Assuming a large sampling of ECG signals, the Binomial distribution can be approximated to Gaussian distribution. Gaussian distribution is a good approximation because the detecting of QRS complex, R-R wave, P complex are independent. The Gaussian (normal) probability density function is given by the Equation (2-7).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})} \tag{2-7}$$

where P(x) is the overall probability of detecting the parameters - QRS complex, R-R wave and P complex, x is the number of times of detecting the parameters, σ is the standard deviation and μ is the mean.

2.2.2 Monitored Signal: Pulse Oxygen Saturation – SpO_2 [18]

Oxygen saturation is a measure of how much oxygen the blood is carrying as a percentage of the maximum it could carry. Oxygen is carried by the blood attached to the hemoglobin molecules. One hemoglobin molecule can carry a maximum of four molecules of oxygen; if a hemoglobin molecule is carrying three molecules of oxygen then it is carrying ¾ or 75% of the maximum oxygen it could carry. One hundred hemoglobin molecules could together carry a maximum of 400 (100 x 4) oxygen molecules, if these 100 hemoglobin molecules were carrying 380 oxygen molecules they would be carrying (380 / 400) x 100 = 95% of the maximum number of oxygen molecules that could carry and so together would be 95% saturated. Oxygen saturation is also referred to as SpO_2.

The color of blood varies depending on how much oxygen it contains. A pulse oximeter shines two beams of light through a finger (or earlobe etc.), one beam is red light, and the other is infrared light.

These two beams of light can let the pulse oximeter detect what color the arterial blood is and it can then work out the oxygen saturation. However there are lots of other bits of a finger which will absorb light (such as *venous* blood, bone, skin, muscle etc.), so to work out the color of the arterial blood a pulse oximeter looks for the slight change in the overall color caused by a beat of the heart pushing arterial blood into the finger.

This change in color is very small so pulse oximeter works best when there is a good strong pulse in the finger (etc.) the probe is on. If the signal is too low the measured oxygen saturation may not be reliable and lower than this the pulse oximeter will not be able to work.

2.2.2.1 Probability Distribution

The detection and analysis of the pulse oxygen saturation can also be modeled as stochastic process with an exponential distribution with uniform sampling rate with a probability density function given by Equation (2-8) as

$$f(x;\lambda) = \lambda_s e^{-x\lambda_s} \tag{2-8}$$

Where λ_s represent the constant rate of detecting the oxygen saturation, x is the pulse oxygen samples.

Various mobility models are available for WBASN designers. Smooth Random mobility model is chosen for this work. The various mobility models and their weaknesses are described below.

2.2.3 Mobility Model

Many different models have been proposed by researchers such as Random Waypoint model [6], Random Direction model [7] and Random Walk Mobility [8]. The commonest model is the random waypoint mobility model. Random Waypoint model and its variants are designed to mimic the movement of mobile nodes in a simplified way. However, they may not adequately capture certain mobility characteristics of some realistic scenarios such as temporal dependence on velocity. That is to say, they are memory-less random process whose current velocities are independent of their previous velocities. Therefore some extreme behavior such as sudden stop, sudden acceleration and sharp turn, may frequently occur in the trace generated by the Random Waypoint model. However, in many real life scenarios, a patient can walk steadily with incremental speed and in addition his direction change is also smooth. In this case Smooth Random Mobility [19] which exhibit temporal dependence on velocity and its model independent of each mobile node is chosen in this work.

2.2.3.1 Random Waypoint Model [6]

The Random Waypoint Model was first proposed by Johnson and Maltz [6]. The implementation of this mobility model is as follows: as the simulation starts, each mobile node randomly selects one location in the simulation field as the destination. It then travels towards this destination with constant velocity chosen uniformly and randomly from $[0, V_{\max}]$, where the parameter $V_{\max}$ is the maximum allowable velocity for every wireless node. The velocity and direction are chosen independently of other wireless nodes. Upon reaching the destination the node stops for a duration defined by the 'pause time' parameter T_{pause}. After this duration the whole process repeats again. As example of the movement traces is shown below:

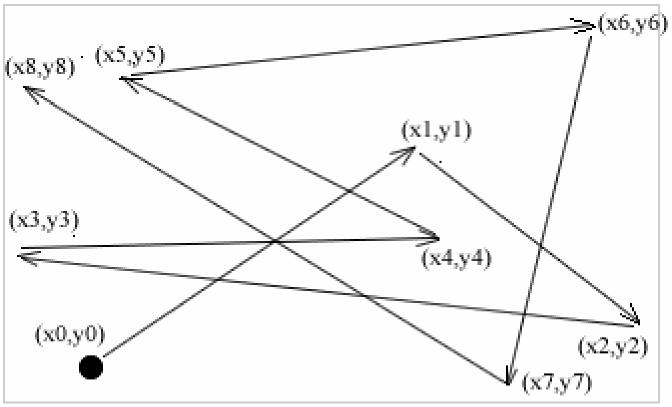


Figure 2-3 Example of node movement in the Random Waypoint Model

The measure of relative speed between node i and j at time t is

$$RS(i,j,t) = |\vec{V}_i(t) - \vec{V}_j(t)| \tag{2-9}$$

Where the parameters $\vec{V}_i$ and $\vec{V}_j$ are the velocities of node i and j respectively. The mobility metric $\overline{M}$ to capture and quantify nodal speed is calculated as the measure of relative speed average over all node pairs and over all time, defined as

$$\overline{M} = \frac{1}{|i,j|}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{1}{T}\int_{0}^{T} RS(i,j,t)dt \tag{2-10}$$

Where $|i,j|$ is the number of distinct node pair, (i,j), n is the total number of nodes in the simulation field and T is the simulation time. This metric allows us to roughly measure the level of nodal speed and differentiate the different mobility scenarios based on the level of mobility. In the following sections the variations of Random Waypoint and their limitations are discussed.

2.2.3.2 Random Walk Model [8]

The Random Walk mobility model was originally proposed to emulate the unpredictable movement of particles referred to as the *Brownian motion.* Because some mobile nodes are believed to move in an unexpected way, Random Walk mobility model is proposed to mimic their movement behaviors. The Random Walk model is a specific Random Waypoint model with zero pause time. However in Random Walk model, the nodes change their speed and direction at each time interval. For every new interval t, each node randomly and uniformly chooses its new direction $\theta(t)$ from $[0,2\pi]$. In similar way the new speed $v(t)$ follows a uniform distribution from [0, $V_{\max}$] as shown the Figure below:

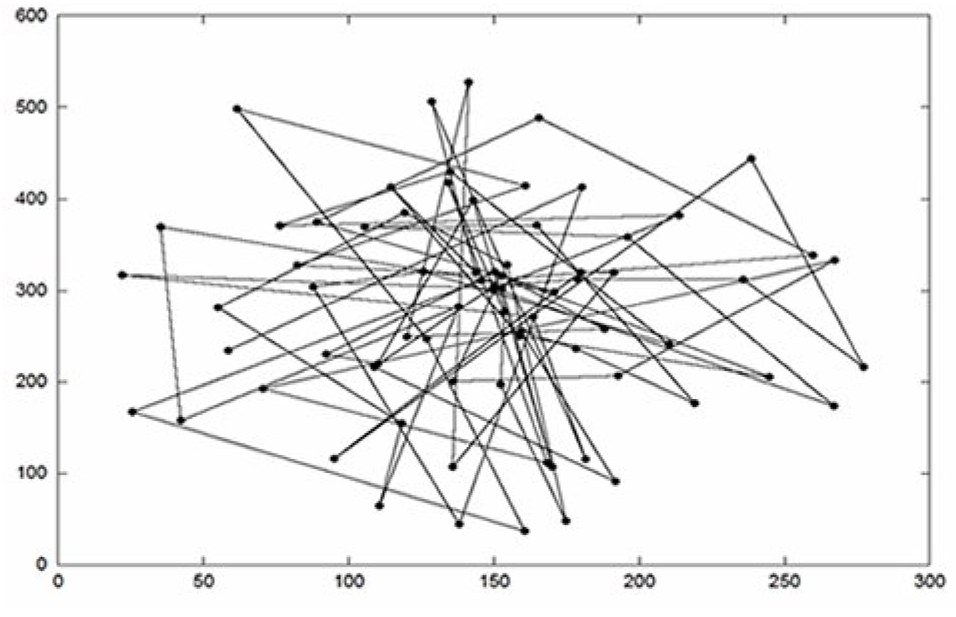


Figure 2-5 Example of nodes movements in the Random Walk Model

During time interval t, the node moves with the velocity vector $(v(t)\cos\theta(t), v(t)\sin\theta(t))$. Like the Random Waypoint model, Random Walk model is a memory-less mobility process where the information about the previous status is not used for the future decision. In order words, the current velocity is independent with its previous velocity and the future velocity is also independent with its current velocity. This is not the case of mobile nodes in many real life applications as discussed in Smooth Random mobility model.

2.2.3.3 Random Direction Model [7]

Another variation of Random Waypoint model is the Random Direction model. It is observed that the distribution of movement angle is not uniform in Random Waypoint; therefore Random Directional Model was proposed by Royer et al [7]. Instead of selecting a random direction within the simulation field, the mobile node randomly and uniformly chooses a direction by which to move along until it reaches the boundary. After the node reaches the boundary of the simulation field and stops with a pause time T_{pause}, it then randomly and uniformly chooses another direction to travel. This way the nodes are uniformly distributed within the simulation field. However, like Random Waypoint and Random Walk model, the Random Direction model is memory-less random process, i.e., the velocity at current epoch is independent of the previous epoch. Thus, some extreme behavior such as sudden stop, sudden acceleration and sharp turn, may frequently occur in the trace generated by the Random Waypoint model. This is the reason why Smooth Random model is chosen in this work.

2.2.3.4 Smooth Random Mobility Model [19]

In Ref. [5], it is found that the memory-less nature of Random Waypoint and its variants may result in unrealistic movement behaviors. It is observed that mobile nodes in real life tend to move at certain preferred speeds $\left\{V_{pref}^{1}, V_{pref}^{2}, \ldots, V_{pref}^{n}\right\}$, rather than at speeds purely uniformly distributed in the range [0, V_{max}]. The probability distribution of node velocity in Smooth Random Mobility model is as follows: the speed within the set of preferred speed values has a high probability, while a uniform distribution is assumed on the remaining part of entire interval [0, V_{max}]. For example if the node has a preferred speed set {0, 0.5V_{max}, V_{max}}, then the probability distribution is

$$P(v) = \begin{cases} P(v=0)\delta(v) & v=0 \\ P(v=0.5V_{\max})\delta(v-0.5V_{\max}) & v=0.5V_{\max} \\ P(v=V_{\max})\delta(v-V_{\max}) & v=V_{\max} \\ \dfrac{1-P(v=0)-P(v=0.5V_{\max})-P(v=V_{\max})}{V_{\max}} & 0<V_{\max}<1 \\ 0 & otherwise \end{cases} \tag{2-11}$$

where $P(v=0)+P(v=0.5V_{\max})+P(v=V_{\max})<1$.

In Smooth Random Mobility model, the frequency of speed change is assumed to be a Poisson process. Upon an event of speed change, a new target speed $v(t)$ is chosen according to the probability distribution function of speed as shown in Equation (2-11). Then the speed of the mobile node is changed incrementally from the current speed $v(t^{'})$ to the targeted new speed $v(t)$ by acceleration speed or deceleration speed $a(t)$. The probability distribution function of acceleration or deceleration $a(t)$ is uniformly distributed among $\left[0, a_{\max}\right]$ and $[a_{\min}, 0]$ respectively as shown in Equation (2-12).

$$P(a) = \begin{cases} \dfrac{1}{a_{\max}} & acceleration: 0 < a \le a_{\max} \\ \dfrac{1}{a_{\min}} & deceleration: a_{\min} \le a \le 0 \\ 0 & otherwise \end{cases} \tag{2-12}$$

For each time slot t, the new speed is calculated as

$$v(t) = v(t-\Delta t) + a(t)\Delta t \tag{2-13}$$

The speed can then be controlled to increase or decrease continuously and incrementally. If $a(t)$ is a small value, then the speed is changed slowly and the degree of temporal correlation is expected to be strong. On the other hand the speed can be changed quickly and the temporal correlation is small.

Unlike speed, the movement direction is purely uniformly distributed in the interval $[0,2\pi]$, as

$$P_{\phi}(\phi) = \frac{1}{2\pi} \qquad for 0 \leq \phi \leq 2\pi \tag{2-14}$$

Once a movement direction is chosen, the node moves in a straight line until the direction changes. The frequency of direction change is assumed to have an exponential distribution. When the direction is about to change, the new movement direction is also selected according to the probability distribution function described by Equation (2-14). The direction difference $\Delta\phi(t)$ between the new direction $\phi(t)$ and the old direction $\phi(t')$ is defined as

$$\Delta\phi(t) = \begin{cases} \phi(t) - \phi(t') + 2\pi & for - 2\pi < \phi(t) - \phi(t') \leq -\pi \\ \phi(t) - \phi(t') & for - \pi < \phi(t) - \phi(t') \leq \pi \\ \phi(t) - \phi(t') - 2\pi & for \pi < \phi(t) - \phi(t') \leq 2\pi. \end{cases} \tag{2-15}$$

Since the value of direction change $\Delta\phi(t)$ is distributed in the interval $[-\pi,\pi]$, this change may be a large value. However, the change of movement direction also should be smooth and incremental. Therefore, the large value of $\Delta\phi(t)$ should be divided into several incremental small direction changes $\Delta\varphi(t)$, such that the value of $\Delta\varphi(t)$ should be small value and it represents the maximum allowable value of direction change per time slot. The direction change can be achieved in $\frac{\Delta\phi(t)}{\Delta\varphi(t)}$ time slots.

For each time slot in the period of direction change, the mobile node only changes its movement direction by $\Delta\varphi(t)$ as follows:

$$\phi(t) = \phi(t - \Delta t) + \Delta\varphi(t) \quad (2\text{-}16)$$

This small change in direction is repeated for $\frac{\Delta\phi(t)}{\Delta\varphi(t)}$ time slots until the node reaches the new direction $\phi(t)$. Then, the node continues to move in the new chosen direction.

In this case Smooth Random mobility model is able to capture temporal dependence on velocity in order to avoid some extreme behaviors such as sudden stop, sudden acceleration and sharp turn, which frequently occur in the trace generated by the Random Waypoint model.

The lifetime of WBASN is determined by partly the transmitting range, mobility model and the routing protocol used. The various routing protocols used are discussed below.

2.2.4 Routing Algorithms

Four routing algorithms were used in this research. Single hop is the simplest. Single hop directly transmits to the sink and this is only efficient and the best if and only if the sensor node is transmitting within its operating range. An algorithm like AODV minimizes hop distances and due to significant overhead such as route establishment and synchronization, it tends to consume more power [20]. In Directed Diffusion the sink propagates an “interest” such as emergency ECG signal, to the source, the source set up a gradient and propagates its data back to the sink through a reinforced path. Multi-hop MAC routing protocol sends packets to the immediate neighbors on the path to the base station. At the start of the simulation, nodes maintain a routing table of the position of the immediate neighbors and continue to update their position. Unlike AODV, nodes do not broadcast RREQ and RREP messages to set up the path. It is assumed that the nodes have GPS to track the location of the individual nodes on their path to the base station. The routing of the packets takes place the Medium Access Control MAC layer. Below is the complete discussion of the routing protocols used in this work.

2.2.4.1 Single Hop

This is the simplest routing algorithm. Each node transmits directly to the destination whenever necessary. The power used for transmission is proportional to the square of distance. Martin et al [23] shows that single-hop algorithm consume less energy in relaying data compared to equivalent multi-hop (such as AODV) due to simpler routing protocols, lower communication overhead, and higher overall efficiency. The predictability of power usage makes this method a good test case.

2.2.4.2 AODV [24]

This protocol is designed to allow nodes in an ad-hoc network to find multi-hop paths to a destination and maintain these paths without being affected by changes in topology. This is an IP based protocol. Each node maintains a routing table that it can use to send message on existing paths. If a destination is not listed in the routing table, the node broadcasts a Route Request (RREQ) packet to its neighbors. When a node receives RREQ packet, it checks its routing table for a route to the destination and adds the source and destination to the table. If the node has a route to the destination, it sends a Route Reply (RREP) packet back to the source of the RREQ. Otherwise, it rebroadcasts the RREQ packet. This is effectively a flooding process. However once a stable routes are established, there is no need to continue flooding.

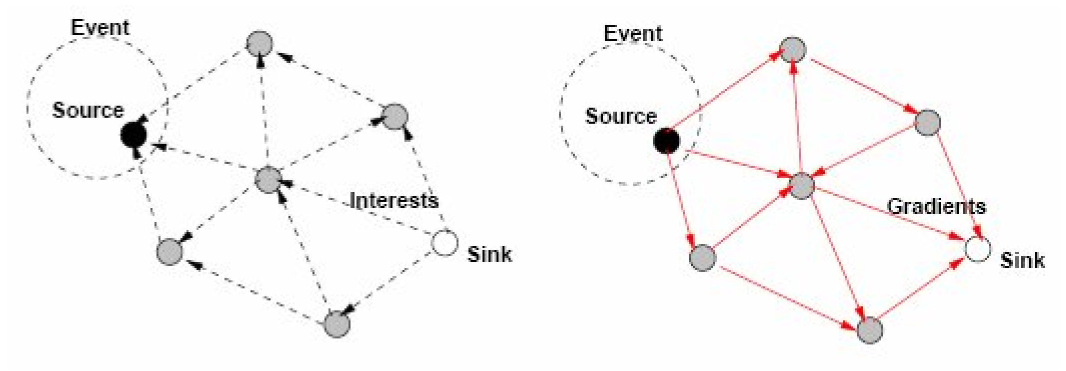
Each node sends a hello packet to its neighbors at regular intervals so that lost routes can be detected. A node that detects a lost route sends a Route Error (RERR) packet toward the source node to ensure that every node has an up to date routing table..

AODV is capable of both unicast and multicast routing. In this work the AODV is configured as a multicast routing protocol.

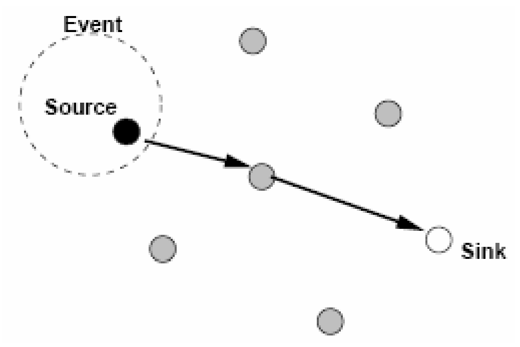
Multicast routes are set up as follow: A node wishing to join a multicast group broadcasts a RREQ with the destination IP address set to that of the multicast group and with the 'J'(*join*) flag set to indicate that it would like to join the group. Any node receiving this RREQ that is a member of the multicast tree that has a fresh enough sequence number for the multicast group may send a RREP. As the RREPs propagate back to the source, the nodes forwarding the message set up pointers in their multicast route tables. As the source node receives the RREPs, it keeps track of the route with the freshest sequence number, and beyond that the smallest hop count to the next multicast group member. After the specified discovery period, the source node unicasts a Multicast Activation (MACT) message to its selected next hop. This message serves the purpose of activating the route. A node that does not receive this message that had set up a multicast route pointer will timeout and delete the pointer. If the node receiving the MACT was not already a part of the multicast tree, it will also have been keeping track of the best route from the RREPs it received. Hence it must also unicast a MACT to its next hop, and so on until a node that was previously a member of the multicast tree is reached.

2.2.4.3 Directed Diffusion [25]

In Directed diffusion, data is named using value-attribute pair. A sensing task is disseminated throughout the sensor network as an “interest” for named data such as “vital ECG signal”. A base station propagates such “interests” to the source through multiple paths. This dissemination sets up gradients within the network designed to “draw” events (i.e. data matching the interest). Events start flowing towards the originators along multiple paths. The sensor network reinforces one or a small number of these paths as shown in Figure 2-6 below.



a) Interest propagation b) Initial gradient setup



c) Data delivery along reinforced path

Figure 2-6: A simplified schematic of directed diffusion protocol

2.2.4.4 Multi-Hop Medium Access Control MAC routing protocol

This is the multi-hop routing protocol uses few hops (at most 3 hops) to relay data to the base station at the MAC layer. The nodes maintain a routing table of the positions of the immediate neighbors. The node chooses the closest node in the direction of the base station as shown in Figure 2-7. This algorithm assumes nodes know the location of all nodes near them. In practice this would require an initial set-up phase where this information is disseminated throughout the network and that each node has a Global Positioning System GPS receiver or other location tracking algorithm such as radio frequency (RF)-based location tracking system to determine the nodes location.

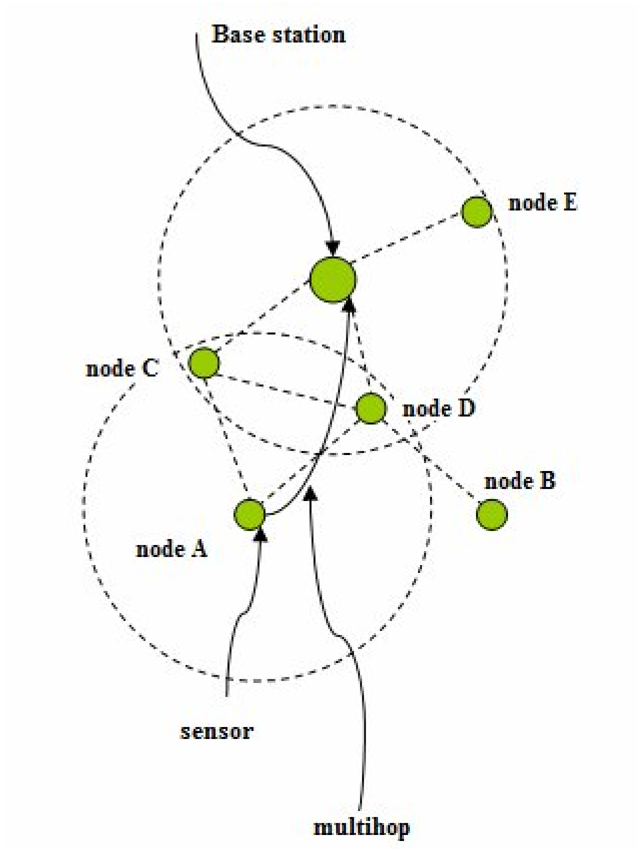


Figure 2-7: Simple multi-hop scheme

The difference between this routing protocol and AODV is that, AODV initially broadcast RREQ and RREP packets to the neighbors and this is effectively a flooding process but unlike a simple Multi-hop protocol, no flooding occurs, it only maintains a routing table of immediate neighbors on the path to the base station. This is done through a localization algorithm which disseminates the locations of the nodes through the network and periodically updates the nodes location. The updating of the nodes location is very important in maintaining active route especially in mobile network where the position of the node changes frequently. In order to prevent a large number of collisions the routing algorithm uses carrier sense multiple access with collision avoidance CSMA/CA scheme. The nodes backoff when the channel is busy at a random time.

CHAPTER 3: MONTE CARLO METHOD

Due to the complexity of the system involved, the Monte Carlo method is best suited to this problem. It is the most basic but most accurate method for probabilistic analysis. The goal of using Monte Carlo for the health monitoring network is to determine how random variations of the WBASN parameters such as detection and propagation of stimuli or mobility affect the performance of the network.

3.1 Monte Carlo Simulation

This method involves generating random inputs vectors with known distribution and then running the simulation with these vectors as input. The resulting output vectors will provide the output distribution. A deterministic model generates the same results no matter how many times the input is recalculated. By using random inputs, a deterministic model is turned into a stochastic model. Monte Carlo simulation can then be performed with the input distributions. The number of iterations N necessary to obtain a probability that is precise to within δp with confidence $1-\varepsilon$ is shown in Equation 3-1.

$$N \geq \frac{2}{p\delta^2(1-\delta)} \ln\left(\frac{2}{\varepsilon}\right) \tag{3-1}$$

3.2 Steps in Monte Carlo Simulation

In order to use Monte Carlo simulation a parametric model of the health monitoring network is created with sets of random inputs (for ECG, pulse oxygen saturation, power of signals source, mobility) generated. The steps in Monte Carlo simulations are shown below:

1. **Create** a parametric model of the health monitoring application, y = *f*(x1, x2 ... xn).
2. **Generate** a set of random inputs (ECG signals, SpO_2, mobility, power of source signals), x*i*1, x*i*2, ..., x*iq*.
3. **Evaluate** the model and store the results as y*i*.
4. **Repeat** steps 2 and 3 for *i* = 1 to *n*.
5. **Analyze** the results using histograms, summary statistics, confidence intervals, etc.

Table 3-1: Steps in Monte Carlo Simulation

3.3 Inputs Randomization and Distributions

The sensor input sources are fixed and generate stimuli at known distributions. The stimuli are ECG and pulse oxygen saturation.

ECG: The sources generate ECG stimuli at constant intervals ΔT_s with exponential distributions given by the Equation 3-2. The exponential distribution was chosen because it represents a constant average rate.

$$f(x;\lambda) = \lambda_s e^{-x\lambda_s} \tag{3-2}$$

where λ_s is the rate parameter and x represents the ECG samples.

The parameters of the ECG stimuli which are the complexes (QRS complex, R complex, T complex and P complex), inter-wave segments and cardiac intervals are sampled as Gaussian process with Gaussian distribution given by Equation 3-3. The Gaussian distribution was chosen because the detection of parameters in the ECG wave is independent of the other.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{(x-\mu)^2}{2\sigma^2})} \quad (3\text{-}3)$$

where P(x) is the overall probability of detecting the parameters - QRS complex, R-R wave and P complex, x is the number of times of detecting the parameters, σ is the standard deviation and μ is the mean.

Pulse Oxygen Saturation (SpO$_2$): The SpO$_2$ stimuli are generated by the sensor input sources (targets) at constant interval and detected by the sensors at constant interval with exponential distribution. Similarly, exponential distribution was chosen because it represents constant average rate as shown in Equation 3-4:

$$f(x;\lambda) = \lambda_s e^{-x\lambda_s} \quad (3\text{-}4)$$

where λ_s is the rate parameter and x represents the SpO$_2$ samples.

Power: The power of the events (ECG and SpO$_2$ signs) generated also has a Gaussian distributions given by Equation 3-5:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{(P_i-\mu)^2}{2\sigma^2})} \quad (3\text{-}5)$$

where P$_i$ is the power of the events at any time t, μ is the expectation (mean) and σ is the standard deviation.

Mobility: The mobile sensor nodes move in a certain preferred speed $\{V_{pref}^1, V_{pref}^2, ..., V_{pref}^n\}$. The mobility model of the node is Smooth Random mobility. The preferred speed set for each node is assumed to be random. If a node has a preferred speed set $\{0, 0.5V_{\max}, V_{\max}\}$, then the probability distribution is

$$P(v) = \begin{cases} P(v=0)\delta(v) & v = 0 \\ P(v=0.5V_{\max})\delta(v-0.5V_{\max}) & v = 0.5V_{\max} \\ P(v=V_{\max})\delta(v-V_{\max}) & v = V_{\max} \\ \dfrac{1-P(v=0)-P(v=0.5V_{\max})-P(v=V_{\max})}{V_{\max}} & 0 < V_{\max} < 1 \\ 0 & \mathit{otherwise} \end{cases} \qquad (3\text{-}5)$$

where $P(v=0)+P(v=0.5V_{\max})+P(v=V_{\max})<1$.

The speed of the mobile node is changed incrementally from the current speed $v(t^{'})$ to the targeted new speed $v(t)$ by acceleration speed or deceleration speed $a(t)$ with a probability distribution function given by Equation (3-6).

$$P(a) = \begin{cases} \dfrac{1}{a_{\max}} & \mathit{acceleration}: 0 < a \le a_{\max} \\ \dfrac{1}{a_{\min}} & \mathit{deceleration}: a_{\min} \le a \le 0 \\ 0 & \mathit{otherwise} \end{cases} \qquad (3\text{-}6)$$

The movement *direction* of the sensor nodes is purely uniformly distributed in the interval $[0, 2\pi]$, with probability distribution given by Equation 3-7

$$P_{\phi}(\phi) = \frac{1}{2\pi} \qquad \mathit{for}\, 0 \le \phi \le 2\pi \qquad (3\text{-}7)$$

The frequency of direction change is assumed to have an exponential distribution. When the direction of a mobile node is about to change, the new movement direction is also selected according to the probability distribution function described by Equation 3-7. The direction difference $\Delta\phi(t)$ between the new direction $\phi(t)$ and the old direction $\phi(t')$ is defined as

$$\Delta\phi(t) = \begin{cases} \phi(t) - \phi(t') + 2\pi & for - 2\pi < \phi(t) - \phi(t') \leq -\pi \\ \phi(t) - \phi(t') & for - \pi < \phi(t) - \phi(t') \leq \pi \\ \phi(t) - \phi(t') - 2\pi & for \pi < \phi(t) - \phi(t') \leq 2\pi \end{cases} \tag{3-8}$$

The node continues to move in the new direction with the given distribution.

Finally random input vectors are generated with the Monte Carlo simulation with the above distribution to produce the resulting output vectors which provide the output distributions.

CHAPTER 4: IMPLEMENTATION OF MONTE CARLO SIMULATION

The mobile nodes in the simulation are modeled to have the power characteristics similar to the CodeBlue. This results in failure times that may be as long as three years. Therefore simulating the entire life of the network will often require a very long time. Since the simulation must be run hundreds to thousands of times to get the precise probabilities, Monte Carlo analysis using full lifetime simulation is impractical. An alternative is to split the simulation into two stages and rely on the power used by nodes being roughly constant in the steady state. The first phase assumes that the nodes are fully charged batteries. The output of this phase is the average power used by each node which is used to determine the energy for the second phase.

4.1 Stage 1 Simulation

1. In stage 1, each node is initialized with the fully charged batteries (i.e. the energy is E_{max}). The output is the energy ΔE_i used by each node in a fixed time (T_e = 500s in this work). The simulation is run for time T_e to obtain the average power used by each node. The time is determined based on the average interval τ as $Te = n\ \tau$ (n =5 was used in this work). The average power used by each node is $P_i = \frac{\Delta E_i}{T_e}$.

4.2 Stage 2 Simulation

2. In stage 2, the powers obtained in stage 1 are used to determine when the first node fails. The second stage uses sequential approximation to reduce the running

time as follows: The energy consumed by each node in stage 2 run is approximated to

$$E_{consumed} = E_{\max}\left(\frac{P_i}{\max\{P\}}\right) \tag{4-1}$$

The starting energies of each node in stage 2 are, therefore

$$E_i = E_{\max}\left(1 - \frac{P_i}{\max\{P\}}\right) \tag{4-2}$$

This approximation ensures that the node using maximum power has zero energy at the start of the next iteration. Monte Carlo method can now be performed using the starting energies in Equation 4-2. It is now obvious that the runs form a geometrical sequence. Then the starting energies of each node before each simulation run are given by sequence:

Run 1:

$$E_i^0 = E_{\max}\left(1 - \frac{P_i}{\max\{P\}}\right) \tag{4-3}$$

Run 2:

$$E_i^1 = E^0\left(1 - \frac{P_i}{\max\{P\}}\right) \tag{4-4}$$

Run 3:

$$E_i^2 = E^1\left(1 - \frac{P_i}{\max\{P\}}\right) \tag{4-5}$$

Then the starting energy for the nth run would be

$$E_i^{n-1} = E\max\left(1 - \frac{P_i}{\max\{P\}}\right)\left(1 - \frac{P_i}{\max\{P\}}\right)\left(1 - \frac{P_i}{\max\{P\}}\right)...\left(1 - \frac{P_i}{\max\{P\}}\right) \tag{4-6}$$

This approximation makes computing the energy use trivial until nodes start to fail. The full simulation process is outlined in Table 4-1.

Table 4-1: Simulation process using power use approximation

```
begin Stage 1
    Run the simulation multiple times for Te simulated time
    Obtain values for $P_i$ from each run
end
repeat Stage 2
    Compute new energies Ei
    Restart each run for $T_e^{'}$ simulated time
    Obtain values for $P$i from each run
until nodes have 0 energy
```

4.3 Estimation of the time for first node to fail

It can be deduced from the Equation 4-6 that before the first node fails, it energy is given by Equation 4-7.

$$E_i^{n-1} = E_{\max}\left(1 - \frac{P_i}{\max\{P\}}\right)^{n-1} \tag{4-7}$$

Where n the number of runs before the node fails

The limit of Equation 4-7 is approximately found to be $E_{\max}$

Therefore the first node is assumed to fail when

$$t = T_{skip} = \frac{E_{\max}}{\max\{P\}} \tag{4-8}$$

Each simulation run M generates the number of working nodes N as a function of time t given by $N(t)$. The change in time for the nodes to reduce from maximum working nodes N to $N-1$ is given as the time for the first node to fail. The full process is outlined in Table 4-2 below:

Table 4-2: Estimating the time for the first node to fail

```
begin
  For each run $M$
     Determine the $average\{T_e\}$ for maximum nodes $N$ given as $t_N$
     Determine the $average\{T_e + T_{skip}\}$ for nodes to reduce to $N-1$ as $t_{N-1}$
     Compute time for first node to fail:
     $t_{first\_node} = t_{N-1} - t_N$
end : where $T_e$ is the set-up time
```

4.4 Stochastic Properties

The simulation runs form a Geometric distribution with expectation given by $1/p$ where p is the probability of first node failing. The probability of the first node failing after the stage 2 is given by Equation 4-9

$$p \le E[P_i]\frac{T_e + T_{skip}}{E_{\max}} \tag{4-9}$$

The expected number (mean or average) of iterations M before a node fails is shown in Equation 4-10

$$E[M_i] = \frac{1}{p} \ge \frac{E_{\max}}{(T_e + T_{skip})E[P_i]} \tag{4-10}$$

4.5 Output Distribution

The results of the Monte Carlo method is a probability distribution of the average power consumption of each node before nodes start to fail, and the distributions of the node loss as a function of time. The particular output distribution depends on the routing protocol. For example, the nodes running single hop routing protocol normally use power uniformly, therefore the simulation of nodes using this routing protocol results in uniform distribution of the average power consumption. Assuming the output distribution is

Gaussian or normal distribution, the cumulative normal distribution is found by the integral [27]

$$N(x) = \int_{-\infty}^{x} n(t)dt \tag{4-11}$$

With $n(t)$ the normal density function

$$n(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{4-12}$$

The integral of Equation 4-11 has no closed form but can be approximated using Algorithm 26.2.17 from Abromowitz and Stegun, Handbook of Mathematical Functions [28] given by Equation 4-13. It has a maximum absolute error of 7.5e^-8.

$$N(x) = 1 - n(x)(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5)$$

$$t = \frac{1}{1 + 0.2316419x} \tag{4-13}$$

with

$$b_1 = 0.319381530$$
$$b_2 = -0.356563782$$
$$b_3 = 1.781477937$$
$$b_4 = -1.821255978$$
$$b_5 = 1.330274429$$

4.6 Scripts

The Monte Carlo method and the sequential approximation method are implemented using Python scripts. The scripts rely entirely on preexisting J-Sim modules with the modifications added by Merizzi [29]. The sources were modified to allow exponential distributions of the ECG and pulse oxygen saturation stimuli. The sensors were also modified to sample the detected ECG stimuli with Gaussian distribution and Pulse

Oxygen saturation stimuli with exponential distribution. The mobility of the sensor nodes is modeled was Smooth Random model with the speed modeled with the distribution according to the preferred speed list of each node, and the direction of movement of the nodes is purely uniformly distributed. There are two versions of each script. The stage 1 version positions the nodes and save the energies. The stage 2 version initializes the node energies in a resume file. The resume file is created by the stage 2 python script. The results from the scripts are a large number of files containing the state of the network for different configurations and times. The timing and distributions of the node failures are obtained using MATLAB and Microsoft Office Excel.

CHAPTER 5: RESULTS

5.1 Simulation Scenario

Networks with 23 bio-sensor nodes, a base station and two (2) target or source nodes were simulated. Simulations are carried out with the J-Sim network simulator [29]. Each patient wearing the bio-sensor moves about in the square grid of 200m by 200m. The motion of each patient is simulated with the Smooth Random Mobility model, which exhibits temporal dependence on velocity. Each patient has a preferred speed set $\{V_{pref}^{1}, V_{pref}^{2}, V_{pref}^{3}, \ldots, V_{pref}^{n}\}$, where V_{pref}^{i} is chosen randomly and lies in the interval given by: $0 \leq V_{pref}^{i} \leq V_{\max}$. The maximum speed $V_{\max}$ chosen for this work is 10m/s.

Each sensor node is injected with the appropriate traffic rate (8Kbits/s for the ECG sensor and 64bits/s for the pulse oximeter sensor). The Constant Bit Rate CBR traffic sources are used in the simulation and the transport agent is the User Datagram Protocol UDP. The J-Sim free-space propagation is used as the propagation model. The wireless physical layer parameters are adjusted according to those of the CodeBlue [5] platform, which utilizes the Chipcon CC2420 radio interface [30]. The initial energy of all nodes is 25200J and the simulator running time is 500s.

Four routing protocols were used to simulate the WBASN: Single-hop, AODV, Directed diffusion and Multihop MAC routing protocols. The single-hop provides the baseline. AODV is capable of both unicast and multicast routing. However, it is configured for multicast routing to conform to the standards of the CodeBlue platform.

The Directed diffusion by definition is a multicast routing protocol. The Multihop MAC routing protocol consists of nodes that relay packets to their immediate neighbors on the path to the database. This is by definition a unicast routing. The number of runs was different for each protocol because of the varying running times for each protocol. A significant amount of time is required to obtain a good result. A good probability estimate can be obtained in about 3-4 days for Single hop and Multihop MAC and 5-7 days for AODV and Directed diffusion.

Three metrics are defined for studying the performance of the WBASN network.

- Power flow: The distribution of the power consumed by each node during the stage 1 simulation.
- Network Failure: The distribution of the number of working nodes for the entire simulation as a function of time.
- Nodes' Lifetime: The time it takes for the first, second, third, …, and nth, nodes to fail.

5.2 Discussions

The simulations of WBASN with J-Sim network simulator reveal interesting results. The power consumed by the mobile nodes is partially due to the transmission range, data rates, routing protocols used and the mobility model. Even though the Smooth random mobility model, which has temporal dependence on velocity, is used, at least some packets are expected to be dropped as a result of a node failing to receive proper acknowledgement ACK from another node on its routing table. Recall that IEEE 802.15.4 drops a packet when three (3) transmissions fail to receive a proper acknowledgement. On the other hand, nodes which by virtue of mobility, lie within an

8m radius of the base station at any point in time would conserve power by transmitting directly to the base station through a Single hop while those at the boundaries of the grid would have to route through multiple hops to the base station.

The energy flow for the ECG sensor is much higher than that of the pulse oximeter because ECG has the greatest data rate and uses a complex algorithm for the detection of its parameters (R complex, T complex, QRS complex, etc.). However, the routing protocols have the greatest influence on the power consumption since a protocol that employs multiple hops and aggregation such as AODV and Directed diffusion consumes more power than the Single hop protocol. The Single-hop power use clearly increases with the square of the distance because it employs direct transmission and this is consistent with the radio transmission model. Much like the Single hop routing protocol, the Multi-hop MAC protocol relays packets through a few hops when the nodes are outside their operating range. The results of the simulation are discussed below.

5.2.1 Power Flow Analysis

This section discusses the distribution of power consumed by each node after the stage 1 simulation by representing the data as a set of Cumulative Distribution Functions CDFs. Figure 5-1 shows a set of CDFs for power use at different times after a WBASN using the AODV protocol is simulated. Considering the distribution for a 50% probability (mean performance), the power consumed by the nodes in the first 100s, 200s, and 300s are 2.6mW, 2.1mW and 2.0mW respectively. More power is consumed by the AODV protocol in the first few seconds. The AODV is configured as multicast routing protocol with no stable routes established initially. Therefore the nodes use more power broadcasting and relaying route request RREQ and route reply RREP messages in the first 100 seconds of the simulation to join the multicast group and update multicast trees. The AODV protocol maintains route for as long as the route is active. This includes maintaining a multicast tree for the entire life of the multicast group. Because the network nodes are mobile, it is likely that many link breakages along a route will occur during the lifetime of that route. However when the network stabilizes, less power is consumed (2.0mW compared to 2.6mW in the initial stages of the simulation).

Figure 5-2 shows a set of CDFs for power use at different times after a WBASN using the Single-hop protocol is simulated. Considering the distribution for 50% probability (mean performance), the power consumed by the nodes in the first 100s, 200s and 300s are 0.75mW, 0.8mW, and 0.70mW respectively. On the average, the power consumed by the Single hop protocol in the 500s of simulation time is far lower than the power consumed by the AODV protocol because the network stabilizes much quicker. Secondly, the Single hop protocol is just direct transmission to the base station with no relaying of RREQ and RREP packets, route discovery, retransmission of dropped packets, multicast tree management, synchronization overhead or updating of route table as in the case of a multicast routing protocol such as AODV. Also, the power consumed in receiving a packet is greater than that of transmitting a similar packet [31]. This is due to the fact that the average power consumption in the interval of 500s of the simulation for the Single hop protocol (which is 0.8mW) is only about one-third of the average for the AODV protocol (which is 2.2mW).

Fig 5-3 shows the CDF of power use at different times after the WBASN using the Multi-hop MAC routing protocol is simulated. During the first 100s of the simulation the nodes consume much less power than during the rest of the simulation. This is because at the start of the simulation the nodes use less power, in sending and receiving Request-To-Send RTS and Clear-To-Send CTS packets from the neighbors to acquire the channel and the path to the base station. If the channel is busy, the nodes wait at a random backoff time (amount of time that the node should wait before trying to retransmit the packet). Few nodes acquire the channel whilst the rest backoff at random time. Less power is drawn from the sensors during backoff time. Much like the Single hop routing protocol, nodes also consume power in the uniform distribution. During the latter stage of the simulation, almost constant power, 0.7mW, is drawn though the nodes periodically update their neighbors and path to the base station.

Figure 5-4 shows the CDFs of power use at different times after the WBASN using the Directed diffusion routing protocol is simulated. Unlike the previous protocols, the Directed diffusion power use is unrealistic. The nodes consume maximum power within the first 100 seconds of the simulation run and turn off completely during the last 200 to 500 seconds of the simulation. Two factors may account for this unexpected behavior. One may be the simplified battery model used and the other factor may be the fact that the nodes do not transmit any signal to the base station due to the low range of the event propagation. However, the same battery model was used for the previous routing protocols and the range of the propagation of the event was set to be a circle with a radius of 250m; therefore, even a node at one corner of the 200m by 200m rectangular grid should be able to receive the event. The only alternative cause is that J-Sim network simulator is not well optimized and/or is inefficient in simulating a complex routing protocol like Directed diffusion.

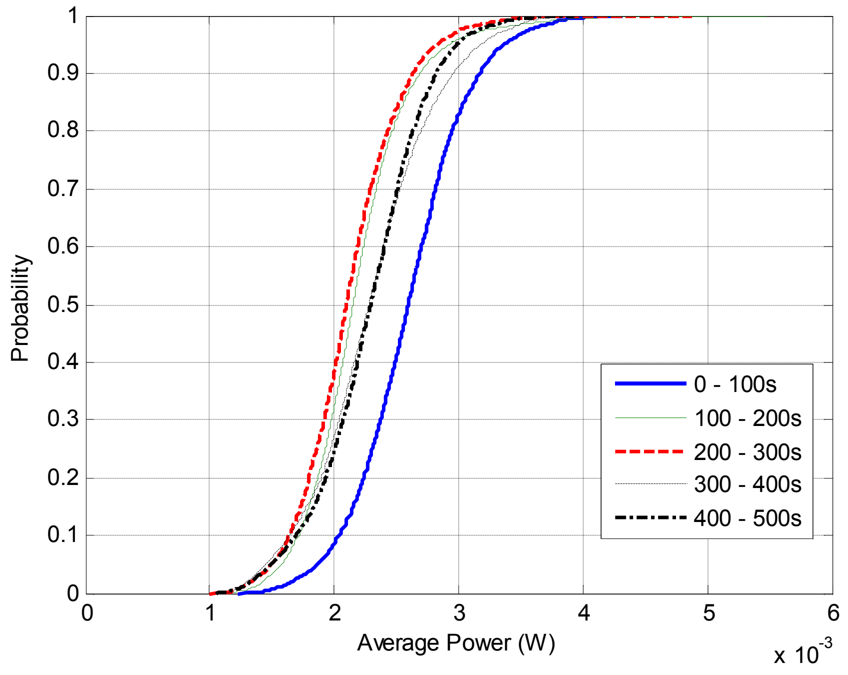


Figure 5-1: CDFs of the average power use by nodes running the AODV protocol.

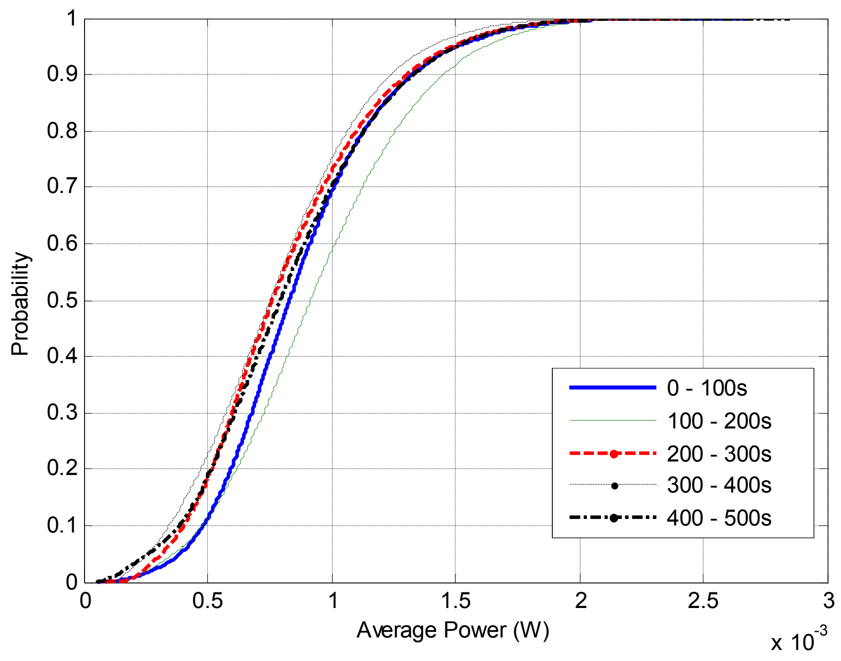


Figure 5-2: CDFs of the average power use by nodes running the single protocol.

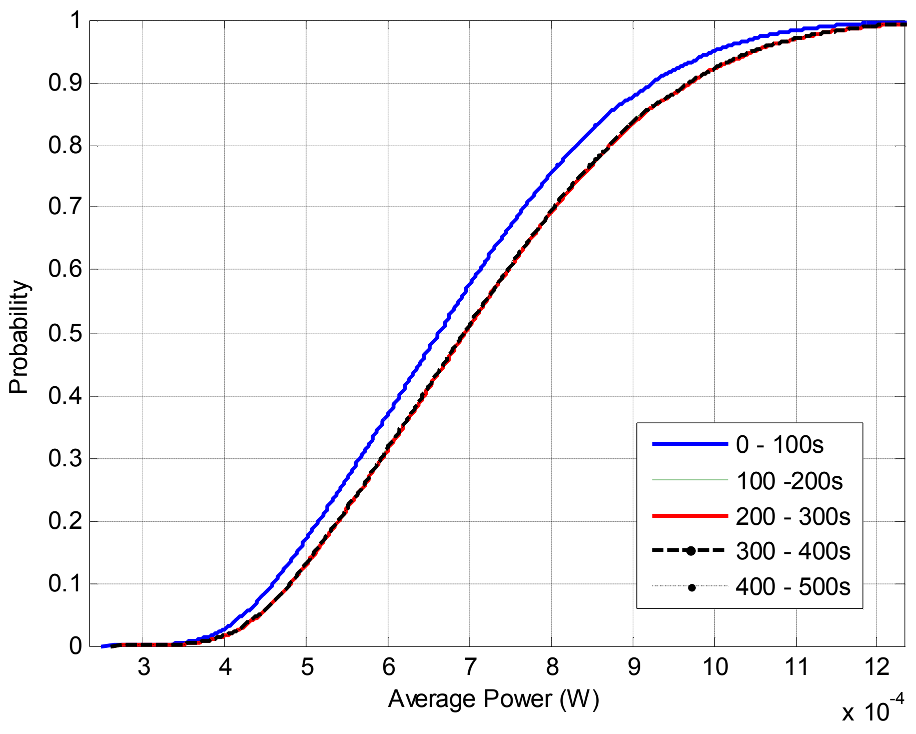


Figure 5-3: CDFs of the average power use by nodes running the Multi-Hop MAC

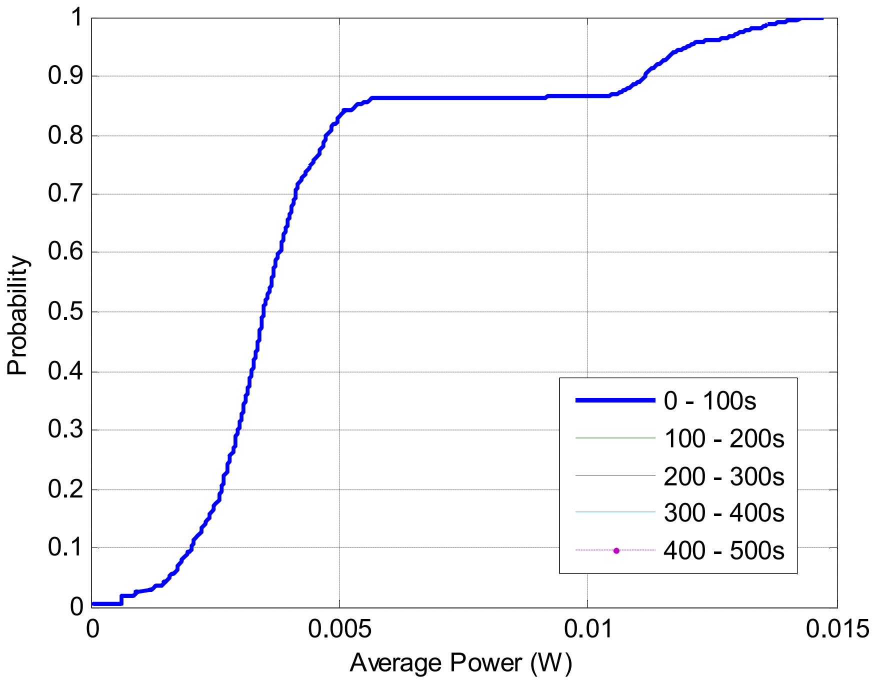


Figure 5-4: CDFs of the average power use by nodes running the directed diffusion protocol.

5.2.2 Network Infrastructure Failure Analysis

This analysis consists of representing the number of working nodes at any given time as contour bands. Figure 5-5 shows the distribution of the number of working nodes for the AODV protocol. Each independent data point is the number of working nodes at some point in time. The points of interest are the bounds rather than the specific paths of the contours. The bands represent different levels on the CDFs of the number of working nodes. The central line (marked 0.5) is the median performance. Likewise, the outer line (marked 0.95) has the probability of 95%; that is, the number of working nodes below the 0.95 line with probability of 95%.

Considering the median performance, some nodes last for 130 days which is quite consistent with the average power usage in Figure 5-1. From Figure 5-1, the average power used when running the AODV protocol is 2.2mW. Then the average lifetime for any node running the AODV protocol is $25.2KJ / 2.2mW \approx 133$ days or 4 months. This confirms that the observed behavior of the results for AODV is similar to what would expect.

Figure 5-7 shows the distribution of the working nodes at some time for the Single hop protocol. Each independent data point is the number of working nodes at some time. Likewise, considering the median performance (marked 0.5), some nodes last for 1600 days ≈ 5 years. From Figure 5-2, the average power consumed when running the Single hop protocol is 0.8mW. The average or expected lifetime for any node running this protocol is calculated as $25.2KJ / 0.8mW \approx 10$ years. This value seems not to be consistent with the average power use. The inconsistency is due to the fact that some nodes consumed power above the average during the stage 2 simulations. In this case localization of the nodes can be a major factor. Recall that the nodes use a Smooth Random mobility model and it makes sense to assume that, for about 70 percent of the time, more than half of the nodes were moving along the border of the 200m x 200m rectangular grid. The operating range of the nodes is 100m.

Figure 5-9 shows the distribution of the working nodes at some time for the Multi-hop MAC routing protocol. Considering the median bound, some nodes last for 1200 days ≈ 3.2 years. This Figure seems more realistic than the estimated lifespan for the nodes running on Single hop (10 years). The number of active nodes drops off smoothly in a network using the Multi-hop MAC protocol as opposed to the rapid drop that occurs when using the AODV protocol (Figure 5-5).

An alternative way to look at the same data is to consider the time to lose a certain fraction of the nodes. Figure 5-6 shows the time obtained for the AODV routing protocol. The data for this distribution is separated by runs. The runs for 10%, 50% and 96% CDFs are shown, which is interpreted as the distribution of the time to lose 90%, 50%, and 4% of the nodes respectively. These distributions can be used to approximately calculate the time that it takes for the first node running AODV to fail. These calculations can be done as follows: Considering that the simulated network consisted of 23 nodes, the runs corresponding to the subplot (in Figure 5-6) that shows “96% of the nodes remaining” mean that $4/100*23 \approx 1$ node has failed. Considering the mean (marked 0.5 on the Probability axis) performance, the time on the “Time (days)” axis is approximately 104 days. This confirms the results obtained for the time the first node fails in Section 5.1.2.3.

The lifetime of the network running AODV is very short. The 50% and 90% CDFs in Figure 5-6 are somewhat similar indicating that the number of nodes drops quickly once the nodes start to fail.

Figure 5-8 shows an alternate representation of the results for the Single hop protocol. As before, the number obtained for 10%, 50% and 96% are shown, which are interpreted as the distribution of the time to lose 90%, 50%, and 4% of the nodes respectively. The time for the first node running Single hop to fail can approximately be found as follows: The runs corresponding to the subplot (in Figure 5-8) for "96% of the nodes remaining" means that $4/100*23 \approx 1$ node has failed. Considering again the mean performance, the time on the "Time (days)" axis is approximately 220 days. Therefore, the observed behavior for the results of the Single hop and AODV routing protocols are similar to what would be expected.

Figure 5-10 shows an alternate representation of the results for the Multi-hop MAC routing protocol. The time for the first node running Multi-hop to fail can approximately be found as follows: The runs corresponding to the subplot for "96% of the nodes remaining" means that $4/100*23 \approx 1$ node has failed. Considering the mean performance, the time on the "Time (days)" axis is approximately 320 days. The network running on Multi-hop MAC has a greater lifespan than the networks of Single hop and AODV routing protocols.

Fig 5-11 shows an alternate representation of the results for the nodes running the Directed diffusion routing protocol. Since the first stage simulation gave unexpected results (shown in Figure 5-4), the sequential approximation in Equation 4-2 employed in its stage 2 simulations also gave unexpected behavior. The results look very noisy and the lifetime for the first node (97200 days or 266 years) is unrealistic.

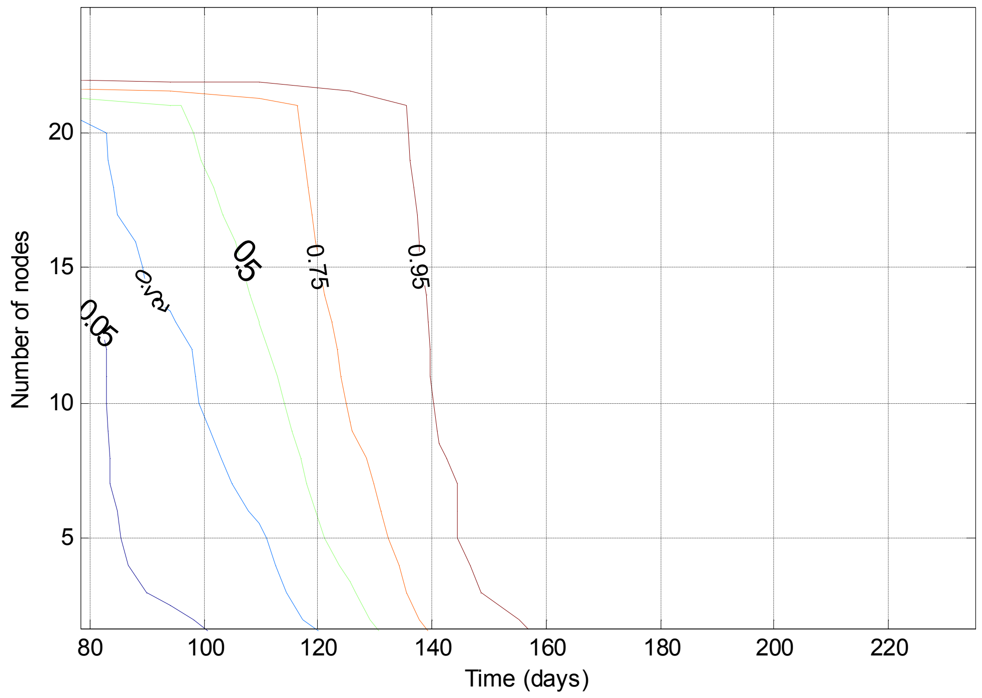


Figure 5-5: Number of active nodes vs. time for AODV protocol. Upper band is 95% probability and lower band is 5%probability.

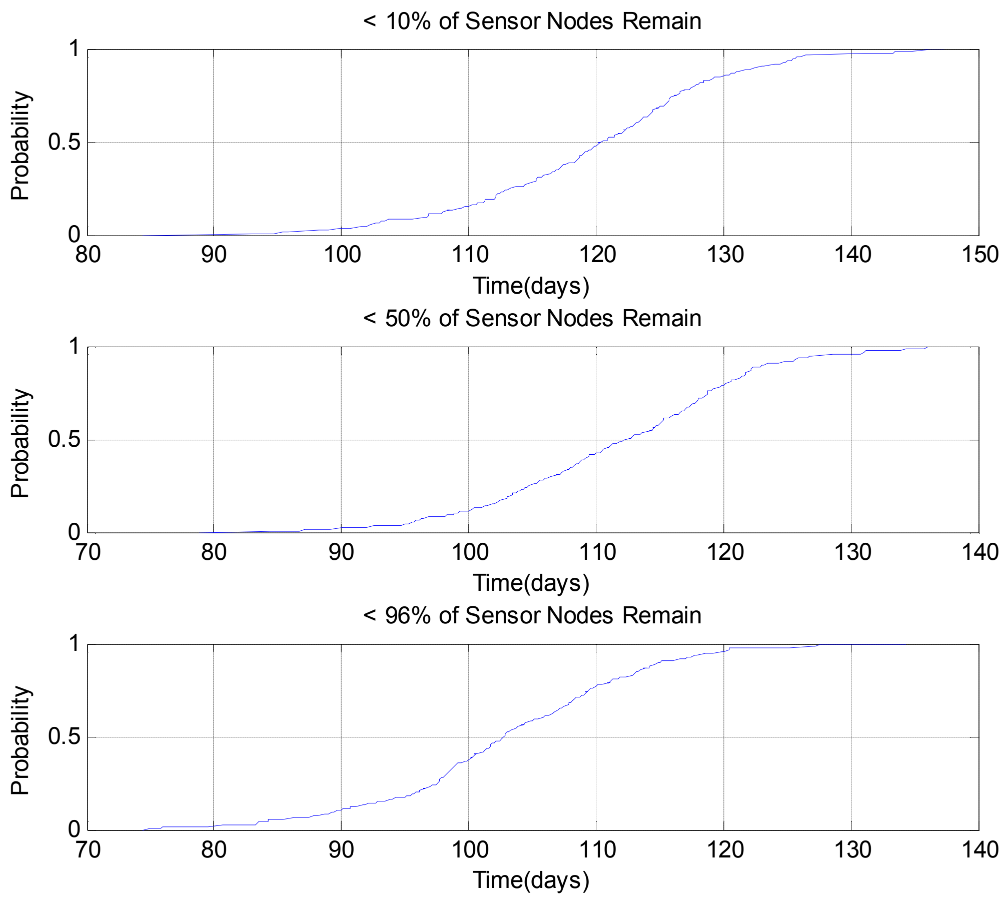


Figure 5-6: Distributions of time required to be left with 96%, 50%, and 10% working nodes for a network using AODV protocol.

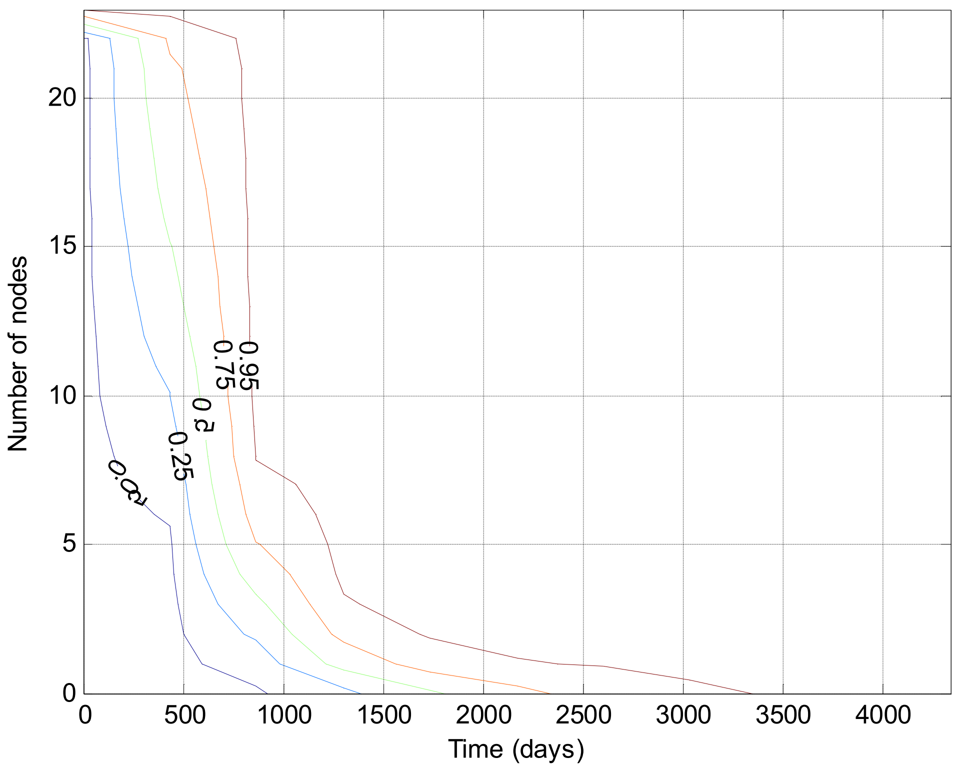


Figure 5-7: Number of active nodes vs. time for single-hop protocol. Upper band is 95% probability and lower band is 5% probability.

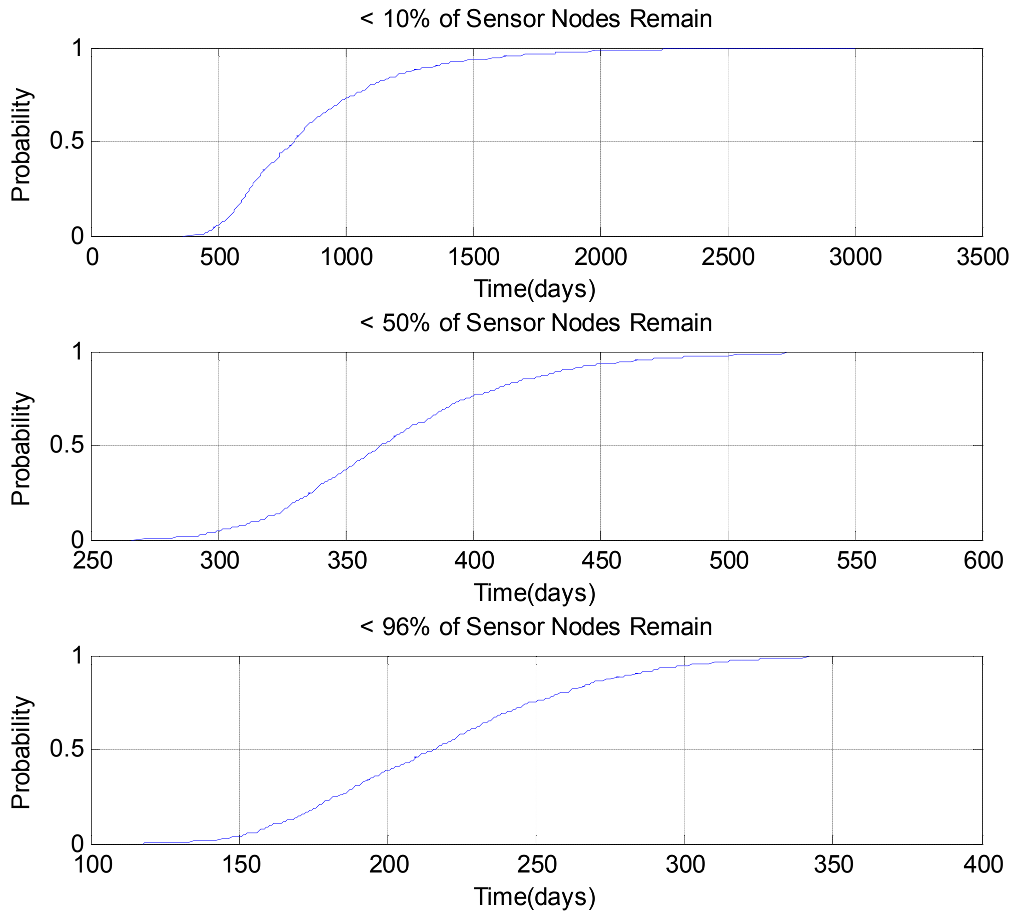


Figure 5-8: Distributions of time required to be left with 96%, 50%, and 10% working nodes for a network using Single hop protocol.

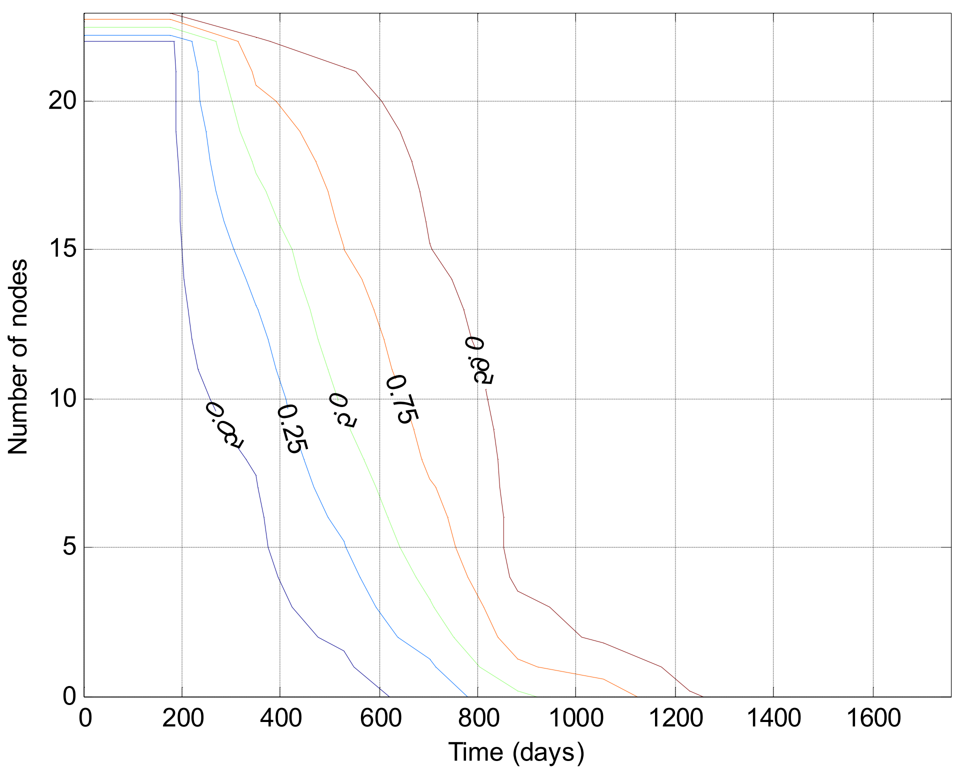


Figure 5-9: Number of active nodes vs. time for multi-hop MAC protocol. Upper band is 95% probability and lower band is 5% probability.

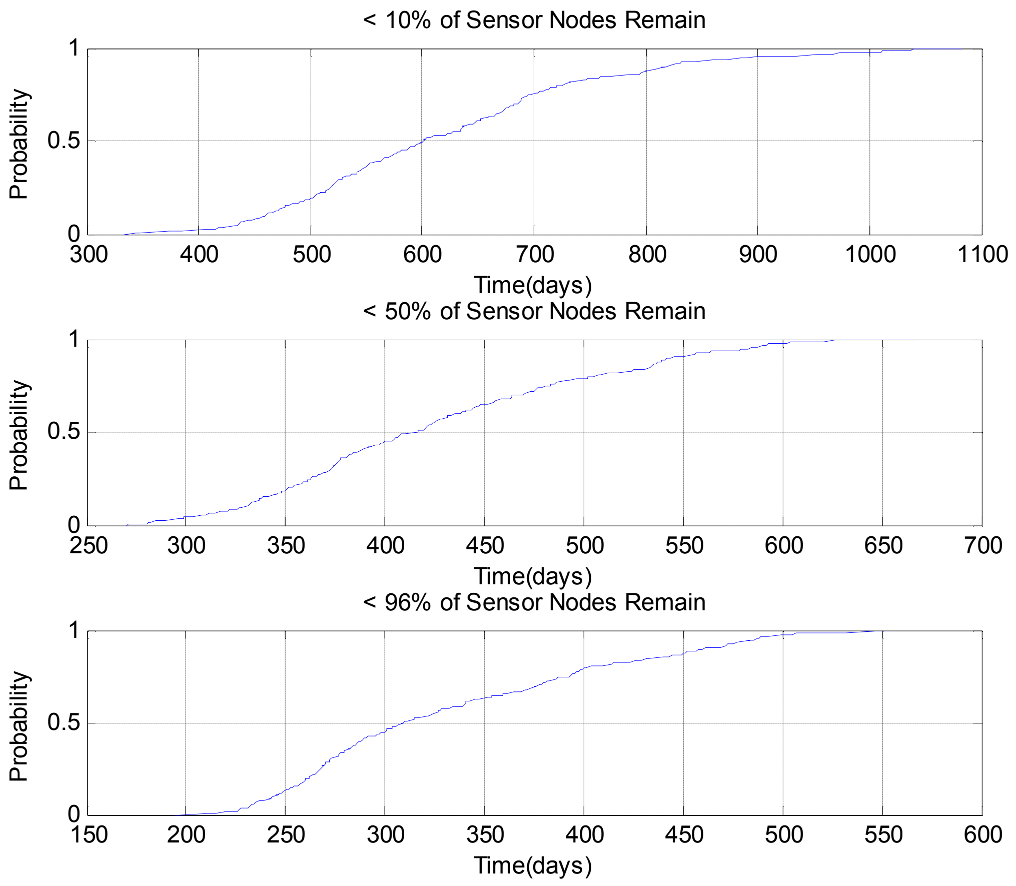


Figure 5-10: Distributions of time required to be left with 96%, 50%, and 10% working nodes for a network using Multi-hop MAC protocol.

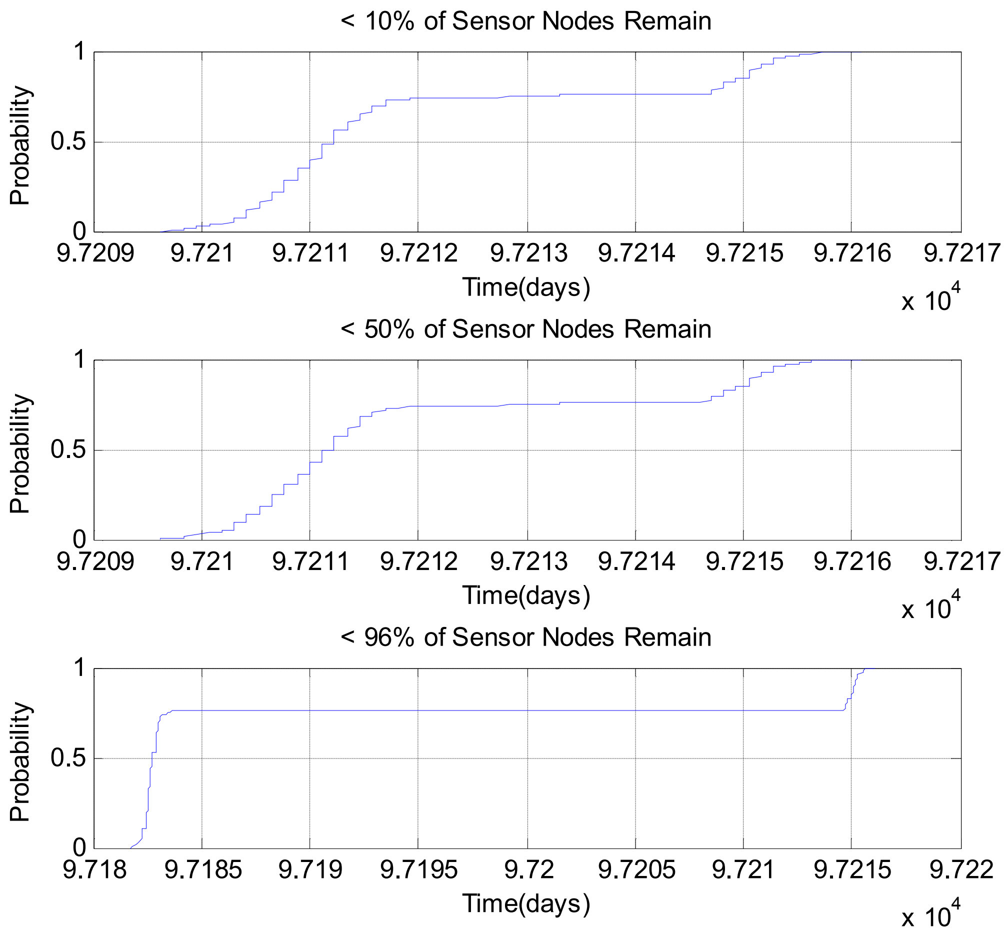


Figure 5-11: Distributions of time required to be left with 96%, 50%, and 10% working nodes for a network using Single hop protocol.

5.2.3 Nodes’ Lifetime

This section discusses the time for the first, second, third, fourth and fifth nodes to fail. Only the time for the first node to fail is needed, but times for up to the fifth node are discussed to determine the failure pattern. Since the network failure is a stochastic process, the mean failure and its standard deviations were determined.

Figure 5-12 shows the nodes’ lifetime for the AODV routing protocol. The mean time for the first node to fail is approximately 100 days with a standard deviation of 13 days. This is roughly in line with the way the nodes consume average power in the stage one simulations. The times for the second, third, fourth and fifth nodes to fail are 103, 104, 105, and 108 days respectively. The differences in times for the nodes to fails are relatively short (approximately 2-day intervals), indicating that the number of nodes drop quickly once the first node fails. This shows that the nodes running the AODV routing protocol configured as multicast routing consume relatively the same power in receiving, broadcasting, and retransmitting. The mean of the standard deviations for the five distributions is approximately 10 days, which is quite reasonable.

Figure 5-13 shows the node’s lifetime for the Single hop routing protocol. The mean time for the first node to fail is 190 days with a standard deviation of 30 days. This is consistent with the way the nodes consume average power in stage one simulations. From the calculation in Section 5.1.2.2, it is estimated that the last node may last for 1600 days or 5 years. The difference in time for the first and last node to fail is significantly wider indicating that the nodes drop slowly once the first node begin to fail. In general, the Single hop protocol consumes twice as much power as the AODV protocol. The time for the second, third, fourth and fifth nodes to fail are 217, 233, 250, and 264 days respectively with approximately 14-day intervals. However, the estimated standard deviation of 30 days for the first node to fail seems too large. The per-run errors, seem in the distributions of the number of nodes are considerably large for Single hop routing protocol.

Figure 5-14 shows the node's lifetime for the Multi-hop MAC routing protocol. The mean time for the first node to fail is 339 days with a standard deviation of 84 days. This is consistent with the way the nodes consume average power in stage 1 simulations. In general the network running the Multi-hop MAC protocol has a longer lifespan due to the simplicity of the protocol as compared to the complexity of the AODV routing protocol. However, a standard deviation of 87 days is too wide to make this protocol a good choice for WBASN. The per-run errors seem in the distribution of the number of nodes is far greater than the Single hop routing protocol. The result for the network running on Directed diffusion in Figure 5-15 does not provide useful information" since its stage 1 and stage 2 results produced unexpected and unrealistic data. A more optimized simulation environment such as NS-2 [11] may give a good result.

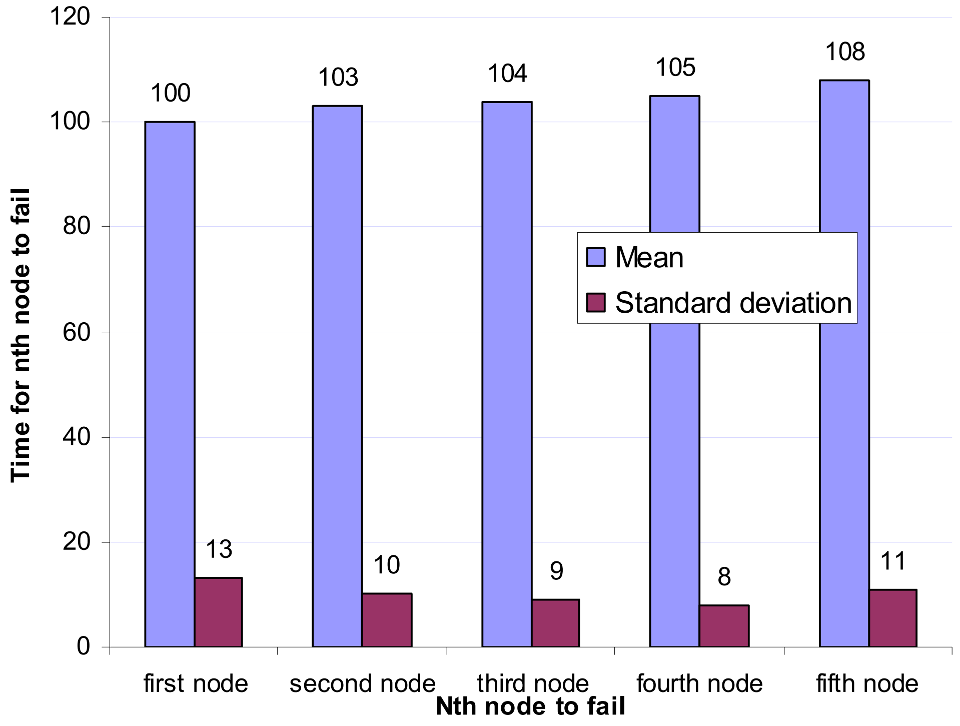


Figure 5-12: Mean Time for 1st, 2nd, 3rd, 4th and 5th nodes to fail for AODV routing protocol.

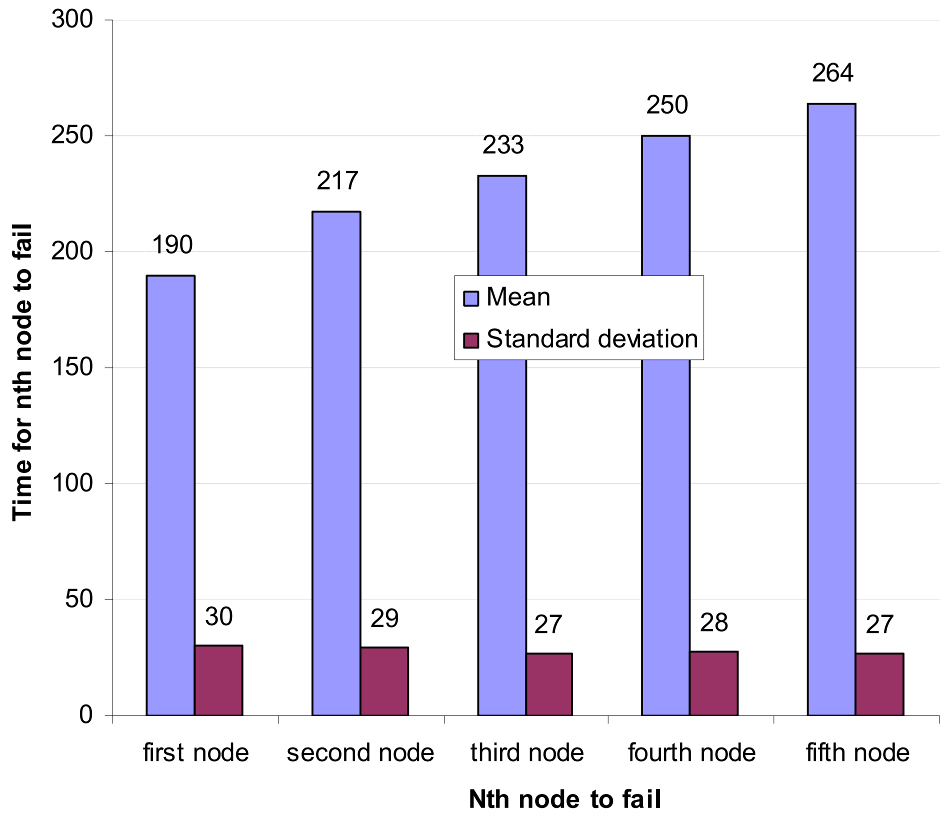


Figure 5-13: Mean Time for 1st, 2nd, 3rd, 4th and 5th nodes to fail for Single hop routing protocol.

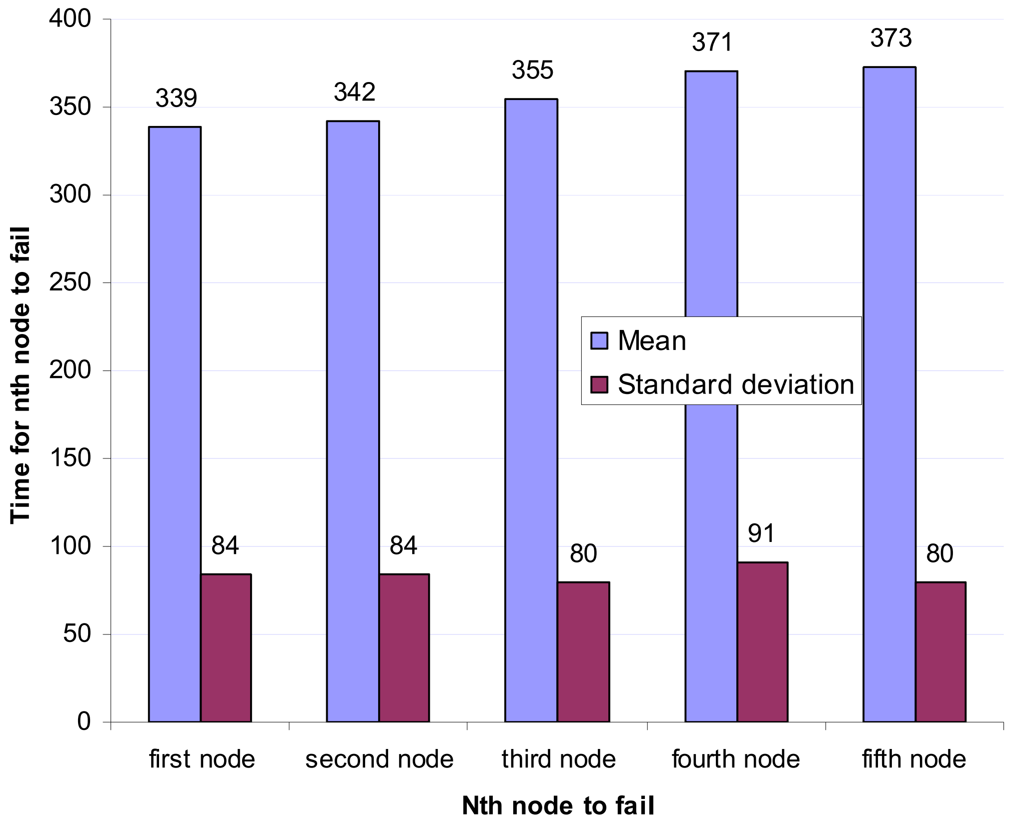


Figure 5-14: Mean Time for 1st, 2nd, 3rd, 4th and 5th nodes to fail for Multi-hop MAC routing protocol.

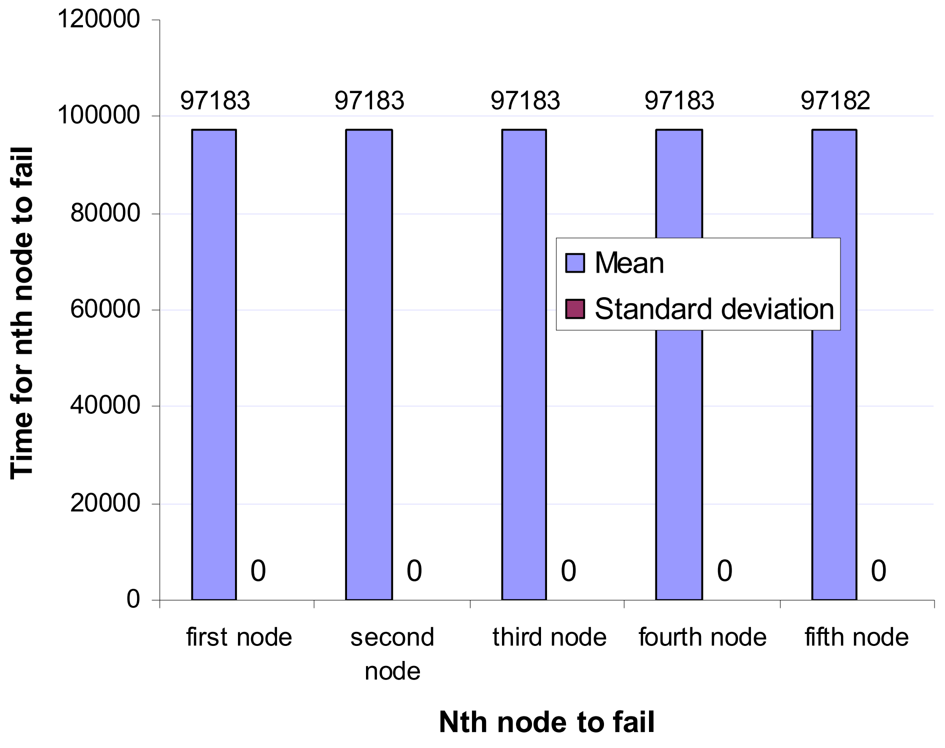


Figure 5-15: Mean Time for 1st, 2nd, 3rd, 4th and 5th nodes to fail for directed diffusion routing protocol.

CHAPTER 6: CONCLUSION

Probability analysis using Monte Carlo method is used as a tool for estimating the lifetime of WBASN for patient health monitoring. Four routing algorithms (protocols) were used to test the method. The network is simulated using J-Sim simulator. It was chosen because of the convenient modular structure, and existing framework for simulating wireless sensor networks. The mobile nodes are modeled with Smooth Random mobility model. The tool produced some interesting results and reveals some setbacks when simulating WBASN with J-Sim network simulator. The results of the probabilistic method for the four routing algorithms are summarized in the table 6-1 below.

Criteria	**AODV**	**Single hop**	**Multi-hop MAC**	**Directed Diffusion**
1. Power consumption during stage 1 runs	Approximately **2.2mW** on the average	Approximately **0.7mW** on the average	Approximately **0.8mW** on the average	Consumed **2.5mW** on the average
2. Lifetime of network*	Very short: 100 days (approx.) with standard deviation of 13 days	About 190 days (approx.) with standard deviation of 30 days	About 339 days (approx.) with standard deviation of 84 days	No useful information

3. Per node statistics	About 250 runs were feasible; this leads to 1250 samples. This leads to a per node probability error of about 9% with a confidence of 95%	About 800 runs were feasible; this leads to 4000 samples. This leads to a per node probability error of about 12% with a confidence of 95%	About 250 runs were feasible; this leads to 1250 samples. This leads to a per node probability error of about 20% with a confidence of 95%	No useful information
4. CPU usage	62% of processor capacity	41% of processor capacity	42% of processor capacity	64% of processor capacity

**Lifetime of the network used here is defined as the duration of time until the first sensor failure due to battery depletion*

Table 6-1: Comparison of the Probabilistic analysis with various routing protocols

Table 6-1 shows the performance of the various routing algorithms with respect to the probabilistic method. The nodes running on Multi-hop MAC protocol has a greater lifetime than that of the nodes running on both Single hop and AODV routing protocols but a standard deviation of 84 days obtained for the network running on the Multi-hop MAC routing protocol shows that the per-errors for the protocol is far greater than that of Single hop and AODV. However, the results obtained from simulating the Directed diffusion routing protocol with J-Sim network simulator did not provide any useful information. The stage 1 simulation of the network running on Directed diffusion produced results that are unrealistic. For the first few seconds of the simulation run, the nodes consume maximum power and turned off completely. Two factors may account for this unexpected behavior. One may be the simplified battery model used and the other factor may be due to the fact that the nodes do not transmit any signal to the base station due to the low range of the event propagation. However, the same battery model was used for previous routing protocols so the simplified battery model may not be actual cause. Also the range of the propagation of the event (target) source may not be the actual cause of the unexpected behavior because the radius of the event source was configured to a radius of 250m; therefore even a node at one corner of the 200m by 200m rectangular grid can receive the event. The only alternative is that J-SIM network simulator is not well optimized and efficient for a complicated routing protocol like Directed diffusion.

For per-node statistics, about 800 runs were feasible for single hop. This leads to 4000 samples, so a probability of 0.05 is accurate within 12% with a confidence of 95%. In the case of AODV and Multi-hop MAC, 250 runs were possible. This leads to a per node probability error of about 9% and 15% respectively. Throughout the execution of the simulation, the processor was running at more than 40% for both single hop and Multi-hop MAC and more than 60% for both AODV and Directed diffusion protocols. This shows that J-Sim is not very well optimized. Monte Carlo based simulation works best with very fast simulations (more runs means smaller per node errors and tighter confidence bounds). If precise distributions are needed, a quicker simulation environment such as Ns-2 may be preferable.

Overall, the probabilistic method proves to be an effective means of estimating the lifetime of the WBASN and the observed lifetimes of the first node of the network running Single hop, AODV and Multi-hop algorithms are similar to what would be expected.

BIBLIOGRAPHY

[1] C. Otto, A. Milenkovic, C. Sanders, E. Jovanov, “System architecture of a Wireless Body Area Sensor Network for Ubiquitous Health Monitoring,” Journal of Mobile Multimedia, Vol. 1, No.4, pages. 307-326, 2006

[2] R. Bults, K. Wac, A. Halteren, D. Konstantas, V. Jones, I. Widya, “Body Area Networks for Ambulant Patient Monitoring Over Next Generation Public Wireless Networks,” 3rd IST Mobile Wireless Communications Summit 2004, Lyon, France, 27-30 June 2004

[3] K. Van Laerhoven, B. P. L. Lo, J. W. P. Ng, S. Thiemjarus, R. King, S. Kwan, H. Gellersen, M. Sloman, O. Wells, P. Needham, N. Peters, A. Darzi, C. Toumazou and G. Yang, “Medical Healthcare Monitoring with Wearable and Implantable Sensors”, International workshop on Ubiquitous Computing for Pervasive healthcare applications (UbiHealth), September 2004

[4] N. Oliver and F. Flores-Mangas, “HealthGear: A Real-time Wearable System for Monitoring and Analyzing Physiological Signals,” Microsoft Research Technical Report, MSR-TR-2005-182, 2005

[5] CodeBlue: Wireless Sensors for Medical Care. http://fiji.eecs.harvard.edu/CodeBlue.

[6] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks, Mobile Computing, pages 153-181. Kluwer Academic Publishers, 1996.

[7] E. Royer, P.M. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. In Proceedings of the IEEE International Conference on Communications (ICC) 2001, Vol 3, pages 857-86, 11-14 June 2001.

[8] KH Chiang, N. Shenoy, "A 2-D random-walk mobility model for location-management studies in wireless networks," *Vehicular Technology, IEEE Transactions on*, vol. 53, no. 2, pages. 413-424, March 2004.

[9] ZigBee Review, FreeScale Semiconductor, www.freescale.com

[10] D. Curren: “A Survey of Simulations in Sensor Networks”, University of Binghamton.

[11] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns.

[12] V. Naoumov and T. Gross. “Simulation of Large Ad Hoc Networks”. Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, San Diego, CA, USA, pages. 50-57, 2003

[13] P. Levis, N. Lee, M. Welsh and D. Culler: "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications". Proceedings 1st international conference on embedded networked sensor systems, Los Angeles, California, pages 126-127, 2003.

[14] S. Park, A. Savvides, and M. B. Srivastava. "SensorSim: A Simulation Framework for Sensor Networks". Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, Boston, Massachusetts, pages 104-111, August 2000.

[15] A. Sobeih and J. C. Hou. : "A Simulation Framework for Sensor Networks in JSim". Technical Report UIUCDCS-R-2003-2386, November 2003.

[16] D. C. Reddy, Biomedical Signal Processing, Principles and Techniques, published by Tata McGraw-Hill Publishing Company Limited, 2005, pages. 265 – 290

[17]W. C. Yang, Ph.D. "Edge Detection Using Template Matching (Image Processing, Threshold Logic, Analysis, Filters)", Duke University 1985, 288 pages; AAT 8523046

[18] Pulse Oxygen Saturation, www.pulseox.info

[19] C. Bettstetter. "Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks," Proceedings of the 4th ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, Rome, Italy, pages 19 -27, July 2001.

[20] Q. Wang, M. Hempstead, W. Yang: "A Realistic Power Consumption Model for Wireless Sensor Network Devices," Sensor and Ad hoc Communications and Networks, 2006. SECON'06. 2003 3rd Annual IEEE Communications Society on, vol. no. 1, pages 286-295, 28-28 Sept. 2006

[21] S. Kellner, M. Pink, D. Meier and E.O Blass: "Towards a Realistic Energy Model for Wireless Sensor Networks," Wireless on Demand Network Systems and Services, 2008. WONS 2008. Fifth Annual Conference on , vol.,no., pages 97-100, 23-25 Jan. 2008

[22] H. Shu, Q. Liang, J. Gao: "Wireless Sensor Network Lifetime Analysis Using Interval Type-2 Fuzzy Logic Systems," Fuzzy Systems, 2005. FUZZ '05. The 14th IEEE International Conference on, vol., no., page(s):19 – 24, May 22-25, 2005

[23] M. Haenggi and D. Puccinelli: "Routing in Ad hoc Networks: A case for Long Hops", Communication Magazine, IEEE, vol. 43, no. 10, pages 93-101, Oct. 2005

[24] E. Belding-Royer and C. Perkins: “Evolution and future directions of the ad hoc on-demand distance-vector routing protocol”. Ad hoc Networks Journal, Vol. 1 no. 1, pages. 125-150, July 2003.

[25] C. Intanagonwiwat, R. Govindan and D. Estrin. : “Directed diffusion: a scalable and robust communication paradigm for sensor networks”, Proceedings of the 6th ACM conference on mobile computing and networking, Boston Massachusetts, pages 56-67, 2000

[26] TECHNICAL SPECIFICATIONS: MAXELL ALKALINE BATTERIES [BUTTON CELL]: http://www.microbattery.com/tech-maxell-alkaline.htm

[27] M.A. (Thijs) van den Berg**,** Calculating the Cumulative Normal Distribution, http://www.sitmo.com/doc/Calculating_the_Cumulative_Normal_Distribution

[28] Abromowitz and Stegun, ”Handbook of Mathematical Functions” , Algorithm 26.2.17, http://www.math.sfu.ca/~cbm/aands/page_932.htm

[29] N. Merizzi and W. F. S Poehlman,”J-Sim and the demand for wireless sensor network simulation”. Tech. Rep. CAS 2005-03-SP, McMaster University

[30] Chipcon CC2420 Zigbee/IEEE 802.15.4 RF Transceiver, www.ti.com

[31] M. Haenggi, "Twelve reasons not to route over many short hops," Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th , vol.5, no., pages. 3130-3134 Vol. 5, 26-29 Sept. 2004

[32] A. Haldar and S. Mahadevan, Probability, Reliability and Statistical methods in Engineering Design. Published by John Wiley and Sons, Inc, 2000

APPENDIX A: MATLAB SCRIPTS DESCRIPTION

A.1 Stage 1 Simulation Matlab Scripts

1. Script name: loaddata1.m

Purpose: This script loads data from the stage 1 log files into the Matlab environment for further analysis.

Prerequisite: This script should be added to the log files generated from the stage 1 simulations.

Parameters: None

Below is loaddata1.m script

```
% Load stage 1 data
files=dir('sensorInfo*.log');

%Declare arrays to hold time, energy, live nodes and
%locations of nodes
t=[];
energy=[];
live=[];
locations=[];

%Scan through all the files in the folder and extract time
%energy, live nodes and location data of the nodes

for i=1:length(files)
    tmp=sscanf(files(i).name,'sensorInfo%06d-%d.log');
    n=tmp(1)+1;
    k=tmp(2)+2;
    f=fopen(files(i).name);
    files(i).name;
    t(k)=fscanf(f,'%g',1);
    tmp=fscanf(f,'%g %g',[2,inf])';
    energy(n,:,k)=tmp(:,1);
    live(n,k)=sum(1-tmp(:,2));
    fclose(f);

end
%Store the energies of the live nodes for next simulation
energy(:,:,1)=25200*ones(size(energy,1),size(energy,2));
```

2. Script name: powerdists.m

Purpose: This scripts plots CDFs of the power use

Prerequisite: The loaddata1.m must be executed first

Parameters: None

Below is powerdists.m script

```
% CDFs of power use
% energy must contain energies of the nodes
power=-diff(energy,1,3)/10;
a=[];
p=[];
N=[];
X=[];
pu=[];
for n=1:size(power,3)
    pu(:,n)=reshape(power(:,:,n),[],1);
    a(:,n)=sort(pu(:,n));
    p(:,n)=(0:size(a,1)-1)/size(a,1);
    [N(:,n),X(:,n)]=hist(a(:,n),100);
end
plot(a,p);
xlabel('Average Power (W)'),ylabel('Probability'),grid

axis([0 0.001 0 1]);
```

A.2 Stage 2 Simulation Matlab Script

1. Script name: loaddata2.m

Purpose: This script loads data from the stage 2 log files into the Matlab environment for further analysis.

Prerequisite: This script should be added to the log files generated from the stage 2 simulations.

Parameters: None

Below is loaddata2.m script

```
% Load stage 2 data.

%Select all sensorInfo files:
files=dir('sensorInfo*-1.log');

%defining empty arrays
t=[]; %array of time offsets
N=[]; %array containing numbers of alive nodes
run=[]; %run indices

for i=1:length(files)
    run(i)=sscanf(files(i).name,'sensorInfo%*03d%03d-1.log'); %get
run index from filename
    f=fopen(files(i).name); %open file
    t(i)=fscanf(f,'%g',1); %get time offset from first line
    tmp=fscanf(f,'%g %g',[2,inf])'; % get array of energies and
statuses
    N(i)=23-sum(tmp(:,2)); % count alive nodes and save in the N
array
    fclose(f);
end
```

2. Scriptname: **tdist.m**
Purpose: This plots the CDFs of the time required to have less than ***q*** nodes alive.
Prerequisites: **loaddata2.m** executed.
Parameters:

- **t** (array) – array containing time stamps, defined in **loaddata2.m**
- **N** (array) – array containing number of nodes alive at various times, defined in **loaddata2.m**
- **run** (array) – array containing run index, defined in **loaddata2.m**
- **q** (array) – array containing percents of alive nodes for plotting. ***Optional value***. Example: q=[10, 50, 90]

Below is tdist.m script

```
% Create a plot CDFs of the time required to have less than q nodes
% t: Sample times
% N: Samples (containing the number of nodes)
% run: the run index of the samples in N

function tdist(t,N,run,q)

runmax=max(run); %count number of iterations
q1=[]; %initialize the array

for i=1:length(q)
    q1(i)=(q(i)*23)/100 %convert percents to numbers
end

figure %create a new graphic window

for a=1:length(q)
    tmin=[]; %initialize array of times when number of nodes is less
than q%

    %this iterations find when the numbers of nodes got below
selected rate
    %for each run
    for i=0:runmax %for each run
        k=find(run==i & N<q1(a)); %find indices when number of nodes
< that q1
        if length(k)>0 %if something is found
            tmin=[tmin,min(t(k))]; %then add lowest time to the
array of times
        end
    end
    %plot CDF of the times when number of nodes got below selected
rate
    subplot(length(q1),1,a);
    plot(sort(tmin)/(3600*24),(0:length(tmin)-1)/length(tmin));
    xlabel('Time(days)'),ylabel('Probability')
    title(sprintf('< %d%% of Sensor Nodes Remain',q1(a)/23*100))
    grid on
end
```

3. Scriptname: **lossdist.m**
Purpose: This plots intervals containing all possible number of nodes at all times.
Prerequisites: **loaddata2.m** executed.
Parameters:

- **t** (array) – array containing time stamps, defined in **loaddata2.m**
- **N** (array) – array containing number of nodes alive at various times, defined in **loaddata2.m**
- **tbins** (array) – array containing desired time bins. ***Optional value***. Example: bord=max(t)/10; tbins=bord:bord:max(t)

Below is lossdist.m script

```
% Create a plot of the interval containing all possible numbers of
nodes
% at all times.
% t: sample times
% N: samples
% tbin: time bins
function [Nbin,map]=lossdist(t,N,tbin)

tmax=max(t); %calculate top maximum lifetime
intervals=length(tbin); %number of intervals
%Nbin - used to calculate map
Nbin=cell(intervals,1); %initialize of the Nbin array
tlast=0; %initialize tlast

%initialize map
map=zeros(24,intervals+1);
map(24,1)=1;
%this iterations
for k=1:intervals %for each interval
    idx=find(tlast<t & t<=tbin(k));  %select times in current time
bin
    Nbin{k}=sort(N(idx)) %get numbers of alive nodes in that times
and sort them
    if length(Nbin{k})>0 %if such numbers are found
        map(:,k+1)=histc(Nbin{k},0:23)'; %then count how many times
each number is present
    end
    tlast=tbin(k); %set interval beginning to the low border of the
next time bin
end
figure %create a new graphic window
%this will make column of ones multiplied by row of map columns sum
S=sum(map)
mapsum=ones(size(map,1),1)*S;
map=cumsum(map)./mapsum;
[X,Y]=meshgrid([0,tbin]/(24*3600),0:23);
He=[0.05,0.25,0.5,0.75,0.95];
He1=[0.05,0.5,0.95];
[C,h]=contour(X,Y,map,He);
clabel(C,h,'manual');
grid on
xlabel('Time (days)'),ylabel('Number of nodes')
```

4. Script name: firstnode.m
Purpose: This script plots the bar graphs of the nodes' lifetimes
Prerequisite: This script should be added to the log files generated from the stage 2 simulations.
Parameters: None
Below is firstnode.m script

```
% Print statistics of the times for the 1st, 2nd and 3rd nodes to
fail
% t: Sample times
% N: Samples (containing the number of nodes)
% run: the run index of the samples in N


files=dir('sensorInfo*-1.log');

%defining empty arrays
t=[]; %array of time offsets
N=[]; %array containing numbers of alive nodes
run=[]; %run indices
D=[];
tmp=[];
for i=1:length(files)
    run(i)=sscanf(files(i).name,'sensorInfo%*03d%03d-1.log'); %get
run index from filename
    f=fopen(files(i).name); %open file
    t(i)=fscanf(f,'%g',1); %get time offset from first line
    tmp=fscanf(f,'%g %g',[2,inf])'; % get array of energies and
statuses
    N(i)=23-sum(tmp(:,2)); % count alive nodes and save in the N
array
    D(i)=sum(tmp(:,2)); % count dead nodes and save in the N array
    fclose(f);
end
length(tmp)
tmin1=[];
tmin2=[];
tmin3=[];
tmin4=[];
tmin5=[];
x= [];
runmax = max(run);

for i=0:runmax %for each run
        k=find(run==i & N==22); %find indices when number of nodes =
that q1
        if length(k)>0 %if something is found
            tmin1=[tmin1,min(t(k))]; %then add lowest time to the
array of times
        end
    end
tset = sort(tmin1)/(3600*24);
tset_min = min((tmin1)/(3600*24));
tset_max = max((tmin1)/(3600*24));
```

```
x(1,1) = mean(tset);
x(1,2) = std(tset);


for i=0:runmax
    k=find(run==i & N==21);
    if length(k)>0
        tmin2=[tmin2,min(t(k))];
    end
end
tset=sort(tmin2)/(3600*24);
tset_min = min((tmin2)/(3600*24));
tset_max = max((tmin2)/(3600*24));
x(2,1) = mean(tset);
x(2,2) = std(tset);


for i=0:runmax
    k=find(run==i & N==20);
    if length(k)>0
        tmin3=[tmin3,min(t(k))];
    end
end
tset=sort(tmin3)/(3600*24);
tset_min = min((tmin3)/(3600*24));
tset_max = max((tmin3)/(3600*24));
x(3,1) = mean(tset);
x(3,2) = std(tset);


for i=0:runmax
    k=find(run==i & N==19);
    if length(k)>0
        tmin4=[tmin4,min(t(k))];
    end
end
tset=sort(tmin4)/(3600*24);
x(4,1)=mean(tset);
x(4,2)=std(tset);

for i=0:runmax
    k=find(run==i & N==18);
    if length(k)>0
        tmin5=[tmin5,min(t(k))];
    end
end
tset=sort(tmin5)/(3600*24);
x(5,1)=mean(tset)
x(5,2)=std(tset)
xsource = {'first node(s)','second node(s)','third node(s)','forth
nodes','fifth node(s)'};
bar(x,'group'),colormap(cool)
```

APPENDIX B: PYTHON SCRIPTS DESCRIPTION

B.1 Stage 1 Simulation Python Scripts

1. Script name: **montecarlo.py**
Purpose: This calls the stage 1 TCL file to generate the stage 1 simulation log files and appends the number of runs to the end of each log file
Prerequisite: This script should be included with the stage 1 TCL file for the routing algorithm
Parameters: None

Below is montecarlo.py script

```
import os
import re

if __name__=='__main__':
    for i in range(0,800):
        cmdline='java drcl.ruv.System ..\OH_thesis\OH_p.tcl %06d' %i
        print cmdline
        os.system(cmdline)
```

B.2 Stage 2 Simulation Python Scripts

1. Script name: **resume.py**
Purpose: This python file initializes the energies of the nodes (after stage 1 simulation runs) in a log files before the sequential approximation is performed.
Prerequisite: This script should be added to the log files generated from the stage 1 simulations.
Parameters: None

```
import os
import re

def makeResumeFile(i,j):
    filename='locations%06d.log' % i
    try:
        f=open(filename,'r')
    except IOError:
        return -1
    locs=[]
    fp_pat=r'[-+]?(?:\d+(?:\.\d*)?|\d*(?:\.\d+)?)(?:[eE][-+]?\d+)?'
    exp=re.compile('(%s) (%s)'%(fp_pat,fp_pat))
    for line in f:
        m=exp.match(line)
        locs.append((m.group(1),m.group(2)))
    f.close();

    files=os.listdir('.')
    maxindex=-1
    exp=re.compile('sensorInfo%03d%03d\\-%s' % (j,i,r'(\d+)'))
    for fn in files:
        m=exp.match(fn)
        if m!=None:
            n=int(m.group(1))
            if n>maxindex:
                maxindex=n




    power=[]
    exp=re.compile('(%s) (%s)'%(fp_pat,fp_pat))
    emax=25200.0

    filename='sensorInfo%03d%03d-%d.log' %(j,i,maxindex)
    f=open(filename,'r')
    lines=f.readlines()
    targetCount=len(locs)-len(lines)+1;
    time1=float(lines.pop(0))
    energy1=[]
    dead=[]
    for line in lines:
        m=exp.match(line)
        if m==None:
            continue
        E=float(m.group(1))
        dead.append(int(m.group(2)))
        energy1.append(E)
    f.close()

    if sum(dead)==len(dead):
        return 1

    filename='sensorInfo%03d%03d-%d.log' %(j,i,maxindex-1)
    f=open(filename,'r')
    f=open(filename,'r')
    lines=f.readlines()
    targetCount=len(locs)-len(lines)+1;
    time2=float(lines.pop(0))
    energy2=[]

    for line in lines:
        m=exp.match(line)
        if m==None:
                continue

            E=float(m.group(1))
            energy2.append(E)
        f.close()

        pmax=0
        for k in range(len(energy1)):
            P=(energy2[k]-energy1[k])/(time1-time2)
            power.append(P)
            if P>pmax and dead[k]==0:
                pmax=P
                kmax=k

        if pmax==0:
            return 1
        filename='resumeState%03d%03d.log' % (j+1,i)
        f=open(filename,'w')
        Tskip=energy1[kmax]/pmax
        f.write('%g\n' % (Tskip+time2))
        for k in range(len(power)):
            if dead[k]==0:
                E=energy2[k]-power[k]*Tskip
            else:
                E=0
            if E<0:
                E=0

            f.write('%s %s %f\n' % (locs[k][0],locs[k][1],E))
        for k in range(len(power),len(locs)):
            f.write('%s %s\n' % locs[k])

        f.close()
        return 0
```

2. Script name: **runToEnd.py**

Purpose: This calls the stage 2 TCL file to generate the stage 2 simulation log files and appends the number of runs to the end of each log file.

Prerequisite: This script should be included with the stage 1 TCL file for the routing algorithm

Parameters: None

```
import resume
import os
from optparse import OptionParser
if __name__=='__main__':
    parser=OptionParser()
    parser.add_option('-s',dest='start',type="int")
    parser.add_option('-e',dest='end',type="int")
    parser.add_option('-t',dest='scriptname')
    parser.set_defaults(start=0,end=800,scriptname='OH_t.tcl')
    (options,args)=parser.parse_args()
    rval=0
    for i in range(options.start,options.end):
        print 'Run %d' % i
        j=0
        while True:
            rval=resume.makeResumeFile(i,j)
            if rval!=0:
                break
            j=j+1
            cmdline='java drcl.ruv.System %s %03d%03d' %
(options.scriptname,j,i)
            print cmdline
            os.system(cmdline)
        if rval==-1:
            print 'Done'
            break
        else:
            print 'End of iteration %d' % j
```

APPENDIX C: MODIFYING THE J-SIM TCL FILES

C.1 Stage 1 TCL script

1. In order to run the stage 1 TCL scripts from command line using Python, a run index and count variable to should be added to the scripts as shown below:

   set run_index [lindex $argv 0]

   The run-index tells the Python scripts the number of iterations the TCL scripts should be run and also append the index to the name of the log files generated
2. The TCL files are run for a total time of 500 seconds in 100second interval, a count variable is needed. This count variable is added as shown:

   set count 0

3. The mobility of the nodes can be set in two ways:
   a) By assigning values to the setPosition variable with the arguments: speed (m/sec), xCoord, yCoord, zCoord as shown below:

   ! n$i/mobility setPosition 10.0 $Xpos $Ypos 0.0

   b) By reading from a scenario file generated from a particular mobility model such as Smooth Random Mobility model as shown below

   ! n$i/mobility setPosition "smooth_mobility.scn"

4. The stage 1 simulation log files are generated from the scripts below:

```
proc file_output { } {
   global sink_id node_num count target_node_num run_index

   #open a file for writting
   set filename "sensorInfo$run_index-$count.log"
   set out [open $filename w]

   puts $out [rt . getTime]

   for {set i [expr $sink_id + 1]} {$i < [expr $node_num -
$target_node_num]} {incr i} {
#     ! n$i/app getLocation
      #get its x and y location
#     set Xloc [! n$i/app getX]
#     set Yloc [! n$i/app getY]

      #get its energy
      set status [expr [! n$i/app isSensorDead] - 0]
      #added: energy
      set eny [! n$i/wphy getRemEnergy]

      #write it to the file
```

```
     puts $out "$eny $status"
   }
   incr count

   close $out
}
```

5. Finally the simulation is started and stopped with the scripts below;

```
puts "Simulation begins...\n"
set sim [attach_simulator .]
$sim stop


#*****************start the sink***********************
script {run n0} -at 0.0000001 -on $sim

#*********print out all the node locations*************
script "sensorLocPrintOut" -at 0.00000015 -on $sim

#**************start the sensors***********************
for {set i [expr $sink_id + 1]} {$i < $node_num} {incr i} {
                     script puts "run n$i" -at 0.000001 -on $sim
}

#*******Check if Sensor Status*************************
script "wsnLoop" -at 1.0 -period 2.0 -on $sim


#***********For Matlab plotting**********************
script "file_output" -at 100.0 -period 100.0 -on $sim

#***********One-time Capture of where energy went******
#script "energy_dist" -at 5999.0 -period 200.0 -on $sim

script exit -at $max_time -on $sim

$sim resumeTo $max_time
```

C.2 Stage 2 TCL script

1. The stage 2 simulation files are run from the Python command line similar to the stage 1 simulation files. The python scripts first initialize the energies of the stage 1 log files into a resume file to be used by the nodes in stage 2 TCL file. The stage 2 TCL file reads the time offset (needed to determine when a node fails) from the resume file using the command below:

```
set filename "resumeState$run_index.log"
set in [open $filename r]
gets $in time_offset
```

2. The remaining energy of the nodes is read from the resume file with the command:

```
! n$i/wphy setRemEnergy $eny
```

3. The stage 2 simulation log files are generated from the scripts below:

```
proc file_output { } {
   global   sink_id   node_num   count   target_node_num   run_index
time_offset

   #open a file for writting
   set filename "sensorInfo$run_index-$count.log"
   set out [open $filename w]

   puts $out [expr [rt . getTime]+$time_offset]

   for  {set  i  [expr  $sink_id  +  1]}  {$i  <  [expr  $node_num  -
$target_node_num]} {incr i} {
#     ! n$i/app getLocation
     #get its x and y location
#     set Xloc [! n$i/app getX]
#     set Yloc [! n$i/app getY]

     #get its energy
     set status [expr [! n$i/app isSensorDead] - 0]

     #added: energy
     set eny [! n$i/wphy getRemEnergy]

     #write it to the file
     puts $out "$eny $status"
   }
   incr count
   close $out
}
```