



***DISTRIBUTED ENERGY CONTROL IN WIRELESS SENSOR NETWORKS  
FOR ENVIRONMENTAL MONITORING BASED ON ADAPTIVE SAMPLING***

***BY***

***ABIUD WAKHANU MULONGO***

**SUMMITTED AT THE**

**SCHOOL OF COMPUTING AND INFORMATICS**

**COLLEGE OF BIOLOGICAL AND PHYSICAL SCIENCES**

**UNIVERSITY OF NAIROBI**

**in partial fulfillment of the requirements for the degree of**

**Master of Science in Computer Science**

**August. 2012**

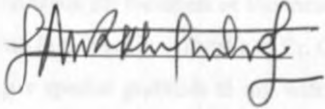
**DECLARATION**

This project, as presented in this report, is my original work and has not been presented for any other award in any other University.

Name: Abiud Wakhanu Mulongo

Reg. No: P58/62085/2010

Sign:



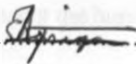
Date

12/10/2012

This project has been submitted as partial fulfillment of the requirements for the degree of Master of Science in Computer Science of the University of Nairobi with my approval as the University supervisor.

Name: Mr. Ayienga Eric

SIGNATURE.....



DATE.....

12/10/2012.....

## ACKNOWLEDGEMENT

My sincere thanks to my thesis advisor and supervisor, Mr. Ayienga Eric. He has been very kind by being there for me even at odd hours when I needed his counsel. His feedback and expert opinions throughout this thesis were invaluable. I also thank all members of the thesis panel whose criticisms, comments and corrections gave me more strength and focus in this work – Thanks to Dr. Opiyo Elisha, Mr. Tony Omwansa and Professor. Elijah Omwenga. I would like to give special gratitude to my wife Jael Ong'asia, first for her encouragement to pursue the master's course and second for providing me with all sort of support throughout the course and the thesis in all seasons of joy and sorrow. I owe my gratitude to my dad Enos Wakhanu and my mum Catherine Namisoo who ever since my childhood have always backed me for every step I took in my life. I would like to thank my colleagues and friends I met during my master's studies at the School of Computing and Informatics of the University of Nairobi with whom we shared good and tough moments and for extending to me help in one way or another. The journey has indeed been very long, tough, painful and with innumerable uncertainties along the way. My gratitude to our God the almighty for the great favour and grace that has kept me in good health, protected me from harm and injury, and enabled me to surmount the hurdles to reach this far.

## ABSTRACT

A common application of wireless sensor networks (WSNs) is in environmental monitoring where a high spatial density of WSNs is deployed in a region to monitor a physical process such as the variation of surface temperature. In some specific applications continuous monitoring is required in order to achieve a high temporal resolution of data. A conflicting requirement in the design of WSNs for such applications is the need to conserve as much energy as possible for prolonged operation (since WSNs are powered by energy constrained batteries) versus the need to address certain application specific quality of service parameters such as data accuracy and timeliness of data delivery. One approach to this dilemma is to use the concept of adaptive sampling. When designing an adaptive sampling algorithm the biggest challenge is to determine the number of samples collected at any given time in order to tradeoff energy consumption and quality of information. The goal of this research was to balance energy consumption across nodes at system level and to balance energy consumption against quality of information at the nodal level. To this end, we proposed DASS (Distributed Adaptive Sensing and Sampling) algorithm. The adaptive sensing protocol was implemented in a simulation environment (OPNET/OpenZB on top of the IEEE 802.15.4/Zigbee model). It was compared to the fixed sampling algorithm in terms of mean energy consumed (joules), Message reduction percentage (MRP) and mean time between samples as the performance metrics. For Quality of information (QoI) we used packet success probability and end to end delay to determine the completeness of information and timeliness of information respectively. Our results demonstrate that DASS can save upto 57% of energy at a PSP of 91% with several factors of improvement in delay compared to the fixed sampling strategy.

**TABLE OF CONTENTS**

**DECLARATION ..... 1**

**ACKNOWLEDGEMENT ..... 2**

**ABSTRACT ..... 3**

**LIST OF FIGURES ..... 7**

**LIST OF TABLES ..... 8**

**LIST OF TERMS..... 9**

**CHAPTER 1: INTRODUCTION ..... 11**

    1.1 Background ..... 11

    1.2 The Problem Statement ..... 12

    1.2 Research Questions ..... 12

    1.4 The Objectives..... 12

    1.5 The Contribution ..... 13

    1.6 Assumptions and Delimitations ..... 13

    1.7 Significance of the Study..... 13

**CHAPTER 2: LITERATURE REVIEW ..... 15**

    2.1 Wireless Sensor Network Applications in Environmental Monitoring ..... 15

    2.2 Architecture of a Sensor Node ..... 15

        2.2.1 Node Level Architecture ..... 16

    2.3 Enabling Wireless Standards ..... 18

    2.4 Key Performance Issues in Wireless Sensor Networks ..... 20

    2.5 Technical Challenges in Wireless Sensor Networks..... 20

        2.5.1 Power Sources/Supply ..... 20

        2.5.2 Design of Energy Efficient Protocols ..... 20

        2.5.4 Quality of Service and Data Quality Guarantees ..... 21

        2.5.6 Security ..... 21

        2.5.7 Deployment and Implementation Issues ..... 22

    2.6 Related Work ..... 22

        2.6.1 Introduction ..... 22

..... 23

..... 23

        2.6.2 A Survey of Adaptive Sampling and Sensing Protocols..... 24

2.7 The Proposed Distributed Energy Control Model.....	27
2.7.1 Introduction .....	27
2.7.2 The System Model .....	28
2.7.3 Node Level Model .....	30
2.7.4 The Adaptive Sensing and Sampling Model.....	33
2.7.4.1 Estimating the Sampling Rate.....	34
2.7.4.2 Signal Frequency Change Detection Model .....	37
<b>CHAPTER 3: METHODOLOGY .....</b>	<b>40</b>
3.1 Introduction.....	40
3.2 Software Development Methodology .....	40
3.3 Performance Evaluation Methodology .....	43
<b>CHAPTER 4: DESIGN AND IMPLEMENTATION .....</b>	<b>48</b>
4.1 Introduction.....	48
4.2 Overview of OPNET.....	48
4.2.1 Modeling Structure and Process .....	48
4.2.2 How to specify Process, Node and Network Models in OPNET .....	50
4.2.3 Data Collection in OPNET .....	50
4.2.4 Data Analysis and Visualization in OPNET .....	50
4.2.5 Modeling Network and Application Specific Traffic .....	50
4.2.6 Limitations of OPNET in relation to our research.....	51
4.3 THE IEEE 802.15.4 OPNET SIMULATION MODEL: OPEN ZB.....	53
4.3.1 Open ZB Physical Layer Model.....	53
4.3.2 Open ZB MAC layer Model .....	53
4.3.3 Open ZB Application layer Model.....	53
4.3.4 Open ZB Battery Model.....	53
4.5 Overview of the FFTW3 C++ Library for Computing Discrete Fourier Transforms.....	54
4.5.1 Fundamental FFTW3 Types - Plans .....	54
4.5.2 Computing One Dimensional DFT Transforms in FFTW3 .....	54
4.5.3 Computing One Dimensional DFTs on Real Data - Real to Complex DFT Transforms .....	55
4.6 Overall System Architecture and Design.....	56
6.7 System Configuration .....	59
6.7.1 Configuration of the Network Model in OpenZB.....	59

4.7.2 Configuration of the Node Model in Open ZB .....	59
6.7.3 Configuration of the Battery Model in Open ZB .....	61
4.7.5 Configuration of Sensory Model in Open ZB.....	64
4.3.8 The Analyzer Node Model Implementation .....	67
4.3.9 The MAC CSMA/CA Programming Model Implementation.....	67
<b>CHAPTER 5: RESULTS AND DISCUSSIONS.....</b>	<b>69</b>
5.1 Introduction.....	69
5.2 Metrics .....	69
5.2.1 Energy Consumption and Network Lifetime .....	69
5.3 Experimental Parameters .....	72
5.3.1 Sensory Traffic Parameters.....	72
5.3.2 Battery and Power Configurations.....	72
5.3.3 Network Configurations.....	73
5.3.4 Simulation Run Configurations .....	73
5.4 Simulation Scenarios. ....	73
5.4.1 A Distributed Adaptive sampling .....	73
5.4.2 Fixed sampling.....	74
5.3 Results and Discussions .....	74
5.3.1 Effect of Adaptive Sampling and Fixed Sampling on Energy Consumption .....	74
5.3.2: Effect of Adaptive Sampling/ Fixed Sampling on the Rate of Sensor Traffic Generation.....	77
5.3.3: Effect of Adaptive Sampling/ Fixed Sampling on the Application Packet Interarrival Time ..	79
5.3.4 Effect of Adaptive Sampling/ Fixed Sampling on End to End Delay .....	82
5.3.5: Effect of Adaptive Sampling/ Fixed Sampling on the Packet Success Probability .....	84
5.3.6: Summary of Energy Consumption and Network Life Time Related Statistics .....	86
5.3.7. Summary of Quality of Information (QoI) Related Statistics .....	86
<b>CHAPTER 6: CONCLUSIONS.....</b>	<b>87</b>
6.1 Introduction .....	87
6.2 Limitations and Challenges .....	87
6.3 Future Work .....	88
<b>REFERENCES.....</b>	<b>89</b>

## LIST OF FIGURES

FIGURE 2-1-A: BASIC ARCHITECTURE OF A SENSOR NODE .....	16
FIGURE 1-1B: DETAILED ARCHITECTURE OF SELECTED SENSOR MOTES – THE WASPMOTE .....	17
FIGURE 4-1: CONCEPTUAL MODEL - SYSTEM CONTEXT DIAGRAM .....	29
FIGURE 4-2: CONCEPTUAL MODEL – DETAILED NODE LEVEL MODEL.....	32
FIGURE 4-3: THE ADAPTIVE SENSING AND SAMPLING MODEL – CONTROL FLOW DIAGRAM.....	39
FIGURE 5-2: AN EXAMPLE MANET NODE MODEL IN OPNET NODE EDITOR .....	52
FIGURE 5-3: AN EXAMPLE MANET IP LAYER PROCESS MODEL IN OPNET PROCESS EDITOR.....	52
FIGURE 5-4: OPNET IEEE 802.15.4/ZIGBEE SIMULATION MODEL .....	53
FIGURE 6-1 : GRAPHICAL USER INTERFACE FOR CONFIGURING SENSOR SOURCE TRAFFIC IN OPENZB.....	66
FIGURE 6-2: SENSOR BATTERY FINITE STATE MACHINE IN IEEE 802.15.4/ZIGBEE OPNET MODEL....	66
FIGURE 6-3: SINK MODULE FINITE STATE MACHINE IN IEEE 802.15.4/ZIGBEE MODEL IN OPNET.....	67
FIGURE 6-4: GRAPHICAL USER INTERFACE FOR CONFIGURATION BATTERY AND POWER MODEL IN IEEE 802.15.4/ZIGBEE OPNET MODEL.....	64
FIGURE 6-5 : SENSOR NODE MODEL IN IEEE 802.15.4/ZIGBEE OPNET MODEL .....	61
FIGURE 6-6: GRAPHICAL USER INTERFACE FOR CONFIGURE IEEE 802.15.4 SETTINGS IN OPNET/OPENZB.....	61
FIGURE 6-7: GRAPHICAL USER INTERFACE FOR CONFIGURING IEEE 802.15.4/ZIGBEE CSMA/CA MAC LAYER PARAMETERS IN OPNET.....	68
FIGURE 7-1: RESULTS – GRAPH OF RATE OF ENERGY CONSUMPTION PER NODE – ADAPTIVE SAMPLING VS FIXED SAMPLING.....	76
FIGURE 7-2 : RESULTS –GRAPH OF CUMULATIVE ENERGY CONSUMED (JOULES) VERSUS TIME – ADAPTIVE VS FIXED SAMPLING (NODE S5).....	76
FIGURE 7-3: RESULTS- GRAPH OF CUMULATIVE ENERGY CONSUMPTION VERSUS TIME - NODE S8 .....	77
FIGURE 7-4 : RESULTS – MEAN SENSOR TRAFFIC GENERATED (PACKET/SECOND) PER NODE.....	79
FIGURE 7-5: RESULTS- MEAN PACKET INTERARRIVAL PER NODE – ADAPTIVE SAMPLING VERSUS FIXED SAMPLING .....	81
FIGURE 7-6: RESULTS- VARIATION OF NETWORK PACKET INTERARRIVAL TIMES WITH TIME .....	82
FIGURE 7-7:RESULTS – MEAN END TO END DELAY (SECONDS) PER SENSOR NODE .....	84
FIGURE 7-8 : VARIATION OF NETWORK PACKET SUCCESS PROBABILITY (%) WITH TIME (SECONDS).....	85



## LIST OF TABLES

TABLE 2-1: UC BERKLEY FAMILY OF SENSOR NOTES – TECHNICAL DATA SHEET .....	18
TABLE 4-1: COMPARISON OF COMMONLY USED WIRELESS SENSOR NETWORK SIMULATORS.....	45
TABLE 6-1: SPECIFYING SENSORY TRAFFIC PARAMETERS IN OPENZB.....	64
TABLE 6-2 : BATTERY AND POWER MODEL SPECIFICATIONS IN IEEE 802.15.4 OPNET MODEL.....	62
TABLE 7-1: SIMULATION PARAMETERS – SENSORY TRAFFIC SETTINGS.....	72
TABLE 7-2: SIMULATION PARAMETERS – NODE BATTERY AND POWER SETTINGS .....	72
TABLE 7-3: SIMULATION PARAMETERS – NETWORK SETTINGS .....	73
TABLE 7-4: SIMULATION RUN SETTINGS .....	73
TABLE 7.5A: RESULTS - RATE OF ENERGY CONSUMPTION (JOULES PER MINUTE) FOR FIXED SAMPLING.....	75
TABLE 7-5B: RESULTS - RATE OF ENERGY CONSUMPTION (JOULES PER MINUTE) - ADAPTIVE SAMPLING.....	75
TABLE 7-6 A : RESULTS- SENSOR TRAFFIC GENERATED (PACKETS/SECOND) - FIXED SAMPLING....	78
TABLE 7-6B: RESULTS – SENSOR TRAFFIC GENERATED (PACKETS/SECOND) – ADAPTIVE SAMPLING .....	78
TABLE 7-7A: RESULTS - MEAN PACKET INTERARRIVAL TIME - FIXED SAMPLING .....	80
TABLE 7-7B: RESULTS – MEAN PACKET INTERARRIVAL TIME – ADAPTIVE SAMPLING.....	80
TABLE 7-8A : RESULTS – MEAN END TO END DELAY (SECONDS) PER NODE – FIXED SAMPLING ....	83
TABLE 7-8B - : RESULTS – MEAN END TO END DELAY (SECONDS) PER NODE – ADAPTIVE SAMPLING.....	83
NODE ID.....	84
TABLE 7-9A: MEAN NETWORK PACKET SUCCESS PROBABILITY (%) - FIXED SAMPLING .....	85
TABLE 7-9B: MEAN NETWORK PACKET SUCCESS PROBABILITY (%) - ADAPTIVE SAMPLING .....	85
TIME (SECONDS).....	85
TABLE 7-10: RESULTS – SUMMARY OF ENERGY CONSUMPTION AND NETWORK LIFETIME RELATED STATISTICS .....	86
TABLE 7-11: RESULTS – SUMMARY OF QUALITY OF INFORMATION RELATED STATISTICS .....	86

## LIST OF TERMS

BS	Base Station: A server that collects and aggregates the sensor data from the sensor nodes
CDMA	Code Division Multiple Access: A method of concurrent access to cellular network resources
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance: An algorithm at the MAC layer for controlling concurrent access to the wireless media
CH	Cluster Head: A high energy node within a Wireless Sensor Node used to coordinate sensing and data transmission.
CUSUM	Cumulative Sums: A statistical change control technique based on the concept of residuals
DD	Direct Diffusion: A protocol used at the WSN routing layer
DFT	Discrete Fourier Transform: A mathematical transform used in the analysis of signals
EED	End to End Delay: Time taken to generate sensor data at the source until when the data is available at the BS
FFT	Fast Fourier Transform: A type of algorithm that computes the DFT in $O(n \log n)$
GUI	Graphical User Interface
IDE	Integrated Development Environment: A programming environment with a GUI and sophisticated debugging capability
LEACH	Least Energy Aware Clustering Hierarchy protocol: An adaptive routing protocol base Hierarchical Clustering
I/O	Input /Output
KP	Kernel Procedure
MAC	Medium Access Layer: The second layer in the OSI model
PAN	Personal Area Network

<b>PEGASIS</b> in WSNs	<b>Power Efficient Gathering in Sensor Information Systems: An adaptive routing protocol</b>
<b>PHY</b>	<b>Physical Layer: The first layer of the OSI communication model</b>
<b>PSP</b>	<b>Packet Success Probability: A measure of the number of packets successfully delivered to the BS</b>
<b>QPSK</b>	<b>Quadrature Phase Shift Keying: One of the modulation techniques</b>
<b>QoI</b>	<b>Quality of Information: A measure of the completeness of information and timeliness of information delivery</b>
<b>QoS</b>	<b>Quality of Service: A network performance measure that is based on varying factors.</b>
<b>SAR</b>	<b>Sequential Assignment Routing: A WSN routing protocol</b>
<b>TCP</b>	<b>Transport Control Protocol /Transmission Control Protocol: The protocol used to guarantee reliable transport of data</b>
<b>TDMA</b>	<b>Time Division Multiple Access: A technique used to achieve simultaneous access to a single channel</b>
<b>UDP</b>	<b>User Data Protocol: A best effort transport protocol that does not guarantee reliable transmission</b>
<b>WPAN</b>	<b>Wireless Personal Area Network:</b>
<b>WSN</b> media	<b>Wireless Sensor Network: A collection of several tiny sensor nodes interconnected by a wireless media</b>
<b>WSAN</b>	<b>Wireless Sensor and Actuator Networks: a WSN with actuation capability</b>

## CHAPTER 1: INTRODUCTION

### 1.1 Background

A wireless sensor network (WSN) is a network of tiny sensors interconnected via a wireless media. The emergence of WSNs has made it possible to monitor the natural environment in unprecedented ways (Martinez, Hart, and Ong, 2004) due to the unique advantages WSNs offer over traditional wired sensor systems- this includes their ability to be deployed in any type of environment such as dangerous battle fields, deep oceans or hostile environment due to their ease of deployment, scalability, flexibility and the ability to self-organize (Ilayas and Mahgoub, 2004). In high resolution data and real time applications, continuous sampling is required to ensure that as many samples as possible are gathered and disseminated to achieve *high data quality* and that critical events in the environment that require instant responses are not missed out implying *timeliness of delivery*.

The need for high data quality has the implication that sensor nodes are supposed to be *ON* most of the time and data transmission is to happen more frequently so more energy is consumed by the sensing unit and the radio. This objective contradicts the key goal in the design of wireless sensor networks- which is to conserve as much as energy as possible since the devices within a WSN are battery driven and energy therefore energy constrained. Further due to requirement of high data quality, too frequent transmission of data puts more congestion on the band limited wireless channel introducing additional delays. This in turn contravenes the objective of *timeliness in data delivery*. Generally, data quality and timeliness form part of an important component in WSNs called *Quality of Information (QoI)*. So how to control energy consumption with QoI guarantees in WSNs for continuous and real time environmental monitoring is still an open research issue. A basic approach to this problem is using the technique of adaptive sampling at the sensing layer. The technique involves varying the frequency of sensing based on various algorithms whose basic principle is to cycle the sensor and radio modules through sleep and active states. This has the potential of saving both the sensing energy and transmission energy. However, in time varying stochastic and dynamic environments, the difficulty in using adaptive sampling is in determining the optimal sampling rate that ensures important events about the physical phenomenon being monitored and at the same time avoid unnecessary wastage of energy (Alippi et al. 2007), (Alippi et al. 2009).

Other techniques exploit the idea of energy aware routing that utilize multihop routing and hierarchical clustering. The idea is motivated by the fact that transmission power accounts for the largest percentage of energy consumption in WSNs and therefore the best approach to minimizing energy consumption is reducing the amount of communication as much as possible. An excellent example of a protocol utilizing this concept is LEACH (Heinzelman et al, 2000; 2002). The development of wireless standards with cross layer designs for low power shortage range communications is also another excellent way of combating energy consumption in wireless sensor networks. The IEEE 802.15.4/Zigbee standard is an attempt towards this direction of cross layer design. Therefore to effectively minimize energy consumption, maximize the network life and to maximize QoI in WSNs for

continuous and real time environmental monitoring, coupled sensing and communication strategies are required at all layers of the sensing and communication protocol stack are required.

The central question we seek to answer in this research is: how can we reduce energy consumption and maximize the lifetime of a real time wireless sensor network designed for continuous monitoring with compromising the QoI of information.

## 1.2 The Problem Statement

The purpose of this research is to develop a distributed coupled energy and QoI control model based on adaptive sensing and sampling at the sensing layer to attempt to further improve the network lifetime of WSNs for continuous environmental monitoring and real time dissemination of data, by reducing as much as possible, the number of messages transported across the network with QoI guarantees.

## 1.2 Research Questions

- a) What is the impact of distributed adaptive sampling on the energy consumption and network life time a WSN?
- b) Can distributed adaptive sampling achieve reduced end to end network delay in a WSN?
- c) Can distributed adaptive sampling reduce the total number of messages disseminated in a WSN within a given time period compared to fixed sampling?
- d) Can distributed adaptive sampling achieve significant improvement in the network lifetime with preserved QoI compared to fixed sampling techniques?

## 1.4 The Objectives

- 1) Formulate a mathematical model for adaptive sampling that concurrently saves energy and preserves quality of information in wireless sensor networks based on ASA.
- 2) Demonstrate how the adaptive sampling model minimizes energy consumption in WSNs with QoI guarantees.
- 3) Implement the model using C++ and the OPNET Modeler.
- 4) Test the model using C++ and OPNET Modeler
- 5) Analyze energy consumption, network lifetime and Quality of Information of the model.
- 6) Compare performance of the adaptive sampling model against fixed sampling.

## 1.5 The Contribution

The main contribution of this project to wireless sensor networks research is: a distributed adaptive sampling technique that simultaneously saves energy in WSNs and achieves improved quality of service.

## 1.6 Assumptions and Delimitations

- a) **At the sensing Layer** : the physical process/phenomena under monitoring is random and highly dynamic
- b) The session layer , transport layer and network layers are abstracted such that there is direct communication between the application layer the MAC layer Transmission packets are small enough such that no fragmentation process is required.
- c) MAC protocols are energy efficient as well of QoI aware. This allows our study to concentrate on the analysis of energy consumption and QoI at the sensor application layer and Physical layer only.
- d) At the physical layer sensor nodes are powered by batteries that are energy constrained.
- e) The measurement noises are negligible compared to the actual sensor readings so that the sensor measurements are tightly coupled to the physical laws governing the physical process under monitoring.

## 1.7 Significance of the Study

A distributed adaptive sampling algorithm based Discrete Fourier Transforms, Sampling Theorem and Statistical Change Techniques for energy conservation in WSNs that fulfills the promise of:

- .Gathering sufficient data about the environmental phenomena in order to discern trends and patterns with improved accuracy and improved timeliness of data delivery.
- Network longevity of the WSN against scarce energy due to sensor nodes that are driven by power constrained batteries

The rest of the report is organized as follows:

**Chapter 1:** Introduces wireless sensor networks and the general issues involved in the design of the WSNs. The research problem is stated as well as the objectives, significance of this study and the contribution.

**Chapter 2:** WSNs are discussed in some much more details. A number of real world sensor applications in environmental monitoring are cited. We also discuss the architecture of a sensor node citing practical sensor nodes. Performance metrics and technical challenges are also touched within this chapter.

**Chapter 3:** We survey the important literature material related to adaptive energy control and briefly introduce our proposed model.

**Chapter 4:** Our proposed energy control model is elaborated. The model covers the system level model, the node level model and the sampling and sensing model. The necessary mathematical background is also introduced and appropriate derivations done.

**Chapter 5:** The methodology, which is the collection of procedures and tools used to implement the solution, is explored. Two levels of methodologies are explored. The software development methodology that concerned with the procedures and techniques of developing the software (in this case the protocol) and the performance evaluation methodology; basically being the simulation of the protocol to analyze performance.

**Chapter 6:** The details of design and implementation of the model are described. In particular, the specific simulation and programming tools used in this research are discussed. Procedures for configuring certain parameters/models within the simulation environments are explored. The language and libraries used to implement are also discussed.

**Chapter 7:** The focus of this chapter is on measuring the performance of DASS and comparing its performance to fixed sampling. The performance metrics introduced in chapter 5 are exemplified and where possible relevant formulae are given to define them. The experimental parameters are outlined. The experiments that were carried out are described. The results are analyzed in depth to provide us with useful lessons.

Other sections of the report include the limitations and challenges, the conclusion and future work that appear in that order.

## CHAPTER 2: LITERATURE REVIEW

### 2.1 Wireless Sensor Network Applications in Environmental Monitoring

WSNs are increasingly being used in environmental monitoring in varied ways: habitat monitoring as in the ZebraNet project (ZebraNet, 2004), disaster monitoring like those in the FloodNet project (Zhou and De Roure, 2007) and GlasWeb (Martinez, 2005) and weather monitoring. WSNs for environmental monitoring are typically used to monitor the environment and collect data such as temperature, chemical concentration, light intensity or atmospheric pressure about a specific physical process in specific locations within the environment (Ilayas and Mahgoub, 2004). Depending on the specific purpose of the application, environmental monitoring applications may simply be required to collect and report the readings about a variable or a set of variables for future trend analysis. Examples of applications in this category are weather monitoring applications in which we may be interested only in computing aggregates on temperature of a place in a given day. In such a case we do not need samples in the order of thousands in order to calculate aggregates like mean temperature. Other environmental applications require sampling at a very high frequency and resolution due to frequent and stochastic variations in the quantity being monitored. A good example is in the monitoring of fresh water quality where we are interested in determining contaminant transport used to assess the exposure levels. Due to the complex spatial temporal variability in hydrological, chemical and ecological parameters, high spatial and temporal sampling rates are required to achieve high accuracy of information (Ilayas and Mahgoub, 2004). This is the case of sensor apps for monitoring of glacial processes (Alippi et al. 2007). For instance in the GlassWeb project, 60000 samples per every 15 seconds were required to accurately determine the optimal sampling frequency. For this kind of applications, there is need to collect data at very close intervals in order to extract logical conclusions about the phenomena being monitored (Hill, 2003) and/ or to react in a timely manner to the events in the environment.

### 2.2 Architecture of a Sensor Node

A wireless sensor network consists of tiny sensor nodes. A single sensor node consists of the following basic components power supply, sensing module, processor, and radio module. The architecture of a sensor node is presented in figure 2-1 below.



### 2.2.1 Node Level Architecture

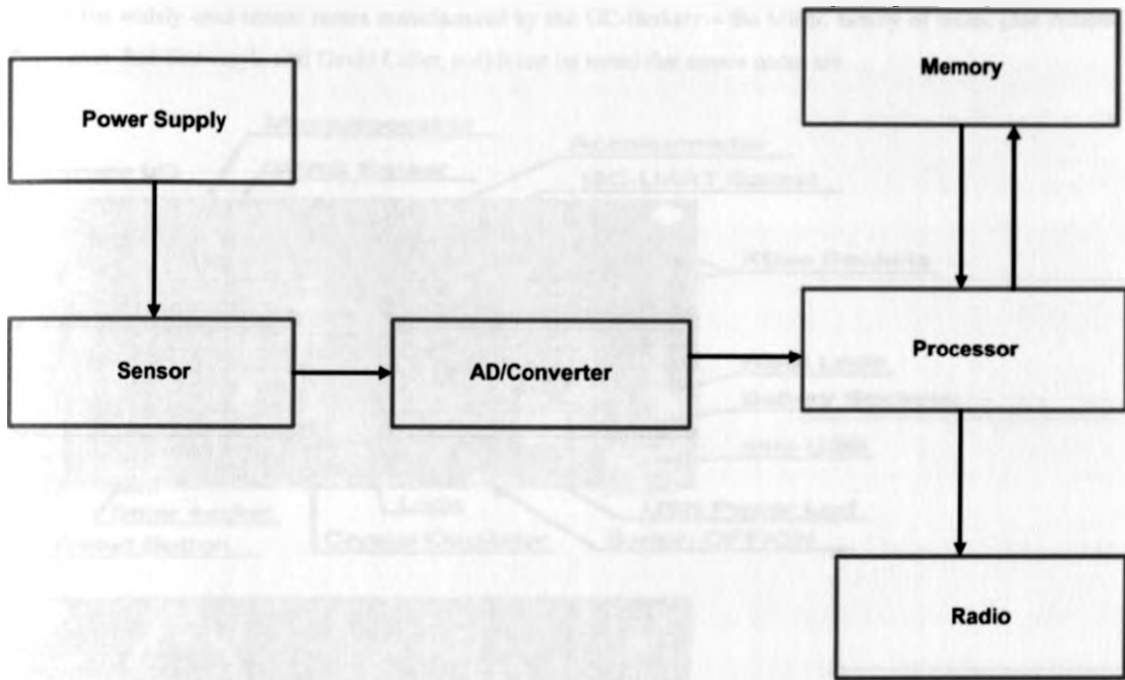


Figure 2-1-a: Basic Architecture of a Sensor Node

- I. **Power Supply** : Inform of a battery attached to the sensor board to drive all the electronic components in the sensor board. Commonly used batteries are the Lithium and Alkaline batteries . see Figure 2-1c for example battery specifications of the MicaZ mote.
- II. **Sensor/Actuator**: Sensor collects signals from the environment and the actuator responds to environment based on some preprogrammed conditions. Depending on the nature of application , the sensor could be a light sensor, temperature sensor , an accelerometer sensor , a pressure sensor, humidity sensor or a combination of these
- III. **A/D Converter** : Connects the sensor to the external environment. Converts analogue signals collected from the environment to digital signals
- IV. **Processor** : a microcontroller that provides intelligence to the sensor . The processor/microcontroller is usually an Intel ARM processor or the ATMEgal processor.
- V. **Radio**: An RF circuitry for data transmission and reception. Specific sensor motes would come equipped with more sophisticated radio modules such as the Zigbee modules ( see figure 2-1b).
- VI. **Memory**: Used to store instructions as well data

Figure 2-1b shows the internal structure of one of the practical sensor motes – the WASPMOTE, a propriety sensor mote manufactured by the Libelium Inc. We can see that this mote for instance comes with an axiullary battery,it's equipped with both GPS and GPRS modules, has a ( the Xbee Sockets) for plugging in the Zigbee radio

devices, an SD card for external data storage and an accelerometer sensor. Figure 2-1c shows the specification of one of the widely used sensor motes manufactured by the UC-Berkely – the MicaZ family of motes (Joe Polastre, Phil Levis, Rob Szewczyk, and David Culler, n.d) It can be noted that sensor nodes are

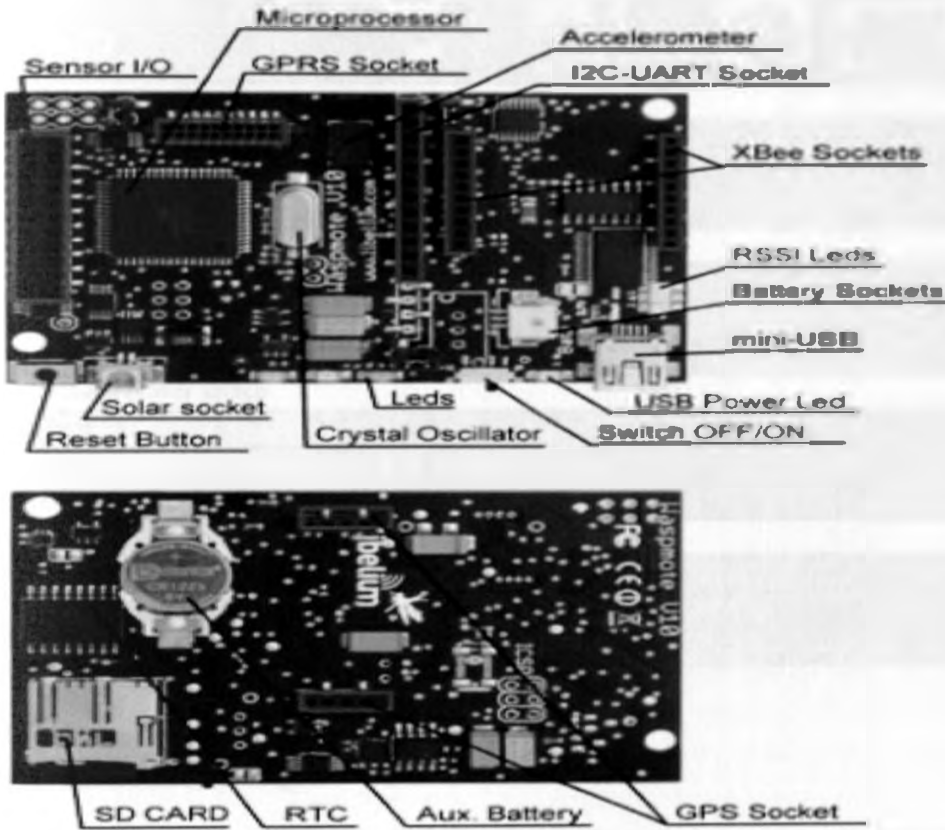







Figure 1-1b: Detailed Architecture of Selected Sensor Motes – the WASPMOTE

Table 2-1: UC Berkley Family of Sensor Motes – Technical Data Sheet

Mote Type	<i>WeC</i>	<i>René</i>	<i>René 2</i>	<i>Dot</i>	<i>Mica</i>	<i>MicaDot</i>
						
<b>Microcontroller</b>						
Type	AT90LS8535		ATmega163		ATmega128	
Program memory (KB)	8		16		128	
RAM (KB)	0.5		1		4	
<b>Nonvolatile storage</b>						
Chip	24LC256			AT45DB041B		
Connection type	I <sup>2</sup> C			SPI		
Size (KB)	32			512		
<b>Default power source</b>						
Type	Lithium	Alkaline	Alkaline	Lithium	Alkaline	Lithium
Size	CR2450	2 x AA	2 x AA	CR2032	2 x AA	3B45
Capacity (mAh)	575	2850	2850	225	2850	1000
<b>Communication</b>						
Radio	TR1000					CC1000
Radio speed (kbps)	10	10	10	10	40	38.4
Modulation type	OOK				ASK	FSK

## 2.3 Enabling Wireless Standards

### 2.3.1 The IEEE 802.15.4/Zigbee

The IEEE 802.15.4 (IEEE, 2003) protocol has recently been adopted as a communication standard for low data rate, low power Consumption and low cost Wireless Sensor Networks (WSNs). This protocol is quite flexible for a wide range of applications by adequately tuning its parameters and it also provides real time guarantees by using the Guaranteed Time Slot (GTS) mechanism. This feature is quite attractive for time-sensitive WSN applications. The IEEE 802.15.4 standard specifies the physical layer and MAC sub-layer for Low-Rate Wireless Personal Area Networks (LR-WPANs). The ZigBee (ZigBee Alliance, n.d) standard is close associated with the IEEE 802.15.4 protocol and specifies the network (including security services) and application (including objects and profiles) layers.

### **2.3.1.1 IEEE 802.15.4 Physical Layer**

The physical layer is responsible for data transmission and reception using a certain radio channel according to a specific modulation and spreading techniques. The IEEE 802.15.4-2003 [1] standard offers three unlicensed frequency bands: 2.4 GHz (worldwide), 915 MHz (e.g. North America) and 866 MHz (e.g. Europe). There is a single channel between 868 and 868.6 MHz, 10 channels between 902 and 928 MHz and 16 channels between 2.4 and 2.483.5 GHz. The data rate is 250 kbps at 2.4 GHz, 40 kbps at 915 MHz and 20 kbps at 868 MHz. In addition to these three frequency band patterns, two high data rate patterns have been added to the 868/915 MHz bands in the last revision of the standard IEEE 802.15.4REVb-2006. The higher data rates are achieved by using of the different modulation formats. This revision of the standard is backward-compatible to the IEEE 802.15.4-2003, meaning that devices conforming to IEEE

802.15.4REVb-2006 are capable of joining and functioning in a PAN composed of devices conforming to IEEE 802.15.4-2003. All of these frequency bands are based on the Direct Sequence Spread Spectrum (DSSS) spreading technique.

### **2.3.1.2 IEEE 802.15.4 Medium Access Control Layer**

The MAC protocol supports two operational modes that can be selected by a central controller of the Person Area Network (PAN), called PAN Coordinator: Beacon-enabled mode: In beacon-enabled mode, the beacon frames are periodically generated by the PAN Coordinator to identify its PAN, to synchronize devices that are associated with it, and to describe the super frame structure. Non Beacon-enabled mode: In non-beacon-enabled mode, the device can simply send their data by using unslotted CSMA/CA mechanism. There is no use of a super frame structure in this mode. The advantages of this mode are a scalability and self-organization. However, the non-beacon-enabled mode cannot provide any time guarantees to deliver data frames. When the PAN Coordinator selects the beacon-enabled mode, it forces the use of a super frame structure to manage communication between the devices that are associated to that PAN. The format of the super frame is defined by the PAN Coordinator. The superframe, corresponding to the Beacon Interval (BI), is defined by the time between two consecutive beacons, and includes an active period and, optionally, a following inactive period. The active period, corresponding to the Superframe Duration (SD), is divided into 16 equally sized time slots, during which data transmission is allowed. Each active period can be further divided into a Contention Access Period (CAP) and an optional Contention Free Period (CFP). Slotted CSMA/CA is used within the CAP. The CFP is activated by the request sent from a device to the sufficient resources and, if possible, allocates the requested time slots. This requested group of time slots is called Guaranteed Time Slot (GTS) and is dedicated exclusively to a given device. A CFP support up to 7 GTSs and each GTS may contain multiple time slots. The allocation of the GTS cannot reduce the length of the CAP to less than the value specified by aMinCAP.length constant. PAN Coordinator. Upon receiving this request, the PAN Coordinator checks whether there are

## 2.4 Key Performance Issues in Wireless Sensor Networks

**Energy Efficiency/Network Lifetime:** Energy is a scarce resource in WSNs due to reliance on power constrained and irreplaceable batteries. The network lifetime, which is a function of the remaining energy of sensor nodes, is defined as the time taken for the first node to die off.

**Data Accuracy:** Collecting accurate information is at the core of most sensor networks. Accuracy can improve through joint detection and detection.

**Latency:** Quite a good number of WSNs require a near real time operation such that data must be delivered to the user in real time. Design of energy efficient protocols should also take this into account.

**Scalability:** The ability of the WSN to sustain its performance with increased network size

**Fault Tolerance:** Collaborative sensing, processing and communication is required to provide necessary redundancy to avoid single point of failure.

**Throughput:** With thousands of nodes involved in sensing and delivery of data across the same network, a heavy traffic load is placed on the network inducing a ripple effect on the system life time, packet delay and scalability.

All the above metrics are tightly coupled. For example high accuracy requires huge amounts of data which not only draws more energy but also causes traffic congestion introducing delays and consequently reducing throughput and scalability. Thus a cross layer design approach, as opposed to the classical layer by layer design, is crucial in order to trade off these issues (Ilyas and Mahgoub, 2004).

## 2.5 Technical Challenges in Wireless Sensor Networks

### 2.5.1 Power Sources/Supply

A key challenge in wireless sensor networks is how to design the network such that the overall energy consumed by the electronic circuitry is minimal. Batteries for example are power limited and may not be the most suitable sources of energy for WSNs designed for continuous and long-term sensing. Current research is exploring alternative sources of energy for WSNs including solar energy, use of free space optical transmission techniques.

### 2.5.2 Design of Energy Efficient Protocols

It has been widely acknowledged that communication energy accounts for the largest percentage of energy expenditure in WSNs when compared to the energy used in sensing and computation. Therefore a natural way of keeping energy consumption at minimum is to reduce as much as possible the energy used in transmission of sensor data. In most cases, individual sensor nodes are not usually in close range with the base station. In such a scenario, the nodes will rely on intermediate nodes to propagate data to the base station and this constitutes routing. A lot of research work is going in the area of hierarchical and cluster based routing, a highly acclaimed technique of adaptively controlling energy consumption in WSNs. Two closely related issues in energy conservation at the sensor layer are: how often should nodes be turned "ON/OFF" (duty cycling) and ii) how many samples should a node

collect in unit time (sampling). The issues are usually addressed by adaptive duty cycling and adaptive sampling respectively. Both strategies in adaptive duty cycling, nodes are cycled through alternating sleep and active periods of varying length. The specific algorithms for determining the sleep duration can vary depending on the application. Adaptive sampling refers to a set of algorithms that control the number of samples a sensor obtains from the environment at a given time instant.

#### 2.5.4 Quality of Service and Data Quality Guarantees

Quality of Service (QoS) as used in the world of communication networks refers to the ability of a network to deliver information in a reliable and timely manner. Formally, QoS is generally defined by the triple:

$(R, P, D)$  Where  $R$  is the throughput,  $P$  is the reliability as determined by either the BER or PSP and  $D$  is the delay (Mohamed, 2004).

There is no standard formal definition of the term "Data quality" as used in wireless sensor networks (Hermans, Dziengel and Schiller, 2009). The authors in (Hermans, Dziengel and Schiller, 2009) identify four important aspects: accuracy of data, consistency of data, timeliness and completeness).By adopting this definition, data quality as used in WSNs can be represented by the four tuple  $(A, C, D, P)$  where;

A, is the data accuracy – the numerical error difference between the value of a single sensor sample and the actual value of the quantity being measured.

C – Consistency: The data stream is said to be consistent if it conforms to a user predefined model that is usually application dependent.

D- Timeliness as measured by the delay: The time taken to get a data stream generated at the source(s) available at the sink/base station

P- Completeness.

The essential requirements of low energy consumption and high quality of information are conflicting in nature and usually a tradeoff between the two is sought.

#### 2.5.6 Security

Security issues in wireless sensor networks range from physical damage given that the nodes may not be well protected and deployed in an unattended regions, message integrity, phishing, masquerading and other security attacks common to communication networks. Security design needs to consider mechanisms for self-defense, detection and prevention of these risks.

### 2.5.7 Deployment and Implementation Issues

Deployment of a massive number of sensor nodes in a target environment is practically challenged by the high unit cost per sensor node.

## 2.6 Related Work

### 2.6.1 Introduction

There have been substantial efforts among WSNs researchers in attempting to design energy aware protocols for wireless sensor networks. The research focus has been directed at various functional layers of a WSN: at the sensor layer, computational layer and communication layer. Within each of the functional layers various energy saving strategies have and are being explored. In figure, the general approaches to energy minimization in wireless sensor networks are presented (figure adapted from Ilayas and Imad Mahgoub ed. 2004, pg. 343)).

Much of the research work on the design of energy efficient protocols in WSNs, have concentrated on efficient data aggregation and routing exploiting various strategies such as the multihop communications, hierarchical clustering, selective transmission eccetra. This focus has been based on the general fact that the radio is the most energy consuming component of a sensor node and any attempt to save as much communication energy as possible significantly leads to overall energy savings in the network. Notably hierarchical clustering with multihop communication is a dual power control strategy that has proven to be a very effective technique at the routing layer. In this strategy, sensor nodes are organized into two or more groups called clusters where data from one cluster is aggregated within the cluster using specific algorithms and transmitted to the base station via intermediate clusters or nodes (multihop communication). An excellent routing and data aggregation protocol in WSNs is LEACH (Heinzelman et al, 200.2002). LEACH is a cluster based algorithm that includes distributed cluster information. The

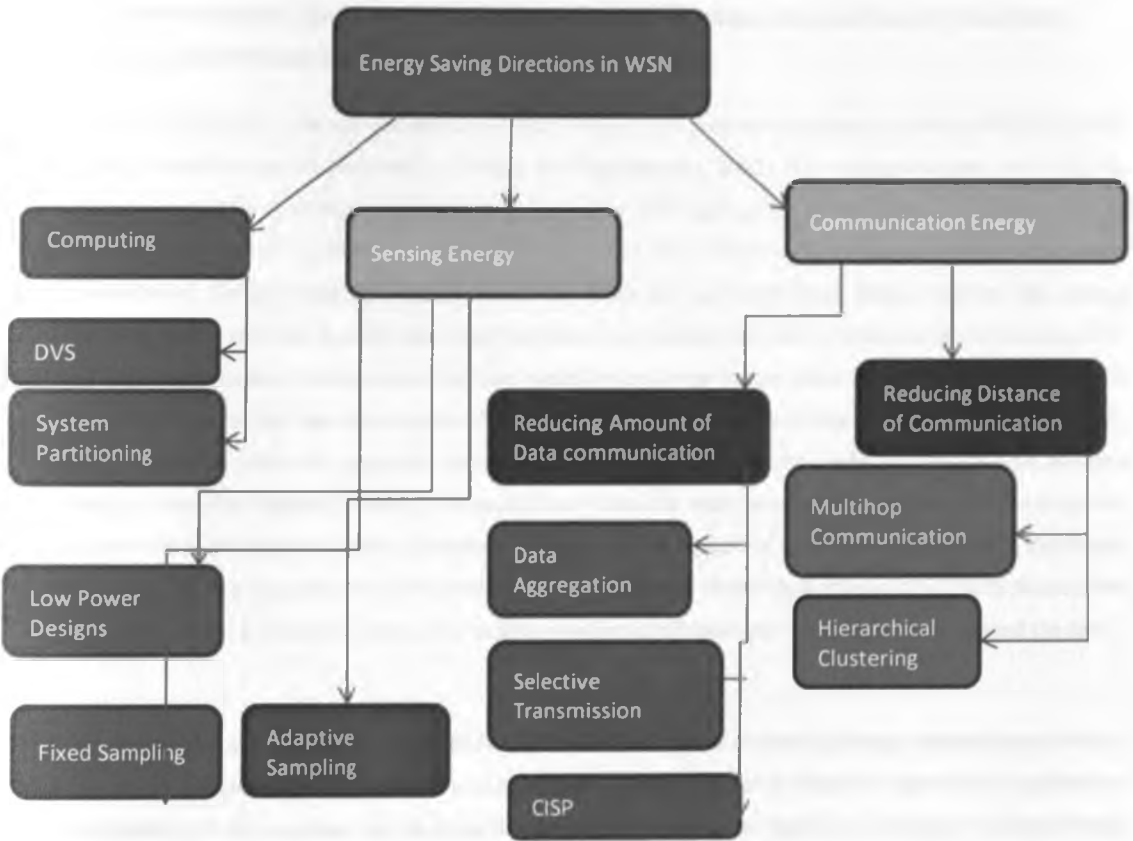


Figure 2: Research Directions In WSN Energy Saving

algorithm randomly partitions a sensor network into self-organizing clusters. In each cluster a cluster head (CH) is elected whose role is to control data transmission and to forward the data to the base station. The remaining nodes simply collect data and relay it to the CH, and the CH (being closer to sink than any of the rest) disseminates the data to the sink. In LEACH, once sensing and transmission commences, it proceeds until certain time limit lapses when the current clusters are collapsed and new ones formed. The idea is to rotate cluster leadership among nodes so as to balance energy consumption fairly across nodes, since those nodes that are cluster heads spent more energy. Despite its advantages, LEACH is faced with a number of issues mainly emanating from the assumptions made in the algorithm design as explained in (K. Chan, F. Fekri, and H. Pishro-Niki), that is: LEACH assumes that each node has sufficient power to reach the base station if needed and that each node has computational power to support different types of MAC protocols. It also assumes that nodes always have data to transmit and that nodes closely located to each other has correlated data. It's not straight forward how the number of predetermined cluster heads



(p) is going to be uniformly distributed across the network. It possible that the elected cluster heads will be concentrated in one side of the network; some nodes in the network far from the new cluster head are likely not to belong to any of the clusters. The protocol also assumes uniform initial energy per sensor node in the network. Variants of the LEACH protocol have been proposed in various literature.

Power-Efficient Gathering in Sensor Information System (PEGASIS) protocol also based on hierarchical clustering and multihop communication is proposed in (Lindsey and Raghavendra, 2002). It is an improvement over LEACH. It allows only one CH to transmit to its local neighbor in the data aggregation phase instead of a node sending sensed data directly to its CH as in the case of LEACH. The main idea of PEGASIS is to form a single chain among sensor nodes using Greedy algorithm. Each node in the chain acts as cluster head which receives data from a neighbor node, fuse it and send it to the next neighbor node closer to the base station. The algorithm make sure that only one node transmits data from source to the base station in any given transmission time frame. This ensures that only relevant data gets to the base station while all redundant data might have been removed down the chain. Energy efficiency is achieved with this approach, since each node acts as CH in the chain and responsible for data transmission to the base station. However, if a node dies within the chain, a new node will be selected based on calculation to avoid transmission failure. Selection of a new node as a result of dead of a node can bring significant overhead especially in a large network. PEGASIS approach avoid s the clustering overhead of LEACH protocol but each sensor node needs to know the status of its neighbor so that it will have the knowledge where to send the data

While the design of energy efficient, as seen above is an effective way of combating energy expenditure in WSNs, designing energy efficient sensing protocols and algorithms can be even more effective especially in application specific scenarios yet the emphasis in the research community has been, as shown, on design of communication energy efficient protocols and less effort has been put towards ways of reducing sensing energy. Particularly C.Allippi in (Alippi, 2007) has proved that designing energy efficient sensing algorithms has the double benefit of reducing both the energy spent due to sensing and energy spent to due data transmission.

In controlling energy due to sensing, two most commonly used techniques is to reduce the frequency of sampling ( how many samples are collected during any time instant) and to reduce idle sensing by cycling the sensor node through alternate ON/OFF states. The former concept is called sampling and latter is called duty cycling. The difficulty in designing a sampling algorithm is to determine the sampling rate while in the duty cycling; the problem is coming up with optimal sleep durations. In an attempt to address this, a number of researchers are making use of the concept of adaptive sampling where the sampling rate is varied based on a number of factors using different approaches. In the following sections we review some recent work that is predominantly related to adaptive sensing and sampling.

#### 2.6.2 A Survey of Adaptive Sampling and Sensing Protocols

In (Willett et al, 2004), an energy efficient as well as information quality aware algorithm called backcasting based on spatial correlation of sensor readings is proposed. Backcasting selects an initial subset of the sensor nodes to

report their sensor readings to a base station. The reported readings are used to estimate of the environment being sensed, meaning that not all the sensor nodes are required in order to achieve the desired level of accuracy. The base station then backcasts information based on the estimate to the entire network and discriminatively switch on more sensor nodes to reach the desired level of error in the estimated parameter. The authors have shown analytically that backcasting can achieve energy savings ten times more than in the case where all sensor nodes are activated at the same accuracy. The limitation of this approach is that it takes a centralized coordination mechanism in which the base station is in charge of making decisions about the appropriateness of activating additional sensors in the network. By extension, backcasting suffers from the limitations of centralized energy control algorithms schemes.

The authors in (Mainland et al, 2005), introduce self-organizing resource allocation (SORA) for determining efficient node resource allocation in WSNs based on pricing mechanisms. In this approach, a virtual market is defined in which buyers (nodes) sell goods (e.g data aggregation, sampling rates, data relaying) in response to global (network wide) price information that is established by the end user. Nodes determine their individual behavior locally and independently by maximizing their own utilities, subject to energy constraints. However, prices are determined and set by an external coordinator agent to induce a desired network's global behavior and consequently this algorithm faces the same challenges as those in the backcasting algorithm. Additionally the question of how to determine optimal price settings is currently unknown.

The Glasweb project researchers in (Martinez, 2005) deployed a wireless sensor network to monitor the Briksdalsbreen glacier movement and behavior in Norway in order to understand climate change involving sea level change due to global warming, eventually to act as a crucial environmental hazard warning system, having recently proposed a decentralized control mechanism for adaptive sampling called Utility based Sensing and Communication (USAC) (Paddy et al, 2006). The algorithm is partitioned into two layers; the sensing protocol and communication protocol. Nodes employ the sensing protocol to adjust their sampling rates depending on the rate of change of its observations. The sensing protocol uses a linear regression predictor model to compute the next data with some confidence interval. If the observed data falls outside the confidence interval, the node adjusts its sampling rate to the maximum possible value. On the other hand if the data observed lies within the confidence interval, then the information value associated with the observed data is termed to be low. Consequently, the node adjusts its sampling rate downwards. The focus in this work is reducing the approximation error in sensor readings.

In the FloodNet project, a flood warning system, (Zou and De Roure,2007) , the authors used a grid based flood predictor model developed by environmental scientists to make flood predictions based on readings of water level collected by a set of sensors. The researchers propose FAR (FloodNet Adaptive Routing) to control energy consumption by adaptively managing the reporting frequency of sensor nodes and in choosing the routes data messages should follow. FAR considers three criteria that is: i) interest of diffusion ii) neighbor status maintenance and iii) adaptive routing algorithm. Interest diffusion involves propagating the predictor model originating from the gateway using a flooding technique on an hourly basis. The process is helps a given node to determine tables of important information about its neighbours. The algorithm then uses a neighbour status table that maintains the each neighbour's unique id, the distance of each of the neighbours from the gateway, the residual battery of and the data

importance (or priority) of the neighbour. The priority of a node is computed by a priority function that is directly proportional to the node's residual energy and inversely proportional to the power required to transmit the message from the current node to the neighbour node. This status information is then used in the adaptive routing stage to make data routing decisions. The basic operation of FAR is as follows. The sensor node sends a data message by selecting the neighbour with the highest priority (use a predefined priority function) among all the neighbours closer to the sink. If no such neighbour exists, the sender relays the data to neighbour with the highest priority among the neighbours with the same distance from the sink. If all neighbours are farther away from the sink, the data is sent to the highest priority neighbour among all the neighbours, the message is dropped otherwise. The general effect of the algorithm is that nodes currently with higher remaining battery power are used as data routers with a lighter reporting task whereas those with less residual power are given less priority of data routing but a higher reporting task. This ensures that energy is equitably distributed and prevents nodes with lower energy from draining their batteries quickly. Additionally by using the concept of data importance, the approach is able to avoid transmission of unnecessary data. The approach however, the approach suffers from a number of issues. First, the cost of maintain each neighbour status table, if not carefully chosen can end up consuming more battery power and supersede the benefits of the overall energy cost savings. Secondly the algorithm is not fault tolerant because should the gateway die either due to power issues or severe environmental conditions or physical damage, then the idea of interest diffusion would not work and consequently the whole protocol would fail. The algorithm also assumes uniform transmission powers for each sensor node. This assumption is far from the reality since the distance for each sensor node from its neighbour or the destination is not the same, noting that transmission power varies among other factors, the distance.

The authors in (Jain and Chang, 2004) propose an adaptive sampling technique based on the result of a Kalman filter. The algorithm is implemented in a distributed manner i.e it's implemented locally on each node. Even though processing Kalman filters is an expensive computational process, if the WSNs under consideration cover a vast geographical area, the distributed implementation saves cost due to communication cost ( if the same algorithm was implemented at a controlling node (BS)). For smaller WSN deployments in which nodes are in line of sight with the base station, the algorithm would not be energy efficient- a centralized approach would be a better alternative.

C. Alippi et al in (Alippi et al, 2007) proposed an adaptive sampling algorithm for minimizing energy consumption for a snow sensor. The proposed algorithm dynamically estimates the current maximum frequency of a signal by using a first set of collected samples and uses a modified version of the CUSUM statistical test for change detection (Zhenghong , 2010), ( Lee et al ,2004), ( Basseville, and Nikoforov, 1993) to detect changes in frequency. The change is detected when the maximum frequency happens to be above or below a threshold. A change in the maximum frequency affects the new sampling frequency which needs to be updated. The algorithm is implemented at the base station and notifies each sensor of their respective sampling frequencies. The authors attribute the implementation of the algorithm at the BS due to the fact that the algorithm is computationally intensive.

Unlike the other adaptive sampling techniques, the proposed algorithm in (Alippi et al, 2007) relies on the well proven Nyquist sampling theory drawn from the information theory (a field of computer science and electrical

engineering), which states that if  $F_{max}$  is the maximum frequency of a signal's power spectrum then the signal can be reconstructed at a sampling rate equal to or greater than twice  $F_{max}$  i.e  $F_{min} \geq 2F_{max}$ , where  $F_{min}$  is the minimum sampling rate/frequency. Thus the algorithm guarantees both energy conservation as well as quality of information (in this case, the ability to reconstruct the original signal with reasonable accuracy. Interestingly the energy savings are both due to sensing and radio communication. However, the algorithm suffers from implementation shortcomings. The authors show experimentally that the algorithm saves energy by 97% compared to static sampling techniques. The authors use a star topology in which it's assumed each sensor node is in communication range with the base station. This assumption would hold if the region being monitored is small enough. Large WSNs deployments would typically require multihop communications. Further, implementing the algorithm at the base station in large scale WSNs may not necessarily yield the benefit of energy expenditure due to computation, instead the reverse effect may be probable – if the nodes are not in communication range with the BS, and the BS needs to multicast new sampling rates to the respective nodes, then the energy consumed due to message transfer may outweigh the energy conservation margins due to computation.

## 2.7 The Proposed Distributed Energy Control Model

### 2.7.1 Introduction

Our proposed distributed cross layer coupled energy and quality of information control is an amalgam of three models: The signal frequency estimation model based on Discrete Fourier Transforms, models Discrete Fourier Transforms (DFTs) and the CUSUM statistical change detection tool.

We adopt an adaptive sampling algorithm akin to (Alippi et al, 2007) and (Martinez, 2005). In both cases, sampling rates are adjusted in response to temporal variations in the environment. However, our proposed model differs from all the rest in the following ways:

- i) **We consider multihop communication network topology as opposed to the star topology adopted in (Alippi et al, 2007).** The star topology in (Alippi et al, 2007) is valid as long as all the sensor nodes have direct communication with the base station or are at least as close to the base station as possible such that there is direct routing of data packets from the sensor nodes to the base station. In practice, if sensor nodes are randomly distributed in a vast field, a number of them may not afford direct communication with the base station, necessitating multihop routing.
- ii) **Distributed Implementation of the Sampling Algorithm as opposed to centralized Implementation**

We consider a distributed implementation of the algorithm in which the sampling algorithm is implemented locally on every sensor node as opposed to the centralized approach. The assumption in (Alippi et al, 2007) is that the sampling algorithm requires high computational energy only afforded by the base station that is considered to have an infinite amount of energy. Again, this is a valid assumption assuming the star topology adopted. With a multi-hop routing, controlling sampling rates from a central point would perhaps reduce the computation energy per sensor

node but would significantly increase the number of control messages sent by BS to each of the sensor nodes requesting them to adjust their sampling rates – it's likely that the communication overhead for this task would supersede the energy savings per sensor node due to transmissions and reception.

By having a distributed implementation of the sampling algorithm, we hope to incur a smaller penalty of computational cost per sensor node but hope to gain a more significant communication energy saving by making use of the concept of coherent processing. Moreover, using efficient software implementation of FFTs has the potential to circumvent the cost that could be incurred in computation by some significant margins.

iii) **Incorporation of adaptive duty cycling:** Sensor nodes undergo sleep for varied length of time. Each node is response of determining locally the duration of their sleep depending on their local environment. The effect is that within a node, sleep times may vary temporally and across nodes sleep durations are still variable within fixed time period. Therefore generally, energy consumption is balanced since not all nodes will be awake at the same time.

### 2.7.2 The System Model

With reference to figure 2.2 we consider a WSN with  $S$  Nodes spatially and randomly distributed in a region  $R$ . We start by assuming that sensor nodes that are in close neighbourhood are spatially correlated and so they may register similar values. Each node  $j$  collects data samples about its local environment and transmits the data to the base station,  $B$ . Since  $j$  may not be within transmission range with the  $B$ ,  $j$  may use intermediate nodes (routers) to forward the data to  $B$ . The communication path over which the data samples are propagated from the source  $j$  to the destination  $B$  is assumed to be error prone such that some data samples may be lost enroute to the destination consequently affecting the accuracy of the data reported at the base station. Moreover, the wireless channel over which the data is transmitted is bandwidth limited. With multiple sensors contenting for the same bandwidth limited channel, each sensor node experiences a delay  $D_j$  when transmitting its data samples to the base station. We also assume that  $R$  is characterized by continuous and abrupt changes both spatially and temporarily. At any time  $t$ , each sensor observes a time varying stochastic signal  $X_j^t$  within its local environment. Since  $X_j$  stochastic, its distribution is unknown a priori.

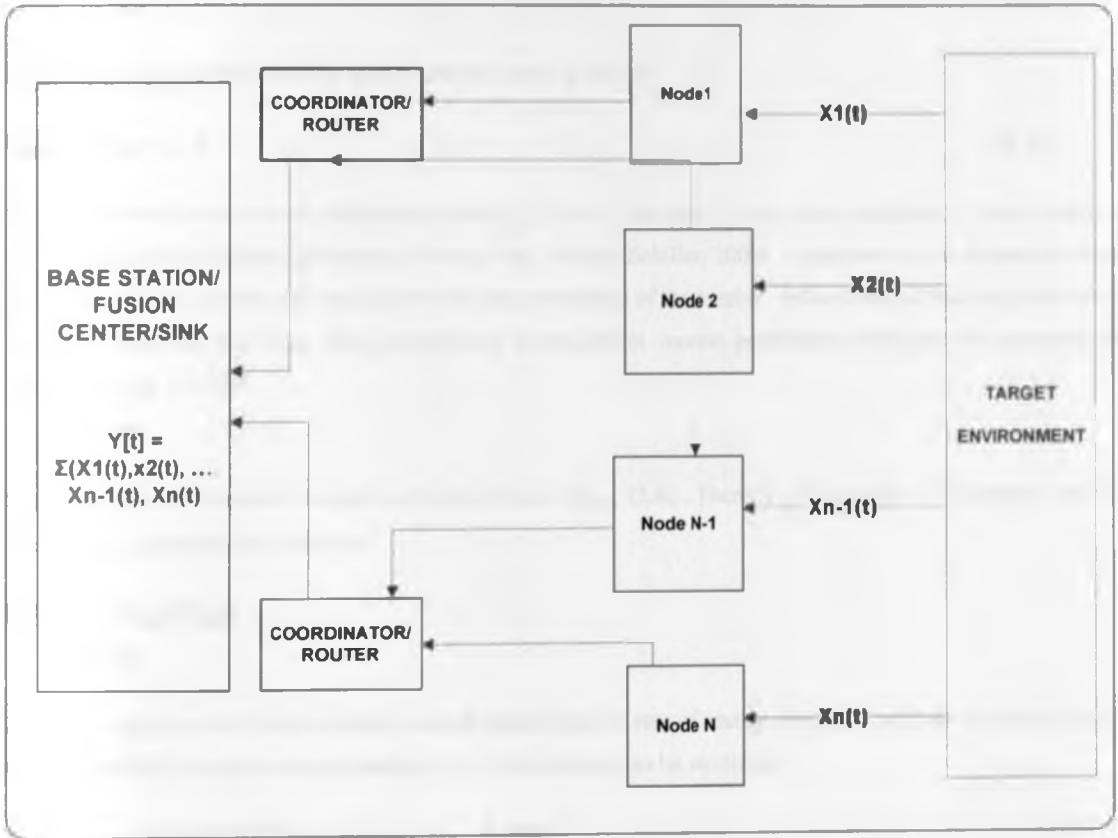


Figure 2-1: Conceptual Model - System Context Diagram

Let  $E_j$  be the total energy consumed by sensor node  $j$  within a time duration  $\Delta t$ ,  $D_j$  be the average delay experienced by the node within this time period and  $\rho$  be the average packet loss experienced by the network during this time.

The total energy spent by the nodes in the network is given by:

$$E_{net} = \sum_{j=0}^S E_j \quad (2.1)$$

And the rate of energy consumption experienced by the network is

$$E_{avg} = ( (E_{net}) / ( \Delta t ) ) \quad (2.2)$$

Since delay is an additive metric, the total delay experienced by the network is given by:

$$D_{net} = \sum_{j=0}^S D_j \quad (2.3)$$

The mean delay experienced in the sensor network is then given by:

$$D_{avg} = ( D_{net} / S ) \quad (2.4)$$

We also define data quality or information quality, Qol as a function of two major attributes of data quality – completeness and timeliness ((Hermans, Dziengel and Jochien Schiller, 2009). Completeness can be expressed as the number of data samples that arrive at the sink as a percentage of the number of data samples that are generated at the source nodes in unit time. This is equivalent to the packet success percentage which can be expressed as

$$P_{avg} = (1 - p ) \times 100 \quad (2.5)$$

Timeliness can be expressed in terms of the mean delay  $D_{avg}$  (2.4) . Therefore the quality of information can be expressed as a function of D and P as:

$$Qol = f ( D_{avg}, P_{avg} ) \quad (2.6)$$

Our goal at system level is to minimize as much as possible the rate of energy consumption (2.4) and maximize as much as possible the quality of information (2.6). This problem can be written as:

$$\text{Min. } E_{navg} , \text{ subject to Max. } (Qol = f(D_{avg}, P_{avg})) \quad (2.7)$$

From our definition of Qol, Qol is a decreasing function of the delay D (since the larger the value of D, the less timely the information becomes and the more meaningless this information becomes) and an increasing value of the packet success percentage P (since the more samples we collect at the sink in a fixed amount of time the more accurate the information becomes). Hence to maximize Qol we need to minimize the network delay and the maximize the packet success probability.

### 2.7.3 Node Level Model

This model is derived from the system model. It details the internal components of a sensor node that are a center of our focus: The power and energy models, the sensory and computation models , the sampling model and the communication models embedded within the node model are discussed here. See section 3.3

#### a) Energy and Power Model

Consists of the following sub-models:

1. **The Battery model.** The model defines the battery characteristics. The model can be thought of as the *energy producer* and so powers the other components within the node. As the battery model powers the

other components, it is also responsible for keeping track of the energy consumed by these components and the remaining energy.

2. **The Transmitter Power Model:** Models the radio transmission power consumption characteristics. This model is an *energy consumer model*.
3. **Receiver Power Model:** Models the radio receiver power consumption characteristics. This model is an *energy consumer model too*.
4. **CPU Energy Model.** Models the energy consumption due to computation and processing.

**b) The Sensing and Computational Model**

Models the sensor layer of the node so as to include the sensor physical and application layers; the layers responsible for collecting sensor data from the environment and transforming these data. This model includes the sampling model.

- c) **Communication Model:** Models the communication protocol stack used for transmission of data over the wireless medium.

The various models are shown in figure 4.3



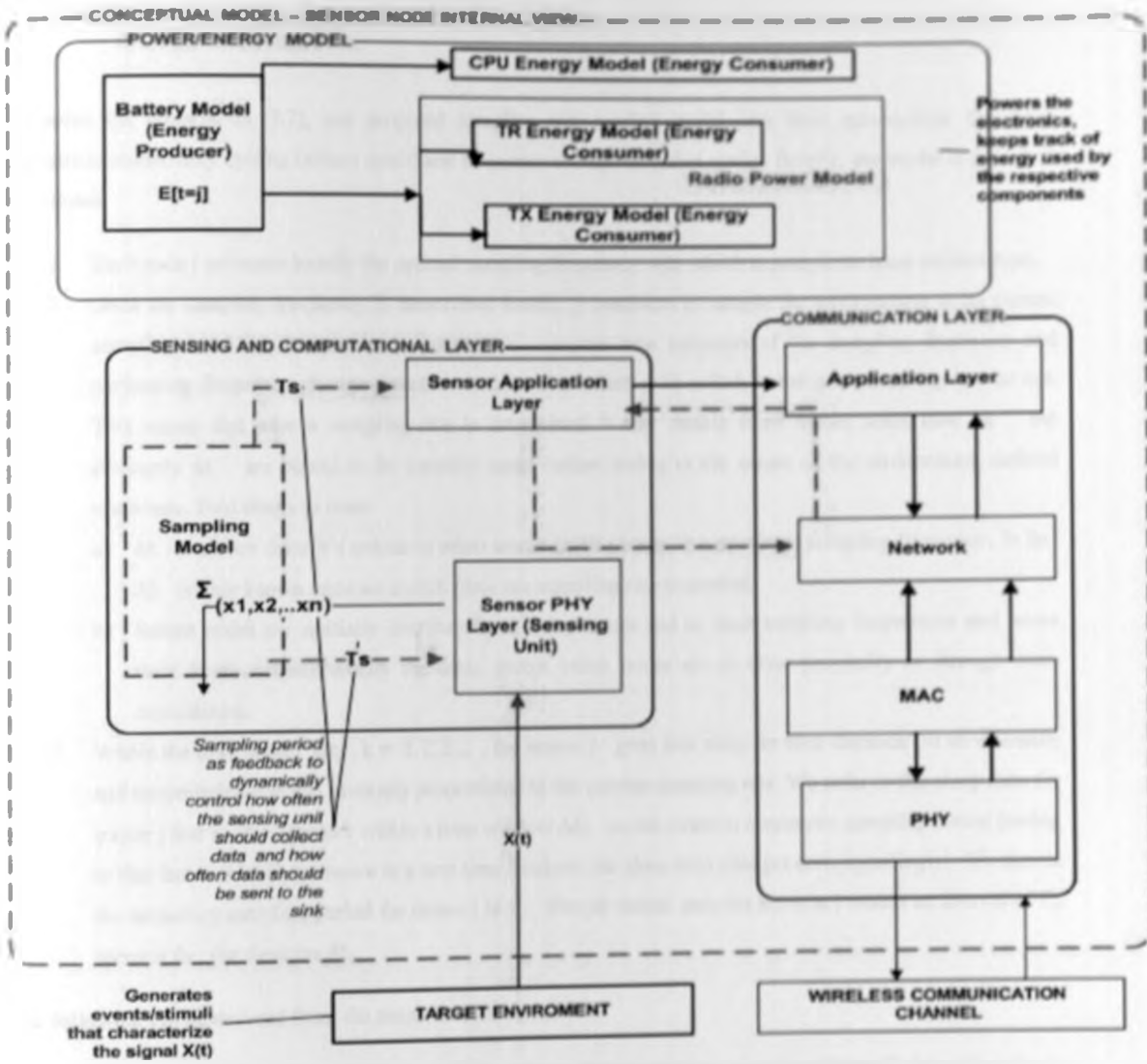


Figure 4-2: Conceptual Model – Detailed Node Level Model

#### 2.7.4 The Adaptive Sensing and Sampling Model

To solve the problem in (2.7), our proposed sampling and sensing model uses three sub-models: frequency estimation model, duty cycling (sleep) model and frequency change detection model. Briefly, the model is described as follows:

1. Each node  $j$  estimates locally the optimal sampling frequency with which to sample its local environment.
2. Once the sampling frequency is determined locally,  $j$  continues to sample the environment at its current sampling rate. Simultaneously,  $j$  continues to compute new estimates of the sampling frequency and performing frequency change detection to determine whether to switch to the new sampling rate or not. This means that once a sampling rate is determined it may remain fixed within some time  $\Delta t$  but obviously  $\Delta t$  are bound to be typically small values owing to the nature of the environment defined elsewhere. Two things to note:
  - a)  $\Delta t$  is random since it's unknown when sensor  $j$  will change to a new local sampling frequency. In fact  $\Delta t$  is only known once we switch from one sampling rate to another.
  - b) Sensor nodes are spatially distributed and independent and so their sampling frequencies and hence their  $\Delta t$  are not necessarily the same except when nodes are in close proximity or through mere coincidence.
3. Within the time period  $\Delta t_k, k = 1, 2, 3, \dots$  the sensor  $j$  goes into sleep for time duration (to save sensing and transmission energy) inversely proportional to the current sampling rate. We refer to this sleep time for sensor  $j$  that is only constant within a time window  $\Delta t_k$  as the (node's) *temporary sampling period* (owing to that fact that when we move to a new time window, the sleep time changes correspondingly). We denote the temporary sampling period for sensor  $j$  as  $T_j$ . Simply stated, samples arrive at  $j$  within an interval of  $T_j$  seconds for the duration  $\Delta t_k$ .

The following can be deduced from the proposed sensing model:

1. The sensing is purely asynchronous and so sensor observations are not "coordinated" but rather "truly independent" observations
2. Processing (estimation) of sampling rate is purely local (coherent) and not centralized like in most strategies. Coherent processing has been proven to be one of the most effective adaptive ways of controlling energy control in wireless sensor networks (ref required) where most computations are done locally on the node before transmission.
3. Adaptive sampling (how many samples the sensor collects from the environment) is complemented by adaptive sensing (in our context) the interval between observations.
4. Sensing is both temporarily adaptive (within a node) and spatially adaptive (across nodes). The former means that for a single node, the sensing and sampling parameters are adjusted with time while the later means that within a fixed timestamp, the sensing and sampling parameters are not necessarily the same.

This has the benefit of balancing energy consumption across nodes due to overlapping and/or adaptively alternating sleep periods and varying “workloads” (sample size at a time instant.).

Each of the models is discussed in the sections that follow.

### 2.7.4.1 Estimating the Sampling Rate

We first establish the necessary mathematical basis as follows:

If  $X(t)$  is a continuous signal with a maximum frequency  $B$ , then the minimum rate at which the signal should be sampled to sufficiently reconstruct the signal is given by:

$$F_{\min} \geq 2B \quad (\text{Shannon, 1949}) \quad (4.5)$$

The above theorem is also called Nyquist-Shannon Sampling Theorem.

So if  $B$  was known, then we can easily derive  $F_{\min}$  from (4.5) as our sampling rate unfortunately this is not the case. We will shortly see how this is achieved.

Next, the Discrete Fourier Transform (DFT) defined on discrete random samples  $X = (x_1, x_2, x_3, \dots, x_n)$  sampled from the time varying signal  $X(t)$  is a mathematical model of the form: (Duhamel and

Vertterli, 1990):

$$* \text{dft}(X) = Y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix} = A \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} \quad (4.6)$$

$\text{dft}(X)$  is the complex output and  $X$  is generally complex too.

Where

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & W_N^3 & \dots & W_N^{(N-1)(N-1)} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & W_N^{(N-1)(N-1)} & W_N^{2(N-1)} & \dots & \dots & W_N^{(N-1)(N-1)} \end{bmatrix}$$

and  $W_N = e^{-j2\pi/N}$

Intuitively computing a DFT according to (4.6) takes time complexity of  $O(N^2)$ . This is indeed an expensive operation for larger values of  $N$ . Fortunately there exists a class of algorithms which use a divide and conquer approach to fulfill the same computation using just  $O(N \log N)$  operations. These set of algorithms are called the *Fast Fourier Transforms (FFT)*. Further, by utilizing certain properties associating input data to the result of a DFT, it's possible to achieve two times speedup in efficiency of FFT implementations .

Additionally, certain implementations of FFT such as FFTW3 that use adaptive forms of FFT can achieve even much better results.

Given that sensor nodes are already energy constrained then there must be a compelling reason for our choice to implement the computationally intensive FFT locally on the node and still hope to save significant amounts of energy instead of implementing the algorithm at the base station with sufficient source of power. First we state the two properties that our sensor network satisfies in order to take advantage of the two properties.

Now, we noted in (4.6) that the output of a DFT is a complex vector whose elements have a mixture of real and imaginary components. The input vector  $X$  is usually complex too. However, in some practical applications such as sensor networks, the input  $X$  (which is a sequence of purely real valued observations) is real (and not complex). In such a case the transformation of the input vector to the output vector constitute a real to complex (R2C) transform and the inverse process entails a complex to real (C2R) transform (). In our thesis, we are interested in the unique properties of the R2C DFT transform.

### Properties of an R2C DFT Transform

1. The  $i$ th and  $(n-i)$ <sup>th</sup> elements of the complex output are identically equal but opposite in sign . This property satisfies the so called "Hermitian Redundancy". (4.7)
2. If the number of elements in the input vector is  $n$  and  $n$  is even, then the  $(0)$ <sup>th</sup> and  $(n/2)$ <sup>th</sup> element of the output vector are purely real ( have no imaginary parts) and are referred to as the DC and Nyquist components in that order. (4.8)

Property one implies that the upper half of the complex output can be done away reducing the computational cost by half. The second property allows us to derive the Nyquist frequency that is a real number and hence a "true" sampling rate.

Our sampling and sensing model dictates that the sensor network satisfies both the above properties. Firstly the sensors in the network observe purely real values, so by designing an optimal implementation of the FFT algorithm, computational cost becomes less critical compared to the communication overhead if sampling was to be controlled at the sink. The second property is satisfied by our choice of sample sizes that are deliberately even numbered

We apply the fundamental results from 4.5 through 4.8 to our sensor network. Specifically;

Let the vector,  $I_j^t$  be a time series of discrete inputs collected by sensor  $j$  at time  $t$  of its local signal  $X_j^t$ . We define  $I_j^t$  according to the relation

$$I_j^t = W = (x_0, x_2, \dots, x_{w-1}) \quad (1a), \text{ if } t=0 \text{ at the start of sensing} \quad (4.9a)$$

$$I_j^t = (x_k, x_{k+1}, \dots, x_{g-1}) \quad (1b), \text{ if } t>0, (1b) \text{ as sensing progresses} \quad (4.9b)$$

$$1 \leq j \leq S$$

$$w \leq k \leq g$$

$$* \quad 0 \leq t \leq \infty$$

$$g = (N - w)/h$$

Where;

- $S$  is the number of sensor nodes in the in the field
- $N$  is the total number of samples collected by the sensor node
- $W$  is the vector containing sensor inputs at  $t=0$  and  $w$  is the size of this vector
- $h$  is a constant representing the number of samples collected from the environment at each time step except when  $t=0$ .

Using the results from (2), we compute the DFT on the sequence of sensor samples as:

$$o_j^t = \text{dft}(I_j^t) = A \times I_j^{tT} \quad (4.10)$$

Where  $\times I_j^{tT}$  is the transpose of the input vector

Conventionally,  $Y [f/2]$ , the Nyquist Frequency is denoted as  $B$ .

From (4.9a) and (4.9b), the sensor input vector  $I_j^t$  is strictly real as it contains practical values such as temperature levels

Further we deliberately choose  $W$  and  $h$  such that the size of  $I_j^t$  is always even.  $I_j^t$  therefore satisfies the properties in (4.7) and (4.8) above and therefore define  $B_j^t$ , as the estimated Nyquist Frequency of  $X(t)$  at time  $t$  as  $B_j^t = o_{j[m/2]}^t$ , where  $m$  is the size of  $o_j^t$  (4.12)

if  $t=0$ ,  $B_j^t = B_j^0$ . We call this, as the initial estimated Nyquist frequency (estimated sensor  $j$ ). Otherwise, we refer to  $B_j^t$  as the estimated current Nyquist frequency for sensor  $j$ .

Let  $F_j^t$  be the sampling rate of sensor  $j$  at time  $t$ . From (4.5) and (4.12):

$$F_j^t = 2 B_j^t \quad (4.13)$$

Thus the sensor j will collect  $f_j^t$  samples per second.

The value  $f_j^t$  will remain fixed until a time duration  $\Delta t$ .  $\Delta t$  is random and therefore is not known a priori.

#### 2.7.4.2 Signal Frequency Change Detection Model

##### 3.2.2.1 Preliminaries

A signal can be generally represented by the equation:

$$y_t = \phi_t + e_t \quad (4.14)$$

Where  $\phi_t$  is the deterministic component of the signal and  $e_t$  is the additive white noise.

(Fredrick Gustaffson, 2000) .

The problem of determining  $\phi_t$  is referred to as estimation while change detection is the process of determining rapid changes in  $\phi_t$ , starting at some unknown time  $t_k$ .  $t_k$  is called change time

We have addressed the estimation task in the previous section which involved estimating the optimal sampling rate of the sensor at any given time. The next task is performing change detection test to establish whether there are any significant changes in the frequency parameter of the signal ( $\phi_t$ ) i.e we would like to determine at each time instant if

$\phi_{(t=k)}$  and  $\phi_{t=(k+1)}$ ,  $k = 1, 2, 3 \dots$  are significantly different.

The change at time k is defined as  $v \triangleq \phi_{(k+1)} - \phi_{(k)}$  (4.15)

Many algorithms exist for determining  $t_k$  . the time at which a change has happened. We use the modified CUSUM test based on (Page, 1952) for our purpose of detecting signal changes. To determine, whether or not to switch to a new sampling rate, we proceed as follows:

1.  $B1 = -0.5 B_j^0$ ,  $B2 = 3/2 B_j^0$
2. IF  $(|B_j^t - B1| < (B_j^t - B_j^0) \parallel (|B_j^t - B2| < (|B_j^t - B_j^0|))$

3 THEN

4 . UPDATE THE SAMPLING RATE using equation (5).

5.  $B1 = B_1^t$ ,  $B2 = 3/2 B_1^t$

The simplified sampling model is shown in figure 4.4 below.

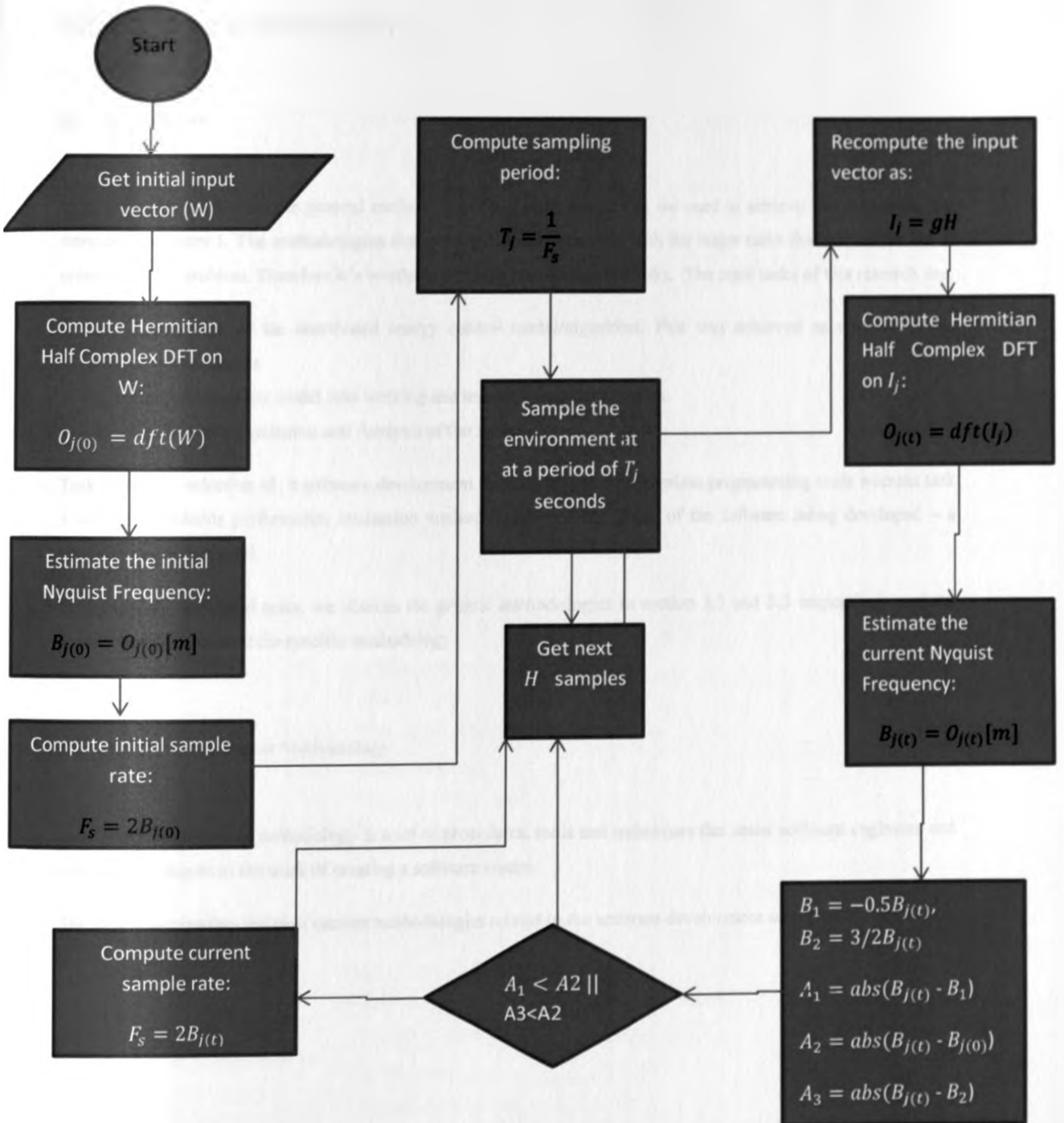


Figure 4-3: The Adaptive Sensing and Sampling Model – Control Flow Diagram



## CHAPTER 3: METHODOLOGY

### 3.1 Introduction

In this chapter we discuss the general methods, tools and instruments that we used to achieve the objectives that were set in chapter 1. The methodologies discussed are directly matched with the major tasks that we carried out in order solve the problem. Therefore it's worthwhile briefly mentioning the tasks. The main tasks of this research are:

1. Formulation of the distributed energy control model/algorithm. This was achieved as detailed in the previous chapter.
2. Conversion of the model into working and testable computer program.
3. Performance Evaluation and Analysis of the model

Task 2 requires selection of a software development methodology and appropriate programming tools whereas task 3 requires a suitable performance evaluation method in view of the nature of the software being developed – a communication protocol.

In light of the identified tasks, we discuss the general methodologies in section 5.2 and 5.3 respectively and the reasons for our choice of the specific methodology.

### 3.2 Software Development Methodology

A software development methodology is a set of procedures, tools and techniques that assist software engineers and software developers in the work of creating a software system.

The below summarizes the most current methodologies related to the software development activities

Table 1: List of Common Software Development Methodologies

Methodology	Main Features	Strengths	Weaknesses
-------------	---------------	-----------	------------

<b>Waterfall</b>	<ul style="list-style-type: none"> <li>▪ .Structured approach to software development</li> <li>▪ Software development flows through phases starting with Analysis, Design, Implementation, Testing and ending in deployment.</li> <li>▪ Each of the phases has a set of well-defined activities</li> </ul>	<ul style="list-style-type: none"> <li>• Disciplined in nature</li> <li>• Requirements well analyzed and translated into designs before coding.</li> <li>• Level of defect injection from one phase to another is minimized</li> </ul>	<ul style="list-style-type: none"> <li>▪ Lacks early feedback</li> <li>▪ When defects are found in requirements they can be costly to fix</li> <li>▪ Inflexible to changes</li> </ul>
<b>Incremental an Iterative Design</b>	<ul style="list-style-type: none"> <li>▪ A functionality or subset of system functionalities are design, built, tested and delivered giving birth to a unique release of the software</li> <li>▪ The cycle of design, build and test is repeated until the desired functionality is achieved</li> </ul>		
<b>Prototyping</b>	<ul style="list-style-type: none"> <li>▪ A few aspects of the main system are simulated and may be completely different from the real system</li> <li>▪ Typically follows the sequence: determine basic inputs and outputs of the system, build the prototype,</li> </ul>	<ul style="list-style-type: none"> <li>• Can be used to quickly gauge accuracy early enough in the project</li> <li>• Suitable for demonstration where building the actual system would take long</li> </ul>	<ul style="list-style-type: none"> <li>▪ The final system may completely deviate from the initial prototype specification</li> </ul>

	review the prototype and enhance the prototype		
<b>Spiral</b>	▪ Combines waterfall and Prototyping	Suitable for large and complex projects where risk is the emphasis	Unsuitable for small projects ( typically < 6months)
<b>Unified Rational Process (URP)</b>	<ul style="list-style-type: none"> <li>▪ Emphasis on following best practices</li> <li>▪ Software developed incrementally and iteratively</li> <li>▪ Requirements managed using use cases</li> <li>▪ Visual modeling using UML</li> <li>▪ Software quality verified</li> <li>▪ Designs through component based architectures</li> </ul>	<ul style="list-style-type: none"> <li>▪ Good where software is done in teams</li> <li>▪ Easy to track changes in requirements</li> <li>▪ Coding is simplified via elaborate visual designs</li> <li>▪ Easy to ascertain software quality</li> </ul>	▪ May not work well for fast track projects that do not need design work.

In this project, we largely followed the prototyping methodology to design, build and test the distributed control energy model. The choice of the model was dedicated by various factors. The project was bound to be completed within six months and system development was just a part. Also, development and testing a fully work solution was impractical since it would call for acquisition of real sensor hardware and related software such as the sensor operating system. Thus by prototyping, we could easily demonstrate only those parts that were relevant to our research using available software tools. The specific steps that were followed in the prototyping process in relation to our work were:

#### 1. Identification of Basic Requirements

Given the main task is protocol development, the requirements for the software development activities were basically nonfunctional (performance) requirements which also coincide with the objectives. The nonfunctional requirements were:

- a) **Energy efficiency:** the developed protocol should reduce amount of energy consumed
- b) **Quality of Information :** the protocol should ensure completeness of information and timeliness of information delivery

Specific measures of energy efficiency and quality of information are discussed in chapter 7 under performance evaluation and analysis.

2. **Design of the software:** We used high level component and context diagrams to show how the protocol fits in the bigger sensor network system. Pseudo codes were used to describe the working logic of the protocol. Details of design are elaborated in chapter 6.
3. **Build the system:** The design was converted into a computer program using C++ and the FFTW3 library. C++ was chosen because of its efficiency and the FFTW3 library was selected based on its adaptivity to varying hardware capabilities. Detailed information about FFTW3 can be found in chapter 6.
4. **Test the software (Protocol):** Testing involved verifying that the output generated at code level by the protocol conforms to the initial model specification underlying the implemented protocol. The main features of the protocol tested were:
  - Generation of correct values of the DFT on various sample sizes
  - Generation of correct values for sampling rates and periods based on the DFT outputs
  - Generation of correct values for sleep time (sampling intervals).

## 5. Evaluation and Analysis

Evaluation and analysis was performed to determine the extent to which the developed protocol is better than standard protocols in the realm of sampling in terms of the established measures of performance; energy efficiency and QoI. To compare the performance, fixed sampling as the most commonly used algorithm was identified as a benchmark. To evaluate performance of DAS against fixed sampling, needed a way of modeling the sensor network that executes the protocols to produce output. In 5.2, we describe the methodology adopted in modeling the entire sensor network system.

### 3.3 Performance Evaluation Methodology

#### 3.3.1 A Survey of Performance Modeling Methodologies in WSNs

##### 3.3.1.1 Analytical Models

Analytical models are based on simple mathematical equations which are usually less involving and relatively cheaper compared to the other two approaches. Analytical modeling can be used to give quick insights into the methods and algorithms designed for wireless sensor network. However, they are less suitable for complex dynamical systems such as wireless sensor networks – particularly when considering the complex interaction of different WSN variables such as limited energy, real time operation and quality of information; hence analytical models may not yield useful results in WSN studies (Chen et al. 2006).

##### 3.3.1.2 Test bed performance measurements

Test bed performance measurements in WSNs involve conducting real world field experiments using physical WSN test beds. Real world performance measurements are the most ideal methods of obtaining realistic WSN performance characteristics. It's a good means of validating analytical and simulation models. However, a

number of challenges exist that hinder successful performance measurements of WSNs algorithms and protocols. Firstly, the cost of WSN nodes is high and we cannot afford it given that this project is to be completed on a lean financial budget. Secondly real WSN nodes are delicate and fragile and susceptible to failure when exposed in the open environment (unless well protected). Practical and successful field deployment of WSNs can be time involving. Lastly, since our research involves a couple of experimentation variables and the system involved is highly dynamic, it's impractical to control the effect of extraneous variables when conducting real life performance measurements (Zeigler, 1976). In consideration of all the foregoing factors, test bed performance measurements are generally inappropriate for our work.

### 3.3.1.3 Simulations

Simulation can be defined as the method of using computer programs or software to model the operation of the real world, processes, systems or events (Law and Kelton, 1991). Other authors define simulation as virtual experiments (Carley, 2001) or as simplified pictures of the real world having part of the characteristics of the real world system (Lave and March, 1975). If the system under study can be simplified to a set of computer algorithms, then simulations can be used to evaluate such a system. Simulation offers a more flexible way of controlling experimental factors (Opiyo, 2010) than other modeling techniques. In relation to our project, simulation will provide the best means of verifying performance of the different WSN protocols under different conditions. Thus, this project will adopt simulation methodology in analyzing and validating the distributed energy and QoS control protocol detailed in section.

In the remaining parts of this chapter, we describe the system/conceptual model, which is a non-software specification of the system to be simulated. We do so by first, describing the problem as outlined in the *problem analysis* subsection, we then identify the simulation objectives consistent with the problem. This is followed by a description of the high level system model showing major components and their attributes. The high level system model is further decomposed into its constituent sub-models to simplify the understanding of the system. The structure of the sub-models is given as well as their behaviour. The behaviour of the various modules is described through detailed process flows and algorithms by means of pseudo-codes. Later on, these algorithms will be converted into the computer code using an appropriate simulation framework.

In order to use simulation methodology effectively, we need an appropriate simulation framework and tools that will enable us to achieve credible results. We therefore discuss the simulation framework (OPNET) that will be used to execute the conceptual model in section 5.3

## 3.4 Survey of Simulation Environments for Wireless Sensor Networks

In order to analyze the performance of the proposed distributed adaptive sampling protocol, we simulate it in an appropriate simulator. Many simulation environments exist for this purpose. A few commonly used simulators for

WSNs are presented in section 3.4. We specifically chose to use OPNET for our modeling purpose for reasons outlined in section the sections that follow. OPNET alone does not provide fully developed models for accurately modeling WSNs. Fortunately there is an existing open source OPNET implementation of the IEEE 802.15.4 standard for WSNs called open-ZB (Jurcik and Koubaa, 2007). The open-ZB model was configured into the OPNET environment and reused for the modeling and simulation of the adaptive sampling

In this project, we will concentrate on building and testing the sensor layer protocol on top of appropriate existing MAC and PHY layer communication protocols. In particular we choose to develop our implementation on top of the IEEE 802.15.4 wireless mesh standard for reasons that will be explained in later sections.

Implementation in which case, the sampling model is implemented as a separate external module then linked to the sensory unit of the simulation model. The former is desirable if the simulation tool has sufficient inbuilt libraries to support an efficient and adaptive implementation of the more resource demanding problem of computing the DFT of an input function.

A number of simulation frameworks exist for modeling and testing WSNs. Some of the tools are have been used to simulate traditional networks while others have been built to meet the unique needs of WSNs. Examples of network simulators for classical networks are NS-2 and OPNET. Examples of WSN specific simulators include OMNET++,

ContikiOS and COOJA, TOSSIM (a simulator for TinyOS operating system. Later on, simulation experiments will be performed under different scenarios to evaluate different performance metrics based on the modeling objectives identified in the conceptual model. Simulation results will be analyzed using appropriate tools outlined in the section that follow. In the below, we summarize the comparison among the most commonly WSN simulation environments.

While we have chosen to use OPNET Modeler in testing the performance of sampling model for reasons explained in the sections that follow, at the time of doing this project and to the best of our knowledge, OPNET Modeler 11.5, the version we used for simulating the performance of the model, did not have inbuilt libraries for computing the DFT of input signals, and more specifically efficient software libraries for computing DFTs for real data. To this end, we chose to use the fastest ever known publicly available software library called FFTW; precisely the version 3 of the library called FFTW3. Details for FFTW3 are discussed in chapter 4

Table 3-1: Comparison of Commonly Used Wireless Sensor Network Simulators

Simulator	Main Features	Limitations
-----------	---------------	-------------

<p>OMNET with CASTALI A</p>	<ul style="list-style-type: none"> <li>▪ Has several MAC protocol implementations</li> <li>▪ Simple tree routing protocol</li> <li>▪ Has several PHY Layer protocol support</li> <li>▪ Has an energy and battery model</li> <li>▪ Flexible parametric physical layer sensing model</li> <li>▪ Modular programming style based on C and the Ned language</li> <li>▪ Open source license</li> </ul>	<ul style="list-style-type: none"> <li>▪ Relatively harder to install compared to OPNET</li> <li>▪ Visualization and statistical tools not easily accessible</li> </ul>
<p>OPNET</p>	<ul style="list-style-type: none"> <li>▪ Supports several PHY layer , MAC and Routing protocols</li> <li>▪ Sensor energy consumption &amp; battery available through the OpenZB model.</li> <li>▪ Sensing model available through OpenZB library</li> <li>▪ Relatively easy to install on windows</li> <li>▪ Modular design</li> <li>▪ Based on C++/Proto-C</li> <li>▪ Rich set of models that can be extended with lots of ease</li> <li>▪ Hundreds of traffic generation models</li> <li>▪ Rich set of statistical analysis tools</li> <li>▪ Simulation visualization tools available.</li> </ul>	<ul style="list-style-type: none"> <li>• Commercial and hence high licensing costs</li> </ul>
<p>N-S2</p>	<ul style="list-style-type: none"> <li>▪ Supports multiple MAC, PHY and Routing layer protocols</li> <li>▪ Sensing model supported through Manassim</li> <li>▪ Open source</li> </ul>	<ul style="list-style-type: none"> <li>• Supports only two MAC protocols</li> <li>• Moderately hard to install and configure</li> <li>• The integration of Tcl makes it relatively hard to learn and use</li> </ul>

**Table 3.3: A list of some of the most popular WSN simulators, their main features and limitations**

From the above, list we opt for OPNET over the rest for the following reasons:-

1. **Object Orientation:** Systems in OPNET are specified as a set of objects. The objects have attributes and behaviour. Attributes can be defined or edited via the GUI. Behaviour can be defined through the process models implemented as processes. The OOP methodology being widely used in the software world makes OPNET easier to grasp.
2. **Hierarchical Models:** Models are grouped into logical entities akin to those found in actual networks and distributed systems.
3. **Graphical specification:** It's possible to define and configure entire models or aspects of a model by simply editing existing models or defining new models or model through the OPNET GUI. This reduces the amount of actual coding to be done. Not quite many simulators support GUI model specification.
4. **Flexibility of customizing models:** Existing OPNET models closely parallel those of real world communication and distributed systems. The simulator provides a wide range of such models that can be quickly customized via the familiar C++ programming language interface.
5. **Application specific statistics:** - OPNET provides many standards in built performance statistics that can be collected automatically during simulation without the headache of programming them. Custom performance statistics are also possible through user defined processes.
6. **Integrated post simulation analysis :** Sophisticated operations for graphical presentation and analysis of simulation output
7. **Automatic generation of simulations**
8. **Easy to install, configure and learn** - Installation on windows is the easiest and is guided by the setup and wizards. Linux installation is slightly more complex though relatively easier when

Despite its numerous advantages, the simulation methodology as a method of evaluating and analyzing wireless sensor network protocols still has a number of limitations:

- i) The field of Wireless Sensor Networks is relatively new compared to the rest of the wireless networks. Fewer simulation software packages currently available have all the features necessary to adequately simulate WSNs. For example, OPNET lacks well defined models for the physical process under monitoring. However, OPNET offers alternative application traffic models that can be customized to model the physical process as closely as possible. Besides, by using user contributed open source models such as the open-ZB model, we are able to achieve a more realistic simulation model of WSNs and hence improved reliability of the results obtained from the simulation experiments.



## CHAPTER 4: DESIGN AND IMPLEMENTATION

### 4.1 Introduction

In the previous chapter we identified the procedures, tools and methods for implementing our proposed model. In this section we will discuss these tools (OPNET and the FFTW3 library) in much more detail and how they were used to implement the solution.

### 4.2 Overview of OPNET

The OPNET modeling architecture is structured in a layered fashion with each layer representing a certain modeling domain. There are three basic domains in OPNET: The process domain, the node domain and the network

#### 4.2.1 Modeling Structure and Process

##### 4.2.1.1 Process Domain Model

It is the lowest layer in the hierarchy. At this layer process *models* are defined. A process model contains the actual code of modeled protocols and algorithms that defines the behaviour of the node model within the node domain. Process models are implemented through executable program units called *processes*. The complete specification of an OPNET process model consists of a FSM, action statements expressed in C/C++, and configurable parameters. Process models are developed using the process editor tools in OPNET. An example OPNET process model is shown in figure 3.6 below.

##### 4.2.1.2 Node Domain Model

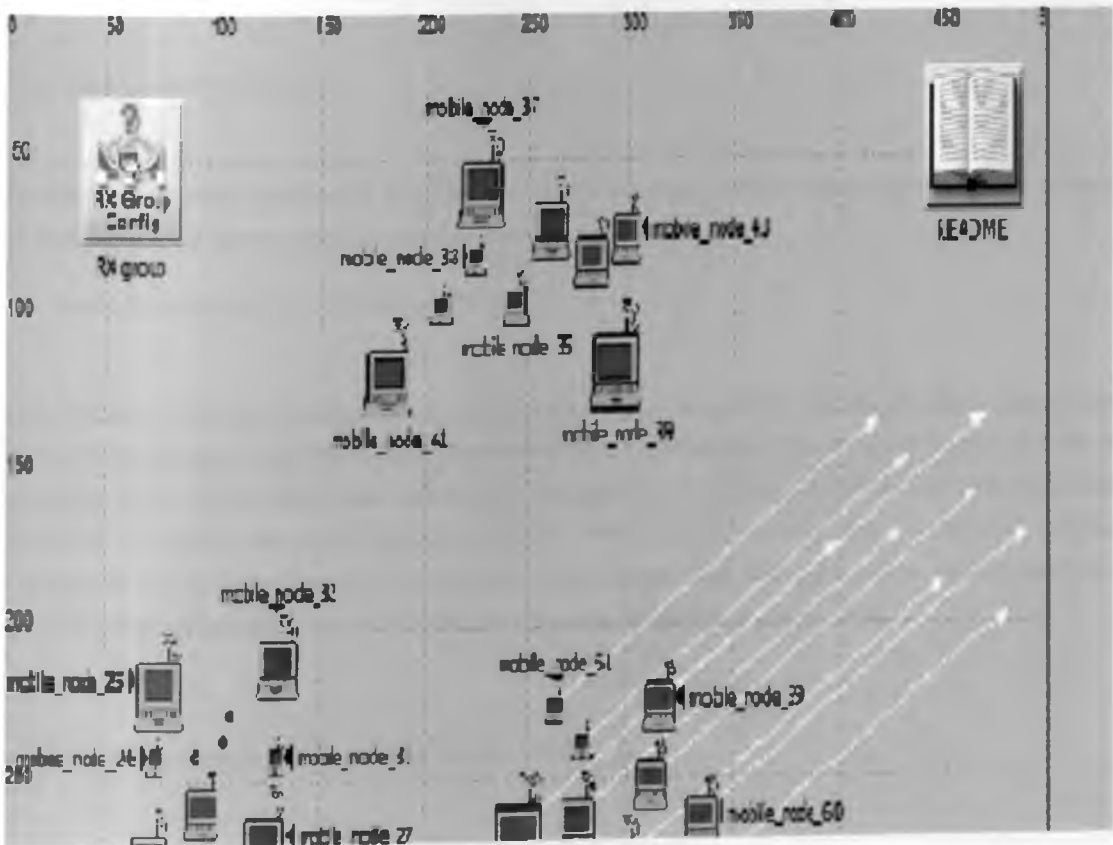
The node domain is responsible for creating models of individual devices known as the node *model*. The node model provides an internal architecture of the node in terms of its constituent elements, their attributes and the data flow between these elements. The smallest element of a node model is called a *module*. A node module can be a transmitter, a receiver, a processor, a queue or an external system. The behaviour of transmitters and receivers can only be changed by configuring inbuilt parameters.

The rest of the modules are highly programmable- their behaviour is defined by attaching them to a predefined process model. Communication between modules in a node is supported via *packet streams*, *statistic wires* and/or *logical associations*. Packet streams allow packets to be relayed from one module to another. *Statistic wires* are used

to relay control information from one module to another. The last type of connection is used to identify a logical binding between modules. In OPNET, logical associations are only allowed between transmitter and receiver modules. Node models are developed using a tool called *node editor*. An example node model is depicted in figure 3.5.

#### 4.2.1.3 Network Domain Model

The network domain defines the topology of a communication network. The individual building blocks at this layer are the nodes with each node defined by its own node model. The model associated with the network domain is the network model. The network model can be viewed as consisting of several node models. From an architectural view point, the network domain is the high level design of the communication system under development. Specifically then, a node is a particular instance of a node model – implying that a single node model may have many node instances. Communication between nodes in the network domain is actualized through communication links. Network models are configured using the project editors. Figure 3.4 below, depicts the high level architecture of an OPNET Model.



#### 4.2.2 How to specify Process, Node and Network Models in OPNET

The task of developing a representation of the system under study. This is equivalent to a conceptual model. Model specification in OPNET is done by complete specification of a network model, the corresponding node models and for each node model, the associated process model(s). The development of the three models is supported by OPNET tools called *project editors*, *node editors* and *process editors* respectively. Other editors in OPNET include: packet format editor for developing packet format models, PDF editors for editing viewing and defining probability distribution functions.

#### 4.2.3 Data Collection in OPNET

Provided that the model specification is representative of the real system, OPNET modeler allows realistic estimates of performance and behaviour to be obtained by executing simulations

#### 4.2.4 Data Analysis and Visualization in OPNET

Data collected during the simulation runs is saved to output vector files. OPNET modeler provides a graphical and numerical processing environment in the results browser of the project editor. One can define the type of results by specifying which statistic they would wish to view. The statistics of interest could be End to end delay, channel utilization, throughput and so on. Moreover, OPNET allows multiple ways of data visualization; individual visualization allows one to view only one statistic at a time, stacked view, allows one to view multiple statistics as separate graphs and overlaid views enables one to view several statistics on the same graph.

#### 4.2.5 Modeling Network and Application Specific Traffic

OPNET supports two basic types of traffic- *explicit* and *background traffic*

Explicit traffic is a packet by packet traffic. Models complete protocol effects. Explicit traffic can be generated through the following ways:

**Packet generation:** Node objects are configured to generate generic packets

**Application demands:** You can create application demands to represent traffic flow between two nodes.

**Application traffic models:** OPNET includes a set of models for generating traffic based on standard applications such as HTTP, voice, FTP etc. You can also use generic custom application model to represent a broad range of applications that do not conform to standard traffic patterns.

Background traffic is analytically modeled traffic that affects performance of explicit traffic by introducing additional delays.

#### 4.2.6 Limitations of OPNET in relation to our research

To the best of our knowledge, at the time of conducting this research, OPNET Modeler (11.5) lacked an expressive native model that can realistically model wireless sensor networks under their unique characteristics. The only model that was available at this time was the Zigbee model. The model was first of all closed hence we could not modify the source code to fit our purpose. This forced us to look into an alternative. We found the open source implementation of the OPNET IEEE 802.15.4/Zigbee model dubbed OpenZB. A brief analysis of OpenZB is highlighted in section 6.3.

Another limitation we noticed was the fact that OPNET Modeler (11.5) did not have signal processing libraries. We identified FFTW3 as an alternative software for implementing the DFT component of the protocol ( see 6.4).

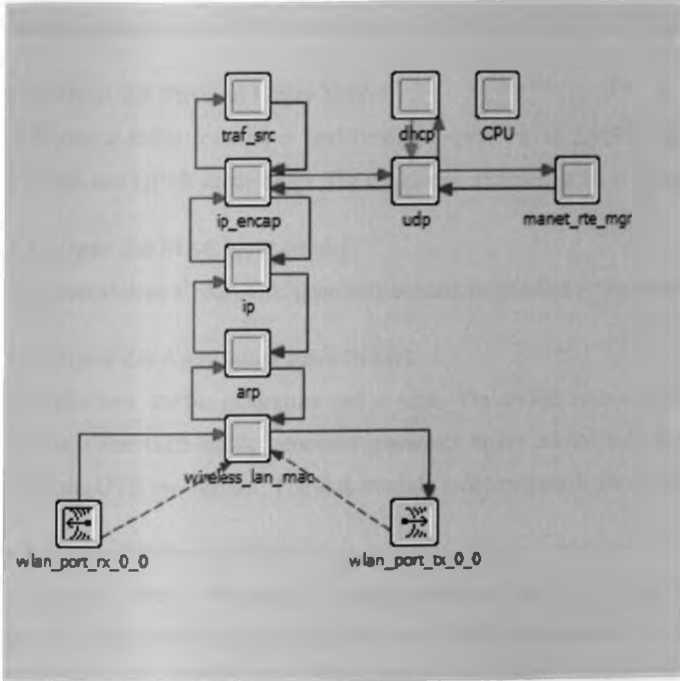


Figure 5-2: An example MANET node model in OPNET Node Editor

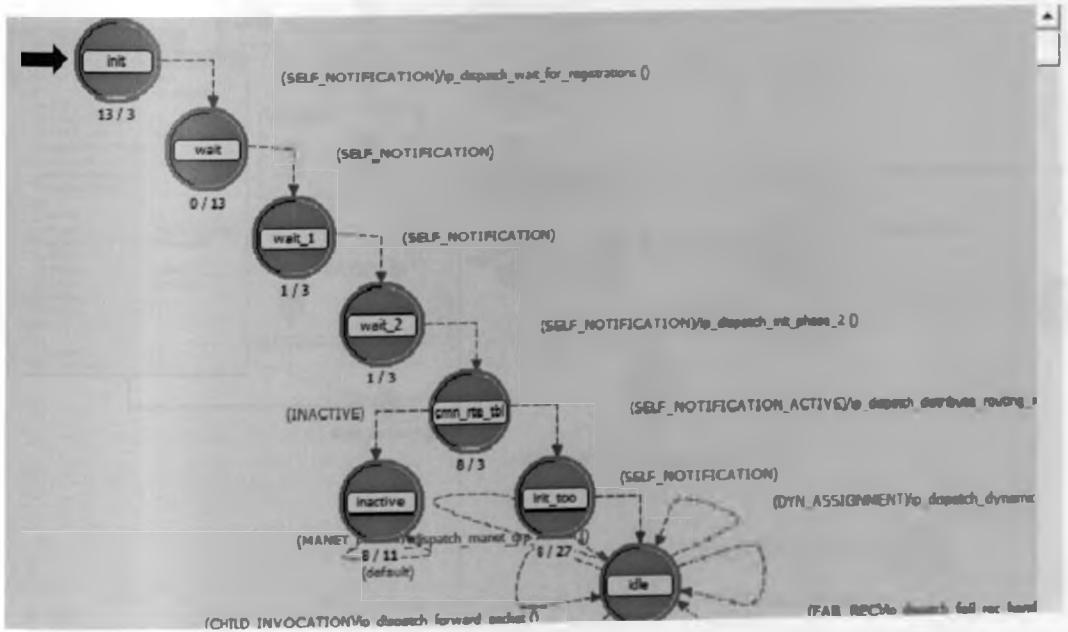


Figure 4-3: An example MANET IP Layer Process Model in OPNET Process Editor

### 4.3 THE IEEE 802.15.4 OPNET SIMULATION MODEL: OPEN ZB

Open-ZB is an open source implementation of the IEEE 802.15.4/Zigbee (Petr Jurcik and Anis Koubaa, (2007). The model implements the PHY, MAC and Application layers of the protocol stack.

#### 4.3.1 Open ZB Physical Layer Model

Contains a radio transmitter and receiver operating at 2.4GHz frequency, 2MHz bandwidth or a data rate of 250kbps and QPSK modulation. The transmission power is set to 1mW.

#### 4.3.2 Open ZB MAC layer Model

Contains slotted CSMA/CA, generates beacon frames and synchronizes nodes.

#### 4.3.3 Open ZB Application layer Model

Contains two traffic generators and a sink. The traffic source generates a sensory data using unacknowledged frames. The GTS traffic generator generates either acknowledged or unacknowledged time critical data frames using the GTS mechanism. The sink module receives frames from lower layers and performs network statistics

#### 4.3.4 Open ZB Battery Model

The battery module computes the energy consumed and the residual energy levels. Default electrical current draws are set to the micaz mote specification even though TelosB and TmoteSky motes battery models are also supported in the latest version. Figure 3.8 below shows the OPNET IEEE 802.15.4/Zigbee model implementation.

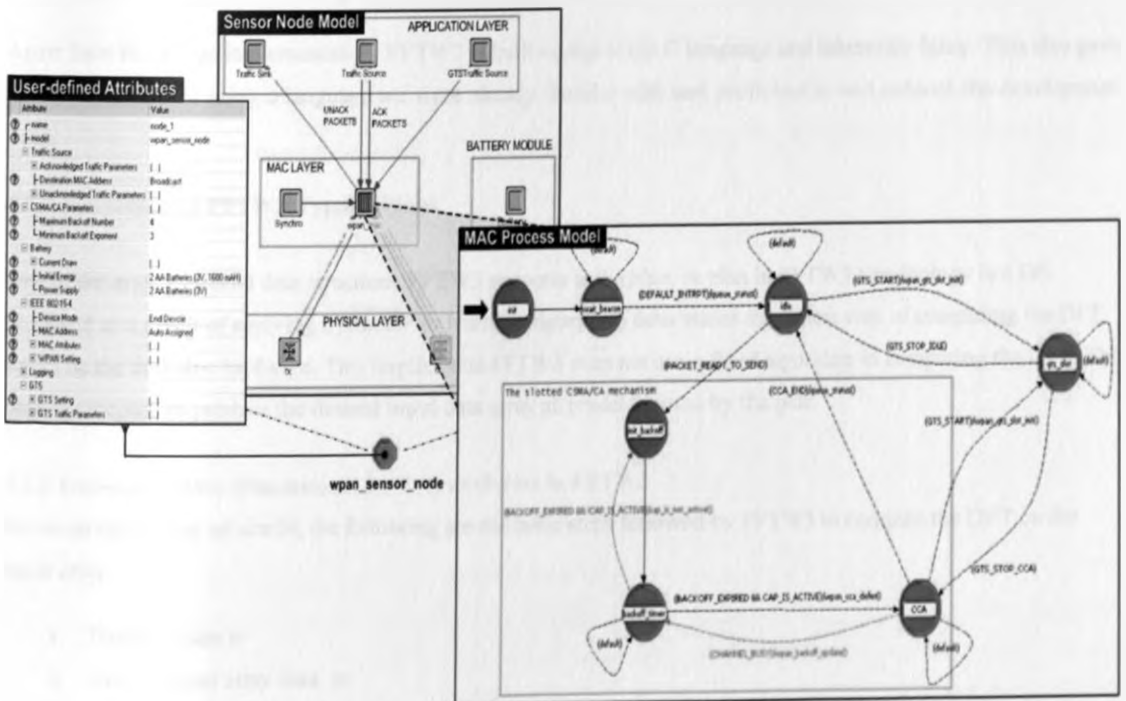


Figure 4-4: OPNET IEEE 802.15.4/Zigbee Simulation Model

## 4.5 Overview of the FFTW3 C++ Library for Computing Discrete Fourier Transforms

The FFTW3 (Frigo, M. and S. G. Johnson, 1998;1999;2005) C++ software library is the world's most known fastest open source software that implements the Fast Fourier Transform (FFT) class of algorithms. The library has more sophisticated routines that can achieve even much faster results. Particularly, FFTW3 makes use of intelligent data structures called *plans* that adaptively optimizes the C/C++ code with respect to the underlying computer architecture. We found the feature of adaptive code optimization key in a wireless sensor network scenario; sensor nodes already are power limited as well memory constrained and need resource efficient algorithm implementations at levels.

Another powerful feature of FFTW3 is its vastness in the number of ways it can compute a DFT and the variety of algorithms it has to handle different special cases that are naturally encountered in the computation of DFTs. Thus the library supports computations such as the Real to Complex (R2C) and the inverse operation Complex to Real for operations involving purely real input vectors. By making use of the Hermitian Redundancy property, FFTW3 does a very efficient implementation of the R2C transform by recognizing that in an R2C transform, the output has the  $i$ th and the  $(n-1)$  th equal but opposite in signs. Consequently, the upper half of the output is simply ignored resulting in both time and memory efficiency gains of up to two times.

Apart from its unique implementation, FFTW3 is built on top of the C language and inherently faster. This also gave us the advantage of using a language we were already familiar with and proficient in and reduced the development time.

### 4.5.1 Fundamental FFTW3 Types - Plans

One of the most powerful data structures FFTW3 supports is the *plan*. A plan in FFTW3 terminology is a DS produced as a result of applying a *planner* (a learning algorithm) determines the fastest way of computing the DFT based on the machine hardware. This implies that FFTW3 does not use a fixed algorithm in computing the DFT. The plan is executed to produce the desired input data array as predetermined by the plan.

### 4.5.2 Computing One Dimensional DFT Transforms in FFTW3

Given an input array of size  $N$ , the following are the basic steps followed by FFTW3 to compute the DFT on the input array:

1. Define a plan  $p$
2. Get the input array data  $in$
3. Apply the plan on the input data
4. Execute the plan to produce the *DFT vector out*

## 5. Release resources after the computation

The plan is defined by declaring a variable of type `fftw_plan`. The input data `in` and output complex `out` are generally declared by using the `fftw_complex` type. The size of the `in` and `out` are dynamically determined at runtime by using the inbuilt function `fftw_malloc`.

The plan is then created by applying the function `fftw_plan_dft_1d` that takes `in` and `out` as one of the inputs. To produce the DFT, the plan is executed by applying the `fftw_execute(p)`. Finally, we release system resources by calling the two functions `fftw_destroy` (for destroying a plan) and `fftw_free` ( for releasing the resources held by the output vector).

The following example directly extracted from the FFTW3 manual illustrates the basic process of computing a DFT of an arbitrary input data array.

```
#include <fftw3.h>

{
    fftw_complex *in, *out;
    fftw_plan p;
    ...
    in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
    out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
    p = fftw_plan_dft_1d(N, in, out, FFTW_FORWARD, FFTW_ESTIMATE);
    ...
    fftw_execute(p); /* repeat as needed */
    ...
    fftw_destroy_plan(p);
    fftw_free(in); fftw_free(out);
}
```

### 4.5.3 Computing One Dimensional DFTs on Real Data - Real to Complex DFT Transforms

The common assumption in computation of DFTs is that the inputs arrays and out arrays `in` and `out` respectively are complex variables. However, in most practical situations, `in` is purely real valued satisfying the "Hermitian redundancy" ( see (4.)). This property as stated earlier makes it possible to achieve two times improvement in



running time and memory usage. The DFT resulting from this special case is called the Half Complex (Hermitian) Real to Complex (R2C) Transform.

By taking advantage of the “Hermitian Redundancy”, FTTW3 implements a special case for computing the R2C transform. The routines to perform real-data transforms are almost the same as those for complex

Transforms: you allocate arrays of double and/or `fftw_complex` (preferably using `fftw_malloc` or `fftw_alloc_complex`), create an `fftw_plan`, execute it as many times as you want with `fftw_execute(plan)`, and clean up with `fftw_destroy_plan(plan)` (and `fftw_free`). The only differences are that the input (or output) is of type double and there are new routines to create the plan. In one dimension: `fftw_plan fftw_plan_dft_r2c_1d` (int n, double \*in, `fftw_complex` \*out, unsigned flags); `fftw_plan fftw_plan_dft_c2r_1d` (int n, `fftw_complex` \*in, double \*out, unsigned flags);

In this work, we have introduced the very basic capabilities of FTTW3. Our main interest was the special case for computing DFTs of one dimensional real valued data. The package has myriad more routines that can be applied in different scenarios depending on the problem being tackled. Those interested in just quickly experimenting with the library can find excellent resource material at the FFTW website. A more formal treatment of FFTW3 can be found in two separate papers (Frigo and Johnson, 1999) and (Frigo and Johnson, 2005).

#### 4.6 Overall System Architecture and Design

System architecture is the set of all attributes and properties of the system visible to the user. The architecture consists a specification of the individual components, their behaviour and interaction. In this section, we describe the system architecture within which the distributed energy control model is implemented. The following architectural components were identified during the development of the system:

1. Adaptive sampler module
2. The process module or signal generator module.
3. A sensory module
4. Node module
5. The network module.
6. A battery and power module
7. The communication protocol stack module

The above modules were designed directly based on the system model that was established in chapter 4. The components are described as below.

##### **Physical process/signal generator**

This component of the system is responsible for generating random signals. It models the environment in which sensors are deployed to collect data. The implementation details are discussed in section 6.7

##### **Adaptive Sampler**

This is core of the overall system. The sampler controls the rate at which each sensor collects data from its local environment. It's also responsible for scheduling the times at which a node goes into sleep. It applies the estimation and change detection models derived in chapter 4. The input to the sampler is the data sequences generated from the signal generator module and the output is the sample rate and sleep duration for the sensor.

### **Sensory Module**

The component is responsible for "sensing" the environment by observing and collecting the signals generated by the signal generator. In the simulated scenario, the main inputs to the sensory module are the sampling rate (generated by the adaptive sampler), the sleep time (generated by the sampler). Other inputs to the sensory module include the start time (the time when sampling starts) and the end time (the time when sampling ends).

### **Battery Module**

This component is responsive for powering all the physical components and also tracking energy used by the devices. The battery design is dictated by the OpenZB model.

### **Node Module**

The component represents the sensor node. The actual design is bound by the OpenZB model.

### **Sink Module**

The component that collects network wide statistics such as the traffic (data) generated/sent by individual nodes. The design is restricted by the OpenZB model.

### **Network Module**

It is the logical ensemble of all components (nodes) interconnected via wireless channel. The network architecture is restricted to the OPNET modeling process as described in the section 6.2.

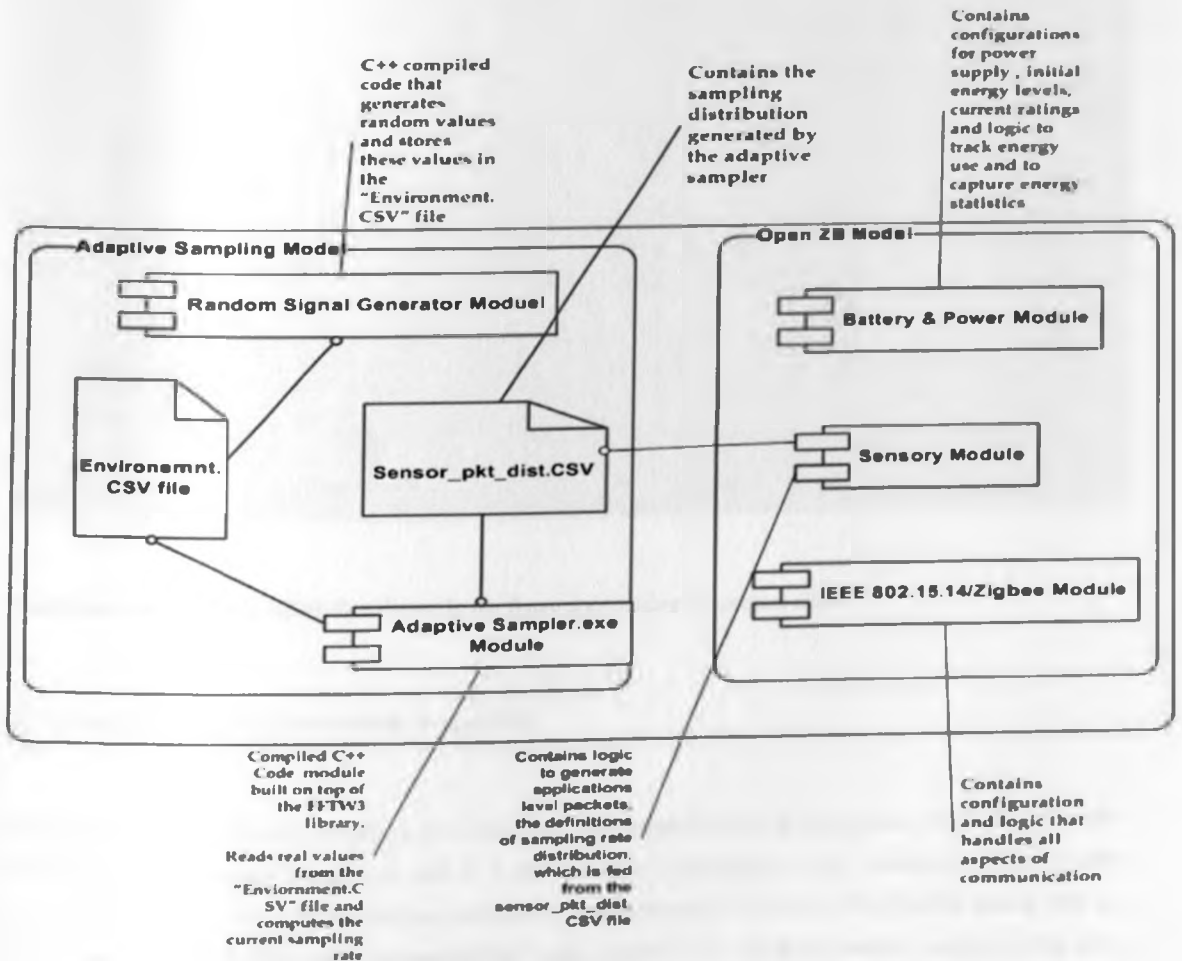
The system context diagram is shown in the figure below. For simplicity and without loss of generality, the system context diagram shows the various modules at a node level only. The architecture is the same for all nodes in the network.

The interaction and behaviour of the components in the node module is as follows:

1. Each node is connected to a distinct signal generator. The reason being each signal generator represents the environment local to a node.
2. The signal generator emits data sequences (that are fed into a file).
3. The adaptive sampler reads the emitted data sequences (from the file) and applies the estimation and change detection model to the data sequence. The output of the computation is the sample rate. The value of the sampling rate is fed into the sampling distribution file.
4. The sensory module constantly reads the sampling distribution file and adjusts the current rate at which generation of data is done.
5. At the beginning of sensing, the node sets a timer that advances as sensing proceeds. Every time the sampler detects a change in the environment, the current clock read and stored in some variable  $t$ . The node

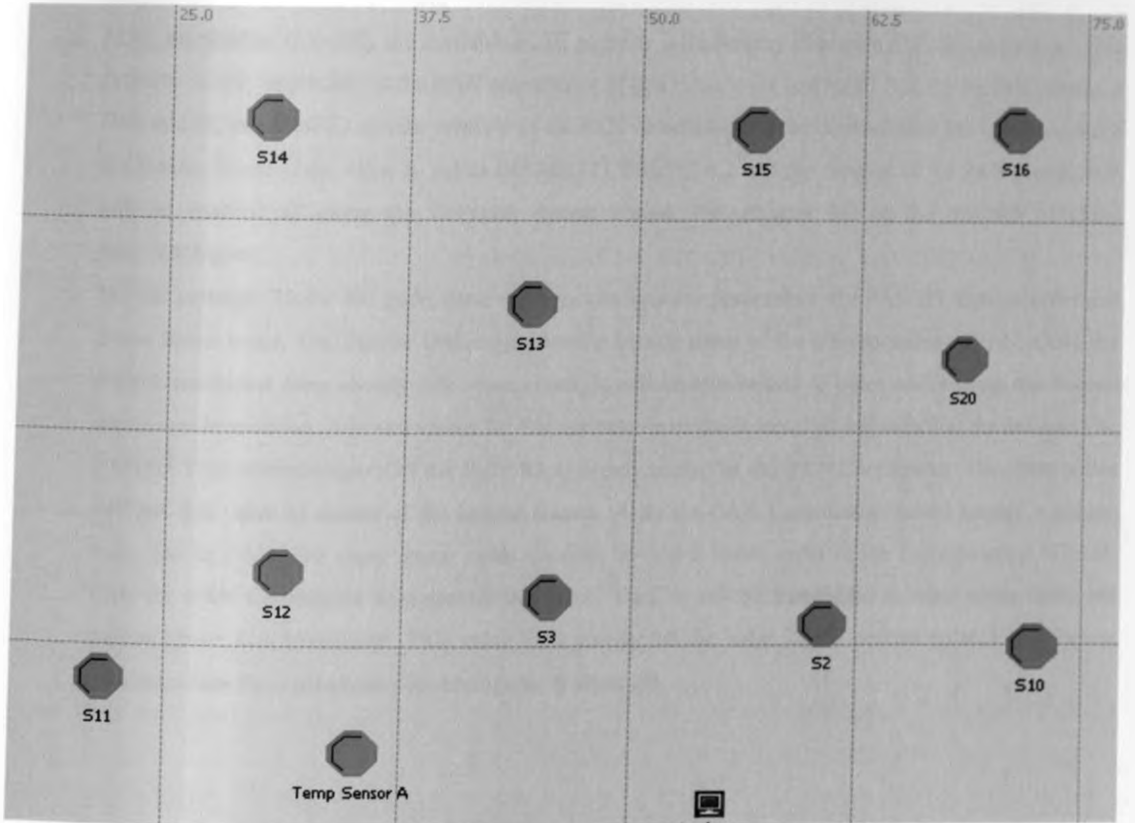
goes to sleep for  $t$  seconds in the current cycle. On waking up, the node resets  $t$  to 0 and the process continues until the end of sensing.

6. As sensing proceeds, the battery module is keeping track of the so far spent by the node as well as the residual energy.
7. The sink module simultaneously keeps track of the global network wide statistics such as the packet success probability.



## 6.7 System Configuration

### 6.7.1 Configuration of the Network Model in OpenZB



Nodes were configured in OPNET as shown in the figure above using the project editor.

### 4.7.2 Configuration of the Node Model in Open ZB

There are two types of nodes models in our work. The first category is the sensing nodes, those that sense and forward data. The second category is that of a data relay or a coordinator. The coordinator controls when transmissions for each of the nodes start and delivers the data to the sink. In OPNET WPAN IEEE model, both the sensing nodes and coordinator nodes are modeled as "wpan\_sensor\_node" using the "model" property of the node. To specify whether a sensor node is "a sensing node" (End Device) or a coordinator, we use the *IEEE 802.15.4* property in the properties window of the node. We expand this property and select the sub property labeled "Device Mode". By default the device type property of a wpan\_sensor\_node is set to "slave". A slave is a non-coordinator – also called an end device in IEEE 802.15.4/Zigbee terminology. The property can be changed to a "PAN coordinator" to specify that the sensor node is a coordinator. Other important properties of a sensor node include:

- 1) **The MAC address of a sensor node:** The unique MAC address for each sensor node. The default setting is 'auto assigned'. There is a possibility of manually editing the MAC address. We choose to use the default value of this property for our work.
- 2) **MAC attributes:** Currently the available MAC property is the Battery Extension (BE) life extension. This property is only applicable to the PAN coordinator. If this value is set to ENABLED, the backoff exponent  $BE = \min(2, \text{macMinBE})$  and the receiver of the PAN coordinator will be disabled after the transmission of the beacon frame. if the value is set to DISABLED, then  $BE = 2$  and the receiver of the PAN coordinator will be enabled all along the Collision Access Period. Ref. chapter 2.0 on the standard of IEEE 802.15.4/Zigbee.
- 3) **WPAN settings:** Under this node, there are three configurable parameters- the PAN ID, Beacon order and Super frame order. The Beacon Order specifies the beacon order of the corresponding WPAN. Only the PAN Coordinator must specify this value. Then, it will be transmitted to other nodes using the beacon frame synchronization. Allowed values for this attribute lie on the interval  $[0, 15]$  with 5 as the default. The PAN ID This attributes specifies the PAN ID. It is only useful for the PAN Coordinator. The other nodes will get this value by means of the beacon frames. Only the PAN Coordinator should specify a unique value for its PAN. The super frame order specifies the super frame order of the corresponding WPAN. Only the PAN Coordinator must specify this value. Then, it will be transmitted to other nodes using the beacon frame synchronization. This value must not exceed the value of the beacon order. Fig || below illustrates how these parameters are configured in Open ZB.

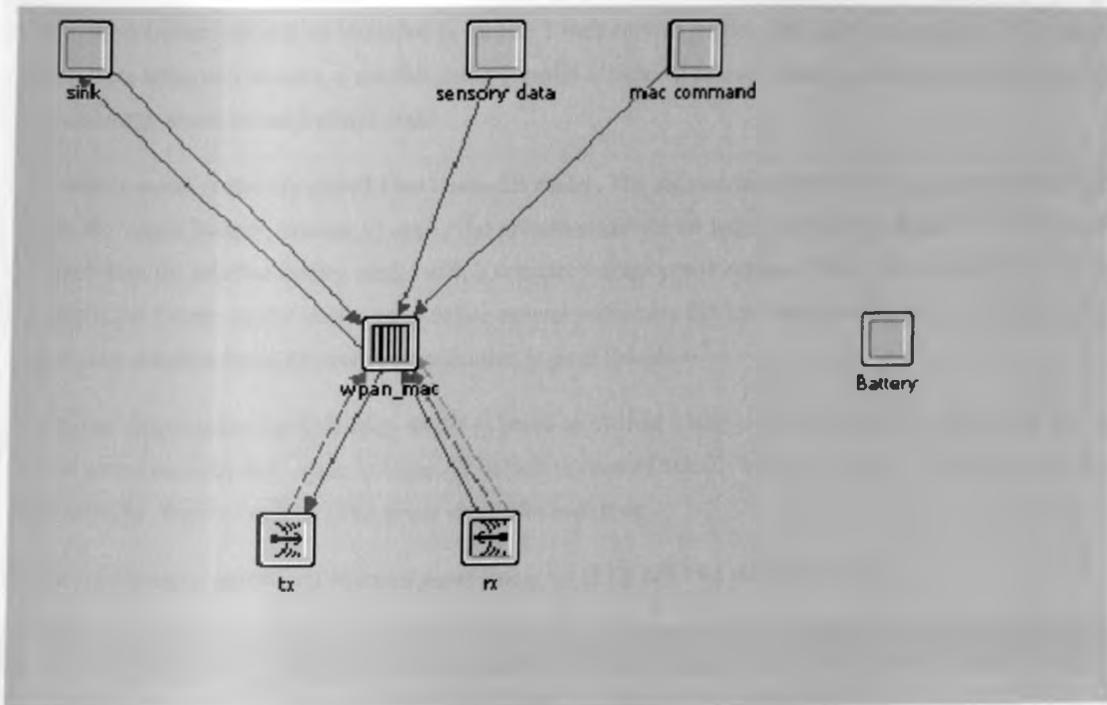


Figure 4-5 : Sensor Node Model in IEEE 802.15.4/Zigbee OPNET model

Each node model, has a source module, a sink module, a MAC command, a receiver module and a transmitter

#### IEEE 802.15.4

③	- Device Mode	Slave
③	- MAC Address	Auto Assigned
③	☐ MAC Attributes	(..)
③	└ Batterie Life Extension	disabled
③	☐ WPAN Settings	(..)
③	- Beacon Order	5
③	- Superframe Order	1
③	└ PAN ID	1

Figure 6-6: Graphical User Interface for Configure IEEE 802.15.4 settings in OPNET/OpenZB

#### 6.7.3 Configuration of the Battery Model in Open ZB

Two of the important metrics we identified in chapter 3 were network lifetime and energy consumption. To measure and evaluate these two metrics, a credible battery model is required to track statistics about the energy usage and residual energy levels for each sensor node.

The battery model is directly picked from Open-ZB model. The code model of the battery model is available in the source file "wpan\_battery\_process\_v1.cpp". The process model for the battery is shown in figure. The batteries are modeled from the alkaline battery model with a constant leakage power equal to 10% of the full power in a year. The Open ZB battery model allows us to define several parameters that are configurable with a set of permissible values. The complete Open ZB battery specification is given in table 6-2.

The power consumption for the battery model is based on the real widely used sensor nodes – MicaZ and TelosB. All the power consumption values in Open ZB default to those of MicaZ. We set our power consumption values to these defaults. We choose 2AA (3V) power supply for each node.

Table 6-2 : Battery and Power Model Specifications in IEEE 802.15.4 OPNET Model

Parameter	Measure	Description
Power supply	Volts(V)	The battery voltage in Volts. There is 1 predefined option: <input type="checkbox"/> <b>2 AA Batteries (3V) = 3 Volts</b> (each battery with voltage 1.5V)
Initial Energy	Joules (j)	The initial amount of battery energy before any activity. The relation between battery voltage (U[V]), battery capacity (C[mAh]) and energy (E[Joule]): $E = U[V] \times C[mAh] = [Watt \times sec] = [Joule]$ There are 3 predefined types of batteries with the following initial energies: <input type="checkbox"/> <b>2 AA Batteries (3.0V, 2300 mAh) = 49 680 Joule</b> <input type="checkbox"/> <b>2 AA Batteries (3.0V, 1600 mAh) = 34 560 Joule</b> <input type="checkbox"/> <b>2 AA Batteries (3.0V, 1200 mAh) = 25 920 Joule</b>
TR Mode	Mill amperes	Represents the current draw of the device in receive mode (in milli-Ampere (mA)). There are 2 predefined options default for MICAz and TelosB nodes: <b>MICAz = 27.7 mA</b> <b>TelosB = 24.8 mA</b>
TX Mode	Mill amperes	The current draw of the device in transmission mode (in milli-Ampere (mA)). The default MicaZ value at 0 dB is used. The current draw of the device when the transceiver is in the transmitting Mode (TX_ON state). There are 6 predefined options corresponding to the nominal transmitting power of the transceiver: <b>MICAz (0 dBm) = TelosB (0 dBm) = 17.4 mA</b>

**MICAz (-5 dBm) = TelosB (-5 dBm) = 14 mA**  
**MICAz (-10 dBm) = TelosB (-10 dBm) = 11 mA**

Idle Mode	Current	The current draw of the device when the transceiver is in idle mode and voltage regulator is on. There are 2 predefined options default for MICAz and TelosB motes: <b>MICAz = 35 <math>\mu</math>A</b> <input type="checkbox"/> <b>TelosB = 26.1 <math>\mu</math>A</b>
Sleep Mode	Current	The current draw of the device when the transceiver is inactive and voltage regulator is off. There are 2 predefined options default for MICAz and TelosB motes: <b>MICAz = 16 <math>\mu</math>A</b> <input type="checkbox"/> <b>TelosB = 6.1 <math>\mu</math>A</b>

To compute the consumed and residual energy for the node, the battery process first initializes the battery by loading the initial battery settings defined – the initial energy, power supply, the power consumption in all the available modes and so on. The initialization is done in the static function "*wpan\_battery\_init()*". During the simulation process, a battery update function called "*wpan\_battery\_update()*" is invoked to update the current values of energy consumption and consequently the remaining energy. The following parameters are considered by the battery update function:

1. Transmission time, *tx\_time*
2. Reception time, *rx\_time*.
3. Data rate, *wpan\_data\_rate*
4. Idle time, *idle\_duration*
5. Packet size
6. No of packets sent/received.

To configure the battery in OPEN ZB in OPNET 11.5, we right click the sensor node of interest in the project's editor and select "advanced edit attributes". We then expand the "Batteries" node and select or define appropriate values for current draw, initial energy and power supply. The figure below illustrates how the battery parameters were configured in OPNET 11.5



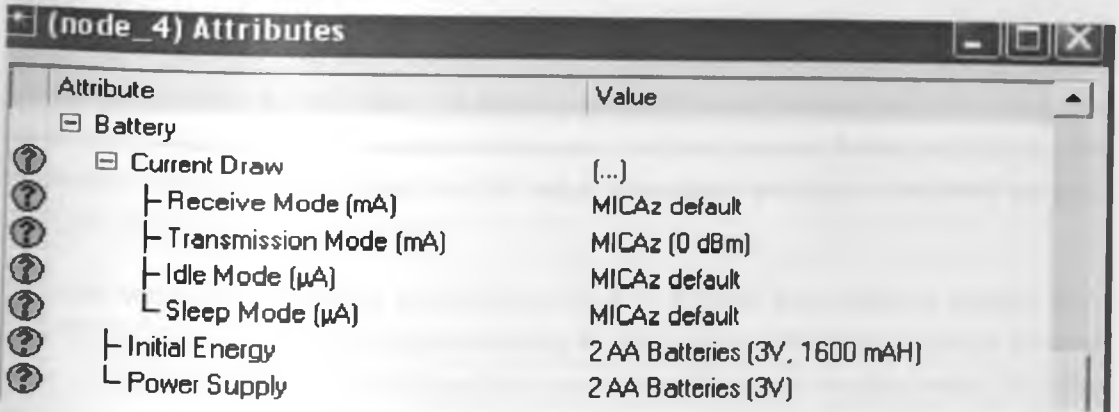


Figure 4-4: Graphical User Interface for Configuration Battery and Power model in IEEE 802.15.4/Zigbee OPNET model

#### 4.7.5 Configuration of Sensory Model in Open ZB

The sensory traffic model in OpenZB is completely specified by the attributes in table 6-1 below

Table 6-1: Specifying Sensory Traffic Parameters in OpenZB

Attribute	Measure	Description
MSDU/Packet Interarrival	Probability Distribution	The inter-arrival time in seconds between two consecutive MSDUs/Packets.
MSDU/Packet Size	PDF [bits]	The size in bits of the generated MSDU/Packet. The value can be from 0 to 1016 bits.
Start Time	[sec]	The absolute simulation time in seconds when the traffic source will <u>start</u> its data generation. Setting the value to <i>Infinity</i> will simply disable the source.
Stop Time	[sec]	The absolute simulation time in seconds when the traffic source will <u>stop</u> its traffic generation. Setting the value to <i>Infinity</i> will make the traffic source generate data until the end of the simulation.

In OPNET 11.5, using the WPAN IEEE 802.15.4 version 1.0 model, traffic source is configured by right clicking the node of interest, going to the advanced edit properties and choosing the node named "traffic source". By

expanding this node, we get the child node named "Sensor y Data". Important parameters to configure here are the "packet inter arrival time" and packet size. Each of these two parameters can be defined by selecting appropriate PDFs that are given from the list of PDFs. The packet arrival time PDF define the arrival process of the data packets -i.e the time interval between two consecutive data packets. A data packet at the application layer is called a MAC Service Data Units (MSDU). The packet size PDF defines the probability distribution of the MSDU size in bits bounded by 1016 bits as the maximum size.

If the application under consideration does not follow any of the PDFs, the model allows one to specify a user defined distribution by supplying the file name containing the desired outcomes (for the arrival process). Our arrival process is determined both by the environment being monitored and the adaptive sampling process. The desired outcomes in our case are the interarrival times between two consecutive MSDUs. We specify the interarrival time by imputing the file name "traffic\_arrival\_process\_ni". During simulation, the value of T in the traffic\_arrival\_process\_ni file changes as changes are detected in the "environment\_process.csv" file. A separate file called "traffic\_arrival\_process\_successive" is maintained to keep track of the sampling frequencies at every instant. The sensory data module of the node model is then used to generate sensory data using unacknowledged MSDUs at a rate determined by the sampling process. Figure [] shows the complete Open ZB version 1.0 traffic source specification. Figure [] below shows how the sensory data traffic source is configured using the Open ZB WPAN IEEE 802.15.4 version 1.0 model in OPNET 11.5

The process model of the sensory data module is illustrated as a FSM in figure []. The FSM is simple in nature. At any time, the generator can be in "init state", "generate state" or "stop state". When the "START time" attributes is set to infinity, and then this particular does not generate any data packets. The generator starts in the init state to initialize basic settings for the traffic source. The parameters include the start time, stop time, packet size and packet interarrival time. If the start time is set to "infinity" then the Sensory data subsystem transitions from the init state straight to stop state because source traffic generation is disabled. If the start attribute is set to a numerical value, then the system calls the `ss_packet_generate()` function and it immediately enters the "generate state". The looping arrow on generate state indicates that the process will remain in this state while generating MSDUs until the stop time is reached, at which point it enters the stop state.

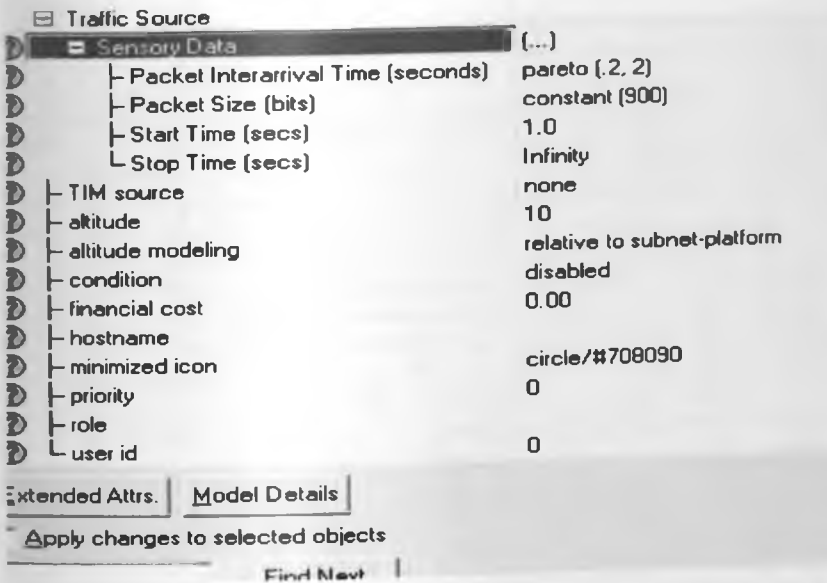


Figure 4-1 : Graphical user interface for configuring sensor source traffic in OPenZB

Figure 6-1: Sensor traffic source in Open ZB IEEE 802.15.4/Zigbee model in OPNET 11.5

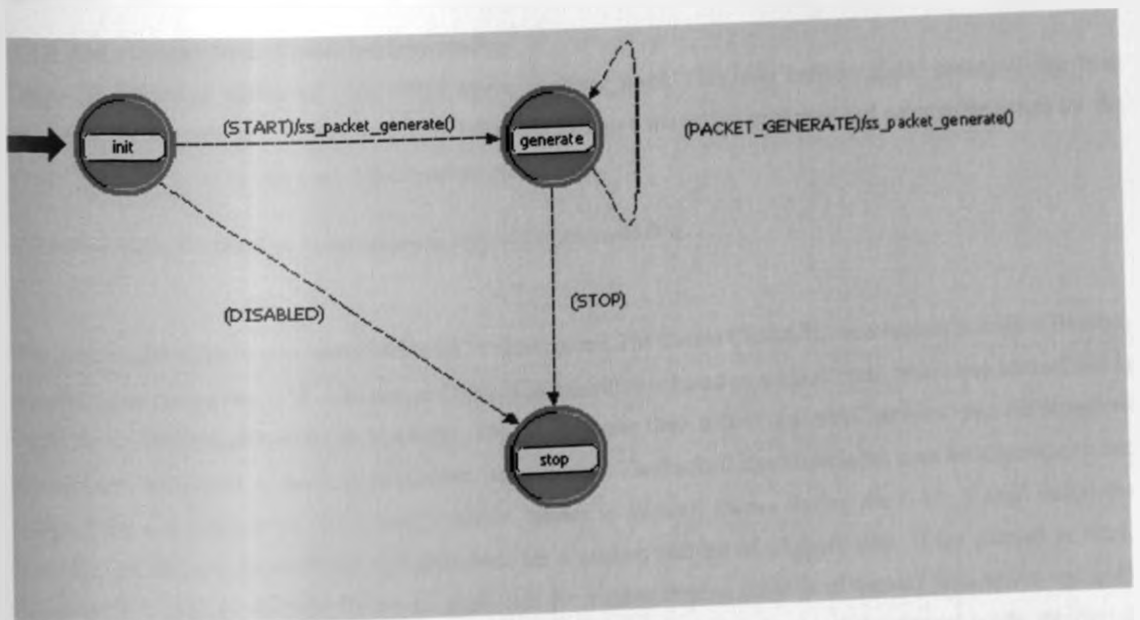


Figure 6-2: Sensor Battery Finite State Machine in IEEE 802.15.4/Zigbee OPNET Model

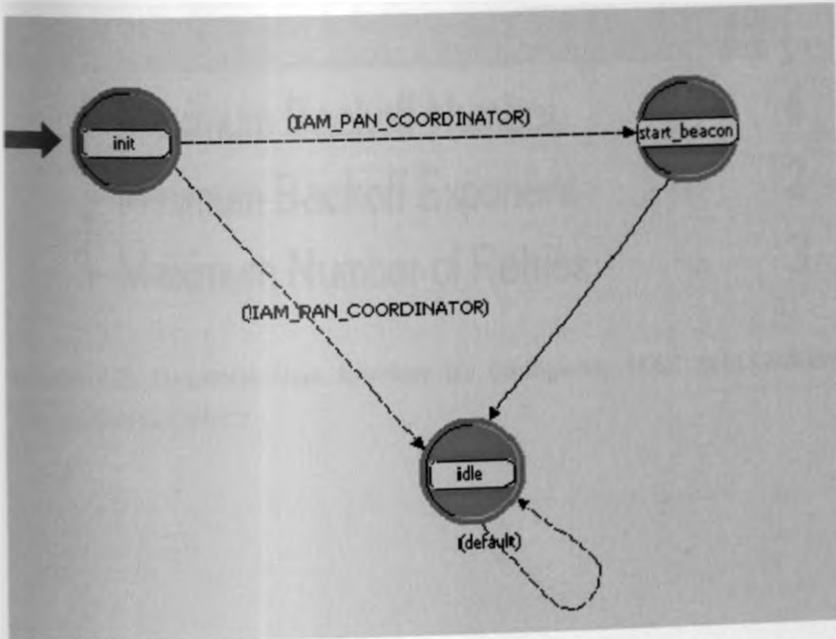


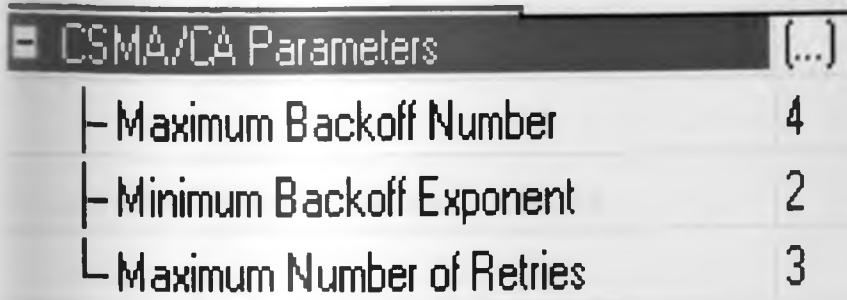
Figure 4-3: Sink Module Finite State Machine in IEEE 802.15.4/Zigbee model in OPNET

#### 4.3.8 The Analyzer Node Model Implementation

Open ZB defines an additional node called **wpan\_analyzer\_node**. This node collects global statistical data from whole PAN (only one node within PAN). The diagram in figure shows the attributes and permissible values for the **Wpan\_analyzer\_node**. In this work default values are adopted.

#### 4.3.9 The MAC CSMA/CA Programming Model Implementation

The parameters related to the slotted CSMA/CA mechanism. The slotted CSMA/CA mechanism is used in Beacon-enabled mode during the CAP. The slotted CSMA/CA algorithm is based on backoff slots, where one backoff slot is equal to a *UnitBackoffPeriod* (20 Symbols). This is the basic time unit of the MAC protocol and the access to channel can only occur at the boundary of the backoff slots. The backoff slot boundaries must be aligned with the super frame slot boundaries. Each time a device wishes to transmit frames during the CAP, it shall locate the boundary of the next backoff slot and then wait for a random number of backoff slots. If the channel is busy, following this random backoff, the device shall wait for another random number of backoff slots before trying to access the channel again (this can be repeated *Maximum Backoff Number* times). If the channel is idle, the device can begin transmitting on the next available backoff slot boundary. CSMA/CA settings were configured according to figure [].



The image shows a graphical user interface window titled "CSMA/CA Parameters (...)". The window contains three rows of configuration parameters, each with a vertical line on the left side indicating a list or tree structure. The parameters and their values are:

Parameter	Value
Maximum Backoff Number	4
Minimum Backoff Exponent	2
Maximum Number of Retries	3

Figure 4-7: Graphical User Interface for Configuring IEEE 802.15.4/Zigbee CSMA/CA MAC Layer Parameters in OPNET

## CHAPTER 5: RESULTS AND DISCUSSIONS

### 5.1 Introduction

In order to evaluate the performance of the adaptive sampling model centered around energy consumption and quality of information, we set forth the following evaluation specific objectives:

1. To compare the average energy consumption (in Joules) per sensor node when using adaptive sampling and fixed sampling.
2. To compare the average energy consumption (in Joules) across the network when using adaptive sampling and fixed sampling from 1 above.
3. To compare the number packets/messages sent per second across the network when using adaptive sampling and fixed sampling.
4. To compare the average EED (in seconds) per sensor node when using the adaptive sampling and fixed sampling.
5. To compare the average EED (in seconds) across the network when using the adaptive sampling and fixed sampling.
6. To compare the packet success probability (percentage) when using adaptive sampling and fixed sampling.
7. To compare the packet interarrival times when using adaptive sampling and fixed sampling

### 5.2 Metrics

#### 5.2.1 Energy Consumption and Network Lifetime

##### 1.) Rate of Energy consumption per Node, $E_{avg}$

The absolute mean of value of energy consumed by an individual node in a unit time (Joules/Minute). The values for average energy consumed per node can be directly obtained from the OPNET simulator.

**2.) Rate of Energy consumption in the network,  $E_{avg}$**

The absolute mean value of energy consumed per unit time in the entire sensor network (Joules/Minute).

$$E_{avg} = \sum_{j=0}^S E_{navg} \tag{6.1}$$

Where: S is the number of nodes in the sensor network.

**3.) Message Reduction Percentage (unit less),  $MRP_{avg}$**

A measure of how many fewer messages or more messages were generated/sent in the network when using the Adaptive Sampling algorithm compared to the fixed sampling strategy. A positive MRP would indicate, a reduction in the total number of messages sent across the network when using adaptive sampling and a negative value means that the adaptive sampling algorithm had more network traffic (and therefore performed worse) compared to the fixed sampling.

Let,  $MSDU_{javg}$ ,  $MSDU_{navg}$  be mean rate of sensor traffic generated by sensor j and the mean rate of packets generated in the entire network respectively.

$$MSDU_{navg} = \sum_{j=0}^S MSDU_{javg} \tag{6.2a}$$

If  $MSDU_{navg(a)}$  and  $MSDU_{navg(f)}$  is the rate of sensor traffic generated when using adaptive sampling and fixed sampling in other order, then:

$$MRP_{avg} = \left\{ \frac{MSDU_{navg(f)} - MSDU_{navg(a)}}{MSDU_{navg(f)}} \right\} \times 100 \tag{6.2b}$$

A positive  $MRP_{avg}$  indicates a reduction in number of messages communicated over the network when using adaptive sampling compared to fixed sampling. 0 values indicate no change in the rate of sensor traffic when using either of the sampling strategies while a negative value indicates that the fixed sampling strategy has a lower sensor traffic load.

**4.) Energy Gain (percentage),  $E_g$**

A measure that was used to compare the relative percentage increase in energy consumption of the entire network under adaptive sampling when compared to the fixed sampling strategy. This means that a negative value for the metric indicates a saving in energy consumption. A positive value indicates the opposite while a 0 value indicates no change in energy consumption when the two algorithms are used.

Let  $E_{avg(a)}$  and  $E_{avg(f)}$  be the mean values for rates of energy consumption in adaptive sampling and fixed sampling respectively:

$$E_g = \left\{ \frac{E_{avg}(a) - E_{avg}(f)}{E_{avg}(f)} \right\} \times 100 \quad (6.3)$$

### 5.) Network Lifetime Gain, $L_g$

The definition of network lifetime in the context of wireless sensor networks takes various forms depending on application nature, goals of performance evaluation etc. For instance ,

A measure of the percentage gain in lifetime of the entire sensor network when using adaptive sampling compared to fixed sampling. Since network lifetime is proportional to the amount of energy saved,  $L_g$  is the same as the absolute value of  $E_g$  , for instance. say  $E_g$  -20% then  $E_g$  is 20%

## 52.2 Quality of Information (Qol)

Two basic measures of Qol identified in chapter 4 will be used: Timeliness and completeness. We choose to measure timeliness using end to end delay whereas for completeness we use packet success percentage.

### 6.) End to End delay in seconds, $D_{avg}$ .

The mean end to end delay experienced by an individual sensor node.

The values for this metric can be directly obtained from the OPNET simulator.

### 7.) End to End delay in seconds, $D_{avg}$ .

The average end to end delay experienced by the entire sensor network.

Therefore  $D_{avg} = \frac{\sum_1^n D_{avg}}{S}$  , where S is the size of the network.

### 8.) Packet Success Probability (ratio/percentage), $PSP_{avg}$

The number of messages actually delivered to the sink as a percentage of the number of messages originated generated at the sensor source. The values for this metric are directly obtained from the simulator.

### 9.) Quality of Information , $QoI_{avg}$

We define,  $QoI_{avg}$ , in terms of Packet Success Probability and End to End delay in our work because the two metrics both affect the quality of information in different ways. Qol has a direct correlation with the PSP in that a higher value for PSP indicates that fewer messages are dropped enroute to the sink and hence higher accuracy of information. On the other hand, the higher the delay the smaller the Qol for time critical applications likes sensor nodes. Therefore we formally define this metric as follows:



$$Qolsavg = \frac{PSPavg}{Dsvg}$$

### 10.) Quality of Information Gain, $Qolg$ .

We defined the metric  $Qolg$  in order to compare the performance adaptive sampling algorithm with the fixed sampling in terms of  $Qolsavg$  defined as follows:  $Qolg = \frac{Qolsavg \text{ in AS} - Qolsavg \text{ in FS}}{Qolsavg \text{ in FS}}$ , where AS stands for adaptive sampling and FS is fixed sampling. A positive value indicates that the adaptive sampling algorithm performed better in terms of both PSP and end to end delay, a negative value indicates the reverse.

## 5.3 Experimental Parameters

### 5.3.1 Sensory Traffic Parameters

Table 5-1: Simulation Parameters – Sensory Traffic Settings

S/No	Parameter	Value
1	Sensing Start Time	1
	Sensing Stop Time	Infinity
2	MSDU/Package Size	900 (Constant)
3	MSDU/Package Interarrival time	Determined by the Adaptive Sampling Algorithm
4	Waiting Window period	Random Distribution

### 5.3.2 Battery and Power Configurations

Table 5-2: Simulation Parameters – Node Battery and Power Settings

S No	Parameter	Value
1	Initial Energy	200 Joules
	Power Supply Voltage	2AA battery rated 1.5 Volts (3Volts)
2	TX Mode current rating	MicaZ default ( 17.4 mA at 0dB)
	TR Mode current rating	MicaZ (19.7mA)
3	Sleep Mode Current rating	MicaZ default

		(16mA)
4	Idle Mode Current rating	MicaZ default (35 Microamps)

### 5.3.3 Network Configurations

**Table 5-3: Simulation Parameters – Network Settings**

S/No	Parameter	Value
1	Number of End Devices	9
2	Number of PAN coordinators	2
3	Number of WPAN Analyzers	1
4	Simulation Area	87m × 87m grid

### 5.3.4 Simulation Run Configurations

**Table 5-4: Simulation Run settings**

S/No	Parameter	Value
1	<b>Simulation Time</b>	<b>3600 Sim seconds</b>
2	<b>Number of runs</b>	<b>3</b>

## 5.4 Simulation Scenarios.

### 5.4.1 A Distributed Adaptive sampling

Simulations were carried out in OPNET Modeler version 11.5 using the WPAN IEEE 802.15.4/Zigbee model (Open ZB) version 1.0 to demonstrate:

- a. The reduced energy consumption due to due adaptive sampling as determined by energy consumed and traffic generated/sent rate.
- b. improved quality of information (QoI) as determined by the ratio of the PSP to delay

The simulation scenario consisted of 11 nodes in a typical office set up randomly distributed in a 87m x 87m square grid as shown in figure 6-1. The wireless sensor network environment R had 9 WPAN end devices and 2 PAN coordinators. All the end devices were assumed to be temperature sensors. Each sensor had an initial energy of 200Joules and powered by two alkaline (2AA) batteries each rated at 1.5V. A power level in each of the five modes was set to the MicaZ default value. With the exception of a PAN coordinator, each device started sensing its local environment at the 1<sup>st</sup> second and stopped at the end of the simulation (indicated by the infinity value against the start time (see table 6-1). A PAN coordinator is not allowed to sense the environment and so the sensing start time is set to infinity.. Each PAN coordinator was responsible for controlling communication of the end devices by assigning transmission slots using the TDMA. Temperature variations within region R were assumed to be spatially correlated and to vary at different times according to a uniform distribution. Random values for temperature were continuously generated during the simulation run and stored in an external file *Aj*. The adaptive sampler continually read the sequence of data from the file, with initial number of samples set at 200 and the sample spacing at 10, computed the appropriate sampling rates, sampling periods (time lapse between two consecutive packet arrivals) and stored the results in a second output file *B*. The values generated in *B* are then read by the respective sensor. The sensor then generated unacknowledged application packets interval at an interval specified by the sampling rate probability distribution *B*. During every node's CAP period, the node sent the generated data to the coordinator and the data is finally relayed to the sink. The simulation was run for a period of 3600 Sim seconds. The experiment was repeated 2 other times under the same conditions with different random seeds during each run. Average values for energy consumed, end to end delay, traffic generated per second and packet success probability were collected.

The experiment was repeated with different values of *W* and *H* in order to determine the ideal respective values. The *W*, *H* pairs were randomly chosen and the combination *W*=200, *H*=10 yielded the best results in the preliminary experiments. After fixing *W* and *H*, the experiment was repeated 3 times with each run having a different random seed and lasting for 1 hour (3600 seconds).

#### 5.4.2 Fixed sampling

The experiment outlined in section 7.4.1 was repeated with all conditions remaining the same with only one exception – the sampling rate was estimated only once based on the first *W* samples yielding a *B* file with a single value only throughout the simulation. It's obvious that in this case, *B* is a constant distribution and therefore each sensor node generated data at a fixed rate.

### 5.3 Results and Discussions

#### 5.3.1 Effect of Adaptive Sampling and Fixed Sampling on Energy Consumption

Table 7-5a and table 7-5b show the rates of energy consumption per node under fixed sampling and adaptive sampling respectively. It's quick to see that the average energy consumed per node per minute for any of the nodes in the network is higher in fixed sampling compared to adaptive sampling by about 130% - see table 7-7a on the summary statistics and figure 7-1 for a graphical view of the results. The maximum indicates the peak energy consumption at a point in time. Figures 7-1b-I and 7-1b-II show the cumulative energy consumption of individual

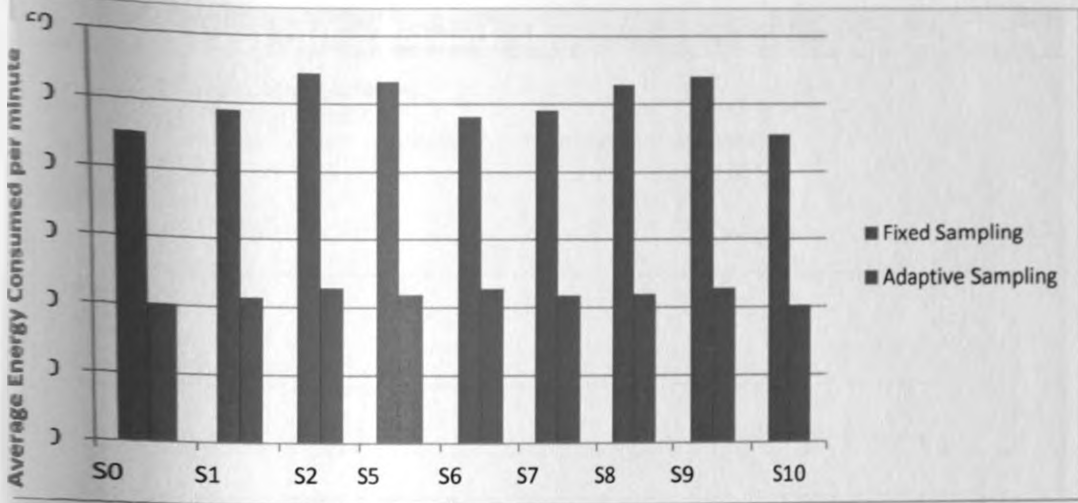
nodes over time. We selected node S5 and node S7 for the purposes of explanation. We see that within the first few minutes, the energy consumption under fixed sampling and adaptive sampling is nearly the same. This is because within this short period of time, changes in the environment are not very significant and therefore the sampling rates of individual nodes under adaptive sampling remain fairly constant. Beyond this time, energy consumption rate per node under the two scenarios start becoming significant and the gap between two scenarios grows larger within time. We attribute this to the increased randomness in the local environment that triggers the nodes under adaptive sampling to adjust their sampling rates and hence their sleep times accordingly minimizing redundant data collection and sensing.

**Table 5.5a: Results - Rate of Energy consumption (Joules per minute) for Fixed Sampling**

Rank	Node ID	Minimum	Average	Maximum	Std Dev
9	S0	0	45.241	90.53	26.135
6	S1	0	48.951	97.94	28.277
1	S2	0	54.459	108.96	31.457
3	S5	0	53.31	106.66	30.791
7	S6	0	48.131	96.29	27.799
5	S7	0	49.071	98.18	28.345
4	S8	0	52.866	105.78	30.539
2	S9	0	54.03	108.1	31.209
8	S10	0	45.518	91.07	26.292

**Table 5-5b: Results - Rate of Energy Consumption (Joules per minute) - Adaptive Sampling**

Rank	Node ID	Minimum	Average	Maximum	Std Dev
9	S0	0	20.201	40.401	11.665
7	S1	0	21.419	42.839	12.369
1	S2	0	22.833	45.668	13.185
5	S5	0	21.883	43.767	12.636
3	S6	0	22.765	45.532	13.146
6	S7	0	21.813	43.624	12.595
4	S8	0	21.968	43.936	12.686
2	S9	0	22.821	45.644	13.178
8	S10	0	20.222	40.442	11.677



Node ID

Figure 7-1: Results – Graph of Rate of Energy Consumption per Node – Adaptive Sampling vs Fixed Sampling

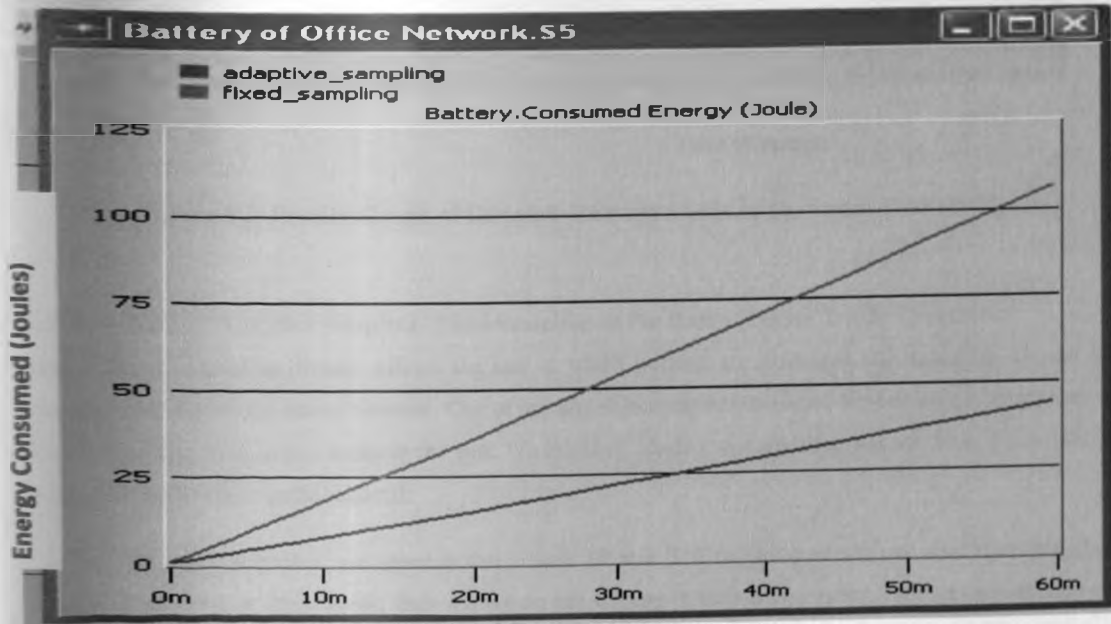


Figure 7-2 : Results –Graph of Cumulative Energy Consumed (Joules) versus Time – Adaptive vs Fixed Sampling (Node S5)

Time (Minutes)

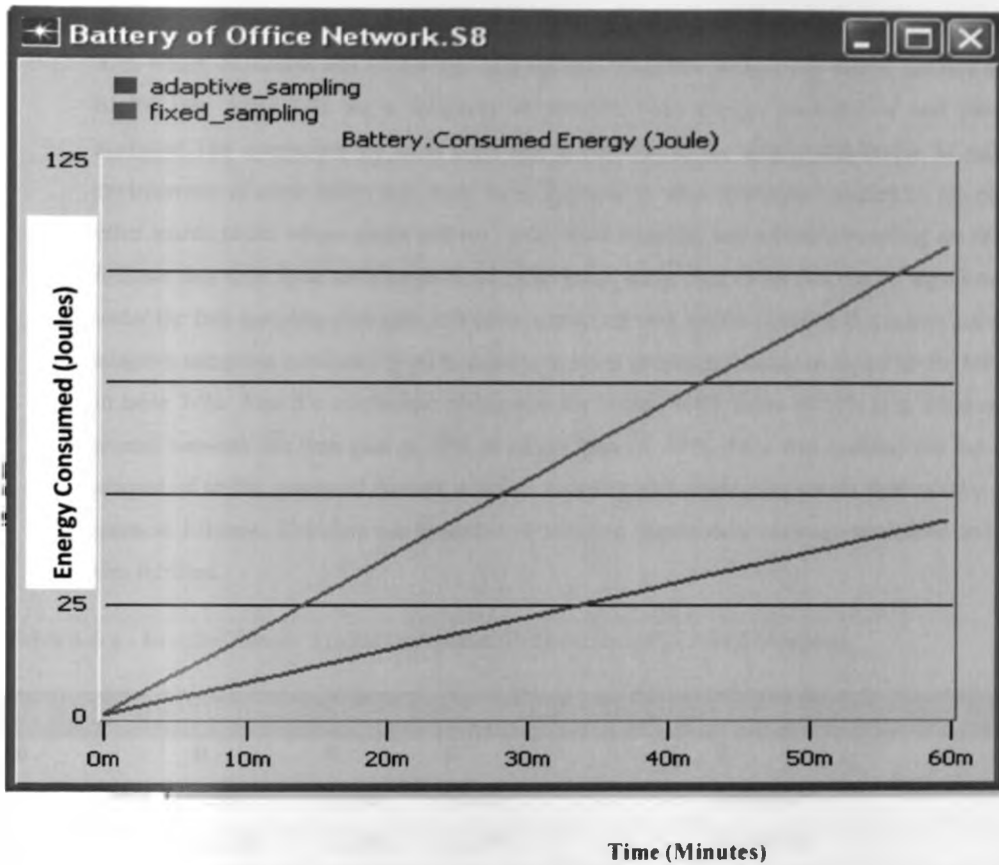


Figure 5-3: Results- Graph of Cumulative Energy Consumption versus Time - Node S8

### 5.3.2: Effect of Adaptive Sampling/ Fixed Sampling on the Rate of Sensor Traffic Generation

The rate of sampling directly affects the rate at which packets are generated and hence the amount of traffic transmitted across the sensor network. One of our key objectives was to reduce the amount of unnecessary message dissemination from source nodes to the sink. We establish whether this objective was achieved. From table 7-2a and 7-2b, the following can be deduced:

- a. In both scenarios, we observe that node S0 and S10 did not generate any messages throughout their lifetime. In other words, they did not do any sensing in their environment. This was an expected case, due to the fact that both nodes were configured as PAN coordinators such that their role in the network was that of coordination and not sensing.
- b. Some nodes in the network exhibited fairly marginal reduction in the mean rate of sensor traffic under the sampling strategies while others showed modest reduction in sensor activity and the rest had extremely reduced message generation activity under adaptive sampling compared to fixed sampling. For instance, Sensor node S6 had a mean sensor traffic rate of 8.364 packets per sec and 7.7558 packets per second.

translating to a percentage reduction of only 7.6%. Sensor node S7 follows with a percentage reduction of 42% which is modest and S8 having 78% message reduction against the overall network wide MRP of 61.2% (see table 7-7a for a summary of network wide energy consumption and network lifetime statistics). The conclusion we draw from this is that due to the spatial distribution of nodes, the local environment of some nodes may not be as dynamic as other locations occupied by the other nodes, in other words nodes whose sensor activity under fixed sampling and adaptive sampling are fairly the same, indicate that their local environments are fairly more stable than those that exhibit significant differences under the two sampling strategies. However, global network statistics reveal that sensor activity under our adaptive sampling is reduced by 61% relative to static sampling strategy as hinted by the MRP percentage in table 7-7a. Also it's worthwhile noting that the overall MRP value of 61% is in close range with the overall network life time gain of 57% or energy gain of -57%. Thus this confirms the fact that reducing amount of traffic generated through adaptive sampling and sensing we obtain algebraically approximates gains in lifetime. Therefore our objective of reducing unnecessary message generation and transmission was fulfilled.

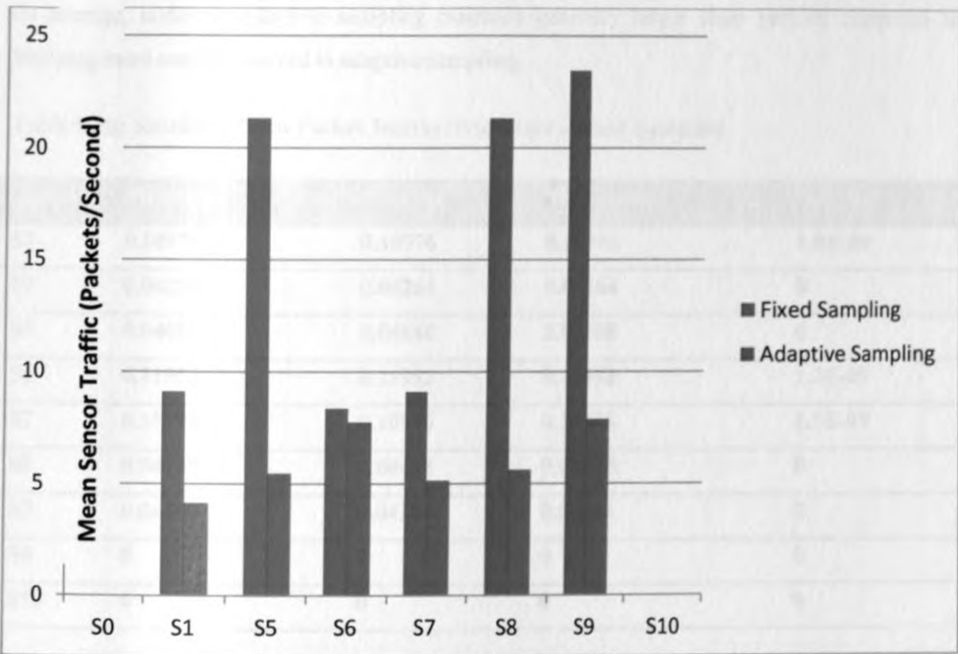
**Table 5-6 a : Results- Sensor Traffic Generated (Packets/Second) - Fixed Sampling**

Node ID	Minimum	Average	Maximum	Std. Dev
S0	0	0	0	0
S1	9.099	9.108	9.111	0.004763
S5	21.301	21.325	21.332	0.011818
S6	8.356	8.364	8.367	0.004444
S7	9.099	9.108	9.111	0.004763
S8	21.301	21.325	21.332	0.011818
S9	23.421	23.446	23.453	0.012515
S10	0	0	0	0

**Table 5-6b: Results – Sensor Traffic Generated (Packets/Second) – Adaptive Sampling**

Node ID	Minimum	Average	Maximum	Std Dev.
S0	0	0	0	0
S1	4.1306	4.1333	4.1361	0.001757
S5	5.4569	5.4639	5.4708	0.004969
S6	7.7417	7.7558	7.7625	0.007318
S7	5.1403	5.1442	5.15	0.003447
S8	5.6292	5.6336	5.6389	0.004064

S9	7.8611	7.8756	7.8819	0.007433
S10	0	0	0	0



**Node ID**

Figure 5-4 : Results – Mean Sensor Traffic Generated (Packet/Second) per Node

**5.3.3: Effect of Adaptive Sampling/ Fixed Sampling on the Application Packet Interarrival Time**

Table 7-3a and 7-3b below show the various packets interarrival times for individual nodes under fixed sampling and adaptive sampling accordingly. The interarrival times for each node indicate the time period between samples. In other terms, the interarrival times are equivalent to the time duration within which the sensor nodes are "OFF". Several conclusions can be inferred from the two tables. One, interarrival times are variable across nodes for both adaptive sampling and fixed sampling. This observation is due to the spatial distribution of nodes, so that each node has their own local sleep periods. A second observation is that the intra node sleep durations are generally constant under the fixed sampling strategy; this is shown by the values of standard deviations that are basically 0 for very node in table 6-3a. Therefore in fixed sampling, once a node establishes their sleep periods initially, it remains fixed throughout their sensing lifetime. On the other hand, the intra node sleep durations as proven by the standard deviation for packet interarrival times are variable in our adaptive sampling algorithm as evidenced by the values of standard deviations show in table 7-3b. The reason for the variability in intra node sleep times directly follows from the algorithm design as described in chapter 4. The sleep periods under our algorithm only remain fixed as long as the sampling rate has not changed but immediately change with change in sampling rate. Some nodes in the network, S0 and S10 have 0 (no) packet interarrival time in both scenarios because they are not configured to do any



sensing. Even though there are certain times when there is some overlap between sleep periods for some nodes in fixed sampling with the same nodes in adaptive sampling e.g node S6 has more less the same value of sleep times in both scenarios. figure 7-3b which shows the variation of system level packet interarrival time with time reveals that on average, nodes in adaptive sampling maintain generally larger sleep periods compared to fixed sampling implying more energy is saved in adaptive sampling.

Table 5-7a: Results - Mean Packet Interarrival Time - Fixed Sampling

Node ID	Minimum	Average	Maximum	Std Dev.
S1	0.10976	0.10976	0.10976	1.9E-09
S2	0.04264	0.04264	0.04264	0
S5	0.04688	0.04688	0.04688	0
S6	0.11952	0.11952	0.11952	1.3E-09
S7	0.10976	0.10976	0.10976	1.9E-09
S8	0.04688	0.04688	0.04688	0
S9	0.04264	0.04264	0.04264	0
S0	0	0	0	0
S10	0	0	0	0

Table 5-7b: Results – Mean Packet Interarrival Time – Adaptive Sampling

Node ID	Minimum	Average	Maximum	Std Dev
S1	0.24167	0.24188	0.24208	0.00013
S2	0.12687	0.12694	0.12703	5.45E-05
S5	0.18277	0.18297	0.1832	0.000157
S6	0.12883	0.1289	0.12899	5.69E-05
S7	0.19418	0.19434	0.19451	0.000126
S8	0.17732	0.17746	0.17767	0.000123
S9	0.12687	0.12694	0.12703	5.45E-05
S0	0	0	0	0
S10	0	0	0	0

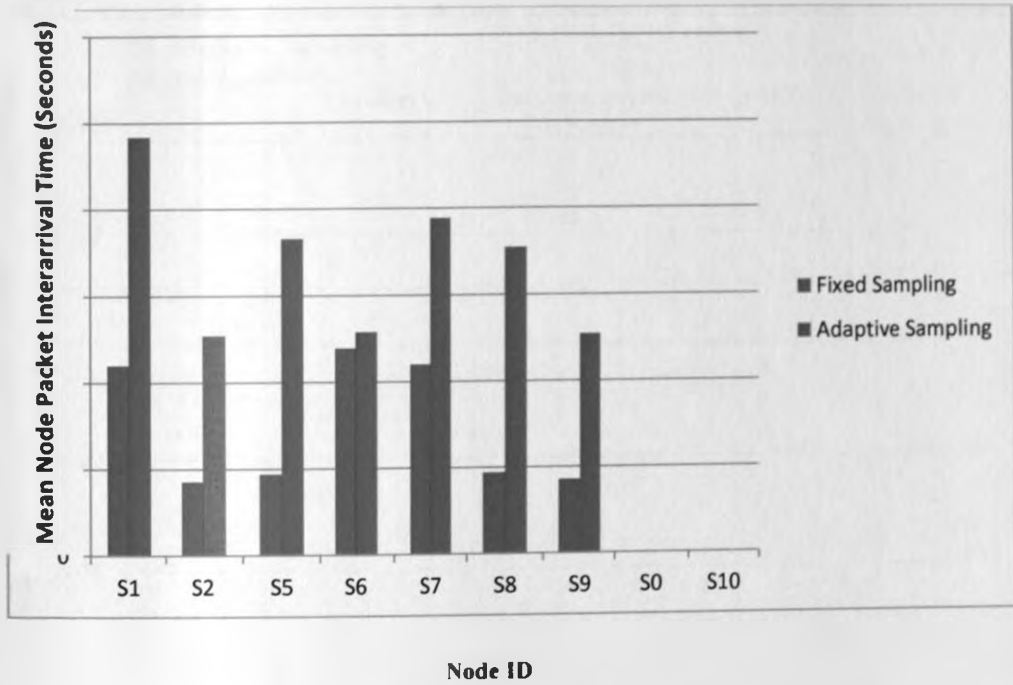


Figure 5-5: Results- Mean Packet Interarrival Per Node – Adaptive Sampling versus Fixed Sampling

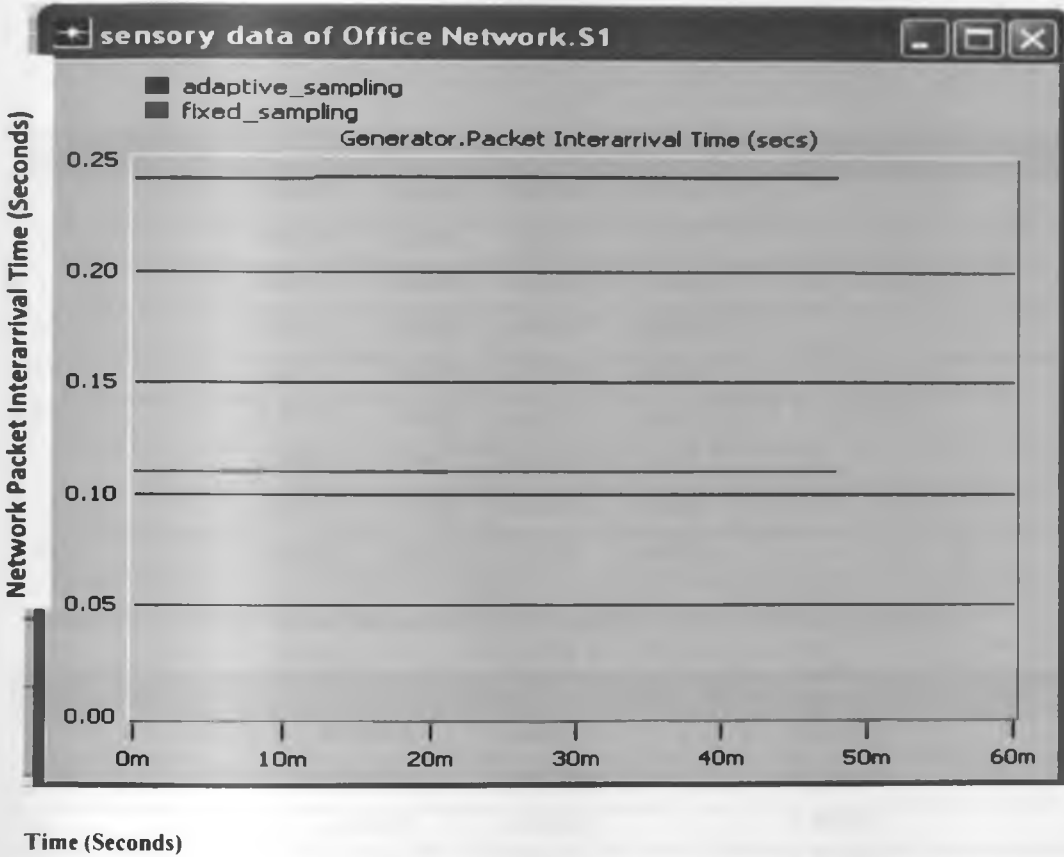


Figure 5-6: Results- Variation of Network Packet Interarrival Times with Time

### 5.3.4 Effect of Adaptive Sampling/ Fixed Sampling on End to End Delay

Table 7-4a and 7-4b show the end to end delay statistics per node under fixed sampling and adaptive sampling. It's obvious to see that the average delay per node under our adaptive sampling algorithm is several factors lower than that of fixed sampling. Figure 7-4a also depicts the graphical results. Assuming all factor fixed, the enormous difference in delay per node can attributed to the adaptive duty cycling per sensor node combined with generally lower sampling rates across nodes which in turn lead to proportionately reduced traffic generation at the source. Consequently, message transmission across the network is also reduced because there is less contention for the band limited wireless channel. What this also signifies is that the environment under observation is highly dynamic and stochastic necessitating frequent adjustments in duty cycle periods and sampling rates. In fairly slowing changing processes, the performance of our algorithm would be nearly comparable with adaptive sampling i.e the two algorithms would be indistinguishable in performance.

**Table 5-8a : Results – Mean End to End Delay (Seconds) per Node – Fixed Sampling**

Node ID	Minimum	Average	Maximum	Std. Dev.
S0	3.0307	14.639	26.374	8.174
S1	3.5197	17.047	30.819	9.552
S2	3.8342	18.507	33.365	10.336
S5	7.375	36.042	64.575	20.099
S6	5.22	25.549	45.829	14.234
S7	5.3216	26.052	46.515	14.497
S8	0.0693	0.09	0.143	0.027
S9	4.1677	20.232	36.571	11.34
S10	3.1456	15.192	27.373	8.492

**Table 7-8b - : Results – Mean End to End Delay (Seconds) per Node – Adaptive Sampling**

Node ID	Minimum	Average	Maximum	Std Dev.
S0	0.021326	0.021478	0.021632	0.000103
S1	0.02273	0.022828	0.022928	6.74E-05
S2	0.023996	0.024125	0.024292	0.000105
S5	0.023926	0.024079	0.024149	8.15E-05
S6	0.024084	0.024276	0.024409	0.000117
S7	0.018274	0.018421	0.018512	8.72E-05
S8	0.01772	0.017816	0.017884	5.63E-05
S9	0.023734	0.023849	0.023983	8.22E-05
S10	0.022009	0.022115	0.022225	7.25E-05

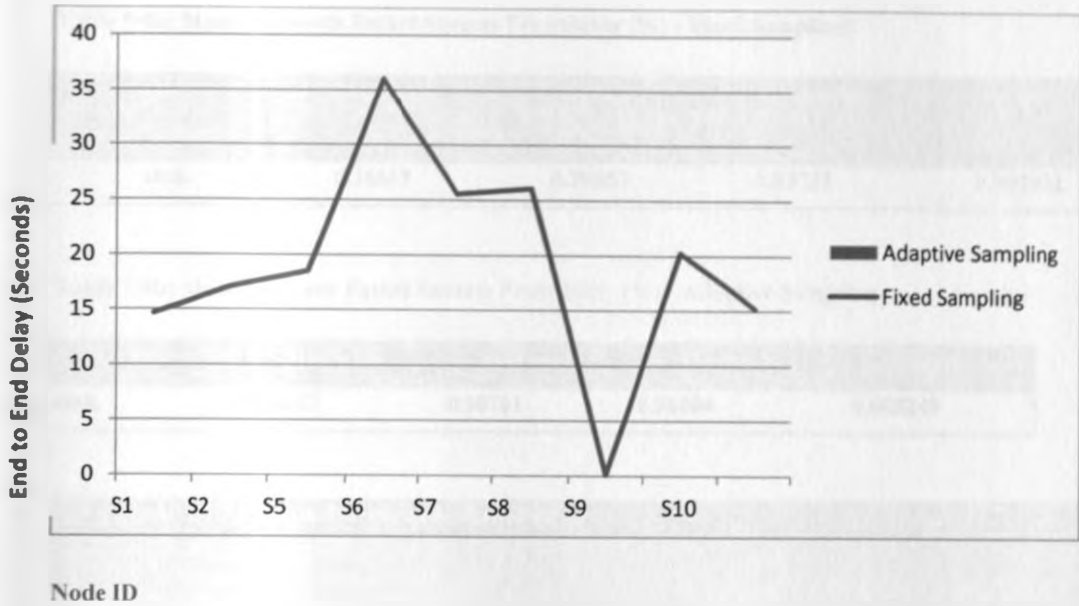


Figure 5-7: Results – Mean End to End Delay (Seconds) per Sensor Node

### 5.3.5: Effect of Adaptive Sampling/ Fixed Sampling on the Packet Success Probability

A second goal in our problem statement was to ensure that in an effort to minimize energy consumption, the quality of information as measured by timeliness and completeness is preserved or maximized. We chose to measure completeness using Packet Success Probability (PSP). In figure 7-5a a graph of PSP (%) versus time is presented for both scenarios. At the beginning of sensing both adaptive sampling and fixed sampling have the same PSP value of 16.7%. In both setups, the PSP sharply rises suddenly above this percentage just beyond the 2<sup>nd</sup> minute of sensing, with adaptive sampling taking lead and the fixed sampling lagging beyond. After the sharp rise in PSP, the PSP steadily increases then plateaus. The peak and average performance reached are 78.66% and 83.72% in fixed sampling respectively and 90.76% and 91.00% in adaptive sampling. Clearly, the reliability of success data delivery is higher in adaptive sampling than that in fixed sampling. Even though in fixed sampling more messages are generated in a unit time as seen in the previous session, only 78.66% get to reach the fusion center compared to 90.76% in adaptive sampling, representing a significant 13% increase in quality of information.

The combined increase in PSP and the tremendous reduction in delay complete our second goal of the thesis – preserving/ maximizing QoI as measured by timeliness (of data delivery) and completeness (as measured by PSP).

Table 5-9a: Mean Network Packet Success Probability (%) - Fixed Sampling

Object Name	Minimum	Average	Maximum	Std Dev
sink	0.16667	0.78657	0.83721	0.001932

Table 7-9b: Mean Network Packet Success Probability (%) - Adaptive Sampling

Object ID	Minimum	Average	Maximum	Std Dev
sink	0.16667	0.90761	0.91004	0.005249

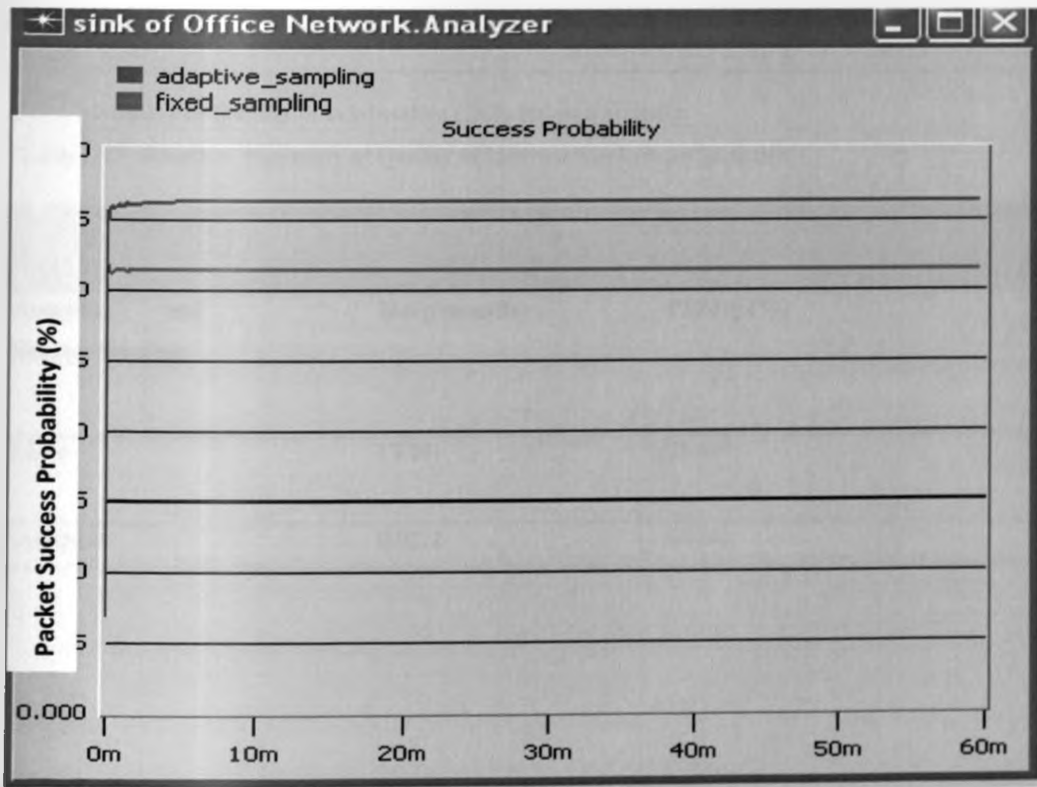


Figure 5-8 : Variation of Network Packet Success Probability (%) with time (seconds)

5.3.6: Summary of Energy Consumption and Network Life Time Related Statistics

Table 5-10: Results – Summary of Energy Consumption and Network Lifetime Related Statistics

Performance Metric (Mean Values)						
Sampling and Sensing Strategy	Enavg(Joules per min)	Traffic Sent per unit time (Packets/Sec)	T (second)	Eg (%)	Lg(%)	MRP (%)
Fixed	451.577	92.676	0.51808			
Adaptive	195.925	36.0064	1.17943			
				-57	57	61.2

5.3.7. Summary of Quality of Information (QoI) Related Statistics

Table 5-11: Results – Summary of Quality of Information Related Statistics

Performance Metric (Mean Values)		
Sampling and Sensing Strategy	Davg(seconds)	PSPavg (%)
Fixed	19.261	78.657
Adaptive	0.0212	90.761

## CHAPTER 6: CONCLUSIONS

### 6.1 Introduction

In this thesis we have attempted to discuss wireless sensor networks and the fundamental issues surrounding WSNs and proposed a solution to combating some of these issues. Therefore anyone wishing to get acquainted with the field of WSNs may find our work relevant. Obviously, several intertwined challenges stand in the way of complete realization of the enormous potential of WSNs and in no way have we exhaustively tackled these issues in our work. Our focus instead has been on the major challenge of WSNs which is energy conservation versus quality of information (QoI) and specifically in the context of dynamic and stochastic environments. We have seen that conserving energy is key to the longevity of WSNs but again longevity when considered in isolation without addressing the core functional aspect of WSNs of collecting and dissemination of data that can be trusted does not fulfill the promise for which these networks are built for. The thesis has tried to address these two contradicting goals by using the adaptive sampling and sensing technique – which is definitely not an uncommon technique among the wireless sensor network research community. By combining concepts from digital signal processing, statistical process control and adaptive duty cycling, we have demonstrated that our algorithm is able to conserve significant amount of energy, while guaranteeing impressive levels of QoI as measured by the completeness and timeliness attributes of data quality.

### 6.2 Limitations and Challenges

The project could not be complete without experiencing some challenges. The decision to implement Discrete Fourier Transforms locally on every sensor node to effectively estimate sampling rates posed a computational challenge: by nature computing DFTs is a computationally intensive process and even more constraining for the already energy limited sensor nodes. This required a careful choice of appropriate software implementation of DFT that is cognisant of the power limitations of the nodes. We learned that such efficient implementations lack in the OPNET modeler 11.5 versions were using for our simulations. It took us time of extensive research to come across the most efficient publically available FFT software implementation – the FFTW3 library. See chapter () for a detailed description of FFTW3 and the reasons why we chose it for the work described in this thesis. By using the FFTW3, we took advantage of the routine, *fftw\_plan\_dft\_r2c\_1d()*, that takes a real valued data vector and maps it to the complex output vector, making use of the Hermitian<sup>†</sup> redundancy and thus leading to an efficiency gain of 2 times in speed. Time on the other hand, was another bottleneck. Despite the fact that we achieved a 57% increase in energy gain at 91% packet delivery success with delay several factors smaller than the fixed sampling, there were much more that we desired to do to obtain better results e.g refining the DFT implementation through distributed memory implementation and implementation of the algorithm on top of energy efficient routing protocols such as LEACH. However, this could not be achieved in the 6 months period that was allotted for the project.



OPNET modeler 11.5 did not have a proper realistic way of modeling wireless sensor networks. At best what we saw was the Zigbee wireless mesh model that offered support for low power wireless communications within which WSNs fall. However, there two major pitfalls with this model that made it not useful to us. First and foremost, the model was closed – the underlying code was inaccessible and so we could not modify it to suit the needs of our project. Secondly even if the model was accessible, it did not have an appropriate model for the sensory layer of sensor networks that addresses issues related to battery models typical with real sensor motes like the MicaZ. We went over this issue when we got the OpenZB – an OPNET implementation of the IEEE 802.15.4 /Zigbee with embedded support for sensor specific battery models – the MicaZ and TelosB sensor mote models.

### 6.3 Future Work

There is a lot that was desired to be accomplished in this work but owing to the practical limitations discussed elsewhere in this work we could not achieve them. First, effective energy conservation with quality of service (QoS) guarantees requires a combination of techniques across the entire sensor and communication protocol stack. We have partially achieved this by attempting to combine several complementary approaches (signal processing, statistical control and duty cycling) that have indeed proven to yield significant results; however, we have only done this at the sensor layer. Future work will include testing our algorithm over existing and perhaps emerging energy efficient network levels protocols such as LEACH and/or MAC layer protocols. Another area of improvement would be considering optimization of the DFT algorithm by implementation of a distributed memory version leveraging the FFTW3 library. An optimization algorithm that determines the  $(W, h)$  pair(s) to yield the most globally optimal results will be investigated. Lastly, we saw under the limitations section that the statistical change technique though powerful suffers from the problem of false negatives and false positives. We will be keen on further improving the effectiveness of this algorithm by combining it with machine learning techniques.

## REFERENCES

- Alippi. C et al (2009). Energy Management in Wireless Sensor Networks with Energy Hungry Sensors, IEEE Instrumentation and Measurement Magazine, Vol. 12(2), pp 16-23, April 2009.
- Alippi C. et al (2007). Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications. Proc. IEEE MASS 2007, Pisa, Italy, 8<sup>th</sup> April.
- Alippi C. et al (2007). Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications. 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems
- Altman E. and Basar T. (1998). Multiuser Based Flow Control, IEEE Transactions on Communications, Vol .46, No. 7, Page 940-949, July 1998.
- Akyildiz F. et al (2002). A survey on sensor networks Communications Magazine, IEEE, vol. 40, pp. 102-114, 2002.
- Ahmedy I. et al (2011). A review on wireless sensor network routing protocol: challenge in energy perspective, Scientific Research and Essays Vol. 6(26), pp.5628-5649, 9<sup>th</sup> November, 2011.
- Alpcan T., Basar T., and Dey S. (2006). a Power Control game based on outage probabilities for multicel Wireless data networks, IEEE Transactions on Wireless Communications, Vol. 5, no. 4, pg 890-899, April 2006
- Bandyopadhyay S. and. Coyle E.J (2003). An energy efficient hierarchical clustering algorithm for wireless sensor networks. Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications, March-30-April, 3, IEEE Societies, pp: 1713-1723.
- Basseville M. and Nikoforov I.V. (1993). Detection of Abrupt Changes: Theory and Application, Prentice Hall, Inc. 1993.
- Cesana M., Gatti N., and Malanchini I.(2008). a Game theoretic approach decentralized power allocation for cellular networks, in the proceedings of 2<sup>nd</sup> International Workshop on Game theory in communication networks, Athens, Greece, Oct. 2008.
- Chan K., Fekri F., and Pishro-Niki H. (2005). Analysis of hierarchical Algorithms for wireless sensor network routing protocols ", IEEE wireless communications and networking conference, Vol. 3, page 180-1835
- Choi D. S. Moh and I. Chung (2011). ARCS: An Energy Efficient Clustering Scheme for Sensor Monitoring Systems. International Scholarly Research Network (ISRN) Communications and Networking Vol. 2011, Article ID 572572, 10 pages
- Cooley, J. W. and J. W. Tukey. "An Algorithm for the Machine Computation of the Complex Fourier Series." *Mathematics of Computation*. Vol. 19, April 1965, pp. 297-301.

Duhamel, P. and M. Vetterli, "Fast Fourier Transforms: A Tutorial Review and a State of the Art," *Signal Processing*, Vol. 19, April 1990, pp. 259-299.

Frigo, M. and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, 1998, pp. 1381-1384.

FFTW (<http://www.fftw.org>), last accessed on the 27<sup>th</sup> of July 2012.

Goldsmith A.J and Wicker S.B (2002). Design Challenges for Energy Constrained ad hoc wireless networks, *IEEE Wireless Communications*, Vol. 9, pp 8-29, 2002.

Heinzelman W.B, Chandrakasan A.P, and Balakrishnan H. (2000): Energy Efficient Communication Protocols for Wireless Micro sensor Networks (LEACH)". HICSS Vol. 8, pg 3005-3014, Maui, Hawaii, January 2005. Available: [citeseer.ist.psu.edu/rabinerhelman00energyefficient.html](http://citeseer.ist.psu.edu/rabinerhelman00energyefficient.html)

Heinzelman W.B., Chandrakasan A.P., Balakrishnan H. (2002). "Application specific protocol architecture for wireless micro sensor networks," *Wireless Communications, IEEE*, vol. 1, pp. 660-670, oct 2002.

Hill J. L. (2003). System Architecture for Wireless Sensor Networks, PhD Thesis, University of California at Berkley.

IEEE 802.15.4 Standard-2003. "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer

(PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)". IEEE-SA Standards Board, 2003.

Jain A. and Chang J.Y (2004). Adaptive Sampling for Sensor Networks, Proc., International Workshop on Management for Sensor Networks, pp 10-16, Toronto , Canada, 30<sup>th</sup> August, 2004.

Kanann R. and Iyengar S. (2004). Game Theoretic Modes for Reliable Path Length and Energy Constrained Routing with Data Aggregation in Wireless Sensor Networks, *IEEE Journal on Selected areas in Communications*, Vol 22, No. 6, August 2004.

Karl H. and Willig A. (2003). A short Survey of Wireless Sensor Networks, Technical Report, TKN-03.018, Telecommunications Networks Group, Technische Universitat Berlin, Oct 2003.

Lindsey S. and Raghavendra C. (2002). "PEGASIS: Power- Efficient Gathering in Sensor Information System," in *Proceedings of the Aerospace Conference*, IEEE, vol. 3, Big Sky, Montana, 2002, pp. 9-16.

Martinez K., Hart J.K, and Ong R. (2004). Environmental Sensor Networks, *Computer Magazine, IEEE*, vol. 37(8), 2004, pp. 50-56.

Mohammad Ilayas and Imad Mahgoub ed. (2004). Handbook of Wireless Sensor Networks: Compact Wireless and Wired Sensing Systems, CRC Press.

Neal Patwari, et al Hierarchical Censoring Sensor for Change Detection

Nallusamy, R. and Duraiswamy K. (2011). Solar powered wireless sensor networks for environmental applications with energy efficient routing concepts: A review. *Inform. Technol. Journal* Vol. 10, pp 1-10.

Oppenheim, A. V. and Schafer R.W. , *Discrete-Time Signal Processing*, Prentice-Hall, 1989, p. 611.

Oppenheim, A. V. and. Schafer R. W, *Discrete-Time Signal Processing*, Prentice-Hall, 1989, p. 619.

Petr Jurcik and Anis Koubaa (2007). HURRAY-TR-070509 Technical report on the IEEE 802.15.4 OPNET Simulation model: Reference Guide v2.0, available at [www.hurray.isep.ipp.pt](http://www.hurray.isep.ipp.pt)

Rader, C. M., "Discrete Fourier Transforms when the Number of Data Samples Is Prime." *Proceedings of the IEEE*, Vol. 56, June 1968, pp. 1107-1108.

Rajgopal Kannan and Sitharama Iyengar S. (2004). Game Theoretic Models for Reliable Path Length and Energy Constrained Routing with Data Aggregation in Wireless Sensor Networks, *IEEE Journal on Selected Areas in Communications*, Vol. 2(6), and August 2004

Urpi, A., M.Bonucelli, and Giordano S. (2003). Modeling Cooperation in Mobile Ad hoc networks: A formal description of selfishness, in the proceedings of the 1<sup>st</sup> WiOpt 2003, France, and March 2003.

Sangyeo Lee et al. The CUSUM Test for Parameter Change in Regression Models with Arch Errors, *J. Japan Statist. Soc.* Vol 34(2), 2004, pp 173-188

Sirinivasan V. et al, 2003). Cooperation in ad hoc wireless networks, in the Proceedings of IEEE INFOCOM 2003, San Francisco, CA, USA, March 2003.

Suraiya Trannum (2010). Energy Conservation Challenges in Wireless Sensor Networks: A Comprehensive Study. *Wireless Sensor Network*, Vol 2, 2010, pp 483-491, [online]. DOI: 10.4236/wsn.2010.26060

Valli R. and Danajayan P. (2010). A Non-cooperative Game Theoretical Approach for Power Control in Virtual

Varga, A. and Fakhmzadeh, B. (1997), "The K-Split Algorithm for the PDF Approximation of Multi-Dimensional Empirical Distributions without Storing Observations", *ESS'97: 9th European Simulation Symposium*: 94--98.

Wang, J, Liu, Y and Das S.K (2010). Energy Efficient Data Gathering in Wireless Sensor Networks with Asynchronous Sampling. *ACM Transactions Sensor Networks* 6, 3, Article 22, June 2010.

Xiao Zhenghong (2010). Anomaly Detection Based on Multiclass CUSUM Algorithm for WSN, *Journal of Computers*, Vol. 5(2), and February 2010.

Zhou, J. and De Roure, D. (2007) FloodNet: Coupling Adaptive Sampling with Energy Aware Routing in a Flood Warning System. *Journal of Computer Science and Technology*, 22 (1). pp. 121-130. ISSN 1000-9000