UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

TITLE: MAXIMISING LIFETIME OF WIRELESS SENSOR NETWORKS USING DISTRIBUTED SCHEDULING ALGORITHMS WITH ADJUSTABLE SENSING RANGE.

BY

Moso Chebet Juliet

P58/70482/2008

SUPERVISOR:

Prof Okelo W. Odongo

AUGUST 2011

Submitted in partial fulfillment of the requirements of the Degree of Master of Science in Computer Science at the University of Nairobi.

Declaration

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

Sign: _____ Date: 29/08/2011

Juliet C. Moso

P58/70482/2008

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

Sign: _____ Date: 30/8/2011

PROF OKELO W. ODONGO

Project supervisor

School of Computing and Informatics

University of Nairobi

Acknowledgements

I would like to express my gratitude to my supervisor Prof Okelo W. Odongo for his support and invaluable guidance throughout my study. His suggestions and insights for this project have been invaluable. Without his support and guidance this project would not have been possible. I am also grateful to the other members of the panel, Mr Samuel Ruhiu and Mr. Stephen Mburu for their advice and valuable time spent in reviewing the project material. Finally, I would like to express my heartfelt gratitude to my family, especially my husband, for their continuous encouragement and confidence in me during the course of this project. Without their support, this project would not have been possible.

Abstract

Optimizing the energy consumption in wireless sensor networks has recently become the most important performance objective. In this project we define the lifetime of a wireless sensor network as the amount of time that the network can effectively cover the targets of interest. Having all the sensors active at all times would ensure coverage but would also significantly reduce the network lifetime as the nodes would discharge quickly. A viable approach taken to maximize the network lifetime is to make good use of the overlap in the sensing regions of individual sensors caused by the high density of deployment. We design a scheduling mechanism in which only a subset of the sensors can be active at any one time, while all other sensors are put to sleep. The members of this active set (cover set) are periodically updated to keep the network alive for a longer duration of time. Also, for each of the cover sets, the goal is to smoothly adjust the sensing range such that a minimum sensing range can be maintained while meeting the target coverage objective. We propose a reliable distributed scheduling algorithm which can smoothly adjust the nodes' sensing range while providing optimal target coverage with the minimal set of active sensors. From the simulation results, the improvement in network lifetime of the Distributed Scheduling Algorithm with adjustable sensing range over the Load Balancing Protocol for sensing with fixed sensing range is about 26% on average in the linear energy model and about 50% on average in the quadratic energy model.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BWRC	Berkeley Wireless Research Center
CEC	Cluster-based Energy Conservation
CMOS	Complementary Metal-Oxide-Semiconductor
CODA	Congestion Detection and Avoidance
DARPA	Defense Advanced Research Projects Agency
DSN	Distributed Sensor Networks
GAF	Geographic Adaptive Fidelity
HEED	Hybrid Energy-Efficient Distributed clustering.
LBP	Load Balancing Protocol for Sensing
LEACH	Low Energy Adaptive Clustering Hierarchy
LP	Linear Programming
LWIM	Low PowerWireless Integrated Microsensor
MEMs	Micro –Electro – Mechanical systems
MIT	Massachusetts Institute of Technology
SensIT	Sensor Information Technology
STEM	Sparse Topology and Energy Management
μAMPS	micro-Adaptive Multidomain Power-aware Sensor
WINS	Wireless Integrated Network Sensors
WSNs	Wireless Sensor Network

CHAPTER 1: INTRODUCTION

1.1 Introduction

A wireless sensor network (WSN) is a network that is made up of hundreds or thousands of sensor nodes which are densely deployed in an unattended environment with the capabilities of sensing, wireless communications and computations (i.e. collecting and disseminating environmental data). These spatially distributed autonomous devices cooperatively monitor physical and environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants (Akyildiz *et al.* 2002; Romer & Mattern 2004; Haenselmann 2006). Recent advances in micro-electro-mechanical systems (MEMs) and low power, highly integrated digital electronics has led to the development of micro-sensors which can be easily deployed in an ad hoc manner and self-organized into a system that monitors the environment and forwards data back to a base station or sink.

The power of WSNs is found in their ability to deploy large numbers of sensor nodes which can assemble and self configure as a network. This network can be easily extended by adding more sensor nodes without any need for reconfiguration. In addition, the network can automatically adapt to compensate for node failures. As shown below in Figure 1.1 (Akyildiz *et al.* 2002) the main components of a sensor node include a power unit (batteries and/or solar cells), a sensing unit (sensors and analog-to-digital converters), a processing unit (along with storage), and a transceiver unit (connects the node to the network). The optional components include a location-finding system, a power generator, a control actuator, and other application-dependent elements. Sometimes a mobilizer is needed to move the sensor node from the current position and carry out the assigned tasks. Since the sensor may be mobile, the base station may require accurate location of the node which is done by the location finding system. The analog signals measured by the sensors are converted to digital signals by analog - to- digital converters and then supplied to the processing unit. Sensor nodes may also have to be disposable, autonomous, and adaptive to the environment.

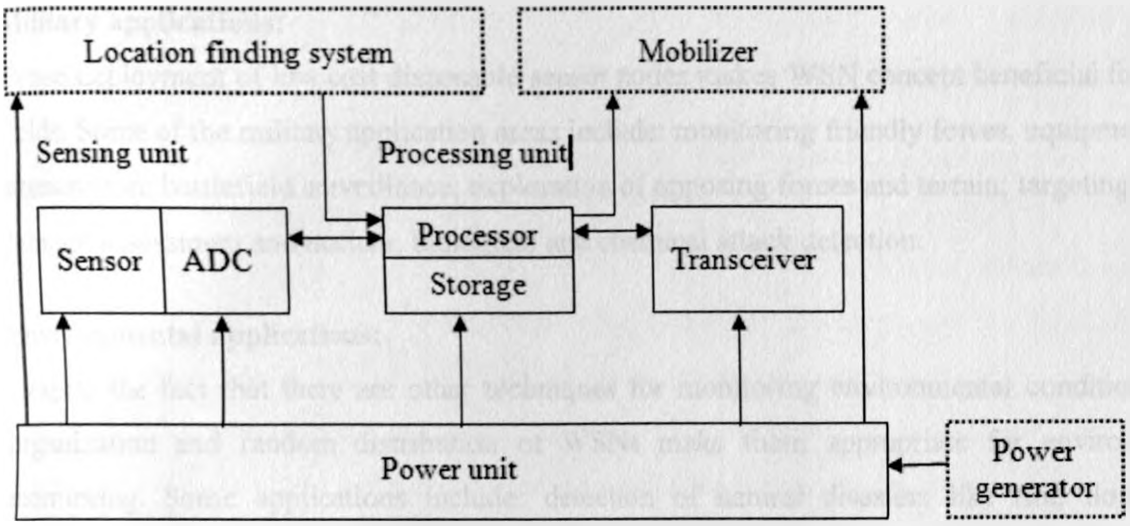

Figure 1.1: Sensor node architecture
(Source: Akyildiz *et al.* 2002)

The main function of a sensor node in a monitored region is to detect events, perform local data processing, and transmit raw and/or processed data. Power consumption can therefore be allocated to three functional domains: sensing, communication, and data processing, each of which requires optimization. Among these tasks transmitting data requires much more energy than processing data (Pottie & Kaiser 2000) and the most recent efforts on optimizing the wireless sensor network lifetime have been focused on routing protocols (i.e., transmitting data to the base and data request from the base to the sensor node). In the context of communications, in a multihop sensor network a node may play the dual role of data collection and processing and of being a data relay point.

1.1.1 Sensor Network Application Areas

The applications of WSNs typically involve some kind of monitoring, tracking and controlling. WSNs come in handy in habitat monitoring, object tracking, nuclear reactor control, fire detection and traffic monitoring. A general categorization of WSN applications may include military applications, environmental applications, health applications and other commercial applications as described by (Akyildiz *et al.* 2002).

Military applications:

Dense deployment of low cost disposable sensor nodes makes WSN concept beneficial for battle fields. Some of the military application areas include: monitoring friendly forces, equipment and ammunition; battlefield surveillance; exploration of opposing forces and terrain; targeting; Battle damage assessment and nuclear, biological and chemical attack detection.

Environmental applications:

Despite the fact that there are other techniques for monitoring environmental conditions, self organization and random distribution of WSNs make them appropriate for environmental monitoring. Some applications include: detection of natural disasters like fire, floods and earthquakes; biocomplexity mapping of the environment; precision agriculture; planetary exploration; habitat monitoring and pollution detection.

Health applications:

The small size and light-weight structure of sensor nodes provides much functionality in the health sector, including: tracking and monitoring doctors and patients; drug administration and telemonitoring of human physiological data.

Commercial applications:

Some of the commercial applications of WSNs include: burglary detection and monitoring; vehicle tracking and detection; interactive museums; monitoring material fatigue; environmental control in buildings; robot control and guidance in automatic manufacturing environments; factory process control and automation; smart structures with sensor nodes embedded inside; interactive toys and machine diagnosis.

1.1.2 Wireless Sensor Network Challenges

WSNs are limited in power, memory and computation capabilities. Node failures in WSNs can adversely affect network performance and may lead to network partitioning or death of the whole network, as either some region of the network will not be covered or there is data transmission failure in the network. With this in mind, energy saving has become a critical issue in WSNs, and the most energy saving must come from energy aware protocols and algorithms. One important perspective is to maximize the network lifetime (Sankar & Liu 2004; Chang & Tassiulas 2004;

Madan, Luo & Lall *et al.* 2005), where the network lifetime usually refers to the time interval between the initialization of the network and the battery exhaustion of the first sensor node.

WSN Sensor nodes operate on battery power typically based on two AA alkaline cells or one Li-AA cell which implies that they have to operate on a limited energy budget (1.2 Volts). To compound this problem further, most of the nodes are deployed in hostile and an inaccessible environment where making any attempt at changing batteries is a very complicated affair. Some networks consist of thousands to millions of sensor nodes in a distributed environment making changing of batteries difficult and recharging almost impossible during operations. This problem has forced node, network and system developers to make changes in the basic WSN architecture with the main aim being to minimize energy consumption especially of the nodes so as to make the network and overall application system more energy efficient (Ali & Partha 2008). Despite the substantial improvement on chip design for energy conservation, advances on battery design still lag behind. The lifetime of a wireless sensor node depends on the available energy sources and its energy consumption. It is inversely proportional to the average rate of information generated and relayed by it.

1.1.3 Our contribution

In this project we define the lifetime of a wireless sensor network as the amount of time that the network can effectively cover the targets of interest. Having all the sensors active at all times would ensure coverage but would also significantly reduce the network lifetime as the nodes would discharge quickly. A viable approach taken to maximize the network lifetime is to make good use of the overlap in the sensing regions of individual sensors caused by the high density of deployment. We design a scheduling mechanism in which only a subset of the sensors can be active at any one time, while all other sensors are put to sleep. The members of this active set (cover set) are periodically updated to keep the network alive for a longer duration of time. Also, for each of the cover sets, the goal is to smoothly adjust the sensing range such that a minimum sensing range can be maintained while meeting the target coverage objective. This problem was introduced by Cardei et al. (2005b), the authors divide the sensors into cover sets, where the sensors of each cover set adjust their sensing range in order to avoid covering the same targets two or more times. They examine the case where the nodes' sensing range has p steps, while they target to maximize the number of cover sets with the aim of extending the network lifetime.

Our formulation differs significantly from theirs since we allow smooth sensing range variation as opposed to their discrete range model. We assume that each sensor can communicate with its neighbors within two times of the maximum sensing range. Initially each sensor broadcasts its energy level and covered targets to its neighbors; this facilitates the formation of the sensor cover schedule. Each sensor can be in any one of four states, active, idle, vulnerable and terminated state. In the initialization phase, all sensors are in the vulnerable state with maximum sensing range. For each sensor, it should change to *Active state* with sensing range *r* if there is a target at range *r* which is not covered by any other active or vulnerable sensors or remain in the *Vulnerable state* but decrease its sensing range to the next furthest target if all targets at range r are covered by other active or vulnerable sensor with greater energy supply. If its sensing range decreases to zero it should change to *Idle state.* When the energy gets completely exhausted it enters the *terminated state* and is deleted from the network. Once all the sensors have made a decision to be active or idle, they will stay in that state for a period of time called shuffle time or until there is an active sensor which exhausts it energy and is going to die. This dying sensor sends a wake-up call to all the sensors in the network causing them to transition to the vulnerable state with their maximum sensing range. When there is a target that cannot be covered by any sensor in the network, the network fails.

The main contribution of this study is a distributed scheduling algorithm with smooth adjustment of sensing range. This algorithm is an enhancement of the load balancing protocol LBP with fixed sensing range described by Brinza & Zelikovsky (2006).

1.1.4 Organization of the chapters

The rest of the paper is organized as follows. In chapter 2 we provide the background and literature review in WSN lifetime and coverage problems. Chapter 3 provides a discussion on energy management in WSNs. Chapter 4 introduces the Load Balancing Protocol for fixed sensing and the proposed distributed scheduling algorithm with adjustable sensing range. Chapter 5 presents the design and implementation of the distributed algorithms. Chapter 6 presents analysis of the simulation results. Chapter 7 summarizes the findings and outlines possible directions for future research.

1.2 Problem Statement

Sensor coverage can be formulated as an optimization problem: Given the number of available sensors *n*, how do we place the sensors so that the sensing range *r* needed to cover the monitoring area is minimized? This is the question this project sets out to answer and or ascertain. We consider the problem of maximizing the network lifetime for adjustable sensing range wireless sensor networks. This problem can be formulated as follows:

Given a region R to be monitored, a set of sensors $s_1,....s_n$, and a set of targets $i_1,...i_m$, and energy supply bi for each sensor, find a monitoring schedule $(C_1, t_1),.....,(C_k, t_k)$ and a range assignment for each sensor in a set C_i such that:

- Lifetime is maximized $(t_1 + t_2 + ... + t_k)$ is maximized.
- Each set cover monitors all targets and
- Each sensor does not appear in the sets for a time more than its initial energy.

This problem was introduced by Cardei et al. (2005b), the authors divide the sensors into cover sets, where the sensors of each cover set adjust their sensing range in order to avoid covering the same targets two or more times. They examine the case where the nodes' sensing range has *p* steps, while they target to maximize the number of cover sets. Our formulation differs significantly from theirs since we allow smooth sensing range variation as opposed to their discrete range model. We define the lifetime of a wireless sensor network as the amount of time that the network can effectively cover the targets of interest. Having all the sensors active at all times would ensure coverage but would also significantly reduce the network lifetime as the nodes would discharge quickly. A viable approach taken to maximize the network lifetime is to make good use of the overlap in the sensing regions of individual sensors caused by the high density of deployment. We design a scheduling mechanism in which only a subset of the sensors can be active at any one time, while all other sensors are put to sleep. The members of this active cover set are periodically updated to keep the network alive for a longer duration of time. Our work is an extension of the techniques used in (Berman *et al.* 2004, 2005 and Brinza & Zelikovsky 2006) where a similar problem was studied for sensor networks with fixed sensing range.

Assumptions

- Nodes are randomly distributed in a defined region.
- Each sensor monitoring a target can collect data from that target without the help of any other sensor.
- Each sensor knows its own coordinates as well as the coordinates of all the covered targets.
- A sensor can vary the sensing range smoothly from zero to a defined maximum range.
- During monitoring mode a sensor can either be in the idle or active state.
- The number of deployed sensors largely exceeds the number of targets to be monitored.

1.3 Purpose of the study

This project aims to study distributed scheduling algorithms with fixed sensing range and implement an enhanced algorithm with adjustable sensing range that provides optimal maximum lifetime of a wireless sensor network.

1.4 Objectives

a. To identify wasteful and unnecessary activities in a wireless sensor node and mitigate their impact.

b. Study and establish a distributed scheduling algorithm with adjustable sensing range that maximizes WSN lifetime.

c. Evaluate and model the distributed scheduling algorithm and verify its performance on target coverage, reliability, scalability and extension of network lifetime through simulation.

1.5 Research questions

(1) Does the implementation of distributed scheduling algorithms with smooth adjustment of sensing range guarantee an improvement in network lifetime when compared with distributed algorithms with fixed sensing range?

(2) In self-organizing monitoring schedules what rules should be used by sensor nodes when deciding to become active or idle and when should nodes make such decisions?

1.6 Hypothesis

Wireless sensor network lifetime increases with increase in number of sensor nodes with adjustable sensing range as compared with sensors with fixed sensing range and decreases with increase in the number of targets to be monitored.

1.7 Significance of the study

A Wireless Sensor Network (WSN) can be used in a variety of applications e.g. Disaster monitoring, Early warning systems (Forest Fires, Tides), contaminant flow monitoring, structural monitoring, military command and control, military surveillance, intrusion detection etc. Amidst all this application areas, the biggest constraint of the WSN is energy due to use of limited non-replaceable batteries. One of the key characteristics in wireless communication is that energy consumption increases with increase in transmission distance. In conventional sensor design, energy spent in sensing has an inverse relationship with the amount of signal energy received by the sensor. Node failures can adversely affect network performance leading to network partitioning and data transmission failure. Therefore, energy saving is a critical issue and must come from energy aware protocols and algorithms

1.8 Justification for adjustment of sensing range

Power saving techniques in wireless sensor networks can be classified into two categories. The first category uses scheduling such that subsets of sensors can alternate between active state and low power sleep mode. The second technique is that of adjusting the sensing range of the wireless sensor nodes. The example used by (Cardei *et al* 2005b) is used here for justification. Figure 1.2 (a) shows an example of four sensors s_1, s_2, s_3, s_4 and three targets t_1, t_2, and t_3. Each sensor has two sensing ranges r_1 and r_2 where $r_1 < r_2$. A node's sensing area is the disk centered at the sensor, with a radius equal to the sensing range. A solid line denotes the range r_1 and a dotted line denotes the range r_2. The coverage relationships between sensors and targets are illustrated in figure 1.2 (b): $(s_1, r_1) = \{t_3\}$, $(s_1, r_2) = \{t_1, t_3\}$, $(s_2, r_1) = \{t_2\}$, $(s_2, r_2) = \{t_1, t_2\}$, $(s_3, r_1) = \{t_2\}$, $(s_3, r_2) = \{t_2, t_3\}$, $(s_4, r_1) = \{t_1, t_3\}$ and $(s_4, r_2) = \{t_1, t_2, t_3\}$. The initial energy at each sensor E=2. The energy required for one unit of time with range r_1, e_1=0.5. The energy required for one unit of time with range r_2, e_2=1.

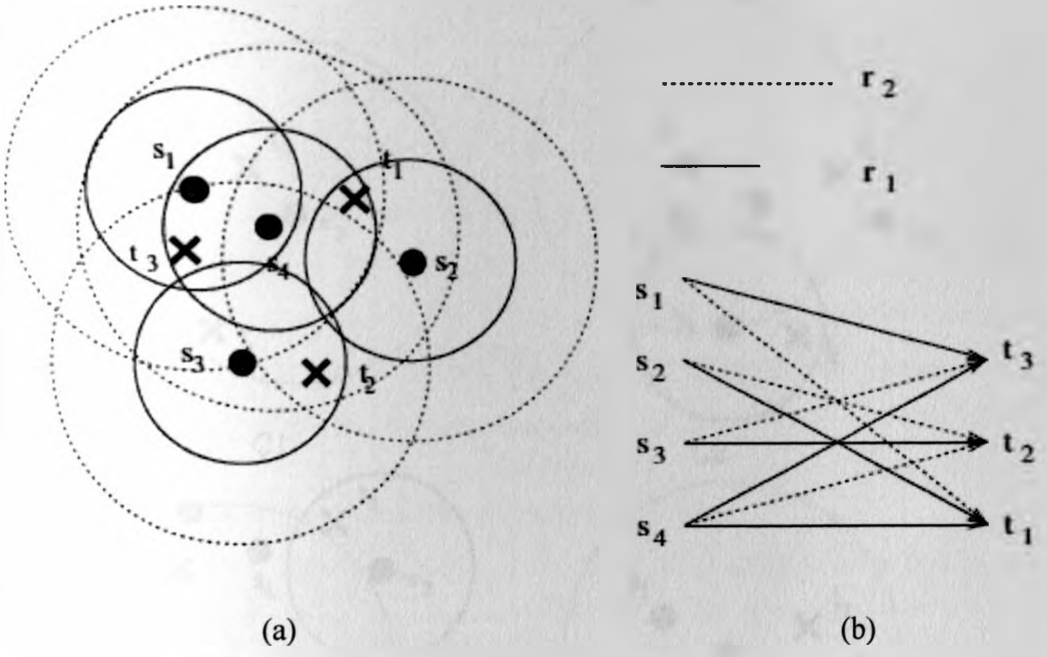

Figure 1.2: Sensor network with four sensors and three targets
(Source: Cardei *et al.* 2005)

In the sensor network shown above, a sensor can be part of more than one cover set and five different cover sets can be obtained using the combinations of the two sensing ranges (r_1 and r_2): C_1= {(s_1, r_1), (s_2, r_2)}, C_2= {(s_1, r_2), (s_3, r_1)}, C_3= {(s_2, r_1), (s_3, r_2)}, C_4= {(s_4, r_2)}, C_5= {(s_1, r_1), (s_2, r_1), (s_3, r_1)}. These five cover sets are illustrated in figure 1.3. Each cover set is active for a unit time of 1 thus giving a maximum lifetime of 6 when using the sequence of cover sets: C_1, C_2, C_3, C_4, C_5 and C_4. After this sequence, the residual energy of each sensor becomes zero as illustrated in table 1.1.

C_1	$S_1 = 2 - 0.5 = 1.5$	$S_2 = 2 - 1 = 1$	
C_2	$S_1 = 1.5 - 1 = 0.5$	$S_3 = 2 - 0.5 = 1.5$	
C_3	$S_2 = 1 - 0.5 = 0.5$	$S_3 = 1.5 - 1 = 0.5$	
C_4	$S_4 = 2 - 1 = 1$		
C_5	$S_1 = 0.5 - 0.5 = 0$	$S_2 = 0.5 - 0.5 = 0$	$S_3 = 0.5 - 0.5 = 0$
C_4	$S_4 = 1 - 1 = 0$		

Table 1.1: Sequence of five cover sets with maximum lifetime of 6 time units

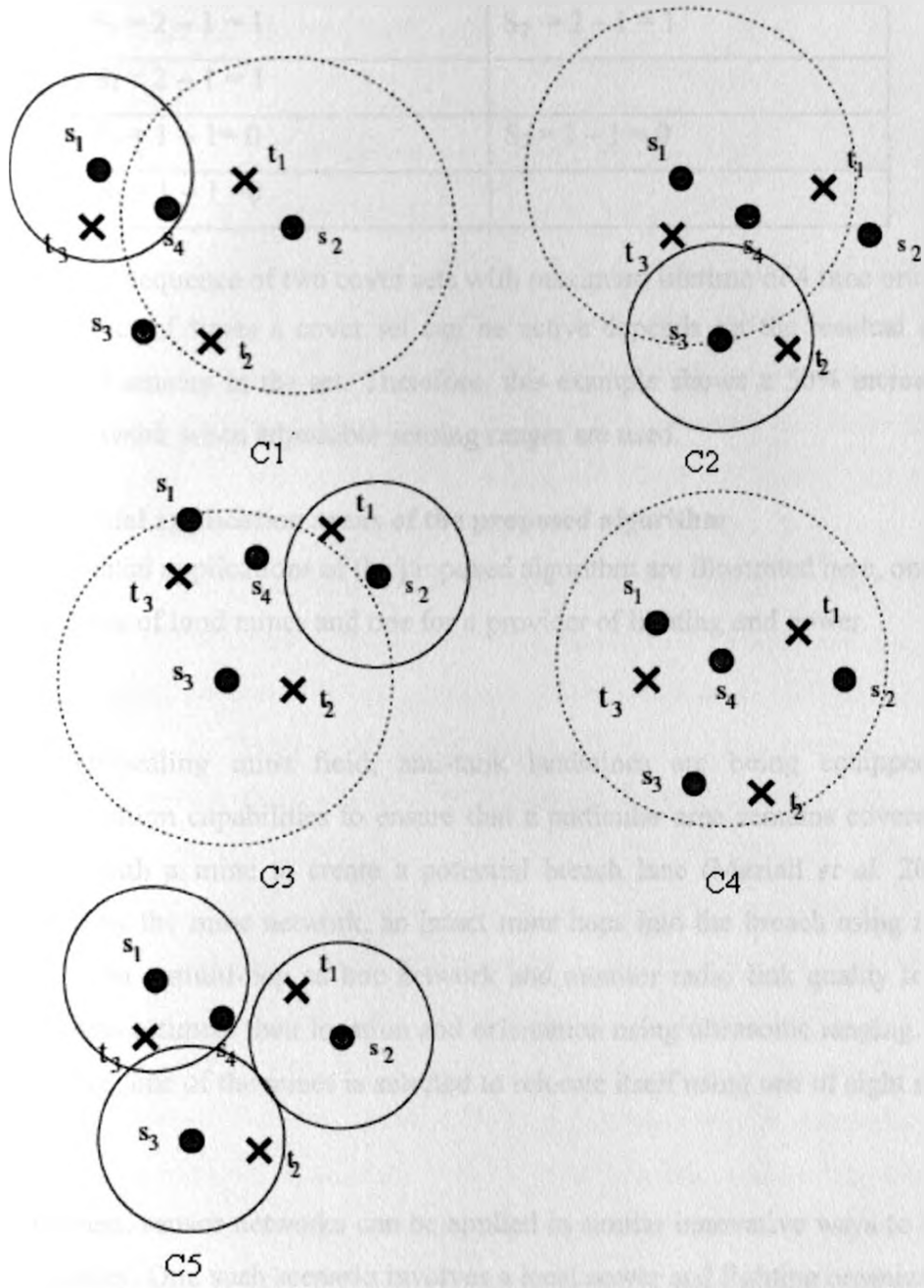

Figure 1.3: Five set covers (C_1, C_2, C_3, C_4, and C_5) with different sensing ranges.
(Source: Cardei *et al.* 2005)

If the sensor nodes do not have adjustable sensing ranges, then we obtain a lifetime of 4 when using sensing range r_2. The sensors can be organized in two distinct cover sets, C_1= {(s_1, r_2), (s_2, r_2)} and C_2= {(s_4, r_2)} with each cover set being active twice as illustrated in table 1.2.

C_1	$S_1 = 2 - 1 = 1$	$S_2 = 2 - 1 = 1$
C_2	$S_4 = 2 - 1 = 1$	
C_1	$S_1 = 1 - 1 = 0$	$S_2 = 1 - 1 = 0$
C_2	$S_4 = 1 - 1 = 0$	

Table 1.2: Sequence of two cover sets with maximum lifetime of 4 time units

The number of times a cover set can be active depends on the residual energy values of the individual sensors in the set. Therefore, this example shows a 50% increase in lifetime of the sensor network when adjustable sensing ranges are used.

1.9 Potential application areas of the proposed algorithm

Two potential applications of the proposed algorithm are illustrated here, one involving military surveillance of land mines and one for a provider of lighting and power.

In a self-healing mine field, anti-tank landmines are being equipped with sensing and communication capabilities to ensure that a particular area remains covered even if the enemy tampers with a mine to create a potential breach lane (Mariall *et al.* 2003). If tampering is detected by the mine network, an intact mine hops into the breach using a rocket thruster. The mines form a multi-hop ad hoc network and monitor radio link quality to detect failed mines. Nodes also estimate their location and orientation using ultrasonic ranging. When a node failure is detected, one of the mines is selected to relocate itself using one of eight rocket thrusters

In business, sensor networks can be applied in similar innovative ways to improve services and save money. One such scenario involves a local power and lighting organization. Today, in order to determine which of thousands of street lights are out or in need of service the power company periodically surveys the lights after sunset or waits for a customer complaint. If the organization monitored its service area with thousands of cheap, light-sensor equipped motes, they can immediately pinpoint the location of non-working lights without incurring the labor and transportation cost of a physical survey. Repairs can be organized in a more systematic manner, complaint calls can be reduced and customers will be more satisfied (Wired 2003).

CHAPTER 2: LITERATURE REVIEW

2.1 Background

Wireless sensor nodes have been in existence for decades and have been used for applications as diverse as earthquake measurements to military surveillance. The main driving force behind WSNs development has been the military. In 1978, the Defense Advanced Research Projects Agency (DARPA) organized the Distributed Sensor Nets Workshop (DAR 1978), focusing on sensor network research challenges such as networking technologies, signal processing techniques and distributed algorithms. DARPA also operated the Distributed Sensor Networks (DSN) program in the early 1980s, which was then followed by the Sensor Information Technology (SensIT) program. In collaboration with the Rockwell Science Center, the University of California at Los Angeles proposed the concept of Wireless Integrated Network Sensors or WINS (Pottie 2001). One outcome of this project was the Low PowerWireless Integrated Microsensor (LWIM), produced in 1996 (Bult *et al.* 1996). This smart sensing system was based on a CMOS chip, integrated multiple sensors, interface circuits, digital signal processing circuits, wireless radio, and microcontroller onto a single chip.

The Smart Dust project (Kahn *et al.* 1999) at the University of California at Berkeley focused on the design of extremely small sensor nodes called *motes.* The goal of this project was to demonstrate that a complete sensor system can be integrated into tiny devices, possibly the size of a grain of sand or even a dust particle. The PicoRadio project (Rabaey *et al.* 2000) by the Berkeley Wireless Research Center (BWRC) focuses on the development of low-power sensor devices, whose power consumption is so small that they can power themselves from energy sources of the environment such as solar or vibrational energy. The MIT μAMPS (micro-Adaptive Multidomain Power-aware Sensors) project also focuses on low-power hardware and software components for sensor nodes, including the use of microcontrollers capable of dynamic voltage scaling and techniques to restructure data processing algorithms to reduce power requirements at the software level (Calhoun *et al.* 2005).

Physical sensor nodes have been able to increase their capability in conjunction with Moore's Law. The chip footprint contains more complex and lower powered microcontrollers. The existing wireless network can be classified into two categories: infrastructure and ad-hoc

network. The infrastructure network architecture contains a wired backbone which is connected to a base station. Communication between wireless hosts must go through the base station. Ad hoc wirelesses network on the other hand is a self-organizing system with wireless hosts that do not depend on any fixed network infrastructure. The nodes in a WSN are generally stationary and self-organized into a network. They perform sensing tasks and send the information to a control center or base station, where the end-user can retrieve the data. In table 2.1 (Dargie & Poellabauer 2010) highlight the major differences between traditional networks and WSNs.

Traditional networks	Wireless sensor networks
General-purpose design; serving many applications.	Single-purpose design; serving one specific application.
Typical primary design concerns are network performance and latencies; energy is not a primary concern.	Energy is the main constraint in the design of all node and network components.
Networks are designed and engineered according to plans.	Deployment, network structure, and resource use are often ad hoc (without planning).
Devices and networks operate in controlled and mild environments.	Sensor networks often operate in environments with harsh conditions.
Maintenance and repair are common and networks are typically easy to access.	Physical access to sensor nodes is often difficult or even impossible.
Component failure is addressed through maintenance and repair.	Component failure is expected and addressed in the design of the network.
Obtaining global network knowledge is typically feasible and centralized management is possible	Most decisions are made localized without the support of a central manager

Table 2.1: Comparison of traditional networks and wireless sensor networks

Self-organized and self-configurable wireless sensor networks have attracted a great deal of interest from researchers and practitioners in recent years for their invaluable potentials in military and civilian applications where distributed sensing, collection and dissemination of information is required (Akyildiz *et al.* 2002). The self organizing and dynamic nature of

wireless sensor networks require energy efficient algorithms and protocols which are capable of handling massive amounts of data generated by the sensors. Wireless sensor network design is influenced by many factors, which include fault tolerance, scalability, production costs, operating environment, network topology, hardware constraints, transmission media, and power consumption (Chong & Kumar 2003). These factors serve as a guideline in design of protocols and algorithms for wireless sensor networks. The computational and energy resources of a sensor node are limited due to size and weight restriction. They are also limited in terms of computation power, storage and support short- range wireless communication. Sensors may be deployed in hostile and inaccessible environments where they are subjected to changing conditions and there is a possibility of them being destroyed as a result of the harsh environmental conditions. Due to the frequent changes in sensor accessibility and power availability the network configuration will be subjected to frequent changes. The limited power supply is the most critical issue in wireless sensor networks. Therefore, the efficient use of available energy resources directly impacts the lifetime and performance of wireless sensor networks and it is very important that the algorithms and protocols used in wireless sensor networks must optimize the sensor energy utilization (Cardei *et al.* 2005a). A typical wireless sensor node can be in four communication modes: transmit, receive, idle or sleep and two states during monitoring: idle and active (Brinza & Zelikovsky 2006).

The sensor node power can be conserved by allowing some sensors to sleep while other sensors are covering the monitoring region of interest. In (Cardei *et al.* 2005a) other energy saving techniques are identified which include:

- Using energy efficient routing and data gathering techniques
- Adjusting the transmitting range of a sensor,
- Reducing the amount of data transmitted and
- Avoiding useless activities.

In this project our goal is to investigate distributed algorithms for energy efficient target coverage. We investigate the situation where sleep-wake scheduling is used to define the communication and monitoring activities of a sensor and there is adjustment of sensing ranges within the WSN. We propose a reliable distributed scheduling algorithm which can smoothly vary sensing range while providing optimal target coverage with a minimal set of active sensors.

2.2 Literature Review and Theory

Energy saving in wireless sensor networks has attracted a lot of attention in the recent years and introduced unique challenges compared to traditional wired networks. Extensive research has been conducted to address these limitations by developing schemes that can improve resource efficiency. Mao *et al.* (2009) proposes an EDGA algorithm to achieve good performance in terms of lifetime by minimizing energy consumption for in-network communications and balancing the energy load. It is based on weighted election probabilities of each node to become a cluster head, which can better handle the heterogeneous energy capacities and adopt a simple but efficient method to solve the area coverage problem in a cluster range. Lee, Yoo and Chuan (2004) suggest a new clustering algorithm CODA. In order to mitigate the unbalance of energy depletion caused by different distance from the sink, CODA divides the whole network into a small number of groups based on the distance from the base station and the strategy of routing and each group has its own number of cluster members and member nodes. The farther the distance from the base station, the more clusters are formed in case of single hop with clustering. It shows better performance than applying the same probability to the whole network in terms of the network lifetime and the dissipated energy.

Heinzelman, Chandrakasan and Balakrishnan (2002) proposed LEACH, a substitute clustering based algorithm. In order to save energy, LEACH deals with the heterogeneous energy condition where the node with higher energy should have larger probability of becoming the cluster head. Each sensor node must have an approximation of the total energy of all nodes in the network to compute the probability of becoming a cluster head but it cannot make the decision of becoming a cluster head only by its local information, so the scalability of this scheme will be influenced. Lindsey and Raghavendra (2002) proposed an algorithm based on chain, which uses a greedy algorithm to form a data chain. Each node, aggregates data from downstream node and sends it to upstream node along the chain and communicates only with a close neighbor and takes turns transmitting to the base station, thus reducing the amount of energy spent per round. Younis and Fahmy (2004) discuss a HEED clustering algorithm which periodically selects a cluster head based on the node residual energy and node degree and a secondary parameter, such as node proximity to its neighbors or node degree. The clustering process terminates in $O(1)$ iterations and it also achieves fairly uniform cluster head distribution across the network and selection of the secondary clustering parameter can balance load among cluster heads.

Kim *et al.* (2008) introduce a cluster head election method using fuzz logic to overcome the defects of LEACH. They inquired that the network lifetime can be prolonged by using fuzz variables in homogeneous network system, which is different from the heterogeneous energy consideration. Recently (Kumar, Aseri & Patel January 2009, March 2009), authors suggested the impact of heterogeneity of nodes in terms of their energy that are hierarchically clustered in WSNs and initiate an energy efficient heterogeneous clustered method for WSNs based on weighted election probabilities of each node to become a cluster head according to the residual energy in each node. For this they suppose a percentage of the population of sensor nodes is equipped with the additional energy resources.

The works of Cardei *et al* (2005b), Dhawan *et al.* (2006) and Lu *et al.* (2009) deal with the target coverage problem where the sensors can adjust their sensing range in order to conserve energy. In (Cardei *et al* 2005b), the authors divide the sensors into cover sets, where the sensors of each cover set adjust their sensing range in order to avoid covering the same targets two or more times. The authors examine the case where the nodes' sensing range has p steps, while they target to maximize the number of cover sets. They propose a Linear Programming (LP) solution to the target coverage problem for non-disjoint cover sets. Although the LP algorithm presents a high complexity $O(m^3n^3)$, where m is the number of cover sets and n the number of sensors, the authors also proposes a greedy algorithm with a lower complexity $O(dk^2n)$, where d is the number of sensors that cover the most poorly covered targets and k is the number of targets. They assume a linear and an exponential energy consumption model for the sensing operation. The simulation results show that the consumed energy can be reduced in half compared to the approach where the sensors have the longest possible sensing range.

Unlike Cardei *et al* (2005b), in (Dhawan *et al.* 2006), it is assumed that each sensor has an infinite number of options concerning its sensing range. The authors propose an approximate algorithm to solve this problem based on Garg and Könemann algorithm (Garg & Könemann 1998). Their simulation results show a significant improvement over the distributed algorithm of Cardei *et al* (2005b). The work of Lu *et al.* (2009) presents an extension of the work of Cardei *et al* (2005b), where given a set of sensors and targets, a sensor can watch only one target at a time.

The objective is to schedule sensors to monitor targets, such that the lifetime of the surveillance system is maximized, where lifetime is defined as the duration of time when all targets are covered. This problem does not belong to the NP class, as it can be solved in polynomial time. The authors present a distributed algorithm that builds a virtual backbone first to satisfy network connectivity, and they ensure coverage based on that backbone. In providing such a virtual backbone, the authors first construct a connected dominating set and prune redundant sensors by applying the Rule-k algorithm (Dai & Wu 2003).

Cardei *et al* (2005a) introduce an energy efficient target coverage mechanism for sensor networks. The idea is to extend the network lifetime by organizing sensors into the maximal number of set covers. These set covers are activated successively such that at any given time only a set is active. The nodes from the active set will be in the active state while all the others will be in the sleep state. A key difference between this approach and (Slijepcevic & Potkonjak 2001) is that the sensor nodes can participate in multiple sets (the covers do not contain mutually exclusive nodes). The single restriction is that the sum of all time weights associated with the sets a node belongs to has to be 1. Thus, in a complete cycle each node consumes the same amount of energy. They formalize the Maximum Set Covers (MSC) problem and they prove it to be NP-complete. In order to find the solutions two heuristics are used: a linear programming (LP) heuristic and a greedy heuristic. The greedy heuristic has a lower complexity than LP and thus a lower execution time. The evaluation confirms that Greedy performs slightly better and that it is more scalable for large sensor networks. They also show that for a specific number of targets the network lifetime increases with the increase of the number of sensors and the increase of sensing range. On the other hand, by using a constant number of sensors and sensing ranges they observe that the lifetime decreases with the increase of the number of targets.

It has also been proven in (Cardei & Du 2005) that the disjoint set coverage problem is NP-complete and has a lower approximation bound of 2 for any polynomial time approximation. In (Berman *et al.* 2004, 2005 and Cardei *et al* 2005a) the disjoint set cover problem is further extended by not requiring the sensor sets to be disjoint (i.e., a sensor can be active in more than one sensor set) thereby, allowing the sets to operate for different time intervals. It is also shown in (Cardei *et al* 2005a) that non-disjoint sensor covers can provide better lifetime when compared to disjoint set covers.

CHAPTER 3: ENERGY MANAGEMENT IN WIRELESS SENSOR NETWORKS

Energy management in WSNs is of critical importance due to the energy limitation inherent in the sensor network. A sensor's durability and reliability depend on its battery's capacity and on the energy consuming tasks it performs in order to fulfill its functions. Energy is a scarce resource in every wireless device but, the problem is amplified in sensor networks due to the following reasons: (Dargie & Poellabauer 2010)

a) Compared to the complexity of the sensing, processing, self management and communication tasks carried out by sensor networks, the sensor nodes are very small in size to accommodate high capacity power supplies required by these tasks.
b) WSNs consist of thousands of sensor nodes in a distributed environment making manual changing, replacement or recharging of batteries almost impossible.
c) While the research community is investigating the contribution of renewable energy and self-recharging mechanisms, the size of the nodes still remains a constraining factor.
d) The failure of a few nodes in a WSN can adversely affect network performance resulting in premature fragmentation of the network or death of the whole network.

The problem of energy consumption can be approached from two perspectives. The first is to develop energy efficient, self - organizing communication protocols which take into account the unique characteristics of WSNs. The second approach is to identify activities in the network which are both wasteful and unnecessary and mitigate their impact.

3.1 Sources of energy wastage in WSNs

During the normal working of a WSN the sensor nodes engage in various activities which result in useful as well as wasteful energy consumption (Dargie & Poellabauer 2010). The activities that relate to useful energy consumption can be due to data transmission or reception, query requests processing and forwarding of queries and data to neighboring nodes. Wasteful and unnecessary activities can be limited to a sensor node or can have a network wide scope. In both cases these activities can be viewed as accidental side-effects or can be as a result of non-optimal software and hardware implementations. Jiang *et al.* (2007) presents observations based on field deployment which reveal that some nodes exhausted their batteries prematurely as a result of

unexpected overhearing of traffic that caused the communication subsystem to become operational for a longer time than originally intended. Similarly, other nodes exhausted their batteries prematurely because they aimlessly attempted to establish links with a network that was no longer accessible to them.

Most of the wasteful and unnecessary activities can be attributed to non-optimal configuration and implementation of hardware and software components. The four major sources of energy waste are: (Ye, Heidemann & Estrin 2002)

Idle listening – This occurs when a node is listening to the channel in order to receive possible traffic, but is neither transmitting nor receiving any data. This process typically consumes much more energy than if the node were to sleep.

Packet collision – When packets collide, energy is wasted because the packet along with associated control overhead must be retransmitted. Thus, the energy used to transmit and receive the colliding packets is wasted. Collisions can be reduced by scheduling or deferring for random intervals from medium access.

Overhearing – This occurs when a node promiscuously listens to packets which are destined for other nodes. Overhearing unnecessary traffic can be a dominant factor of energy waste when traffic load is heavy and node density is high. In this case, the node could sleep rather than idly listening to the channel.

Control packet overhead – Aside from the data packet, which has additional header bytes prepended, usually other MAC layer packets add overhead. For example, an acknowledgement packet is usually required from the receiver to verify whether the data transmission was a success or a failure. A minimal number of these packets should be used to make a data transmission so as to reduce control packet overhead and conserve energy since transmission, reception and listening to these packets will result in energy consumption. Over-emission which is caused by transmission of a message when the destination node is not ready should be avoided as it wastes energy.

3.2 Energy conservation techniques in WSNs

The first step toward developing energy conservation techniques and strategies is the understanding of how energy is consumed by the different subsystems of the wireless sensor node. This knowledge enables wasteful activities to be identified and avoided as well as enabling one to estimate the overall energy dissipation rate in a sensor node and how this rate affects the lifetime of the entire network. It is therefore critical to minimize the energy consumption of sensor nodes while meeting the application requirements. Several energy techniques have been identified in literature, but, for the purpose of this study we will concentrate on duty cycling or sleep – wake scheduling.

3.2.1 Sleep – Wake scheduling

In order to extend the working lifetime of individual devices in a sensor network, it is common practice that some node elements are deactivated, including the radio transceiver which contributes significantly to the node's overall energy consumption. These devices should remain inactive for most of the time and should only be activated when they are required to transmit or receive messages from other nodes. The radio transceiver can operate in one out of four modes (Shwe, Jiang & Horiguchi 2009), which differ in the consumption of energy necessary for proper operation. During transmission mode when signals are transmitted to other nodes the greatest energy is consumed, approximately 80 mW. Approximately the same energy is consumed in the reception of messages from other nodes and idle listening mode when the transceiver is inactive, turned on and ready to change to data transmission or receiving state (i.e. 30 mW). In the sleep mode the radio transceiver is off and therefore consumes the least energy, approximately 0.003 mW.

In event detection applications where occurrence of events is rare, the sensors spend most of the time in idle mode which consumes energy and effectively reduces the lifetime of the network. To mitigate such kind of a scenario it is prudent to turn the radio off and only activate it when an actual event occurs so as to conserve energy and extend the network lifetime. This is made possible by the use of sleep – wake scheduling whereby sensors alternate between sleeping and waking periods. Only the waking sensors actively sense events in the environment while the sleeping sensors avoid idle listening and overhearing. A sensor in the sleeping mode is unable to transmit packets to other nodes but it can be able to receive packets destined to it.

Active sensors on the other hand can be able to make measurements, perform computations and relay the information to base stations or sinks. Despite the fact that sleep – wake scheduling can considerable improve energy efficiency and prolong network lifetime, it introduces an additional delay when a node has to wait for its next hop neighbor to wake up which could be unacceptable for the delay sensitive applications in WSNs. Shwe, Jiang & Horiguchi (2009) classify sleep – wake scheduling under three broad categories: on-demand, scheduled rendezvous and asynchronous schemes.

a) On-demand scheduling scheme

This scheme is based on the idea that a sensor node should be woken up when a packet from the neighboring nodes is received. This minimizes energy consumption and is particularly suitable for applications like fire detection, machine failure surveillance and event driven scenarios where very low duty cycles are required. In such applications the sensor nodes are in the monitoring state for most of the time and only transition to the transmission state when an event is detected. The main aim of this scheme is to reduce energy consumption in the monitoring state while ensuring a limited latency for transitioning in the transfer state.

b) Scheduled rendezvous scheme

Individual nodes define their own sleeping schedules and share these schedules with their neighbors to facilitate coordinated sensing and efficient inter-node communication. This synchronous sleeping scheme requires that all the neighboring nodes wake up at the same time. Nodes are also required to wake up periodically to check for potential communication within the network after which they return to sleep until the next rendezvous time. The major advantage of this scheme is that it allows sending of broadcast messages since it guarantees synchronized wake up of all neighboring nodes. The main disadvantage is that neighbors need to synchronize time as well as their sleep – wake schedules which is an energy intensive process.

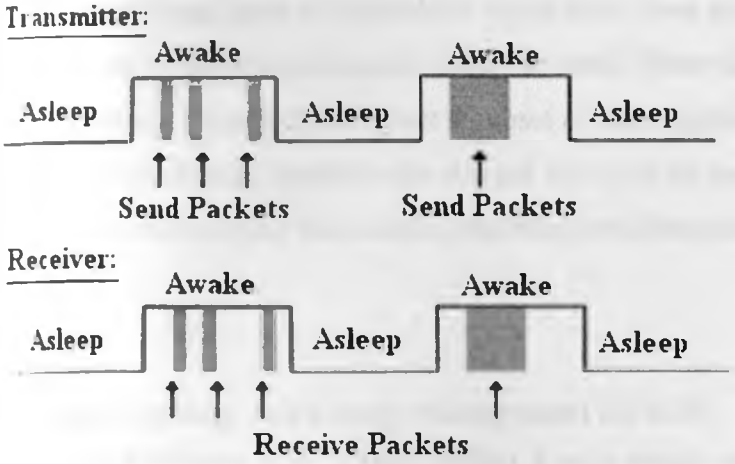

Figure 3.1: Scheduled rendezvous scheme.
(Source: Shwe, Jiang & Horiguchi 2009, p.22)

c) **Asynchronous scheduling scheme**

This scheme avoids the tight synchronization among sensor nodes required by the scheduled rendezvous scheme. Individual nodes keep their sleeping schedules to themselves and a node that initiates a communication should first send a preamble and wait for an acknowledgement from the receiving node before continuing with the transmission of the data as shown in figure 3.2. Each node is allowed to wake up independently of the others by guaranteeing that neighbors always have overlapping active periods within a specified number of cycles. The main drawback is that it has a latency side-effect on data transmission.

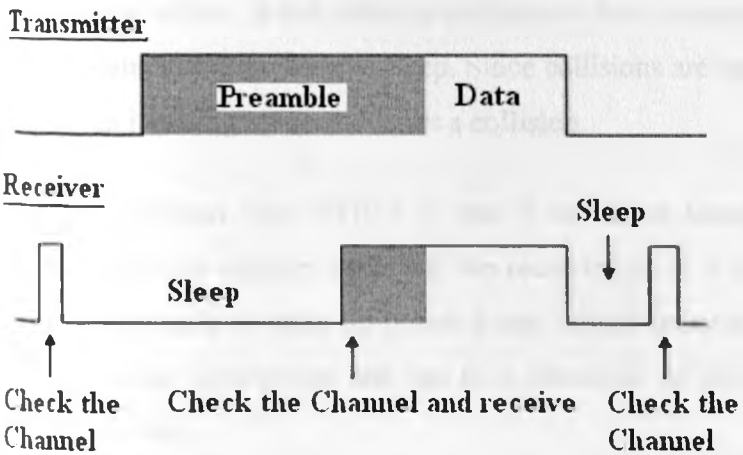

Figure 3.2: Asynchronous scheduling scheme.
(Source: Shwe, Jiang & Horiguchi 2009, p.22)

There are different types of algorithms which have been developed to help conserve energy in sensor nodes by allowing redundant nodes to sleep. These algorithms determine which nodes in the network are redundant and select a subset of these nodes to sleep for a predefined period of time, after which they should wake up and take part in the network activities. Some of these algorithms which employ sleep-wake scheduling are discussed below:

3.3 Sparse Topology and Energy Management (STEM)

In STEM (Schurgers *et al.* 2002a, 2002b), a node spends most of its time monitoring data and only powers on the radio transceiver when it needs to forward some data to neighboring nodes. Bearing in mind that the radio is the largest energy consumer in a WSN, turning it off when it is not in use will conserve the limited energy resource. The more time spend by the network in monitoring environmental conditions, the more energy STEM will save. STEM adds a second radio to the sensor nodes. The primary radio is used for the reception and transmission of messages, whereas the second radio is a low duty cycle radio used for the transmission of "wakeup messages". Wakeup messages are messages that are periodically sent out to neighboring nodes informing them that the sending node would like to communicate with them and therefore they must wake up for communication to be possible. A sleeping node will periodically wake up to see if another node is trying to communicate with it. If a node is trying to communicate with it, it will wake up and receive the communication. If there is no node trying to communicate, it will go back to sleep. Since collisions are bound to occur, a node will also wake up if it is in listening mode and hears a collision.

The main problem with STEM is that it sacrifices latency. If a particular node wants to communicate with another node, and the receiving node is in sleep mode, the transmitting node must wait for node to wake up before it can initiate transmission of packets. This delay is very critical is some applications and can be a drawback in delay sensitive applications of wireless sensor networks.

3.4 Geographic Adaptive Fidelity (GAF)

GAF is a localized distributed topology management algorithm that allows nodes to sleep with the main aim of conserving energy (Xu, Heidemann & Estrin 2001 and Xu *et al.* 2003). It uses location information generated by a GPS device to organize redundant nodes into groups. Some of the redundant nodes will be put to sleep. It divides the network area into virtual grids. A virtual grid is "defined such that, for two adjacent grids A and B, all nodes in A can communicate with all nodes in B and vice versa" Xu, Heidemann and Estrin (2001). Since all nodes in adjacent grids can communicate with each other the nodes in these two grids are equivalent to the routing algorithm. The nodes can be in three different states, sleeping, discovery or active state. The main concept of GAF is to maintain only one node with its radio turned on per grid square. An active node has the responsibility of relaying all the network traffic on behalf of its grid square. The state transitions of GAF are illustrated in figure 3.3.

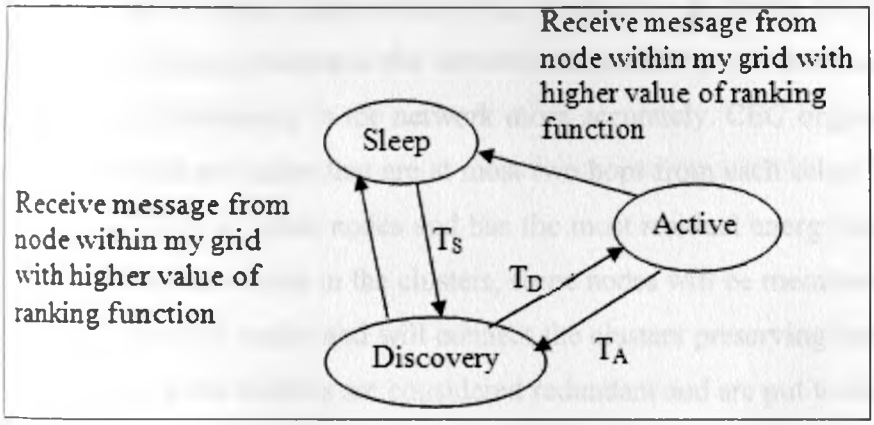

Figure 3.3: State Transitions in GAF algorithm.
(Source: Xu, Heidemann & Estrin 2001)

All nodes begin in the discovery state where the radio transceiver is turned on and is pending to switch to active state. In this state a node sends out discovery messages and receives replies back in order to determine the nodes in its same grid. The node spends a fixed amount of time T_D in the discovery state then it transitions to the active state. After spending a fixed amount of time T_A in the active state, the node switches back to the discovery state. Every time a node changes state to active or discovery, it sends out a broadcast message containing node id, grid id and the value of the ranking function.

If a node in the active or discovery state receives a message from a node in the same grid with a higher value of the ranging function, it is allowed to change its state to sleep and turns its radio off for a time T_S. The ranking function and timers T_D, T_A, T_S can be used to tune the algorithm. The ranking function usually selects nodes with the longest expected lifetime as the active nodes. GAF algorithm suffers from quite a number of drawbacks. The first drawback is that it requires location information for it to operate effectively. It guesses at connectivity instead of directly measuring it thus requiring more nodes to be active than may be necessary. GAF is also completely independent of the routing algorithm used, meaning it may allow a node to sleep even if that node is actively participating in routing causing interruptions in communication and increasing latency.

3.5 Cluster-Based Energy Conservation (CEC)

Cluster-based Energy Conservation (Xu, Heidemann & Estrin 2001) is an algorithm based on GAF but it directly measures the network connectivity, thus not requiring location information and finding redundancy in the network more accurately. CEC organizes nodes into overlapping clusters, which are nodes that are at most two hops from each other. The node in the cluster that is one hop form all other nodes and has the most residual energy selects itself to be the cluster head. Due to the overlap in the clusters, some nodes will be members of multiple clusters. These nodes are gateway nodes and will connect the clusters preserving network connectivity. The rest of the nodes in the clusters are considered redundant and are put to sleep. After a specified period of time, these nodes wake up and a cluster head will once again be selected. The role of the cluster head is rotated among all nodes in the cluster so all nodes have a chance to sleep and conserve energy. CEC is an acceptable sleep cycle management algorithm for applications where there is no location information available. It would be a good solution for network deployments that have variable radio ranges since it directly measures the network connectivity. Despite the fact that CEC conserves more energy than GAF, it does not perform well if the network topology changes frequently.

3.6 Span

Span (Chen *et al.* 2001) is based on the fact that a WSN can be connected with only a subset of the deployed nodes being active with a sufficient density of nodes. The nodes that remain active in Span are called coordinators and are used by the routing protocol. The rest of the nodes are put in power-saving sleep mode with their radios turned off. Coordinators are selected based on local information only. All nodes maintain a list of neighbors which includes a list of coordinators. This neighbor list is maintained by each node periodically broadcasting a HELLO message which states that the node is alive. This message contains information about the node, such as if it is a coordinator or not, the current coordinator list, and the current neighbor list. A node becomes a coordinator if two of its neighbors cannot reach each other. Nodes can reach each other if they can communicate directly or by using one or two coordinators.

Coordinators periodically demote themselves in order to go into power-saving mode. A demoted coordinator can no longer be designated a coordinator but will continue to participate in routing until a non-coordinator node replaces it. This means that there is no increase in latency. The replacement of coordinators is made possible by non-coordinator nodes periodically waking up and determining if there is need for them to become coordinator (if two of its neighbors cannot reach each other). Changing the role of coordinators among all the nodes helps to prolong the network lifetime. By allowing nodes to reach each other via one or two coordinators, fewer coordinators can be used, which also increases the network lifetime since more nodes can be put in power-saving mode. Span uses only local information (no centralized decisions); however, it requires nodes to send HELLO messages which uses bandwidth and energy.

3.7 Naps

Naps (Godfrey & Ratajczak 2004), is based on message broadcasts, with each node cycling through a time period. At the beginning of the time period a node broadcasts a HELLO message and sets a counter to zero. The node then goes into a listening state where it listens for HELLO messages from other nodes. Each time it receives a HELLO message from another node it increments its counter. When the counter reaches a threshold value, which is a parameter to the algorithm, the node goes to sleep. When the time period expires, the node will wake up and start the time period again. If the node's counter does not reach the specified threshold then the node

remains awake for the time period. Naps algorithm does not require any location information but, it does introduce extra traffic into the network with its use of HELLO messages. It also does not minimize the number of awake nodes since it assumes reliable transmission of the HELLO messages, which is not always the case due to collisions. The role of which nodes are put to sleep is not rotated as the nodes that are put to sleep are just the ones that receive the threshold number of HELLO messages from the neighbors. Due to the proximity of the nodes to each other, it may always be the same nodes that are allowed to be put to sleep. As network density increases, more nodes will be put to sleep; therefore, low density networks may not benefit from Naps since few nodes will be allowed to sleep meaning minimal energy savings.

CHAPTER 4: DISTRIBUTED SCHEDULING ALGORITHMS

In this chapter we present a discussion of the Load Balancing Protocol for sensing (LBP) with fixed sensing range and propose an enhanced Distributed Scheduling Algorithm with adjustable sensing range (DSA) which is an extension of LBP.

4.1 Load Balancing Protocol for Sensing (LBP)

The main idea of LBP (Brinza & Zelikovsky 2006) is that by means of load balancing a maximum number of sensors are kept alive for as long as possible by ensuring that if a certain sensor is being overused as compared to its neighbors, then it should get a break and be allowed to sleep. Sensors are allowed to freely exchange *on* and *off* states. Each sensor node can be in three different states:

***On state*:** the sensor is actively monitoring its targets and does not waste energy.

***Off state*:** idle and sleepy modes, the sensor stops wasting any energy.

***Alert state*:** active and listening mode, the sensor monitors its targets and should change its state to either active or idle in order to conserve energy.

Each alert sensor knows all the states its neighbors are on due to the fact that any state transition is immediately broadcasted to the neighbors together with the current energy supply. When a sensor is in the *Alert state* there are two rules which describe its state transitions:

1) **On – rule** – whenever a sensor s has a target that is solely covered by it and there is no other *on* or *alert* sensor covering it, it should change its state to *on* and cover the target.

2) **Off – rule** – whenever a target covered by a sensor s is also covered by another *on* or *alert* sensor with a larger battery supply, s should change its state to *off*.

For a certain period of time depending on the energy spent on communication, all nodes are alerted using wake-up calls and each sensor has to decide whether to change its state to *on* or *off* in a process called global reshuffle. During global reshuffle, each sensor sends out two broadcasts to its neighbors. The first broadcast includes the targets it is covering and the battery supply (energy level). The second broadcast informs the neighbors of the state transition it will be taking, either *on state* or *off state*. If an active sensor exhausts its energy and is about to die, it sends out a wake-up call to its neighbors. A minimal subset of neighbors in *off state* changes their state to *on* and effectively substitute the dead sensor.

It has been shown by Berman *et al.* (2004) that LBP satisfies the following properties:

a) Each global reshuffle of LBP requires two broadcasts to the neighbors and the resultant set of all active sensors forms a minimal sensor cover.

b) LBP is a reliable protocol. This is due to the fact that the **off-rule** allows a sensor to switch itself off only if it does not have the maximum battery supply for any of its targets covered by on sensors with greater battery supply. This means that, for any target, there is a sensor covering it with the largest battery power.

"The main drawback of LBP is that it balances the load of sensors (while it does not matter if they are alive or dead) instead of balancing the energy of sensors covering the same target. Brinza & Zelikovsky (2006)."

4.2 Proposed Distributed Scheduling Algorithm with adjustable sensing range (DSA)

The main objective of this algorithm is to maximize the time that the sensors can actively monitor the targets by using sleep-wake schedules and smooth adjustment of the sensing range. DSA should be able to keep as many live sensors as possible by means of load balancing and allowing them to die simultaneously. There are three questions which should be answered by the distributed scheduling algorithm with adjustable sensing range:

a) What rules should be used by each sensor node to decide whether to turn on or off?

b) When a sensor decides to turn on and be active, what should be its sensing range?

c) When should nodes make such kind of decisions?

To answer these three questions, we describe the states and the transition rules of the sensors. Each sensor can be in one of three different states at any point in time.

- **Active:** the sensor monitors its targets.
- **Idle:** the sensor sleeps and does not monitor any targets.
- **Vulnerable:** the sensor monitors its targets and should change its state to either active or idle state soon.

We assume that each sensor can communicate with its neighbors within two times of the maximum sensing range. Initially each sensor broadcasts its energy level and covered targets to its neighbors; this facilitates the formation of the sensor cover schedule. Each sensor will change its state based on the following transition rules. When a sensor is in the vulnerable state with range r, then it should change its state to

- **Active state** with sensing range *r* if there is a target at range *r* which is not covered by any other active or vulnerable sensors.
- **Vulnerable state** but decrease its sensing range to the next furthest target if all targets at range r are covered by other active or vulnerable sensor with greater energy supply.
- **Idle state** if its sensing range decreases to zero.
- **Terminated state** if its energy level gets completely exhausted.

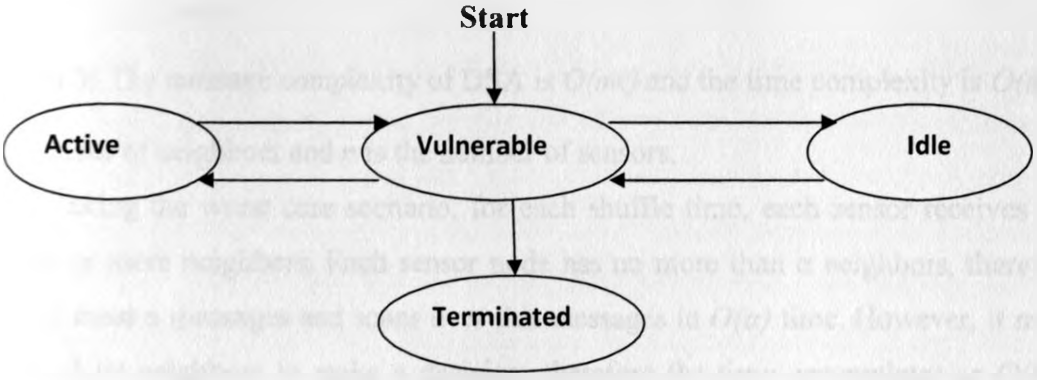

Figure 4.1: State transitions in the proposed algorithm.

Once all the sensors have made a decision to be active or idle, they will stay in that state for a period of time called shuffle time or until there is an active sensor which exhausts it energy and is going to die. This dying sensor sends a wake-up call to all the sensors in the network causing them to transition to the vulnerable state with their maximum sensing range. When there is a target that cannot be covered by any sensor in the network, the network fails.

As in Berman *et al.* (2005), we can show that DSA satisfies the following properties:

Theorem 1: Each global reshuffle needs 2 broadcasts (to the neighbors) from each sensor and the resulted set of all active sensors forms and minimal sensor cover.

Proof: Taking the assumption that each sensor knows the coordinates of all its neighbors and all the targets to be monitored. For each reshuffle there is one broadcast informing the neighbors of the sensors available energy level and the covered targets and the second broadcast informs the neighbors that the sensor changes its state to one of the 3 non- vulnerable states. The resulting set of active sensors is minimal and covers all targets since a sensor can change its state to idle only when its sensing range reaches zero, that is, all of its targets are covered by other active sensors with greater energy supply.

Theorem 2: DSA is a reliable protocol.

Proof: Following the fact that the active sensors form a minimal sensor cover as shown in theorem 1 above, we can show that DSA is a reliable protocol. In DSA, a sensor can only change its state to idle when its sensing range reaches zero, meaning it will only sleep if all its targets are covered by other active sensors with greater energy supply. Therefore, for any target i, there is a sensor with the largest energy supply to cover it.

Theorem 3: The message complexity of DSA is $O(n\alpha)$ and the time complexity is $O(\alpha^2)$ where α is the number of neighbors and n is the number of sensors.

Proof: Taking the worst case scenario, for each shuffle time, each sensor receives a message from one or more neighbors. Each sensor node has no more than α neighbors, therefore, it can receive at most α messages and scans over this messages in $O(\alpha)$ time. However, it may have to wait on all its neighbors to make a decision, therefore the time accumulates as $O(\alpha + \alpha + \alpha + \ldots.. + \alpha)$ in the decision phase. This implies that the time complexity is $O(\alpha^2)$. Each sensor has at most α neighbors and can broadcast at most two messages to its neighbors per shuffle time, so each sensor sends at most $O(\alpha)$ messages in the decision phase. This means that the message complexity is $O(n\alpha)$.

CHAPTER 5: DESIGN AND IMPLEMENTATION

5.1 Methodology

The network model used in this project is similar to the models described in (Cardei *et al.* 2005b and Dhawan *et al.* 2006). We assume that sensors are deployed over a monitoring region where each sensor is in charge of monitoring assigned targets from where it can collect data without the help of any other sensor. We also assume that each sensor knows its own coordinates as well as the coordinates of the targets it is covering. All the sensors are able to smoothly vary their sensing range and they largely exceed the number of targets to be monitored so that some sensors can turn themselves off and go into low power sleep mode and save energy. A sleeping sensor cannot hear any packets but can be woken up using wakeup mechanisms. To ensure that the network is fault tolerant, the sensors should be able to broadcast just before the battery exhaustion so that the neighboring sleep nodes can wake up and replace the dead sensor. For any target there is at least one sensor which cannot become idle unless there is an active sensor covering that particular target. This requirement in the network model guarantees the correctness of the distributed algorithm and also ensures complete target coverage.

We will test the performance of the distributed algorithm by simulating it over a wide range of simulation parameters. Two energy models will be used as defined by Cardei *et al.* 2005b. The linear model defines the energy e_d needed to cover a target at distance d as $e_d = \Theta(d)$ The quadratic model defines the energy e_d needed to cover a target at distance d as $e_d = \Theta(d^2)$. In the simulation the adjustable parameters will be the number of sensors, the number of targets and the sensing range. The simulator will take inputs like, target id, sensing range, shuffle time, maximum battery power, sensor id and sensor position. The sensors will be able to change their sensing range based on the number of targets to be covered and the sensing range of the neighboring sensors. The sensors communicate through packets and alternate between sleep and active mode. At any one time all targets should be covered and the sensing range of each sensor should be minimal so as to conserve the sensors power. The simulation will terminate when a target cannot be covered by any of the sensors in the network. The output will be the network lifetime, which is the performance measure.

5.2 Design

To evaluate the performance of the proposed Distributed scheduling algorithm with adjustable sensing range and to make comparison with the Load balancing protocol described by Brinza and Zelikovsky (2006), both algorithms are implemented using Microsoft Visual Studio C# 2008 in Windows XP operating system. The target and sensor input file are generated using Research Randomizer (Version 3.0) designed by Urbaniak & Plous (2011). The main classes in DSA are highlighted in figure 5.1 below.

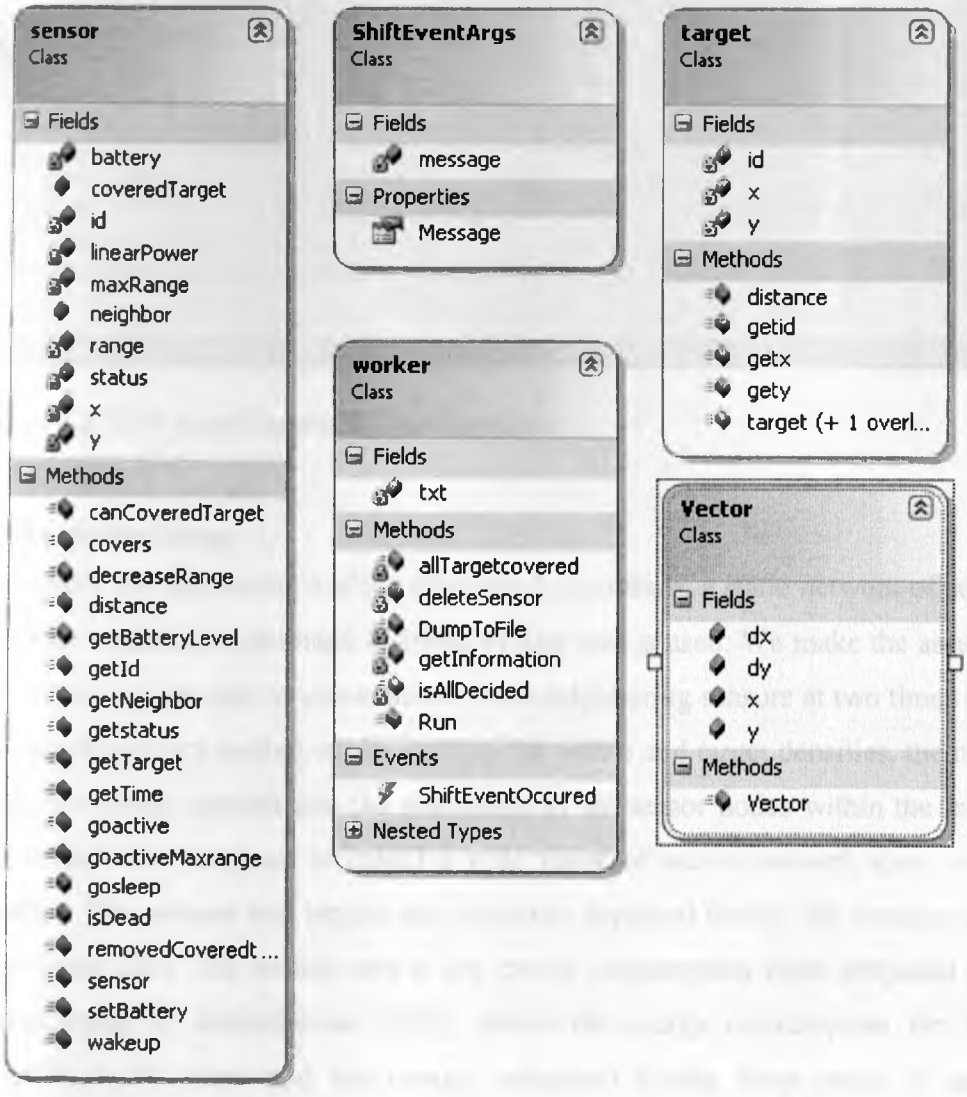

Figure 5.1 DSA Class diagram

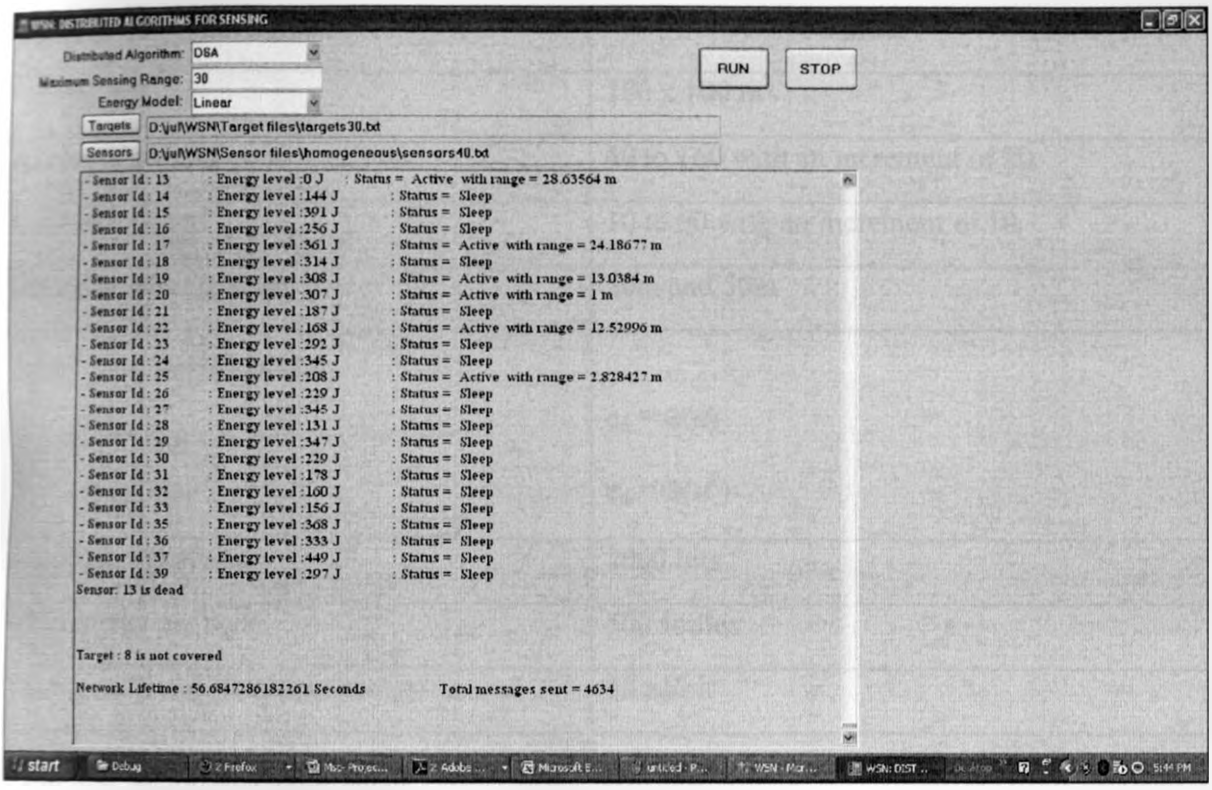

Figure 5.2 DSA main Graphical User Interface

5.3 Simulation Setup

To evaluate the performance of the distributed algorithms, a static network of sensors and targets which are randomly distributed in 100m x100m area is used. We make the assumption that each sensor node can be able to communicate with neighboring sensors at two times its sensing range. The simulations are carried out by varying the sensor and target densities, the maximum sensing range, the energy models and the placement of the sensor nodes within the monitoring region. The simulator is designed to model a wide range of sensor network sizes with varying node densities. The sensors and targets are randomly deployed during the creation of the sensor and target input files. The model uses a low power consumption radio proposed by (Heinzelman, Chandrakasan & Balakrishnan 2000), where the energy consumption for transmission and reception is 50 nJ/bit and the energy consumed during sleep mode is zero. The tunable parameters during the simulation runs are summarized in table 5.1:

Parameter	Value
Network size	100 x 100 m^2
Number of sensors	40 to 160 with an increment of 20
Number of targets	10 to 60 with an increment of 10
Maximum sensing range	30m and 50m
Energy consumption model Linear Quadratic	 $e_d = \Theta(d)$ $e_d = \Theta(d^2)$
Broadcast packet size	2000 bits
Initial energy per node	500 Joules
Transmission and reception energy	50 nJ/bit

Table 5.1: Simulation parameters.

For each of the algorithms, LBP and DSA, the following steps are required for the simulation:

1. Generate the target and sensor files which contain the target id, target x and y coordinates, sensor id, sensor energy level and the sensor x and y coordinates.
2. On the main form, select the algorithm and energy model, input the maximum sensing range and provide the location of the target file and sensor file.
3. Using these data and parameters, start the simulation.
4. The simulation runs until a target cannot be covered.
5. The simulation stops, and the lifetime of the network is generated as the final result.

5.4 Explanation of the proposed algorithm (DSA)

In DSA, each sensor node has to decide whether they can go to sleep or become active and cover the targets in the monitoring region. Each sensor knows all its neighboring sensors and covered

targets. After exchanging their energy levels and covered targets using the rules in chapter 4, each sensor makes a decision on whether to be active or idle. The simulation steps are as follows:

1. Targets and sensors are read into memory.
2. Sensor nodes are in the vulnerable state and know their neighbors and covered targets.
3. Each sensor checks with each neighboring sensor starting with the furthest target whether that target can be covered by the neighboring sensor with larger energy supply. If the neighbor can cover the target it removes that target from its target list and reduces the sensing range to the next target. If the sensing range reaches zero, the sensor goes to sleep. This process stops after all sensors make a decision.
4. After all sensors make a decision to be either active or idle, they stay in that state for a period of time called shuffle time or until an active sensor exhausts its energy and is going to die. A wake-up call is sent to all sensors causing them to transition into the vulnerable state with the maximum sensing range and repeats the process from step 3.
5. The simulation is repeated until a target cannot be covered by any of the sensors.
6. The process terminates and the lifetime of the network is printed out.

5.5 Experimental study

The main purpose of carrying out the experiments is to evaluate the performance of DSA and LBP by varying the node and target densities, the maximum sensing range, the energy models and the placement of the sensor nodes within the monitoring region, with the main aim being to find out which algorithm performs better than the other in terms of lifetime maximization. The first four simulations are aimed at finding out the impact of increase in the number of sensor nodes and the maximum sensing range on the lifetime of the sensor network. We conduct the simulations with 30 randomly deployed targets, 40 to 160 randomly deployed sensors with an increment of 20, maximum sensing range of 30m and 50m with the linear and quadratic energy model.

The fifth and sixth simulation focuses on the impact of increase in the number of targets to be monitored on the sensor network lifetime. We conduct the simulations with 60 randomly deployed targets, 40 to 160 sensors with an increment of 20, maximum sensing range of 30m with linear and quadratic energy model. The seventh also focuses on the impact of increase in the number of targets to be monitored on the sensor network lifetime. We conduct the simulation with 100 randomly deployed sensors, 20 to 50 randomly deployed targets, and maximum sensing range of 30m with the linear and quadratic energy model. The eighth simulation is conducted with 30 randomly deployed targets, 60 sensors and maximum sensing range of 30m with linear and quadratic energy model. The main aim of this simulation is to find out if sensor node placement in an area affects the network lifetime. It is usually important to ensure that enough output data have been obtained from the simulation in order to estimate the model performance with sufficient accuracy. With reference to the *rule of thumb* by Robinson (2004) where he identifies a reasonable number of replications to be at least three to five, we use five samples of 60 randomly generated sensors. The ninth simulation is conducted with 30 randomly deployed targets, 40 to 160 sensors with an increment of 20, maximum sensing range of 30m with linear and quadratic energy model. The main aim of this simulation is to evaluate the message complexity of LBP and DSA

5.6 Limitations of the design

The model considers a low power consumption radio proposed by (Heinzelman, Chandrakasan & Balakrishnan 2000) that provides a commonly used starting point, however, the model has not been verified against the behavior of a physical radio in a wireless sensor network. When computing node energy consumption, the CPU and the sensors are consumers that may or may not be neglected, depending on the nature of the application. So, the radio model must be used jointly with some figure of the energy consumption of those elements, because in the end, power supply must feed all the system and not just the radio.

CHAPTER 6: SIMULATION RESULTS AND ANALYSIS

6.1. Presentation of results

The simulations are carried out by varying the sensor and target densities, the maximum sensing range, the energy models and the placement of the sensor nodes within the monitoring region. The first simulation is conducted with 30 randomly deployed targets, 40 to 160 sensors with an increment of 20, maximum sensing range of 30m with linear energy model. The results are shown in table 6.1 and figure 6.1. The results show a significant increase in network lifetime with the increase in number of sensor nodes due to the fact that each target can be covered by more sensors. The increase in the number of nodes results in an increase in the number of redundant nodes which are put to sleep thus conserving energy and increasing network lifetime. When comparing the performance of DSA against LBP there is a significant increase in lifetime with smooth adjustment of sensing range as opposed to using fixed range sensor nodes.

Number of sensors	40	60	80	100	120	140	160
DSA	56.7	109.2	181.6	207.0	260.6	309.7	389.3
LBP	49.0	83.3	133.3	150.0	166.7	216.7	283.3

Table 6.1: Lifetime variation with 30 targets, 30m sensing range and linear energy model.

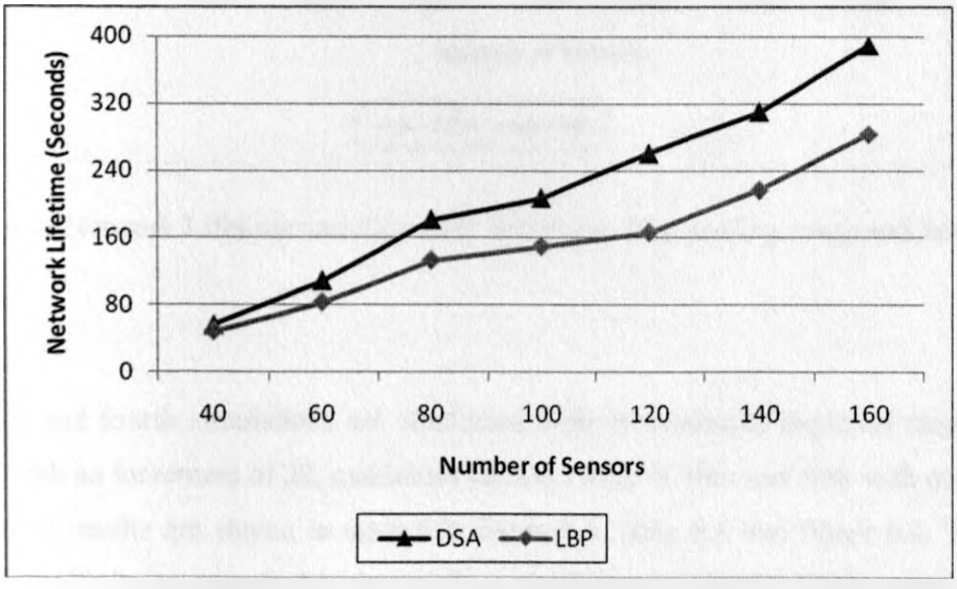

Figure 6.1: Network Lifetime variation with 30 targets, 30m sensing range and linear energy model.

In the second simulation, we increase the maximum sensing range to 50m and run the simulation using the same number of sensors, targets and linear energy model. The results show a significant increase in network lifetime due to the fact that with an increase in sensing range fewer sensors are active at any one time, meaning that more sensors are able to conserve energy by transitioning to the sleep state. Table 6.2 and figure 6.2 shows the corresponding results.

Number of sensors	40	60	80	100	120	140	160
DSA	86.4	151.4	231.3	262.3	317.7	418.0	524.3
LBP	60.0	130.0	160.0	180.0	210.0	279.0	340.0

Table 6.2: Lifetime variation with 30 targets, 50m sensing range and linear energy model.

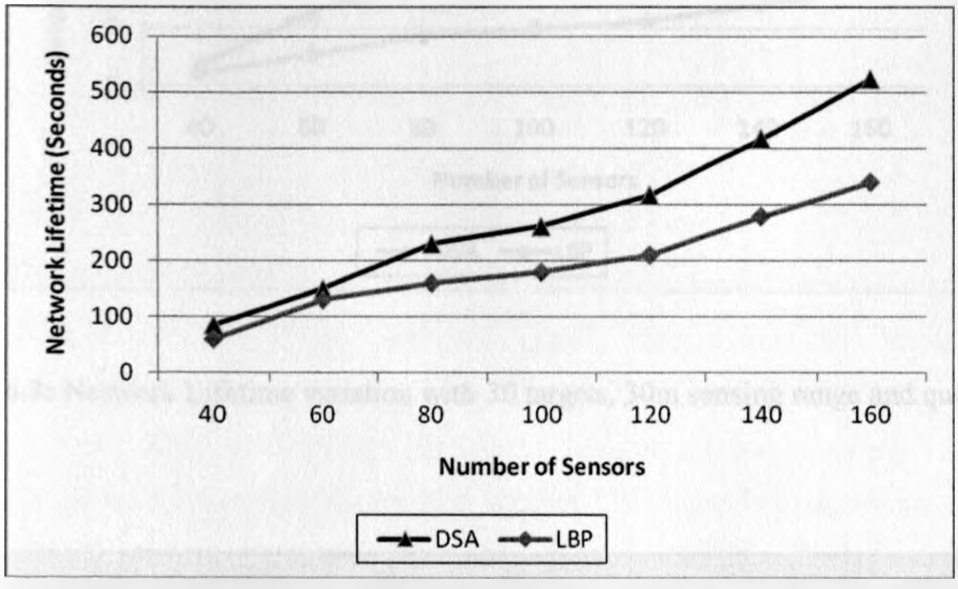

Figure 6.2: Network Lifetime variation with 30 targets, 50m sensing range and linear energy model.

The third and fourth simulations are conducted with 30 randomly deployed targets, 40 to 160 sensors with an increment of 20, maximum sensing range of 30m and 50m with quadratic energy model. The results are shown in table 6.3, figure 6.3, table 6.4 and figure 6.4. The results are consistent with those generated in the previous simulations under the linear energy model. There is however, a significant improvement in network lifetime with DSA under the quadratic model as compared with the performance of LBP.

Number of sensors	40	60	80	100	120	140	160
DSA	2.0	6.4	10.2	14.4	17.3	21.7	25.8
LBP	1.1	2.8	4.4	5.0	5.6	7.2	9.4

Table 6.3: Lifetime variation with 30 targets, 30m sensing range and quadratic energy model.

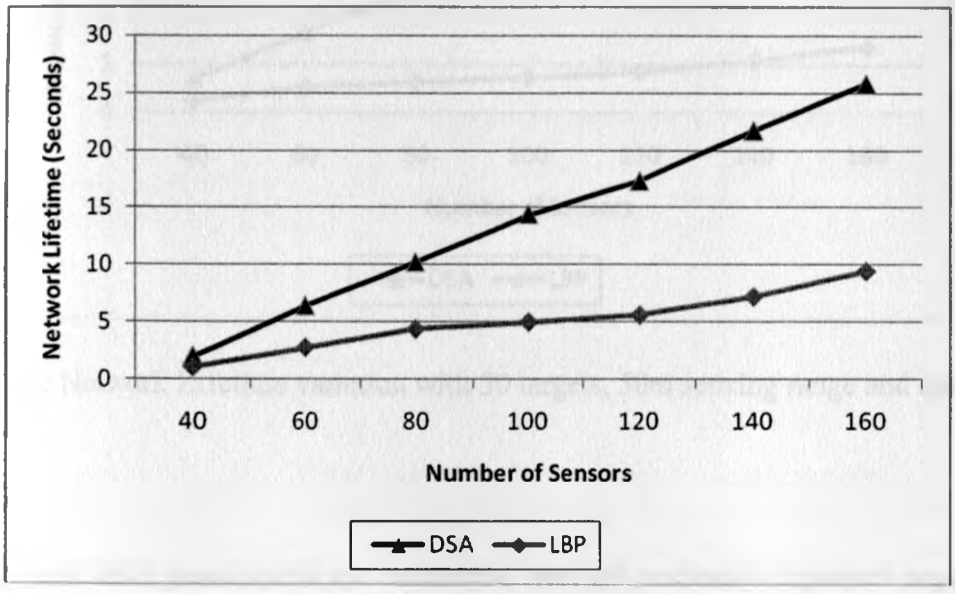

Figure 6.3: Network Lifetime variation with 30 targets, 30m sensing range and quadratic energy model.

Number of sensors	40	60	80	100	120	140	160
DSA	3.5	8.3	12.9	18.3	21.4	26.4	31.5
LBP	1.2	2.6	3.2	3.6	4.2	5.4	6.8

Table 6.4: Lifetime variation with 30 targets, 50m sensing range and quadratic energy model.

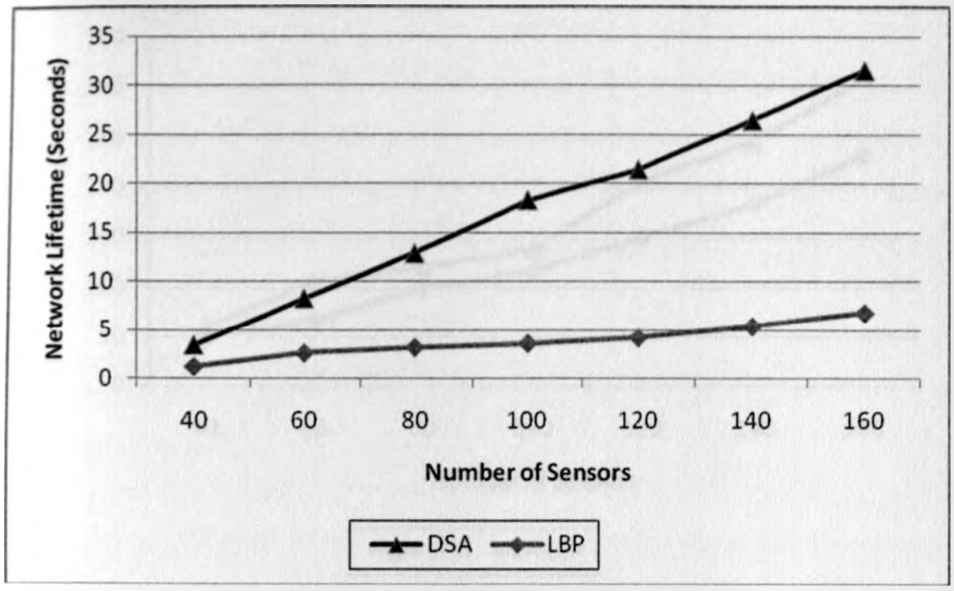

Figure 6.4: Network Lifetime variation with 30 targets, 50m sensing range and quadratic energy model.

The fifth and sixth simulations are conducted with 60 randomly deployed targets, 40 to 160 sensors with an increment of 20, maximum sensing range of 30m with linear and quadratic energy model. The results are shown in table 6.5, figure 6.5, table 6.6 and figure 6.6. The results show a decrease in network lifetime with the increase in number of targets to be monitored. When comparing the performance of DSA against LBP there is a significant increase in the number of active sensors per round in the execution of the algorithms as compared with the case when the targets are 30. This increase in the number of active sensors is more notable in DSA as compared with LBP.

Number of sensors	40	60	80	100	120	140	160
DSA	59.9	105.9	120	138	205.6	245.9	309
LBP	50	66.7	100	116.7	150	182.3	232.3

Table 6.5: Lifetime variation with 60 targets, 30m sensing range and linear energy model.

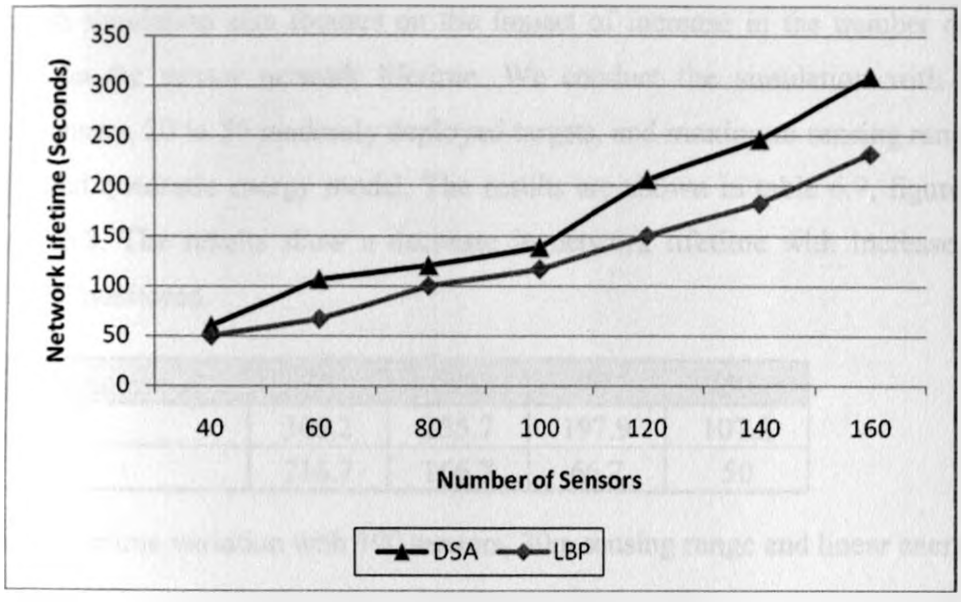

Figure 6.5: Network Lifetime variation with 60 targets, 30m sensing range and linear energy model.

Number of sensors	40	60	80	100	120	140	160
DSA	3.9	5.4	7.4	8.8	13.5	16.2	20.8
LBP	1.7	2.2	3.3	3.9	5	5.5	7.2

Table 6.6: Lifetime variation with 60 targets, 30m sensing range and quadratic energy model.

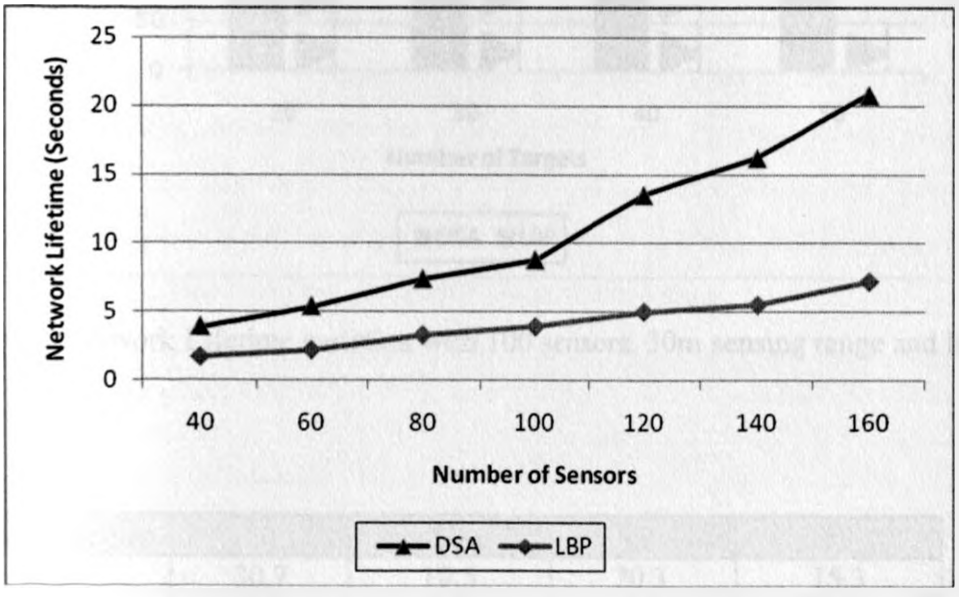

Figure 6.6: Network Lifetime variation with 60 targets, 30m sensing range and quadratic energy model.

The seventh simulation also focuses on the impact of increase in the number of targets to be monitored on the sensor network lifetime. We conduct the simulation with 100 randomly deployed sensors, 20 to 50 randomly deployed targets, and maximum sensing range of 30m with the linear and quadratic energy model. The results are shown in table 6.7, figure 6.7, table 6.8 and figure 6.8. The results show a decrease in network lifetime with increase in number of targets to be monitored.

Number of targets	20	30	40	50
DSA	305.2	255.7	197.9	107.2
LBP	216.7	166.7	66.7	50

Table 6.7: Lifetime variation with 100 sensors, 30m sensing range and linear energy model.

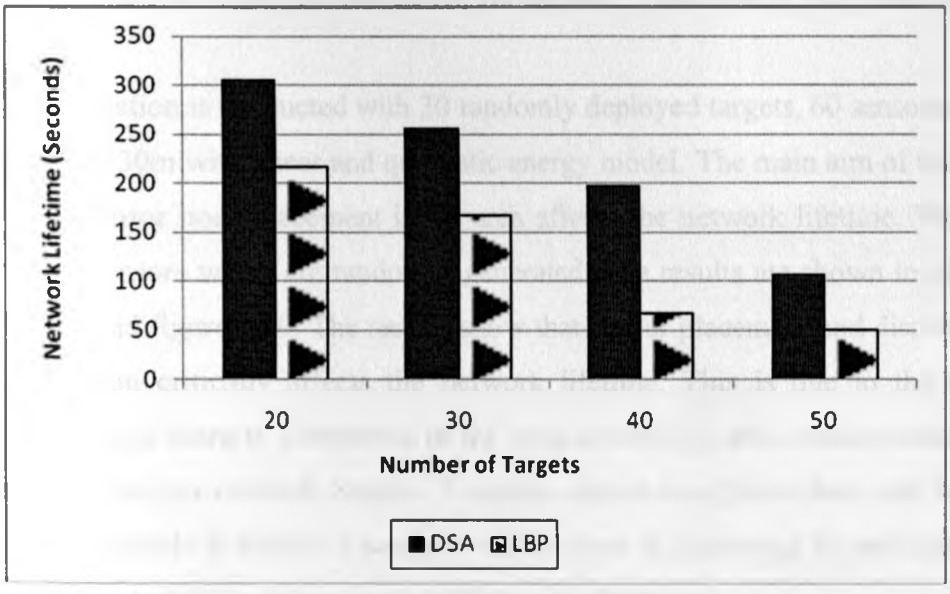

Figure 6.7: Network Lifetime variation with 100 sensors, 30m sensing range and linear energy model.

Number of targets	20	30	40	50
DSA	30.7	19.5	20.3	15.3
LBP	7.2	5.6	2.2	1.7

Table 6.8: Lifetime variation with 100 sensors, 30m sensing range and quadratic energy model.

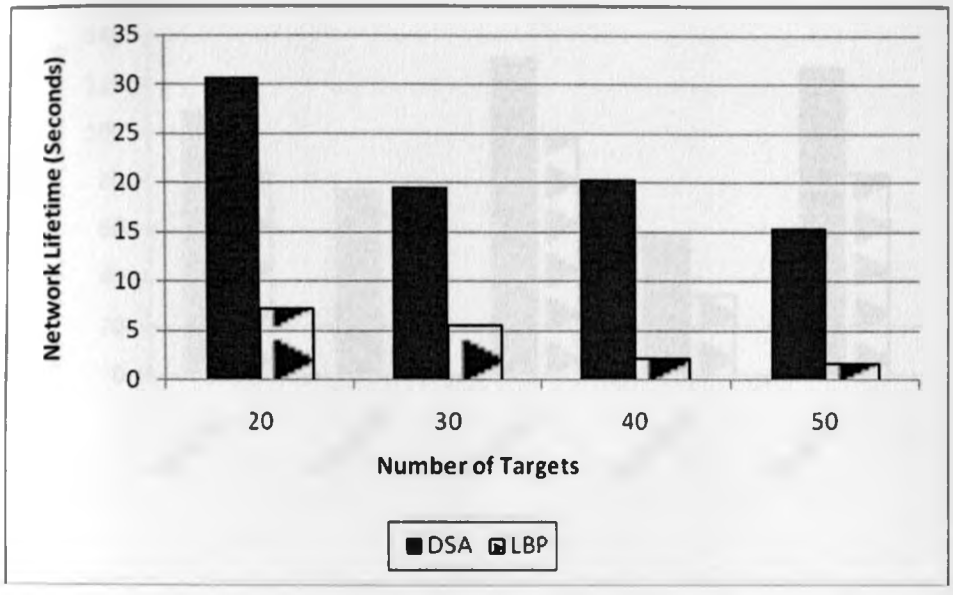

Figure 6.8: Network Lifetime variation with 100 sensors, 30m sensing range and quadratic energy model.

The eighth simulation is conducted with 30 randomly deployed targets, 60 sensors and maximum sensing range of 30m with linear and quadratic energy model. The main aim of this simulation is to find out if sensor node placement in an area affects the network lifetime. We use different samples of 60 sensors which are randomly generated. The results are shown in table 6.9, figure 6.9, table 6.10 and figure 6.10. The results show that sensor placement and distribution within a monitoring region critically affects the network lifetime. This is due to the fact that with placement variation there is a variation in the area covered by each sensor which translates to variation in the targets covered. Sample 3 depicts almost an optimal area and target coverage scenario while sample 4 depicts a scenario where there is clustering in particular regions and exposure of other regions resulting in poor lifetime performance.

Placement	Sample 1	Sample2	Sample 3	Sample 4	Sample 5
DSA	109.2	76.9	130.8	58.3	127.5
LBP	83.3	66.7	100	33.3	83.3

Table 6.9: Lifetime variation with 30 targets, 30m sensing range, different samples of 60 sensors and linear energy model.

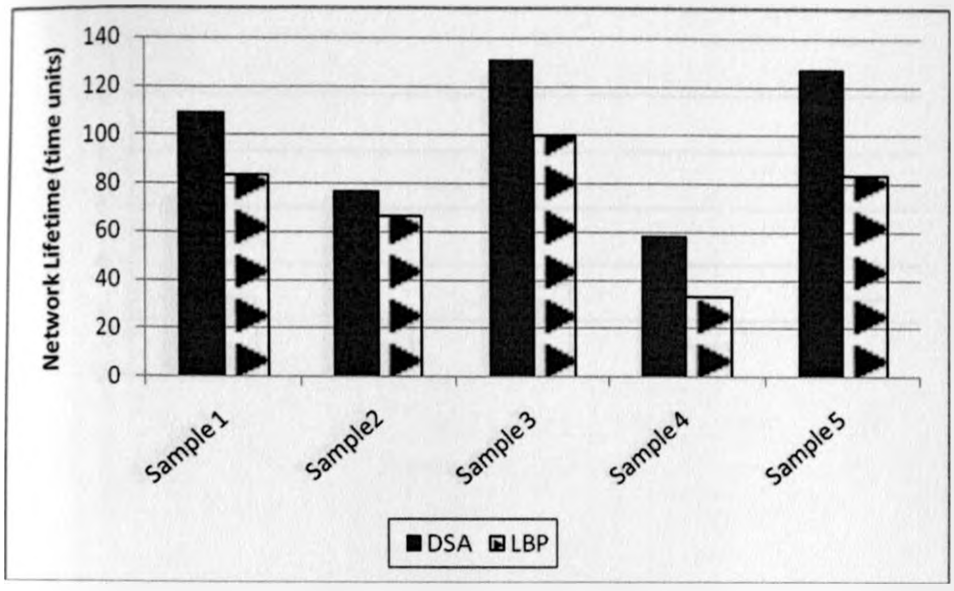

Figure 6.9: Network Lifetime variation with 30 targets, 30m sensing range, different samples of 60 sensors and linear energy model.

Placement	Sample 1	Sample2	Sample 3	Sample 4	Sample 5
DSA	6.4	6.3	10.7	2.5	9.1
LBP	2.8	2.2	3.3	1.1	2.8

Table 6.10: Lifetime variation with 30 targets, 30m sensing range, different samples of 60 sensors and quadratic energy model.

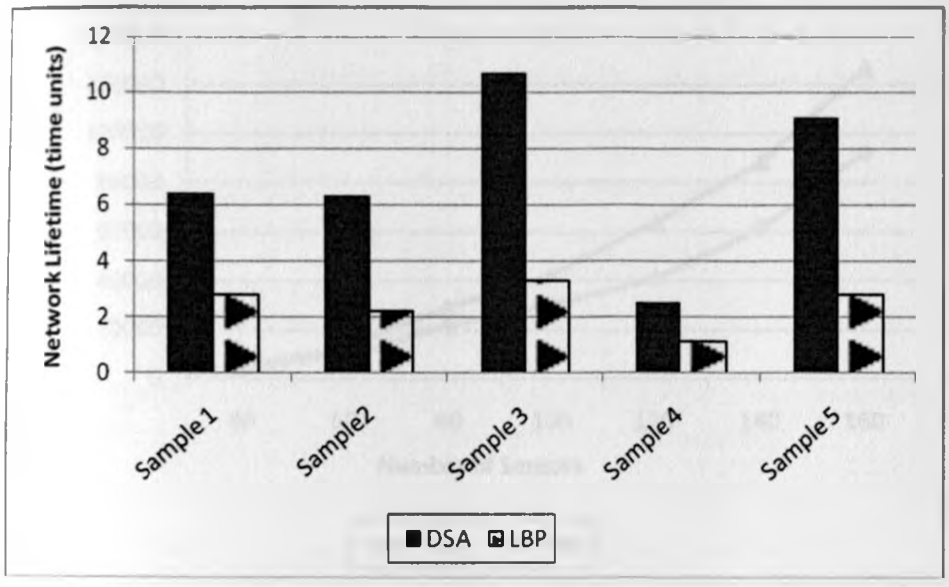

Figure 6.10: Network Lifetime variation with 30 targets, 30m sensing range, different samples of 60 sensors and quadratic energy model.

The ninth simulation is conducted with 30 randomly deployed targets, 40 to 160 sensors with an increment of 20, maximum sensing range of 30m with quadratic energy model. The main aim of this simulation is to evaluate the message complexity of LBP and DSA. The results are shown in table 6.11, figure 6.11, table 6.12 and figure 6.12. The results show an increase in the message complexity in both algorithms with increase in the number of deployed sensors. This is due to the fact that with an increase in the number of sensors there is an increase in the number of broadcast messages and an increase in the number of neighbors per sensor.

Number of sensors	40	60	80	100	120	140	160
DSA	4634	13420	29528	43160	63924	88698	126734
LBP	3994	10160	21704	30528	40710	61724	92208

Table 6.11: Message complexity with 30 targets, 30m sensing range and linear energy model.

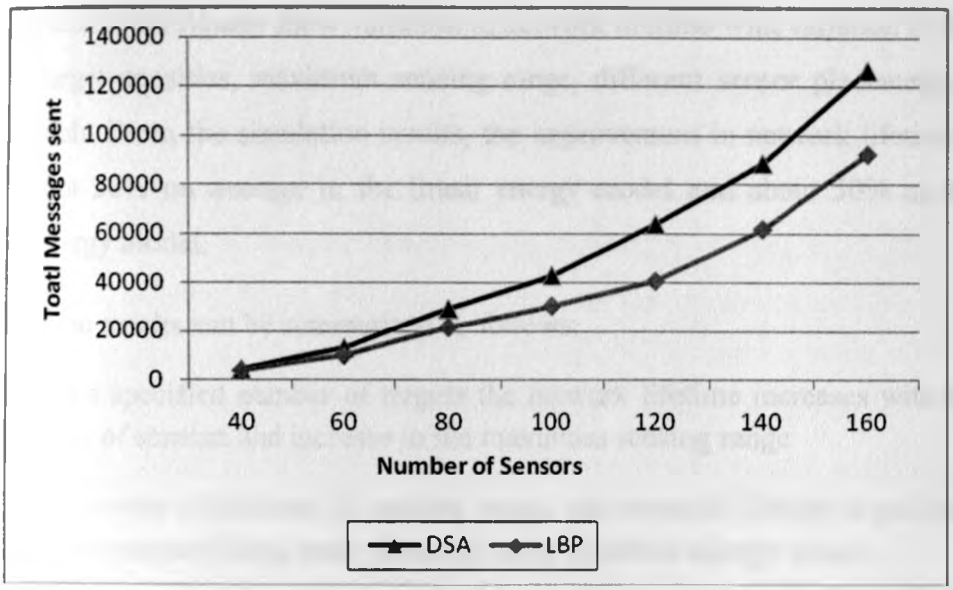

Figure 6.11: Message complexity with 30 targets, 30m sensing range and linear energy model.

Number of sensors	40	60	80	100	120	140	160
DSA	314	1158	2328	4068	5604	8190	10894
LBP	148	476	910	1338	1792	2584	3600

Table 6.12: Message complexity with 30 targets, 30m sensing range and quadratic energy model.

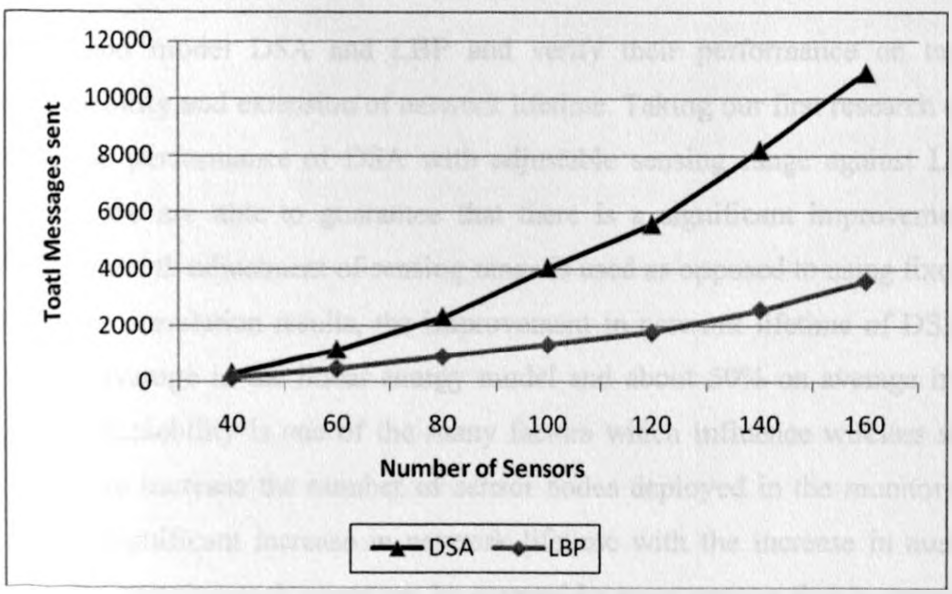

Figure 6.12: Message complexity with 30 targets, 30m sensing range and quadratic energy model.

The above tables and figures show variation in network lifetime with variation in the sensor node densities, target densities, maximum sensing range, different sensor placements and different energy models. From the simulation results, the improvement in network lifetime of DSA over LBP is about 26% on average in the linear energy model and about 50% on average in the quadratic energy model.

The simulation results can be summarized as follows:

- Given a specified number of targets the network lifetime increases with increase in the number of sensors and increase in the maximum sensing range
- With smooth adjustment of sensing range, the network lifetime significantly increases with the increase being more dramatic in the quadratic energy model.
- By using a constant number of sensors and sensing ranges the network lifetime decreases with the increase of the number of targets to be monitored.
- Sensor placement variation has a significant impact on network lifetime.
- Message complexity increases with increase in the number of sensors.

6.2. Discussion of the results

We evaluate and model DSA and LBP and verify their performance on target coverage, reliability, scalability and extension of network lifetime. Taking our first research question, when we compare the performance of DSA with adjustable sensing range against LBP with fixed sensing range, we are able to guarantee that there is a significant improvement in network lifetime when smooth adjustment of sensing range is used as opposed to using fixed range sensor nodes. From the simulation results, the improvement in network lifetime of DSA over LBP is about 26% on average in the linear energy model and about 50% on average in the quadratic energy model. Scalability is one of the many factors which influence wireless sensor network design. When we increase the number of sensor nodes deployed in the monitoring region, the results show a significant increase in network lifetime with the increase in number of sensor nodes due to the fact that each target can be covered by more sensors thus increasing the number of redundant nodes which are put to sleep conserving energy and increasing network lifetime.

On the other hand when we increase the number of targets to be monitored, the results show a decrease in network lifetime due to the fact that more sensors are required to be active so as to ensure complete target coverage. With the increase in the number of active sensors, there is a significant increase in the energy consumed per sensor resulting in a decrease in the network lifetime. When we take the case of increasing the maximum sensing range, the results show an increase in the network lifetime. This is due to the fact that dense deployment means overlap in the monitoring regions of the sensors, with the increase in the sensing range the overlap is more paramount and each sensor is able to cover more targets resulting in an increase in the number of redundant nodes which are effectively put to sleep conserving energy and increasing network lifetime.

The second research question seeks to find out the rules used by self-organizing monitoring schedules to make decisions on whether to become active or idle and the conditions under which nodes make such decisions. To answer this question the proposed algorithm DSA uses a set of transition rules. In the initialization phase, all sensors are in the vulnerable state with maximum sensing range. For each sensor, it should change to *Active state* with sensing range *r* if there is a target at range *r* which is not covered by any other active or vulnerable sensors or remain in the *Vulnerable state* but decrease its sensing range to the next furthest target if all targets at range r are covered by other active or vulnerable sensor with greater energy supply. If its sensing range decreases to zero it should change to *Idle state*. Once all the sensors have made a decision to be active or idle, they will stay in that state for a period of time called shuffle time or until there is an active sensor which exhausts it energy and is going to die

Our hypothesis for this study states that wireless sensor network lifetime increases with increase in number of sensor nodes with adjustable sensing range as compared with sensors with fixed sensing range and decreases with increase in the number of targets to be monitored. From the simulation results we prove that this hypothesis is true since, given a specified number of targets the network lifetime increases with increase in the number of sensors and increase in the maximum sensing range, and by using a constant number of sensors and sensing ranges the network lifetime decreases with increase in the number of targets to be monitored. We are also able to show that for all the simulations carried out DSA always out performs LBP in terms of network lifetime

CHAPTER 8: CONCLUSION AND FUTURE WORK

Energy saving in WSN has attracted a lot of attention and introduced quite a number of challenges as compared to traditional wired networks. Extensive research has been conducted with the main aim of developing schemes that can improve resource efficiency and mitigate some of the unique challenges faced by WSNs. The main problem addressed in this paper is the determination of network lifetime when all targets are covered and sensor energy resources are constrained. We identify wasteful and unnecessary activities in sensor networks and provide a discussion of duty-cycling or sleep-wake scheduling as an energy conservation technique. We provide a problem formulation for the lifetime maximization problem with smooth adjustment of sensing range and use of sleep-wake scheduling. We propose a distributed scheduling algorithm with adjustable sensing range and compare its performance with the load balancing protocol for sensing with fixed sensing range. We also provide the analysis to show the correctness and reliability of DSA and demonstrate it using the simulation results.

The simulation results show that with smooth adjustment of sensing range and use of sleep-wake schedules, the network life time is significantly improved. We prove that sensor node placement significantly affects the lifetime of a wireless sensor network. We also show that given a specified number of targets, increasing the number of sensors and the sensing range results in an increase in the network lifetime. We also prove that given a specified number of sensors and sensing range, increase in the number of targets to be monitored results in a decrease in the network lifetime. When comparing the performance of DSA and LBP under linear and quadratic energy models, DSA presents a more dramatic improvement in network lifetime in the quadratic energy model.

The future work will include simulation of the proposed algorithm with the combination of communication protocols and improvement of its performance by reducing the inefficiency in load balancing among the sensors and better integration of adjustable sensing range into the proposed algorithm. DSA takes into account the energy consumed during message broadcasts within a one hop network, once it is combined with a communication protocol there will be an additional cost in communication in the relay of data from the sensing nodes to the base station. This increase in communication cost will affect the lifetime of the network. Therefore, the

communication protocol should have mechanisms in place to reduce this cost as much as possible so that the lifetime of the network is not adversely affected when routing is introduced. DSA can also be extended to accommodate mobile targets and also allow for sensor node mobility. This ability will make it more desirable in dynamic battlefields where the targets are mobile and multiple unpredictable events occur.

REFERENCES

Abrams, Z, Goel, A & Plotkin, S 2004, 'Set k-cover algorithms for energy efficient monitoring in wireless sensor networks', *Third International Symposium on Information Processing in Sensor Networks*, pages 424–432.

Akyildiz, IF, Su, W, Sankarasubramaniam, Y & Cayirci, E 2002, 'Wireless Sensor Networks: A Survey', *Computer Networks Elsevier Journal*, Vol. 38, No. 4, pp. 393-422

Bali, SJ & Partha, R 2008, 'Energy Saving Methods in Wireless Sensor Networks (Based on 802.15.4)', *Technical report*, IDE0814.

Berman, P, Calinescu, G, Shah, C & Zelikovsky, A 2004, 'Power Efficient Monitoring Management in Sensor Networks', *IEEE Wireless Communication and Networking Conference (WCNC'04)*, Atlanta, pp. 2329-2334.

Berman, P, Calinescu, G, Shah, C & Zelikovsky, A 2005, 'Efficient Energy Management in Sensor Networks', *In Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing*, Vol. 2, Xiao, Y & Pan, Y (Eds.), Nova Science Publishers.

Cerpa, D & Zelikovsky A 2006, 'DEEPS: Deterministic Energy-Efficient Protocol for Sensor Networks', *ACIS International Workshop on Self-Assembling Wireless Networks (SAWN'06) Proc. of SNPD*, pp. 261-266.

Hill, K, Burstein, A, Chang, D, Dong, M., Fielding, M, Kruglick, E, Ho, J, Lin, F, Lin, TH, Kaiser, WJ, Marcy, H, Mukai, R, Nelson, P, Newburg, FL, Pister, KSJ, Pottie, G, Sanchez, H, Sohrabi, K, Stafsudd, OM, Tan, KB, Yung, G, Xue, S, & Yao, J 1996, 'Low power systems for wireless microsensors', *Proc. of the International Symposium on Low Power Electronics and Design.*

Kalhoun, BH, Daly, DC, Verma, N, Finchelstein, DF, Wentzloff, DD, Wang, A, Cho, SH, & Chandrakasan, AP 2005, 'Design considerations for ultralow energy wireless microsensor nodes,' *IEEE Transactions on Computers* **54** (6), pp.727–749.

Cardei, M & Du, D-Z 2005, 'Improving Wireless Sensor Network Lifetime through Power Aware Organization', *ACM Wireless Networks*, vol. 11, No. 3.

Cardei, M, Thai, MT , Li, Y & Wu, W 2005a, 'Energy-efficient target coverage in wireless sensor networks', *In Proceedings of IEEE INFOCOM*, vol. 3, pp. 1976- 1984.

Cardei, M, Wu, J, Lu, M & Pervaiz, M 2005b, 'Maximum network lifetime in wireless sensor networks with adjustable sensing ranges', *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications* (WiMob'2005), Vol. 3, pp. 438 – 445.

Chang, JH & Tassiulas, L 2004, 'Maximum lifetime routing in wireless sensor networks', *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609–619.

Chen, B, Jamieson, K, Balakrishnan, H & Morris, R 2001, 'Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks', *MobiCom 2001*, Rome, Italy, pp. 70–84.

Chong, CY & Kumar, SP 2003, 'Sensor Networks: Evolution, Opportunities and Challenges', *Proceeding of the IEEE*, vol. 91, no. 8.

Dai, F & Wu, J 2003, 'Distributed dominant pruning in ad hoc networks', *Communications, ICC '03. IEEE International Conference on*, vol. 1, pp. 353 – 357.

DAR 1978, *Proceedings of the Distributed Sensor Nets Workshop*, Pittsburgh, PA, Department of Computer Science, Carnegie Mellon University.

Dargie, W & Poellabauer, C 2010, *Fundamentals of Wireless Sensor Networks: Theory and Practice*, John Wiley & Sons Ltd.

Dhawan, A, Vu, C, Zelikovsky, A, Li, Y & Prasad, S 2006, 'Maximum lifetime of sensor networks with adjustable sensing range', *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. SNPD 2006. Seventh ACIS International Conference*, pp. 285 –289.

Garg, N & Könemann, J 1998, 'Faster and simpler algorithms for multicommodity flow and other fractional packing problems', *Proc. of 39th Annual IEEE Symposium on Foundations of Computer Science, IEEE*, pp. 300–309.

Godfrey, BP & Ratajczak, D 2004, 'Naps: Scalable, robust topology management in wireless ad hoc networks', *ACM* ,ISPN '04, Berkeley, CA, April 26–27.

Haenselmann, T 2006, '*Sensornetworks*', GFDL Wireless Sensor Network textbook, http://pi4.informatik.uni-mannheim.de/~haensel/sn_book, retrieved 2011-03-29.

Heinzelman, WR, Chandrakasan, A & Balakrishnan, H 2000, 'Energy-efficient communication protocol for wireless microsensor networks', *IEEE Hawaii International Conference on Systems Sciences*.

Heinzelman, W, Chandrakasan, A & Balakrishnan, H 2002, 'An Application-Specific Protocol Architecture for Wireless Microsensor Networks', *IEEE Transactions on Wireless Communications*, Vol. 1, No. 4, pp. 660-670.

Jiang, X, Taneja, J, Ortiz, J, Tavakoli, A, Dutta, P, Jeong, J, Culler, D, Levis, P, & Shenker, S 2007, 'An architecture for energy management in wireless sensor networks', *SIGBED* ,Rev. 4 (3), pp. 31–36.

Kahn, JM, Katz, RH, & Pister, KSJ 1999, 'Mobile networking for smart dust', *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*.

Kim, JM, Park, SH, Han, YJ & Chung, TM 2008, 'CHEF: Cluster Head Election mechansim using Fuzzy logic in Wireless Sensor Networks', *Proc. of ICACT*, pp. 654- 659.

Kumar, D, Aseri, TS, Patel, RB 2009a, 'EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks', *International Journal of Computer Communications*, Elsevier, 2008. 32(4): 662-667.

Kumar, D, Aseri, TS, Patel, RB 2009b, 'EECHE: Energy efficient cluster head election protocol for heterogeneous Wireless Sensor Networks', *in Proceedings of ACM International Conference on Computing, Communication and Control-09 (ICAC3'09),* Bandra, Mumbai, India, 23-24, pp. 75-80.

Lee, SH, Yoo, JJ & Chuan, TC 2004, 'Distance-based Energy Efficient Clustering for Wireless Sensor Networks', *Proc. of the 29th IEEE Int'l Conf. on Local Computer Networks* (LCN'04).

Lindsey, S & Raghavendra, CS 2002, 'PEGASIS: power efficient gathering in sensor information systems', *in Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana , Vol.3, pp. 3-1125- 3-113.

Lu, M, Wu, J, Cardei, M & Li, M 2009, 'Energy-efficient connected coverage of discrete targets in wireless sensor networks', *Int. J. Ad Hoc Ubiquitous Computing*. 4(3/4): pp. 137–147.

Madan, R, Luo, ZQ & Lall, S 2005, 'A distributed algorithm with linear convergence for maximum lifetime routing in wireless sensor networks', in *Proc. of the Allerton Conference on Communication, Control and Computing.*

Mao, Y, Liu, Z, Zhang, L, & Li, X 2009, 'An Effective Data Gathering Scheme in Heterogeneous Energy Wireless Sensor Networks', *International Conference on Computational Science and Engineering,* cse, vol. 1, pp.338- 343.

Meriall, WM, Newberg, F, Sohrabi, K, Kaiser, W & Pottie, G 2003, 'Collaborative Networking Requirements for Unattended Ground Sensor Systems', In *Proc. IEEE Aerospace Conference.*

Pottie, GJ & Kaiser, WJ 2000, 'Wireless integrated network sensors', *Communication ACM,* 43(5): pp. 51-58.

Pottie, GJ 2001, Wireless integrated network sensors (WINS): The web gets physical. *National Academy of Engineering: The Bridge* **31** (4), pp. 22–27.

Rabaey, J, Ammer, J, da Silva, Jr JL, & Patel, D 2000, 'Picoradio: Ad hoc wireless networking of ubiquitous low-energy sensor/monitor nodes', *Proc. of the IEEE Computer Society Annual Workshop on VLSI.*

Robinson, S 2004, *Simulation: The Practice of Model Development and use*, John Wiley & Sons.

Römer, K & Mattern, F 2004, 'The Design Space of Wireless Sensor Networks', *IEEE Wireless Communications* 11 (6):54–61, doi:10.1109/MWC.2004.1368897.

Sankar, A & Z. Liu, Z 2004, 'Maximum lifetime routing in wireless ad hoc networks', in *Proc. IEEE Infocom*, pp.1089–1097.

Schurgers, C, Tsiatsis, V, Ganeriwal, S & Srivastava, M 2002a, 'Optimizing sensor networks in the energy-latency-density design space', *IEEE Transactions on Mobile Computing*, vol.1, No.1.

Schurgers, C, Tsiatsis, V, Ganeriwal, S & Srivastava, M 2002b, 'Topology management for sensor networks: exploiting latency and density', *MOBIHOC'02*, Lausanne, Switzerland, ACM.

Shwe , HY, Jiang, X, & Horiguchi S 2009, 'Energy saving in wireless sensor networks', *Journal of Communication and Computer*, vol. 6, No. 5 (Serial No. 54).

Slijepcevic, S & Potkonjak, M 2002, 'Power efficient organization of wireless sensor networks', *IEEE International Conference on Communications*, vol. 2, 2001, pp. 472–476.

Urbaniak, GC, & Plous, S 2011, Research Randomizer (Version 3.0) [Computer software]. Retrieved on April 22, 2011, from http://www.randomizer.org/

Wired 2003, 'Slow, Stupid Networks Pack Punch, *Wired News,* http:// www.wired.com/news/wireless/0,1382,60335,00.html, (Sept.2003).

Xu, Y, Bien, S, Mori, Y, Heidemann, J & Estrin, D 2003, 'Topology control protocols to conserve energy in wireless ad hoc networks', *Technical Report 6*, University of California, Los Angeles, Center for Embedded Networked Computing.

Xu, Y, Heidemann, J & Estrin, D 2001, 'Geography-informed energy conservation for ad hoc routing', *Proceedings of the 7th annual international conference on Mobile computing and networking*, Rome, Italy, pp. 70 – 84, 16-21.

Ye. W, Heidemann, J & Estrin, D 2002, 'An Energy-Efficient MAC Protocol for Wireless Sensor Networks', *in IEEE INFOCOM.*

Younis, O & Fahmy, S 2004, 'Distributed Clustering in Adhoc Sensor Networks: A Hybrid, Energy-Efficient Approach', *In Proc. of IEEE INFOCOM.*

APPENDIX – A

C# code for DSA worker class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Windows.Forms;

namespace wsndsa
{
  public class ShiftEventArgs
  {
    string message;

    public string Message
    {
      get { return message; }
      set { message = value; }
    }
  }
  public class worker
  {
    public delegate void ShiftEventHandler(object sender, ShiftEventArgs e);
    public  event ShiftEventHandler ShiftEventOccured;
    private static List<string> txt;
    static void DumpToFile(string path, string[] linesToWrite)
    {
      using (StreamWriter sw = File.AppendText(path))
      {
        //File.WriteAllLines(path, linesToWrite );
        foreach (string text in linesToWrite)
          sw.WriteLine(text);
      }
    }
    public void Run(int maxrange, int choice, string path, string sensors, string targets)
    {
      //string path = "dump.txt";
      if (File.Exists(path))
        File.Delete(path);
      txt = new List<string>();
      txt.Add("Algorithm: DSA\n");
      List<target> t = new List<target>();
      List<sensor> s = new List<sensor>();
```

```
target T;

double batteryTime, nbatteryTime;
int targetid;
bool coveredbyother = false;
bool decreaseRange = false;
double shuffleTime = 1;
int shift = 1; // schuffle index
double shiftTime; // time for one shift
double monitorTime = 0; // total monitor time
double msgcount = 0; // messages sent per sensor
double totalmsgcount = 0;// total messages sent
double noofneighbors; //number of neighbors
double recenergy = 0.000000050;// reception energy per bit = 50nj/bit
double bpacketsize = 2000;// broadcast packet size = 2000 bits
double reccost; //reception communication cost
double transenergy = 0.000000050;//transmission energy per bit = 50nj/bit
double transcost;// transmission communication cost
double commcost = 0; //communication cost
double elevel; // energy level
double cbattery; // current battery level


// get target and sensor information
getInformation(ref t, ref s, maxrange, choice, sensors, targets);

string rv = "";
foreach (string ts in txt)
   rv += ts;

while (allTargetcovered(t, s))
{
   // for each sensor, find target and neighbor sensor
   for (int i = 0; i < s.Count; i++)
   {
      s[i].wakeup();
      // get target
      s[i].getTarget(t);
      // find neighbor sensor
      s[i].getNeighbor(ref s);
   }
   // start by assume that next shift time = shuffle time
   shiftTime = shuffleTime;
   // shuffle process
   while (!isAllDecided(s))
```

```
{
  // for every sensor
  for (int i = 0; i < s.Count(); i++)
  {
    if (s[i].getstatus() < 0)
    {
      decreaseRange = false;

      if (s[i].coveredTarget.Any())
      {
        batteryTime = s[i].getTime();

        // check the first target: the one with maximum length
        targetid = s[i].coveredTarget[0].getid();
        coveredbyother = false;
        // check with all neighbors
        for (int j = 0; j < s[i].neighbor.Count; j++)
        {
          // every covered target of neightbor j
          for (int k = 0; k < s[i].neighbor[j].coveredTarget.Count; k++)
          {
            // the target can be covered by neighbor j
            if (targetid == s[i].neighbor[j].coveredTarget[k].getid())
            {
              coveredbyother = true;
              nbatteryTime = s[i].neighbor[j].getTime();
              if (batteryTime < nbatteryTime || (nbatteryTime == batteryTime &&
        s[i].getId() < s[i].neighbor[j].getId()) || s[i].neighbor[j].getstatus() > 0)
              {
                s[i].decreaseRange();
                decreaseRange = true;
                break;
              }
            }
          }
          if (decreaseRange)
          {
            break;
          }
        }
        if (!coveredbyother)
        {
          s[i].goactive(t);
          if (shiftTime > batteryTime)
          {
            shiftTime = batteryTime;
```

```
                }
                // notify with all neighbors
                for (int j = 0; j < s[i].neighbor.Count; j++)
                {
                    // every covered target of neightbor j
                    while (!s[i].neighbor[j].coveredTarget.Any() == false)
                    {
                        T = s[i].neighbor[j].coveredTarget[0];
                        if (s[i].covers(T))
                        {
                            s[i].neighbor[j].decreaseRange();
                        }
                        else

                            // not covered
                            break;
                    }
                }
            }
        }
        else
        {
            s[i].gosleep();
        }
    }
  }
}
// end of each shift
monitorTime += shiftTime;
msgcount = 2 * s.Count;
totalmsgcount += msgcount;
for (int i = 0; i < s.Count; i++)
  s[i].setBattery(shiftTime);

// print each shift detail

txt.Add ("Shift # " + shift + " ,\t Shift time = " + shiftTime + " Second.\t Life time = "
  + monitorTime + " Seconds." + " " + " \tMessages sent = " + msgcount + " ");
//tbox.Select(txt.Length, 0);
for (int i = 0; i < s.Count; i++)
{
  //communication cost
  noofneighbors = s[i].neighbor.Count;
  transcost = 2 * transenergy * bpacketsize;
  reccost = 2 * recenergy * bpacketsize * noofneighbors;
  commcost += transcost + reccost;
```

```
            elevel = s[i].getBatteryLevel();

            if (elevel < commcost )
               cbattery = s[i].getBatteryLevel();
            else
               cbattery = Convert.ToInt16(elevel - commcost);

            //print sensor details
            txt.Add(" - Sensor Id : " + s[i].getId() + "\t : Energy level :" + cbattery + " J " + "\t :
Status = ");
            if (s[i].getstatus() == 0)
               txt[txt.Count - 1] = txt.Last() + (" Sleep ");
            else
         txt[txt.Count - 1] = txt.Last() + (" Active " + " with range = " + s[i].getstatus() + " m ");
          }
          DumpToFile(path, txt.ToArray());

          ShiftEventOccured(new object(), new ShiftEventArgs
          {
Message = rv + "\nShift # " + shift + " , \t Shift time = " + shiftTime + " Seconds.\t Life time = "
            + monitorTime + " Seconds." + " " + " \t Messages sent = " + msgcount + " "
          });

          txt = new List<string>();
          shift++;
          // delete dead sensor from vector sensor
          deleteSensor(ref s);
          txt.Add ("");
          txt.Add ("");
          txt.Add ("");

       }
txt.Add ("Network Lifetime : " + monitorTime + " Seconds \t\t Total messages sent = " +
totalmsgcount + "");
       txt.Add("");
       txt.Add("");
       DumpToFile(path, txt.ToArray());
    }
 static void getInformation(ref List<target> t, ref List<sensor> s, int maxrange, int choice, string
sensors, string targets)
    {
       float xpos, ypos, maxb, maxr;
       string filename;

       int tid, sid/*, choice*/, numSensor, numTarget;
       bool linearPower;
```

```
// read target information

filename = targets;//"targets.txt"

if (File.Exists(filename))
{
   string[] readText = File.ReadAllLines(filename);

   string[] vt = new string[3];

   numTarget = int.Parse(readText[0]);

   txt.Add ("Number of Targets :" + numTarget + "\n");

   for (int i = 1; i <= numTarget; i++)
   {
      vt = readText[i].Split(' ');
      tid = int.Parse(vt[0]);
      xpos = float.Parse(vt[1]);
      ypos = float.Parse(vt[2]);
      // add new target to vector
      t.Add(new target(tid, xpos, ypos));
   }
}
else
{
   txt.Add ("Can not open file " + filename);
}
// read sensor information

filename = sensors;//"sensors.txt"
//"Maximum Range :"
maxr = maxrange;

if (choice == 1)
   linearPower = true;
else
   linearPower = false;
if (File.Exists(filename))
{
   string[] readText = File.ReadAllLines(filename);
   string[] vt = new string[4];

   numSensor = int.Parse(readText[0]);
   txt.Add ("Number of Sensors :" + numSensor + "\n");
   for (int i = 1; i <= numSensor; i++)
```

```
        {
            vt = readText[i].Split(' ');
            sid = int.Parse(vt[0]);
            maxb = float.Parse(vt[1]);
            xpos = float.Parse(vt[2]);
            ypos = float.Parse(vt[3]);
            // add the new sensor to vector
            s.Add(new sensor(sid, xpos, ypos, maxb, maxr, linearPower));
        }
    }
    else
    {
        txt.Add ("Can not open file " + filename);
    }
}
static bool isAllDecided(List<sensor> s)
{
    for (int i = 0; i < s.Count(); i++)
    {
        if (s[i].getstatus() < 0)
            return false;
    }
    return true;
}
static void deleteSensor(ref List<sensor> s )
{
    int numDead = 0;
    // find number of dead sensor
    foreach (var itr in s)
    {
        if (itr.isDead())
        {
            numDead++;
        }
    }
    // delete sensor
    for (int i = 1; i <= numDead; i++)
    {
        for (int j = 0; j < s.Count; j++)
        {
            if (s[j].isDead())
            {
                txt.Add ("Sensor: " + s[j].getId() + " is dead ");
                s.RemoveAt(j);
                break;
            }
```

```
        }
      }
    }
    static bool allTargetcovered(List<target> t, List<sensor> s )
    {
      bool tcovered;
      for (int i = 0; i < t.Count(); i++)
      {
        tcovered = false;
        for (int j = 0; j < s.Count; j++)
        {
          if (s[j].canCoveredTarget(t[i]))
          {
            tcovered = true;
            break;
          }
        }
        if (!tcovered)
        {
          txt.Add ("Target : " + t[i].getid() + " is not covered\n");
          return false;
        }
      }
      return true;
    }
  }
}
```

APPENDIX – B

1) Sample targets file

The first line shows the total number of targets and each successive line includes the target's id, x coordinate and y coordinate of each target.

```
20
0    97    43
1    43    5
2    6     85
3    42    88
4    30    24
5    23    74
6    62    31
7    21    65
8    5     94
9    81    49
10   63    13
11   82    48
12   4     45
13   53    55
14   66    22
15   88    26
16   99    34
17   39    32
18   86    6
19   92    67
```

2) Sample sensor file

The first line shows the total number of sensors and each successive line includes the sensor's id, total energy level, x coordinate and y coordinate of each sensor.

```
20
0    500    71    78
1    500    47    48
2    500    85    54
3    500    98    11
4    500    76    97
5    500    19    10
6    500    81    61
7    500    56    73
8    500    32    57
9    500    44    15
10   500    6     20
11   500    75    50
12   500    41    26
13   500    13    93
14   500    57    23
15   500    80    58
16   500    34    18
17   500    43    27
18   500    84    81
19   500    74    46
```