



**UNIVERSITY OF NAIROBI**

**SCHOOL OF COMPUTING AND  
INFORMATICS**

**ROUTING ALGORITHM FOR OPTIMAL  
LIFETIME OF WIRELESS SENSOR  
NETWORK**

**BY**

**Okiro, Joseph Onyango**

**P56/9009/2005**

**Supervisor**

**PROFESSOR WILLIAM O. ODONGO**

**November 2010**

---

Submitted in partial fulfillment of the requirements of the Master of Science in  
Computer Science

University of NAIROBI Library



0378862 7


## DECLARATION

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

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

JOSEPH ONYANGO OKIRO

P56/9009/2005

SIGN 

PROFESSOR WILLIAM O. ODONGO

PROJECT SUPERVISOR

SIGN 

## **ACKNOWLEDGEMENTS**

A special word of thanks to my supervisor Professor William Okello Odongo for his guidance and advice throughout the research.

Special thanks to my mother Jane, wife Grace, son Veinard and daughter Vinvella for your support, motivation, understanding and love during the period of the research.

To my late father Samuel Okiro; in your honour I strive.

To all from whom I have learned.

Most important of all: Thanks to GOD who deserves all the credit. For nothing is possible without him.

## **ABSTRACT**

Advances in wireless sensor networks have led to many new routing protocols specifically designed for sensor networks where energy awareness is an essential consideration. Routing in wireless sensor networks has received increasing attention and research in recent years whereas most routing protocols concentrate on finding and maintaining routes in the face of changing topology caused by mobility or other environmental changes. The nodes in wireless sensor network have limited initial amounts of energy that are consumed at different rates while forwarding, receiving and processing messages and the intended receiver which can be more than one hop away. Energy aware routing protocols and algorithms have been developed to address the issue of limited energy of the nodes in a wireless sensor networks. These energy aware routing protocols tries to ensure that the time until the batteries of the nodes drain-out is maximized. This project considers and examines energy aware routing algorithms with the goal of establishing one which maximizes the lifetime of a wireless sensor network. The comparison is performed by evaluating and modeling the routing algorithms and verifying the performance through simulation.

# TABLE OF CONTENTS

DECLARATION.....	I
ACKNOWLEDGEMENTS.....	II
ABSTRACT.....	III
TABLE OF CONTENTS.....	IV
LIST OF FIGURES.....	VI
LIST OF TABLES.....	VII
LIST OF ABBREVIATIONS.....	VIII
<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 PROBLEM STATEMENT.....	3
1.2 ASSUMPTIONS.....	5
1.3 METRICS FOR LIFETIME OF A WIRELESS NETWORKS .....	5
1.4 STATEMENTS OF HYPOTHESIS .....	6
1.5 PROJECT AIM AND OBJECTIVES.....	6
1.5.1 PROJECT AIM.....	6
1.5.2 PROJECT OBJECTIVES.....	6
<b>2. LITRATURE REVIEW.....</b>	<b>7</b>
<b>3. ENERGY AWARE ROUTING ALGORITHMS.....</b>	<b>9</b>
3.1 DIRECTED DIFFUSION ROUTING FOR WIRELESS SENSOR NETWORKS.....	9
3.2 SHORTEST PATH ROUTING ALGORITHMS.....	10
3.2.1 K-SHORTEST PATH ALGORITHMS AND DYNAMIC PROGRAMMING .....	10
3.2.2 BASE STATION MOBILITY .....	11
3.2.3 PATH REROUTING, NEIGHBOR SWITCHING AND PATH REPAIR.....	12
3.2.4 REAL TIME POWER AWARE ROUTING(RPAR) .....	16
3.2.5 SINGLE PATH WITH REPAIR ROUTING SCHEME(SWR).....	16
3.3 MINIMUM TRANSMITTED ENERGY (MTE) ROUTING ALGORITHMS .....	18
3.3.1 STATIC AND DYNAMIC POWER SCHEDULE.....	18
3.3.2 PROBABLISTIC ENERGY METRIC.....	18
3.3.3 DISTRIBUTED BINARY SEARCH.....	19
3.4 MAXIMUM RESIDUAL ENERGY PROTOCOL (MREP) ROUTING ALGORITHMS .....	21
3.4.1 MATHEMATICAL PROGRAMMING AND GREEDY MODEL.....	21
3.4.2 CLUSTERING METHODS.....	22
3.4.3 EXCEPTION MESSAGE .....	23
3.4.4 GRADIENT BASED ROUTING (GBR).....	24
3.4.5 PERMITIVITY AND HEURISTIC COST FACTOR .....	24
3.4.6 DELAY-CONSTRAINT, NODES LIFETIME AND DISTANCE BASED METRICS.....	25
3.4.7 DISTRIBUTED POWER-AWARE ALGORITHMS .....	26
3.4.8 ZONE BASED ROUTING.....	27
<b>4. ALGORITHM MODELLING.....</b>	<b>29</b>
4.1 THE BELLMAN-FORD ALGORITHM [34].....	30
4.2 MAXIMUM RESIDUAL ENERGY PATH (MREP).....	31
4.3 MINIMUM TRANSMITTED ENERGY (MTE).....	32
<b>5 DESIGN AND IMPLEMENTATION .....</b>	<b>35</b>
5.1 ASSUMPTIONS.....	35

- 5.2 DESIGN DESCRIPTION ..... 35
- 5.3 INPUT VARIABLES ..... 35
- 5.4 NODES AND TARGET DEPLOYMENT ALGORITHM ..... 36
- 5.5 UNIVERSAL MODELLING LANGUAGE (UML) DIAGRAM ..... 37
- 5.6 DEVELOPMENT TOOLS ..... 38
  - 5.6.1 ADVANTAGES OF C# OVER JAVA, C AND C++ ..... 39
- 6 SIMULATION, RESULTS AND CONCLUSIONS ..... 41
  - 6.1 SIMULATION WITH 50 SENSOR NODES ..... 43
  - 6.2 SIMULATION WITH 100 SENSORS ..... 47
  - 6.3 SIMULATION WITH VARIABLE NUMBER OF SENSORS ..... 51
  - 6.4 DISCUSSIONS AND CONCLUSIONS ..... 57
- APPENDIX A: REFERENCES AND BIBLIOGRAPHY ..... 59
  - REFERENCES ..... 59
  - BIBLIOGRAPHY ..... 63
- APPENDIX B: USER MANUAL ..... 64
- APPENDIX C: SAMPLE PROGRAMS ..... 67

## **LIST OF FIGURES**

<b>Figure 1. 2 : System architecture of a typical wireless sensor node</b>	<b>3</b>
<b>Figure 3. 1: Simplified schematic for directed diffusion.</b>	<b>10</b>
<b>Figure 3. 2 : Demonstration of unevenness in energy dissipation.</b>	<b>13</b>
<b>Figure 3. 3 : Common neighbor switching</b>	<b>14</b>
<b>Figure 3. 4 : Demonstrations of EERS</b>	<b>15</b>
<b>Figure 5. 1 : UML diagram</b>	<b>37</b>
<b>Figure 6. 1 : Simulator at a glance.</b>	<b>42</b>
<b>Figure 6. 2: A view of simulation in progress</b>	<b>43</b>
<b>Figure 6. 3: Graph of lifetime with 50 nodes</b>	<b>45</b>
<b>Figure 6. 4: Graph of live sensors at Lifetime x 2 with 50 nodes</b>	<b>45</b>
<b>Figure 6. 5 : Graph of residual energy at lifetime x 2 with 50 nodes</b>	<b>46</b>
<b>Figure 6. 6 : Graph of lifetime with 100 nodes</b>	<b>48</b>
<b>Figure 6. 7 : Live sensors at lifetime x 2 with 100 nodes</b>	<b>49</b>
<b>Figure 6. 8 : Graph of residual energy at lifetime x 2 with 100 nodes</b>	<b>49</b>
<b>Figure 6. 9 : Simulator with 47 nodes – Low density</b>	<b>51</b>
<b>Figure 6. 10 : Simulator with 410 nodes – high density.</b>	<b>52</b>
<b>Figure 6. 11 : Graph of lifetime with variable nodes.</b>	<b>54</b>
<b>Figure 6. 12: Graph of active nodes at lifetime * 2</b>	<b>55</b>
<b>Figure 6. 13: Graph of % active nodes at lifetime * 2</b>	<b>55</b>

**LIST OF TABLES**

**Table 1. 1 : Sensor network application domains. 2**  
**Table 6. 1 : Table of simulations results for MREP using 50 nodes 44**  
**Table 6. 2 : Table of simulation results for MTE using 50 nodes 44**  
**Table 6. 3 : Simulation results for MREP with 100 nodes. 47**  
**Table 6. 4 : Simulation results for MTE with 100 nodes. 48**  
**Table 6. 5: Table of simulation results for MREP using variable nodes. 53**  
**Table 6. 6 : Table of simulation results for MTE using variable nodes. 54**



## **LIST OF ABBREVIATIONS**

ACM	Association for Computing Machinery.
AODV	Ad hoc On-Demand Distance Vector.
CLR	Common Language Runtime.
CNS	Common Neighbor Switching.
DCEERP	Delay-constrained energy-efficient routing problem.
DD	Directed Diffusion.
DISJ2	2-disjointed paths without repair.
DSDV	Dynamic Destination-Sequenced Distance-Vector Routing.
DSR	Dynamic Source Routing.
EERFC	Energy efficient data relaying fixed clustering.
EERL	Energy efficient routing longest rerouting.
EERS	Energy efficient routing shortest rerouting.
ERR	Extended rerouting request.
FA	Flow Augmentation.
GBR	Gradient based routing.
GPS	Global Positioning System.
IEEE	Institute of Electrical and Electronics Engineers.
IL	Intermediate Language.
JVM	Java Virtual Machine.
LEACH	Low-energy adaptive clustering hierarchy.
LEFC	Low-energy fixed clustering.
MAC	Media Access Control.
MINAVG	Minimizing the average energy consumption.
MINMAX	Minimizing the maximum energy consumed by an active sensor.
MINREL	Minimizing relative energy consumption.
MREP	Maximum Residual Energy Path.
MSIL	Microsoft Intermediate Language.
MTE	Minimum transmitted Energy.
NeLMUK	Network lifetime maximization using K-shortest simple path algorithm.
PAN	Private Area Network.
RD	Rate Distortion.
RPAR	Real time power aware routing.
SFP	Shortest First Path.
SNLM	Statistical network lifetime measure.
SWOR	Single path without repair.
SWR	Single path with repair routing.
TDMA	Time division multiple access.
UML	Universal Modeling Language.
WSN	Wireless Sensor Network.

# 1. INTRODUCTION

Advances in microelectronics fabrication have allowed the integration of sensing, processing and wireless communication capabilities into low cost and small form factor embedded systems called sensor nodes. The availability of such small, cheap micro sensors and low power wireless communications has enabled the deployment of large arrays of wireless sensor networks allowing us to monitor and eventually control many aspects of the physical world.

“Wireless sensor networks could advance many scientific pursuits while providing a vehicle for enhancing various forms of productivity, including manufacturing, agriculture, construction, and Transportation.” – David Culler, university of California, Berkeley [1].

Wireless Sensor network consist of many inexpensive wireless sensor nodes each capable of collecting, processing, storing and communicating information (sending data).

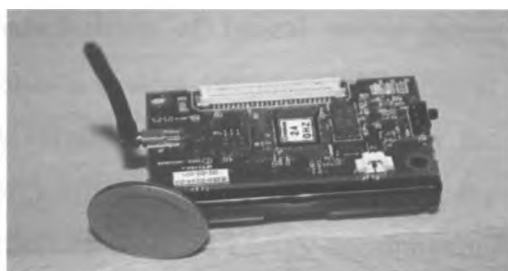


Figure 1. 1 : Sample sensor nodes/motes.

These sensors may be easily deployed in a manner and self-organized thus monitors the environment and forwards data back to a central node.

Military	Enemy tracking and detection, security detection, detection of nuclear, biological and chemical attacks and presence of hazardous material. Monitoring friendly forces, equipments and ammunition.
Habitat	Animal tracking.
Health	Remote patients monitoring, tracking and monitoring doctors inside hospital, identifying pre-defined symptoms by telemonitoring human physiological data.
Environment	Environmental data tracking to launch warning, Forest fire monitoring, flood

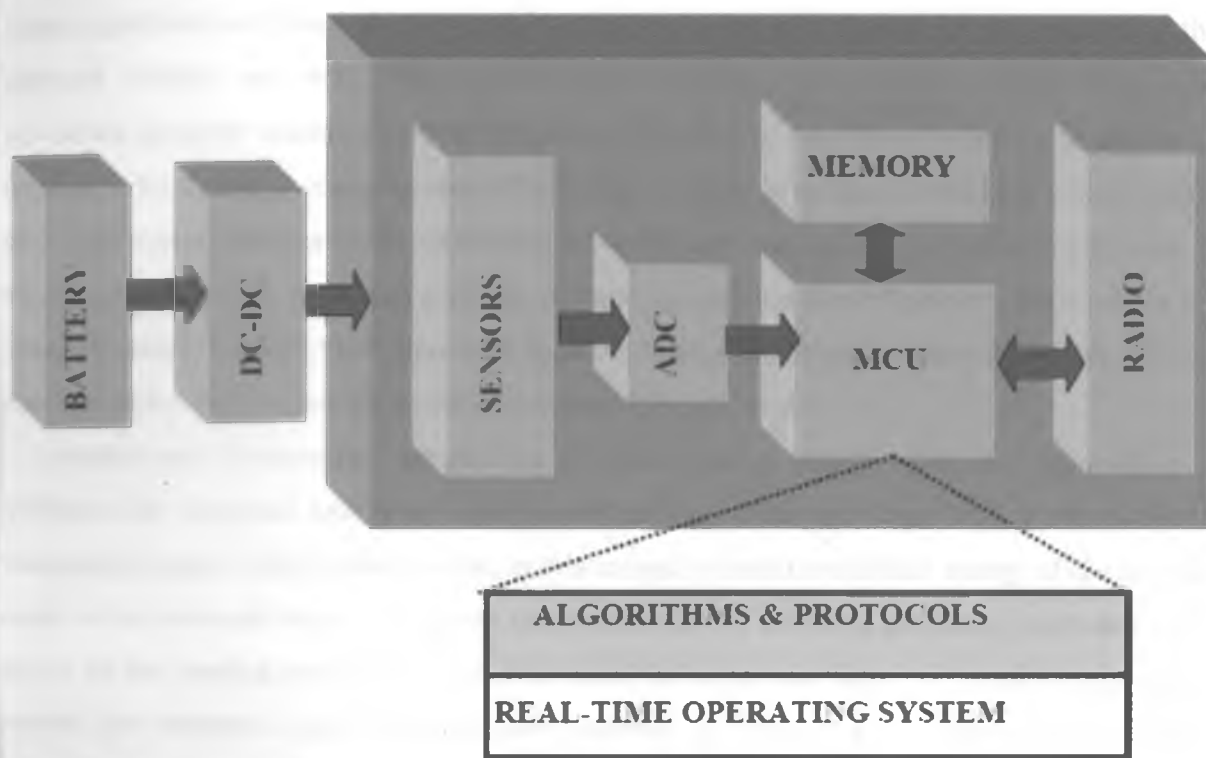
	detection, earthquake detection.
Smart home, office, classroom	Life quality improvement
Industry and business	Machine monitoring, inventory system.
Civilian	Traffic monitoring, available parking slots, security surveillance in banks and shopping malls, infrastructure.
Scientific	Under-sea exploration, study of cosmic radiation, space exploration

Table 1. 1 : Sensor network application domains.

V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless micro sensor networks. IEEE Signal Processing Magazine[2] Described architectural and algorithmic approaches that designers can use to enhance the energy awareness of wireless sensor network. They did an analysis of the power consumption characteristics of typical sensor nodes architectures, and identified the various factors that affects system lifetime. While power management of individual sensor nodes reduces energy consumption, it is important for the communication between nodes to be conducted in an energy efficient manner as well. Since the wireless transmission of data accounts for a major portion of the total energy consumption, power management decisions that take into account the effect of inter-node communication yield significantly higher energy savings. Further, incorporating power management into the communication process enables the diffusion of energy awareness from an individual sensor node to a group of communicating nodes, thereby enhancing the lifetime of entire regions of the network.

A salient feature of battery-powered WSN is its extremely constrained source of energy supplied by batteries coming with sensor nodes, because sensor nodes are typically small and thus use tiny batteries. In many scenarios, it seems infeasible to replace or recharge batteries of sensor nodes. At each node some type of information is generated as the sensors detect the target object, the information that needs to be delivered to some nodes designated as gateway nodes for further processing or delivery to a possibly larger network for retrieval by users. These wireless sensor network nodes have the capability of generating and/or relaying incoming packets to one of its neighboring nodes. Upon or before a new arrival of information either generated at the node itself or forwarded from other nodes, routing decision has to be made so that the nodes knows

which of its neighboring nodes to forward its data to. The routing decision and the transmission energy level selection are fundamentally connected in this power controlled wireless sensor network since the power level will be adjusted depending on the location of the next hop node. Many energy aware routing algorithms and protocols have been developed for wireless sensor network. This project aims to study and model these algorithms to establish the routing algorithm that provides optimal maximum lifetime of wireless sensor network



**Figure 1. 2** : System architecture of a typical wireless sensor node

## **1.1 PROBLEM STATEMENT**

Which of the routing algorithms maximizes the lifetime of a wireless sensor network? This is the question this project sets out to answer and or ascertain. Many Routing algorithms for wireless sensor networks have been developed by various researchers with many of them claiming their algorithms as the most energy efficient and therefore prolong the network lifetime. Pioneer researchers developed routing algorithms based on shortest path algorithms where they presented simple implementable algorithms which guarantees strong connectivity and assumes limited node ranges [3]. These Algorithms based on traditional shortest path algorithms did little,

if not nothing, in prolonging the network lifetime, as a result some researchers came up with some improvements. Algorithms were proposed where the base stations (the sink nodes) moves depending on the residual power of the nodes closest to it [4], [5], [6] the base stations will query the residual energies of the nodes then move closer to the nodes with highest residual energy. All these algorithms still used shortest path routing and were still faulted for not improving the network lifetime and will only perform best if energy was unlimited. Some researchers developed dynamic routing algorithms which use the idea of predictive re-routing to cope with unpredictable topology changes; one of the technique used is neighbor switching where a node with low power switches traffic to a neighbor with more power, and path rerouting in case of topology changes [7], [8]. Some researchers came up with the idea of path repair [9] where the network quickly finds an alternative path against a broken link. Most of these algorithms still use shortest path routing where the number of hops is the path length.

Another set of researchers focused on minimum energy routing where the approach is to minimize the consumed energy to reach the destination. These algorithms include the minimum transmitted energy (MTE) where some, in this category, used the residual energy of the sending node as the dominant factor [10], some used probability to choose a path using residual energy factor of the sending node [11], [12] some used the edge cost factor by employing a binary search for minimum and maximum cost required to connect to the network [13]. MTE algorithms can possibly partition the network faster since it will always route through the minimum transmission energy link, their performance can be improved by augmenting them with rerouting strategies. Quite a number of researches have been done on maximum residual energy path (MREP) algorithms, some work involved partitioning the nodes into clusters with the cluster head being closed to the sink node [14], [15]. The cluster heads are rotated based on their residual power level. Others involved an exception message where receivers tell the sending nodes to choose a different neighbor when their energy reduces below a given threshold [16], [17]. Several researchers used a permittivity and cost heuristic factor where the cost of routing through areas with heavy activities is increased, changing the permittivity factor to high in places with high residual energy nodes and setting it low otherwise [18]. Others used heuristic functions like linear programming to get the route from a node to the base in which the residual power is maximized among all the routes [19], [20], [21], [22]. MREP algorithms will ensure all the nodes have near uniform energy levels at all times.

With all the factors around routing for optimal lifetime of a wireless sensor networks, some researchers have shown that the maximum lifetime routing problem in general case may lead to multiple optimal solutions [23], while others concluded that the problem of maximizing network lifetime while preserving connectivity in general is NP-hard[24]. As a result answering this research question will be guided by the following assumptions.

## **1.2 ASSUMPTIONS**

- a. Nodes are randomly distributed in a defined region.
- b. Each node has a uniform initial energy level.
- c. Energy used in sensing, processing, idle periods is not a bottleneck and thus constant.
- d. Nodes route information towards specified gateway nodes.
- e. Mobile targets are randomly generated and move randomly across the plane of the defined region.

## **1.3 METRICS FOR LIFETIME OF A WIRELESS NETWORKS**

- a. Energy consumed per packet
- b. Time to network partition
- c. Variance in node power levels
- d. Cost per packet
- e. Number of alive sensors as a function of time
- f. Time until the first node dies
- g. Time span from the sensor deployment to the first loss of coverage i.e. the time when some area initially covered by the network is not sensed by any active node any more.

A protocol is energy efficient if it minimizes the accumulative energy consumption for fulfilling its tasks. It has been established that an energy efficient protocol does not necessarily maximizes the network lifetime.

The lifetime metric for this project will be the time until the first node dies; which encompasses time to network partition and time span from the sensor deployment to the first loss of coverage.

This project will study and use graph theory to model these algorithms and use simulation to study the performances and establish those which actually prolongs the lifetime of a wireless sensor network within the bounds of the above assumptions.

## **1.4 STATEMENTS OF HYPOTHESIS**

- a. Routing algorithms is a factor in the lifetime of energy constrained sensor network.
- b. Shortest path routing, minimum transmission energy routing and maximum residual energy routing algorithms perform differently in prolonging the lifetime of energy constrained wireless sensor network.

## **1.5 PROJECT AIM AND OBJECTIVES**

### **1.5.1 PROJECT AIM**

This project aims to establish the routing algorithm that provides optimal maximum lifetime of wireless sensor network.

### **1.5.2 PROJECT OBJECTIVES**

- I. Study and establish the routing algorithms with the best link cost function and shortest path calculation for optimal lifetime of a sensor network.
- II. Develop a computer program simulator to determine performance measures of the routing algorithms on the lifetime of a sensor network.

## 2. LITRATURE REVIEW

The growing interest in power aware routing algorithms in wireless sensor network inspired some previous research in comparing and classifying those algorithms in such a technical area. Visu et al, in their work , Energy-Efficient Routing in Wireless Sensor Networks Based on Data Reduction, during 2006 World Congress in Computer Science Computer Engineering, and Applied Computing Las Vegas, Nevada, USA [25] did an excellent work in defining data compression techniques e.g. coding by ordering, pipelined in-network compression and data reduction techniques e.g. information dissemination via label forwarding, differential coding, which can be used to reduce power consumption during routing, but in the end they did not provide which routing algorithm will result into more network lifetime using those techniques leaving it all to the developers to apply any algorithm. This means if an algorithm is not suitable for optimal network lifetime then the gains from those data compression and reduction techniques may be eroded. Similarly K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks," in the Elsevier Ad Hoc Network Journal, Vol. 3/3 pp. 325-349, 2005 [26] studied the approaches on data routing in sensor networks and classified the approaches into three main categories, namely data-centric, hierarchical and location based. In their classification of routing protocols in wireless sensor networks they indicated the routing protocols which utilizes data aggregation will achieve energy saving and traffic optimization. They neither established nor provided which of the algorithms is more energy efficient and/or will prolong the lifetime of a wireless sensor network when used with data aggregation.

N. Narasimha Datta and K. Gopinath "A survey of routing algorithms for wireless sensor networks," [27] classified the routing protocols into two broad categories namely flat and hierarchical, Hierarchical protocols organize the network nodes into several logical levels through a process called cluster formation. Flat routing protocols, on the other hand, attempt to find good-quality routes from source nodes to sink nodes by some form of flooding. They concluded their work with a good comparison of the routing protocols based on whether they incorporate the following metrics GPS required, Multi-path routing, MAC scheduling (TDMA), Mobility aware, Event driven, Energy distribution, Flooding involved, Intrusion tolerant, and Failure recovery. As much as their work addressed the properties and structures of the algorithms, they did not tackle energy aware routing and prolonging the network lifetime. Chansu Yu, Ben Lee and Hee Yong Youn "Energy efficient routing protocols for mobile ad hoc



networks,” in wireless communication mobile computing 2003; Volume 3: issue 8 [28] Surveyed and classified the energy aware routing protocols proposed for mobile ad-hoc networks. In their study they showed that energy aware routing protocols minimize either the active communication energy required to transmit or receive packets using transmission power control approach and load distribution approach or the inactive energy consumed when a mobile node stays idle but listens to the wireless medium for any possible communication requests from other nodes using the sleep/power-down mode approach. They showed that in many cases, it is difficult to compare these routing protocols directly since each method has a different goal with different assumptions and employs different means to achieve the goal. They concluded that more research is needed to combine and integrate some of the protocols presented to keep wireless sensor networks functioning for a longer duration. Similar work was also done by Jamal N. Al-Karaki and Ahmed E. Kamal, in their research “Routing Techniques in Wireless Sensor Networks: A Survey,” IEEE Wireless Communications, Volume: 11, Issue: 6, 26- 28, Dec. 2004 [29].

With the many algorithms developed for routing in wireless sensor networks and especially those that are energy aware, there is a clear need to establish if they achieve optimal maximum lifetime of a wireless sensor network, which is the contribution of this project. It is evident that not much has been done to establish whether the many algorithms, as claimed by their developers, offers the optimal lifetime of a wireless sensor network under general or specific conditions.

### **3. ENERGY AWARE ROUTING ALGORITHMS**

#### **3.1 DIRECTED DIFFUSION ROUTING FOR WIRELESS SENSOR NETWORKS**

Most of these algorithms compares heavily to directed diffusion concept. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and Fabio Silva, in their work Directed Diffusion for Wireless Sensor Networking, for the IEEE/ACM Transactions on Networking [30] , gave a good account for the directed diffusion concept.

Directed diffusion is data centric in that all communication is for named data; all nodes in directed diffusion based network are application aware. This enables diffusion to achieve energy savings by selecting empirically good paths and by caching and processing data in-network (data aggregation).

In directed diffusion for wireless sensor networks data generated by sensor node is named by attribute-value pairs. A node requests data by sending interests for named data, data matching the interest is then “drawn” down towards that node. An important feature of directed diffusion is that interest and data propagation and aggregation are determined by localized interactions (message exchange between neighbors or nodes within same vicinity).

Directed diffusion consists of several elements: interests, data messages, gradients and reinforcements. An interest message is a query or an interrogation which specifies what a user wants; each interest contains a description of a sensing task that is supported by a sensor network for acquiring data. A sensing task is disseminated throughout the sensor network as an interest for named data. This dissemination sets up gradient within the network designed to draw events (i.e. data matching the interest). Specifically a gradient is direction state created in each node that receives an interest; the gradient direction is set towards the neighbor node from which the interest is received. Events start flowing toward the originators of interest along multiple gradient paths; the sensor network reinforces one or a small number of these paths.

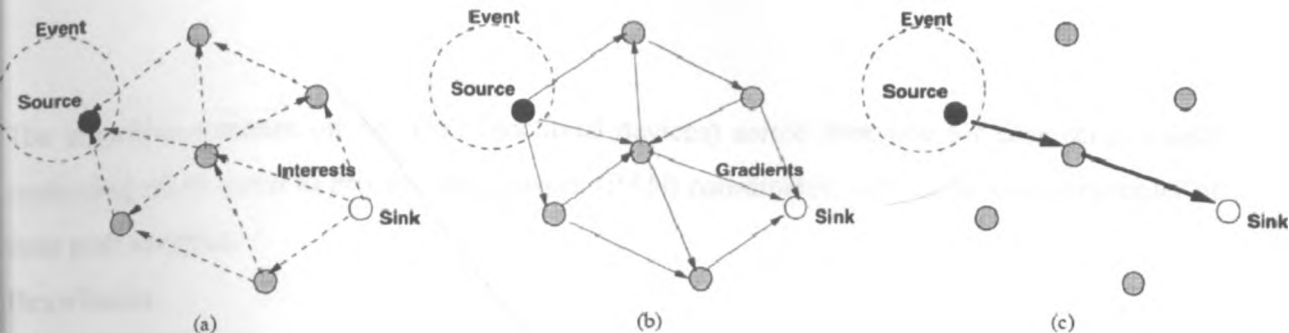


Figure 3. 1: Simplified schematic for directed diffusion.

(a) Interest propagation. (b) Initial gradients setup. (c) Data delivery along reinforced path

## 2. SHORTEST PATH ROUTING ALGORITHMS

### 2.1 K-SHORTEST PATH ALGORITHMS AND DYNAMIC PROGRAMMING

Muhammad U. Ilyas and Hayder Radha, in their work titled Increasing Network Lifetime of an IEEE 802.15.4 Wireless Sensor Network by Energy Efficient Routing [3] proposed one of the modified shortest algorithms. Their algorithm uses the K-shortest path algorithms [31], [32] and dynamic programming method rooted in operational rate distortion (RD) theory. Shortest first path (SFP) routing algorithms such as highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV), Dynamic Source Routing (DSR), Ad hoc On-Demand Distance Vector (AODV) or directed diffusion will cause over-utilization of some paths and energy starvation of the nodes on such paths.

They proposed the joint use of mean of energy consumption rate and variance of energy consumption rate in the form of ordered pair tuple referred to as statistical network lifetime measure (SNLM). SFP algorithms only minimize the mean energy consumption rate, jointly minimizing the variance of energy consumption rate ensures that the energy consumption rates are more uniform and that the traffic load on nodes closer to the base stations is reduced. In their work all nodes are considered equally important. They proposed the network lifetime maximization using K-shortest simple path algorithm (NeLMUK) algorithm which employs the K-shortest simple path algorithm together with an operational rate distortion (RD) algorithm to automatically reduce computational complexity.

The algorithm operates on  $N_{\text{ffd}}$  (full functional devices) sorted lists, one for each sensor, each containing all its paths to private area network (PAN) coordinator, ordered in ascending order of their path energies.

### **Drawbacks**

- a. It is possible the algorithm does not find the optimal operational mean of energy consumption rate and variance of energy consumption rate under certain scenarios.
- b. The sorting process is very complex.

### **3.2.2 BASE STATION MOBILITY**

D. Vass, Z. Vincze, R. Vida and A. Vidács, in their work Energy Efficiency in Wireless Sensor Networks Using Mobile Base Station [4] brought in the concept of base station mobility. Nodes have no mobility after deployment; the network is divided into small clusters in which it is assumed the sensors are distributed uniformly.

Employs two strategies for moving the base station

- a. Minimizing the average energy consumption (MINAVG)

This causes the network to always spend the minimum energy for the communication between the active sensor and the base station, i.e. the total energy of the network will remain the highest compared to other strategies.

- b. Minimizing the maximum energy consumed by an active sensor (MINMAX)

Minavg doesn't take into account the interests of the individual sensors i.e. if most of the active sensors are close to base station while one or more nodes are far from it, these sensors use much more energy than the others and deplete their battery sooner, so they have less lifetime.

Minmax is equivalent to minimizing the maximum distance between the base station and every active sensor in the network. They used an event driven network where a sensor sends data only when sensing an event, the experiment was done in a lab setup not through a computer program.

Minmax is good if the main goal is to have the first node die at the latest possible moment in time.

Minavg is good when the goal is to maximize the lifetime of the majority of the nodes.

If the number of simultaneous events is low, then Minmax and Minavg outperforms fixed Base stations in prolonging the network lifetime. Minmax is good if the network is operable only while every sensor is alive, Minavg is good if the operation of the network is able to tolerate the depletion of some sensors.

In their work they did not consider multi-hop network. They also only considered an event being sensed by only one sensor not multiple sensors.

Vass and A. Vidács in "Positioning Mobile Base Station to Prolong Wireless Sensor Network Lifetime" in CoNEXT 2005 Student Workshop, pp. 300-301, Toulouse, France, 24-27 October, 2005 [5]; introduced minimizing relative energy consumption (MINREL).

MINREL takes into account the current status of the sensor nodes, thus it is able to protect from depletion of those nodes that have already sensed and reported many events and their power are getting exhausted. This is a weakness in Minavg and Minmax.

Minrel works well when the main goal is to have the first node die at the latest possible moment in time. With Minrel the total energy of the network decreases more rapidly, on the longrun the minavg proves a better choice.

Jun Luo, Jean-Pierre Hubaux, in "Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks" during the proceedings INFOCOM 2005 [6] also introduced mobility of base stations.

A base station can become mobile thanks to the advances made in the field of robotics. First fix the routing strategy to shortest path routing and search for the optimum mobility strategy, then based on the optimum mobility strategy search for the routing strategy that performs better than short path routing.

### **3.2.3 PATH REROUTING, NEIGHBOR SWITCHING AND PATH REPAIR**

Some researchers developed algorithms based on path rerouting, neighbor switching and path repair routing concepts.

Wei Ding, S. Sitharama Iyengar, Rajgopal Kannan and William Rummler, in their work "Energy equivalence routing in wireless sensor networks" [7] proposed neighbor switching and path rerouting concept in prolonging the lifetime of a wireless sensor network. Neighbor switching utilizes density and path redundancy in wireless sensor networks. Neighbor switching substitutes

the node with a neighbor outside the original routing tree according to a given criterion. It changes routing tree at very small scale so energy uniformity is achieved with least energy cost.

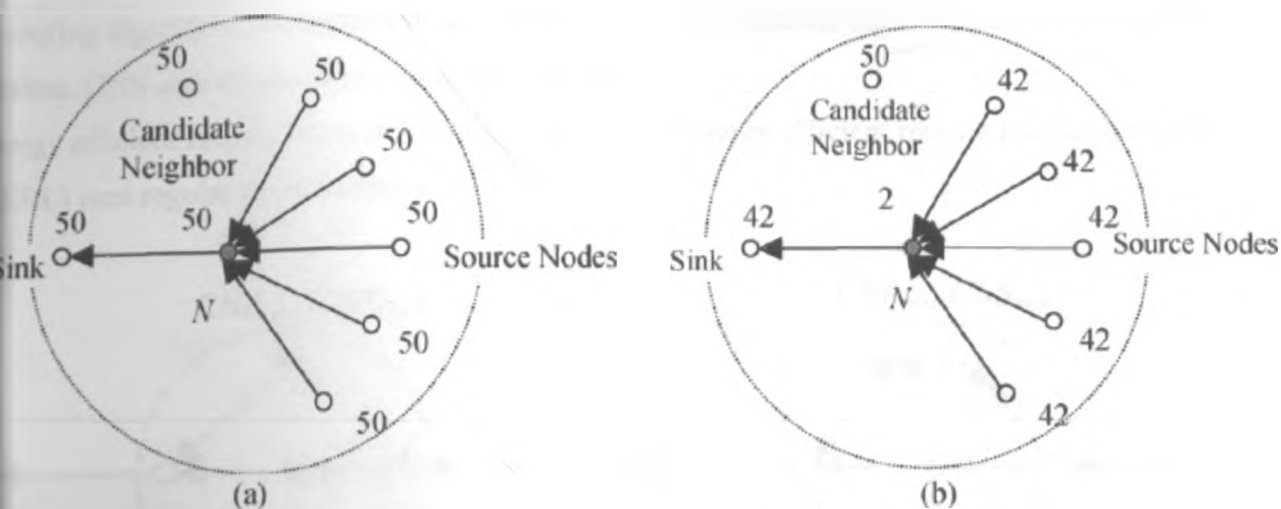


Figure 3.2 : Demonstration of unevenness in energy dissipation.

(a) Initial energy 50 units for all sensor and (b) residual energy after 80 time units. N is a converging node, sensing interval = 10 time units, transmission energy = reception energy = 1.

source and sink nodes could not be switched and the prerequisite of neighbor switching is adequate density or redundancy in vicinity of replaced node. In single neighbor switching N is replaced by one single neighbor with most energy residue which is common neighbor of N (switched node), P (preceding node) and every node in a set of succeeding nodes (SS). In double neighbor switching N is replaced by  $|SS|+1$  double neighbors; one is common neighbor of P and N, the rest are common neighbors of N and every S member of SS.

With rerouting links an outside substitute neighbor back to the original routing tree in which a single neighbor switching is used. Rerouting is still a flooding, it costs much more energy than neighbor switching. Shortest rerouting links the replacing neighbor to the nearest descendant node, while longest rerouting links to the farthest descendant node. The nearest descendant node is the first descent in the path which does not need to be replaced.

The rerouting procedure follows the directed diffusion (DD) protocol but with the constraint that every node in the rerouting path should need no rerouting i.e. it doesn't have a neighbor with energy difference beyond the threshold.

Common neighbor switching (CNS) uses double switching and is the most efficient. It has minimal additional energy cost and minimal time complexity since it never uses flooding in route maintenance.

Routing algorithms are inefficient as rerouting overhead balances out their gain in the network lifetime. CNS uses extended rerouting request (ERR).

Energy efficient routing shortest rerouting (EERS) and energy efficient routing longest rerouting (ERL) uses regular reroute request

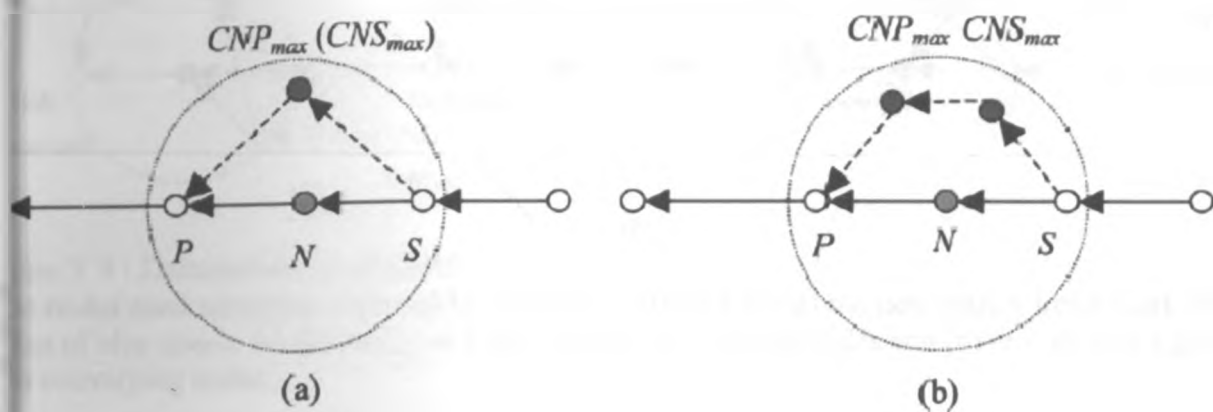
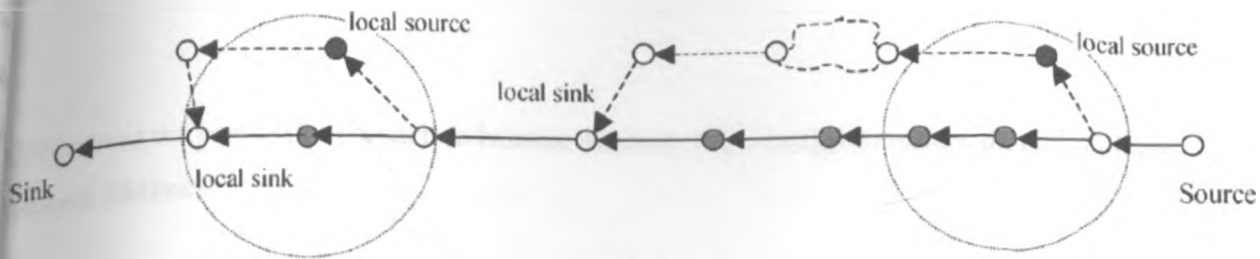
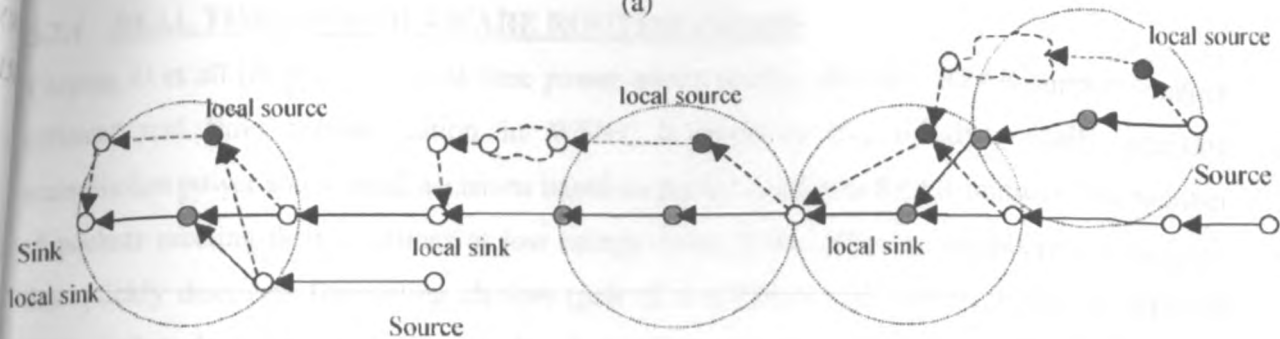


Figure 3.3 : Common neighbor switching when (a)  $CNP_{max} \frac{1}{4} CNS_{max}$  and (b)  $CNP_{max} - CNS_{max}$ .



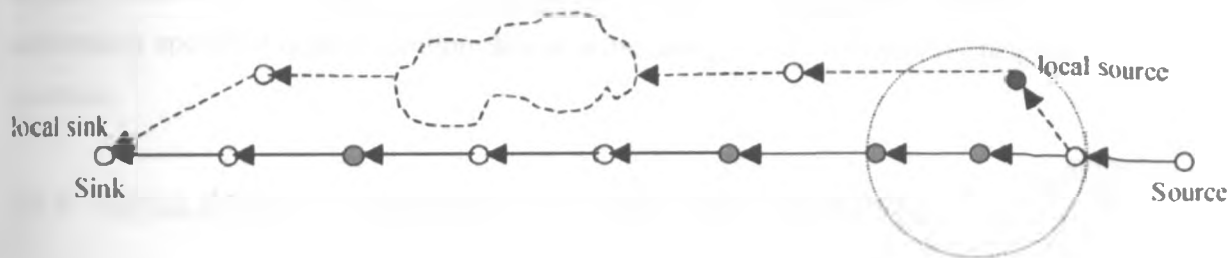
(a)



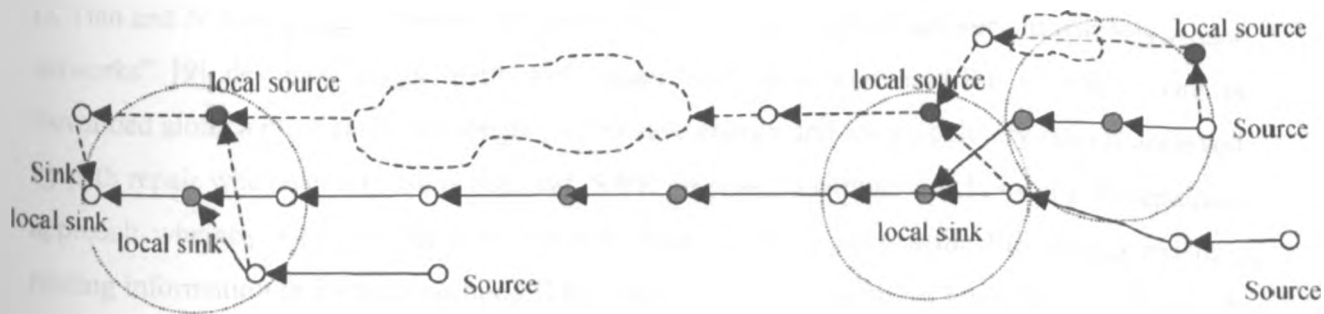
(b)

Figure 3.4 : Demonstrations of EERS

Blue nodes need rerouting, replaced by red nodes. Dashed line is the new path. Circles mark the radius of blue nodes. (a) Rerouting in a path without converging nodes and (b) rerouting in a path with converging nodes.



(a)



(b)

Figure 3.5 : Demonstrations of EERL

Blue nodes need rerouting, replaced by red nodes. Dashed line is the new path. Circles mark the radius of blue nodes. (a) Rerouting for a path without branches and (b) rerouting for a path with branches.



CNS proved the best EER approach to balance network wide energy consumption and prolong network lifetime.

#### **3.2.4 REAL TIME POWER AWARE ROUTING(RPAR)**

Chipara, O et all [8] proposed Real time power aware routing (RPAR). RPAR supports energy efficient real time communication in WSNs. It achieves this by dynamically adapting transmission power and routing decisions based on packet deadlines RPAR improves the number of packets meeting their deadlines at low energy costs, it has efficient neighborhood manager that quickly discovers forwarding choices (pair of a neighbor and a transmission power) that meet packet deadlines while introducing low communication and energy overhead. RPAR addresses important practical issues in WSNs like lossy links, scalability, and server memory and bandwidth constraints. It is based on the hypothesis that there is an inherent tradeoff between transmission power and communication delay.

RPAR intergrates novel real time routing and dynamic power adaptations algorithms to achieve application specified communication delays at no energy cost. RPAR uses neighbor switching protocol.

#### **3.2.5 SINGLE PATH WITH REPAIR ROUTING SCHEME(SWR)**

D. Tian and N. Georganas. "Energy efficient routing with guaranteed delivery in wireless sensor networks" [9] designed Single path with repair routing scheme (SWR). In SWR Data is forwarded along a pre-established single path to save energy and a high delivery ratio is achieved by path repair whenever a break is detected. SWR proposed a simple ,quick, local path repairing approach whereby a pivot node can skip over path break by only using the already existing routing information in its neighborhood. The scheme sets out to achieve high delivery ratio with low energy consumption.

In single path routing , for each data packet, there is only one copy traveling along one path in the network. In multipath routing multiple copies of one packet are transmitted in parallel along different paths to the same destination.

Single path routing is simple and consumes less energy however a single path failure will cause a break of transmission and hence no delivery. Multipath routing schemes have shown higher

resilience to single path failures however determining the width of multipath routing before transmission is not so easy, because sensor network topologies often change unpredictably. It also causes unnecessary energy waste, may cause network congestion but it guarantees high delivery.

SWR achieves high delivery ratio with low energy consumption by forwarding data along a single path and repair the path whenever a break is detected. SWR proposed a path repairing approach which can quickly find an alternative path against a broken link by doing a small survey around the break and only using already existing routing information.

The proposed schemes performances were based on the following metrics:

- a. Delivery ratio which is the ratio of the number of data packets successfully received by sink nodes to the total number of data packets sent by the source nodes.
- b. Average energy consumption which is the ratio of the total energy dissipation to the total number of delivered data packets.
- c. Energy consumed in path repair.

SWR performed better against the following schemes:

- a. Data forwarding along single path without repair (SWOR).
- b. Data forwarding along 2-disjointed paths without repair (DISJ2)- two copies of the data packets are delivered separately along two predetermined paths.
- c. Mesh data forwarding (MESH\_1.3, MESH\_0.13)- Each packet carries a credit which specifies the amount of extra cost allowed beyond the source node's minimum cost to the sink nodes. Multiple copies can go along different paths that interleaves and form a mesh.

### **3.3 MINIMUM TRANSMITTED ENERGY (MTE) ROUTING ALGORITHMS**

Minimum transmitted energy routing algorithms aims to minimize the consumed energy to reach the destination; the residual energy of the sending node is the dominant factor.

#### **3.3.1 STATIC AND DYNAMIC POWER SCHEDULE**

P. Floréen, P. Kaski, J. Kohonen, P. Orponen: in their publication "Lifetime maximization for multicasting in energy-constrained wireless networks" [10] designed an algorithm that tries to maximize the lifetime of a given multicast connection in a wireless network of energy constrained nodes by choosing ideal transmission power levels for the nodes relaying the connection. Their work distinguishes two basic operating modes:

- a. Static power assignment where the power levels of the nodes are set at the beginning and remain unchanged until the nodes are depleted.
- b. Dynamic power schedule: the powers can be adjusted during operation.

They showed that while lifetime maximizing static power assignments can be found in polynomial time, for dynamic schedules the problem becomes NP-hard. They introduced two approximation heuristics for the dynamic case and verified that the lifetime of a dynamically adjusted multicast connection can be made several times longer than what can be achieved by the best possible static assignment.

The power assignment algorithms require some degree of centralised control of the network, either all the nodes need to be aware of the networks complete structure and initial energy state, or they need to communicate with some central coordinating node. The algorithm is useful in determining energy levels of the nodes.

#### **3.3.2 PROBABLISTIC ENERGY METRIC**

Shah R., C., and Rabaey J., M., in "Energy Aware Routing for Low Energy Ad Hoc Sensor Network" [11] developed a routing protocol that keeps a set of good paths and chooses one based on a probabilistic fashion, a communication would use different paths at different times. Multiple paths are found between source and destinations and each path is assigned a probability of being chosen depending on the energy metric. Every time data is to be sent from source to destination one of the paths is randomly chosen depending on the probabilities.

It is a destination-initiated protocol where the consumer of data initiates the route request and maintains the route subsequently; the energy metrics used to evaluate the routes are cost of using the path, energy levels of the nodes along the path.

The protocol was compared against direct diffusion routing.

Proactive routing protocols: Have the distinguishing characteristic of attempting to maintain consistent up-to-date routing information from each node to every other node in the network. Every node maintains one or more routing tables that store the routing information, and topology changes are propagated throughout the network as updates so that the network view remains consistent. Destination Sequenced Distance Vector (DSDV) routing protocol is an example.

Reactive routing protocols: Creates routes only when desired. An explicit route discovery process creates routes and this is initiated only on an as-needed basis, it can either be source initiated routing where the source node initiates the discovery process or destination initiated where the destination node begins the discovery process. Ad-hoc On-demand Distance Vector Routing (AODV) is reactive source initiated, Dynamic Source Routing (DSR) is reactive source initiated. Directed diffusion is reactive destination initiated.

### **3.3.3 DISTRIBUTED BINARY SEARCH**

André Schumacher, Pekka Orponen, Thorn Thaler and Harri Haanpää, in their work "Lifetime Maximization in Wireless Sensor Networks by Distributed Binary Search," [12] proposed a protocol in which, in the first stage nodes collect neighborhood information and estimate link costs by transmitting and receiving beacon messages, the reference nodes also obtains a count of the number of nodes in the network.

The second stage performs a binary search over the range of possible transmission power levels. The final stage consists of a network broadcast in which the reference node notifies all the other nodes of the global termination of the algorithm and the resulting minimum maximum power level.

At each iteration of the binary search algorithm, the reference node initiates the computation of a rooted tree spanning the nodes, that can be reached from the reference node using paths with maximum edge cost. After the search has terminated the reference nodes informs all other nodes about the termination and the minimum edge cost necessary to connect all nodes. This cost can then be used to locally determine the transmission power level required at each node. After the

global termination each node has information of all other nodes in its maximum transmission range, as well as the costs of the incident edges. Additionally the reference node knows the total number of nodes within the network, with each run the maximum edge cost is provided so only edges with at most the maximum cost are returned.

The algorithm is thus based on a binary search for the minimum maximum edge cost that is required to connect the network.

### 3.4 MAXIMUM RESIDUAL ENERGY PROTOCOL (MREP) ROUTING ALGORITHMS

The idea behind MREP is to route packets through paths with maximum residual energy so that energy consumption in all paths will be balanced.

#### 3.4.1 MATHEMATICAL PROGRAMMING AND GREEDY MODEL

A. Alfieri, A. Bianco, P. Brandimarte and C. F. Chiasserini, in their work “Maximizing system lifetime in wireless sensor network” [13] designed an algorithm where they defined network lifetime as the time spanning from the instant when the network starts functioning properly i.e. satisfying the target level of coverage of the area of interest until the same level of coverage cannot be guaranteed anymore due to lack of energy in sensors.

They proposed two approaches, first one based on a mathematical programming model and the second one is a greedy approach that should be more easily implemented in a distributed way in a realistic scenario.

**Column generation** : Generates a set of network configurations (subnets) each of which is connected and meets the minimal coverage requirements, then determines how much time each subnet is used. The columns are selected and the length of the time interval a subnet is used is decided subject to energy budget constraints for each node, this is done by the subnet selection (master) problem. The subnet generation sub problem aims at finding a feasible subnet ensuring the minimal required covering; the objective is to cover the set of points with minimum costs subject to quality constraints.

**Greedy approach:** Only a subset of sensors is active for a given period of time, named scheduling period, whereas all other sensors are in inactive state saving energy for future scheduling periods. Their proposed approach consists of the following steps:

1. Form a proper (i.e. covering) subset of sensors that will be switched on (active sensors); if the subset does not guarantee the required level of coverage, discard this instance and repeat step 1;
2. Put all other sensors in off state (inactive sensors);
3. Determine a suitable minimum cost routing to transfer the sensed information from all active sensors to the gateway node; if this is not possible with the selected subset of sensors to guarantee full connectivity, i.e., if not all of the active sensors are able to

communicate, possibly via multi-hop transmission, with the gateway node, the subset is discarded and the process is restarted from step 1;

4. Determine the scheduling period duration, i.e., the amount of time for which this sensor subnet lasts, while guaranteeing the target level of coverage;
5. Compute sensors energy consumption over this time interval, subtract it from each sensor energy budget, and eventually consider some of the sensors as unavailable in the future due to energy depletion;
6. Iterate through this process until no other subset of covering and fully connected sensors can be found.

### **3.4.2 CLUSTERING METHODS**

Taewook kang, jangkyu yun, hoseung lee, icksoo lee, hyunsook kim, byunghwa lee, byeongjik lee, kijun han, in their study "A Clustering Method for Energy Efficient Routing in Wireless Sensor Networks," [14] proposed a new cluster-based routing protocol in order to distribute cluster heads evenly over the network and reduce energy dissipation. Tries to evenly distribute cluster heads over the whole network and avoid creating redundant cluster heads within a small range so that it can increase the network lifetime.

Sensor nodes are randomly deployed and some of them initially selected as candidate nodes. The nodes not selected as cluster heads are chosen to become candidate nodes, one of the candidate nodes broadcast an advertisement message within its range, nodes receiving this advertisement message are ruled out the qualification of candidate nodes. Ordinary nodes decides the cluster to which it will belong based on the signal strength of the advertisement message, after each node has decided to which cluster it belongs nodes must transmit its date to the appropriate cluster head.

Simulation results showed the scheme offers a better performance than the Low-Energy adaptive clustering hierarchy (LEACH) protocol in terms of network lifetime.

#### **Low-energy adaptive clustering hierarchy (LEACH).**

LEACH randomly selects a few nodes as cluster heads and rotates this role to balance the energy dissipation of the sensor nodes in the networks. It offers no guarantee about the placement and or number of cluster heads.

LEACH-C uses a centralized clustering algorithm and produces a better performance by dispensing the cluster heads throughout the network. It attempts to minimize the amount of energy for the ordinary nodes to transmit their data to the cluster head.

Y.-F. Huang et al in their study "Lifetime Performance of an energy efficient clustering algorithm for cluster-based wireless sensor networks," [15] proposed a novel energy efficient data relaying scheme which improves energy efficiency for a cluster based wireless sensor networks. The fixed clustering algorithm uniformly divides the sensing area into clusters where the cluster head is deployed to the centered of the cluster area. To perform energy efficient data relaying fixed clustering (EERFC), the cluster head is deployed as close to the sink as possible. In EERFC cluster heads are deployed to the closest site to the base station.

#### **Low-energy fixed clustering (LEFC)**

Yung-Fa Huang, Neng-Chung Wang, Ming-Che Chen in their work Performance of a Hierarchical Cluster-Based Wireless Sensor Network [33] combined fixed clustering algorithms and LEACH and proposed low-energy fixed clustering scheme to improve energy efficiency and prolong the network lifetime. LEFC can not only compromise the balancing on energy consumption in performing cluster head (CH), but also improve the energy efficiency of the sensing nodes. Therefore, it is easily observed that the proposed LEFC outperforms the LEACH and Direct schemes. The proposed LEFC gives uniform area of cluster area for the WSN and save the energy dissipation of normal sensor nodes in the cluster. Simulation results show that the LEFC can efficiently cluster the sensing nodes to minimize the energy dissipation and then outperform LEACH.

#### **3.4.3 EXCEPTION MESSAGE**

Ioan Raicu, Loren Schwiebert, Scott Fowler and Sandeep K.S. Gupta, in their research "e3D: An Energy-Efficient Routing Algorithm for Wireless Sensor Networks," [16] introduced the concept of exception message in MREP.

They introduced e3D; a diffusion based algorithm using location, power and load as metrics. Each node makes a list of suitable neighbors and ranks them in order of preferences, each time a node changes neighbours the sender will require an acknowledgement for its first message which will ensure that the receiving node is still alive. Receivers can issue exceptional messages telling



sending nodes to stop sending and let the sender choose a different neighbour, exception messages are generated in three instances:

1. Receiving node queue is too large.
2. The receivers' residual energy is less than the senders' residual energy.
3. The receiver has passed a certain threshold which means that it has very little energy left.

The receiver will analyse the receiving packets for the senders' energy levels.

#### **3.4.4 GRADIENT BASED ROUTING (GBR)**

C. Schurgers and M. B. Srivastava, in their study "Energy efficient routing in wireless sensor networks," [17] defined the network lifetime as the worst case time until a node breaks down. They introduced gradient based routing (GBR). When a network is being flooded, the 'interest' message records the number of hops taken, this allows a node to discover the minimum number of hops to the user called the nodes height. The difference between the nodes height and that of its neighbor is considered the gradient of that link; a packet is forwarded on the link with the largest gradient.

When a node detects that its energy reserve has dropped below a certain threshold it discourages others from sending data to it by increasing its height, it in turn informs other nodes and these updates are propagated as far as is needed to keep all the gradients consistent.

#### **3.4.5 PERMITTIVITY AND HEURISTIC COST FACTOR**

Some set of researchers introduced the concept of permittivity and heuristic cost factors in MREP algorithms.

Mehdi Kalantari, Mark Shayman, in their study "Energy Efficient Routing in Wireless Sensor Networks," [18] proposed an energy efficient routing scheme based on matching the routes to energy constraints in order to increase the network lifetime. When the energy of the sensors in some area of the network is low due to heavy communication in the past, an increase in the cost of routing through this area to protect the sensors from early depletion.

The routing scheme is based on changing the permittivity factor to a higher value in places with high residual energy of the nodes and set it to low value for places with low residual energy nodes. The central nodes collect all the information like the position of the sensors and the

residual energy to find the routes, the routes can then be updated once in a while when considerable change in the residual energy has occurred.

### **3.4.6 DELAY-CONSTRAINT, NODES LIFETIME AND DISTANCE BASED METRICS**

D. Ranganathan, P. K. Pothuri, V. Sarangan, and S. Radhakrishnan, in their study “Energy-efficient routing in wireless sensor networks for delay sensitive applications,” [19] proposed a heuristic solution to delay-constrained energy-efficient routing problem (DCEERP) where in a given delay of  $d'$  seconds the task is to find a path from a sensor node to the sink node with the lowest energy consumption such that the total transfer delay incurred along the path is less than  $d'$  seconds.

Stojmenovic, I., Lin, X., in “Power-Aware Localized Routing in Wireless Networks,” [20] Proposed a power cost metric based on the combination of both nodes lifetime and distance based power metrics.

Types of metrics

- a. Metrics based on remaining battery power at nodes – cost aware.
- b. Power aware metrics where transmission power depends on distance between nodes and corresponding shortest power algorithms

Power aware routing algorithm attempts to minimize the total power needed to route a message between a source and a destination. Cost aware routing algorithm is aimed at extending the battery's worst case lifetime at each node. The combined power-cost localized routing algorithm attempts to minimize the total power needed and to avoid nodes with short battery remaining lifetime.

If nodes have information about the position and activity of all other nodes then the optimal power saving algorithm that will minimize the total energy per packet can be obtained by applying Dijkstra's single source shortest weighted path algorithm SP power algorithm.

The source for an intermediate node B should select one of its neighbours to forward the packet towards the destination with the goal of reducing the total power needed for the packet transmission. The localized cost efficient routing algorithm can be described as follows:

If the destination is one of source or intermediate node neighbours or currently holding the packet, then the packet will be delivered to the destination. Otherwise the source/intermediate

will select one of its neighbours. The algorithm proceeds until the destination is reached or until a node selects the neighbour the message came from as its best option to forward the message. Both power and cost considerations may be incorporated into a single routing algorithm – power-cost metric. Control messages to update positions of all nodes to maintain efficiency of routing algorithms.

### **3.4.7 DISTRIBUTED POWER-AWARE ALGORITHMS**

Qun Li and Javed Aslam and Daniela Rus. in Distributed Energy-Conserving Routing Protocols for Sensor Network. [21] Developed three distributed power-aware algorithms:

#### **a. Distributed minimal power algorithm**

Developed a distributed version of Dijkstra's algorithm that is guaranteed to be a minimal-power routing path algorithm by giving messages variable propagation delays. The idea is to have messages traveling along short paths move faster than messages traveling along longer paths. Thus messages traveling along shorter paths will arrive faster than messages traveling along longer paths i.e. the algorithm will select the shortest paths. In this case Dijkstra distance corresponds to energy consumption.

The idea is implemented by augmenting each message with a record of how far it traveled from the base to the current node. This information is represented by a variable attached to the message that measures the cost (distance representing power consumption). This algorithm produces the minimal power-consumption path for each node; the running time of the algorithm is proportional to the longest shortest distance from the base node to any node.

#### **b. Distributed Max-Min algorithm**

Max-Min path is the route from a node to the base on which the minimal residual power of the nodes is maximized among all routes. Max-Min paths are found by using a modified version of the distributed bellman-ford algorithm. Upon computing a new max-min value, each node broadcast it. The neighbours compute their max-min value according to the new incoming value and broadcast the result only if the value is changed. This algorithm can be improved further using binary search.

#### **c. Distributed max-min $zP_{min}$**

Motivation is to define a routing algorithm that optimizes the overall lifetime of the network by avoiding nodes of low power while not using too much total power. There is a tradeoff between

minimizing the total power consumption and maximizing the minimal residual power of the network. They proposed to enhance a max-min path by limiting its total power consumption. Every time the route information of a node changes the information is broadcast until the system achieves equilibrium.

It is possible to improve the number of message broadcasts by using timing variables to suppress some of the messages, two specific approaches are:

- a. In the max-min part let the message carry the total power consumption on the path and use the power consumption to decide if the max-min value should be accepted.
- b. In the minimal power path part incorporate the max-min value in the waiting time.

### 3.4.8 ZONE BASED ROUTING

Qun Li and Javed Aslam and Daniela Rus. In their study Hierarchical Power-aware Routing in Sensor Networks [22] developed an approximation algorithm called max-min zPmin that has a good empirical competitive ratio, they introduced a hierarchical algorithm called zone-based routing.

Their work focused on a global metric by maximizing the time to the partition of the network, modeled as the time to the failure of the first node, this metric is very important for ad-hoc network where messages have to be delivered at high rates. They also proposed an online approximation algorithm for power aware message routing that optimizes the lifetime of the network. The algorithm combines the benefits of selecting the path with the minimum power consumption and the path that maximizes the minimal residual power in the nodes of the network.

It would be desirable to route messages along the path with the maximal minimal fraction of remaining power after the message is transmitted, this path is called max-min path. A concern with the max-min path is that going through the nodes with high residual may be expensive as compared to the path with the minimal power consumption, too much power consumption decreases the overall power level of the system and thus decreases the lifetime of the network.

The two extreme solutions to power-aware routing for one message are:

- a. Compute a path with minimal power consumption  $P_{min}$  and
- b. Compute a path that maximizes the minimal residual power in the network.

Their algorithm relaxes the minimal power consumption for the message to be  $zP_{min}$  with parameter  $Z \geq 1$  to restrict the power consumption for sending one message to  $zP_{min}$ , the algorithm proposed is called max-min  $zP_{min}$  that consumes at most  $zP_{min}$  while maximizing the minimal residual power fraction. The optimal strategy was computed by using a linear programming package.

The max-min  $zP_{min}$  algorithm requires accurate power level information for all the nodes in the network. For large scale sensor networks this is not a feasible assumption.

Zone based routing clusters together groups of sensors and estimates the overall routing power of the cluster for the purpose of the max-min  $zP_{min}$  algorithm. The idea is to group together all the nodes that are in geographical proximity as a zone, treat the zone as an entity in the network and allow each zone to decide how to route a message across. A global controller for message routing manages the zones, this may be the node with the highest power, can employ other schemes e.g. round robin.

If the network can be divided into a relatively small number of zones, the scale for the global routing algorithm is reduced, the global information required to send each message across is summarized by the power level estimates of each zone.

#### **Zone power estimation**

The power estimate for each zone is controlled by a node in the zone; the controller node polls each node for its power level followed by running the max-min  $zP_{min}$  algorithm. The return value is then broadcasted to all the zones in the network; the power estimation is done relative to the direction of message transmission.

#### **Global path selection**

Given power levels for each possible direction of message transmission, it is possible to construct a small zone-graph that models the global message routing problem. In addition the message direction vertices are connected to the neighboring zone vertices of the current zone and can go to the next neighboring zone in that direction.

#### **Local path selection**

The max-min  $zP_{min}$  algorithm is used directly to route a message within a zone.

The zone-based routing algorithm does not require as much information as would be required by max-min  $zP_{min}$  algorithm over the entire network.

#### 4. ALGORITHM MODELLING

Consider directed graph  $G(N,A)$  where  $N$  is the set of all nodes and  $A$  is the set of all directed links  $(i,j)$  where  $i,j \in N$ . Let  $S_i$  be the set of nodes that are in the transmission range of node  $i$ , link  $(i,j)$  exists if and only if  $j \in S_i$ . Each node has the initial battery energy of  $E_i$  and the amount of energy consumed in transmitting a packet across link  $(i,j)$  is denoted by  $e_{ij}$  where  $j \in S_i$ . Let  $Q_i$  be the rate at which information is generated at node  $i$ , the rate at which information is transmitted from node  $i$  to node  $j$  is called the flow  $q_{ij}$ . We have a set of origin nodes  $O$  where the information is generated and a set of destination nodes  $D$  among which any node can be reached in order for the information to be considered done.

The conservation of flow condition at each node  $i$  is assumed i.e. the sum of all incoming flow must be the same as the sum of all outgoing flow,

$$\sum_{j: i \in S_j} q_{ji} + Q_i = \sum_{k \in S_i} q_{ik}, \quad \forall i \in N - D$$

The time it takes for the battery of node  $i$  to drain out under flow  $q = \{q_{ij}\}$  is given by

$$T_i(q) = \frac{E_i}{\sum_{j \in S_i} e_{ij} q_{ij}}$$

**System lifetime:** The lifetime of the system under flow  $q$  is defined as the minimum battery lifetime over all nodes.

$$T_{sys}(q) = \min_{i \in N} T_i(q)$$

$$= \min_{i \in N} \frac{E_i}{\sum_{j \in S_i} e_{ij} \sum_{c \in C} q_{ij}^{(c)}}$$

The problem can be expressed as follows:

$$\max_{\mathbf{q}} T_{sys}(\mathbf{q}), \quad \text{or equivalently,} \quad \max_{\mathbf{q}} \min_{i \in N} \frac{E_i}{\sum_{j \in S_i} c_{ij} q_{ij}}$$

The problem of maximizing the system lifetime can be expressed as a linear programming problem given by:

Maximize  $T$

$$\hat{q}_{ij} \geq 0, \quad \forall j \in S_i, \forall i \in N \quad D$$

$$\sum_{j \in S_i} c_{ij} \hat{q}_{ij} \leq E_i, \quad \forall i \in N \quad D,$$

$$\sum_{j: i \in S_j} \hat{q}_{ji} + TQ_i = \sum_{k \in S_i} \hat{q}_{ik}, \quad \forall i \in N \quad D,$$

where  $\hat{q}_{ij} = Tq_{ij}$  is the amount of information transferred during  $T$  from node  $i$  to  $j$ .

**Theorem 1 (Necessary optimality condition)** If the minimum lifetime over all nodes is maximized then the minimum lifetime of each path flow from the origin to the destination with positive flow has the same value as the other paths.

#### 4.1 The Bellman-Ford Algorithm [34]

The Bellman-Ford Algorithm will be able to compute shortest paths even if negative costs exist and thus find another route. If the negative cost is used in a path calculation a shortest path can not be found, and would result in a packet never reaching its destination.

If no negative weight loop exists the algorithm will find the shortest path. This is accomplished by repeatedly decreasing an estimated cost to vertex  $v$ , until the actual shortest path from  $s$  to  $v$  is found. The pseudo code is as follows:

**Bellman-Ford(G,w,s)**

for all  $v$  in  $V$

$v.d = \text{infinity}$

$v.p = \text{NIL}$

$s.d = 0$

for  $i$  to  $|V|-1$

```

for each edge (u,v) in E
    if v.d > u.d + c(u, v) then
        u.d <-- u.d+c(u,v)
        v.p <-- u
for each edge u,v in E[G] do
    if v.d > u.d + c(u,v)
then return FALSE
return TRUE

```

First, all of the vertices are initialized. Then for every vertex in the graph an edge is selected and determines if adding this edge is worth while. If so it adds it to the shortest path. Otherwise it moves on to the next edge. Finally, the algorithm checks to see if negative weights exist. If they do it will return false otherwise true.

#### 4.2 MAXIMUM RESIDUAL ENERGY PATH (MREP)

The idea behind MREP is to route packets through paths with maximum residual energy so that energy consumption in all paths will be balanced. For the purposes of calculating the maximum residual energy path we define the following:

Let  $P_i$  be the set of all paths from node  $i$  to the destination node  $d$ . For a path  $p \in P_i$ , we define the path length  $L_p$  as a vector of link costs  $c_{jk}$  where  $\text{link}(j,k)$  is in the path  $p$ .  $c_{jk}$  is the reciprocal of the residual energy at node  $j$  after the route will have been used by a packet i.e

$$c_{jk} = \frac{1}{\underline{E}_j - e_{jk}}$$

Where  $\underline{E}_j$  is the residual energy at node  $j$ . By using the lexicographical ordering i.e by comparing the largest elements first and so on, shortest path from each node  $i$  to the destination can be obtained using a slightly modified version of the distributed Bellman-Ford algorithm. The shortest path thus obtained is the path whose minimum residual energy is the largest among all paths.

Another modification can be expressed as the distance update equation at step  $n$  of the distributed bellman-ford algorithm given by



$$D_i^{(n+1)} = \min[\min_{j \in S_i} \{\max(c_{ij}, D_j^{(n)})\}, D_i^{(n)}]$$

Where  $D_i^{(n)}$  is the distance of the path from node  $i$  to the destination at step  $n$ . We can also use the conventional Bellman-Ford equation given by

$$D_i^{(n+1)} = \min[\min_{j \in S_i} (c_{ij} + D_j^{(n)}), D_i^{(n)}]$$

The following link cost reflects the residual capacity of the link in terms of number of packets that can be delivered with the remaining energy; this link cost is one over the residual capacity of the link and is given by

$$c_{ij} = \frac{e_{ij}}{E_i}$$

#### 4.3 MINIMUM TRANSMITTED ENERGY (MTE)

##### FLOW AUGMENTATION (FA) ALGORITHMS [35]

Flow augmentation algorithms uses shortest cost path; we will use FA to describe and define MTE algorithm. Let each node  $i$  have the initial battery energy  $E_i$ , and let  $Q_i^{(c)}$  be the rate at which information is generated at node  $i$  belonging to commodity  $c \in C$ , where  $C$  is the set of all commodities. Assume that the transmission energy required for node  $i$  to transmit an information unit to its neighboring node  $j$  is  $e_{ij}$ , and the rate at which information of commodity  $c$  is transmitted from node  $i$  to node  $j$  is called the flow  $q_{ij}^{(c)}$ . Further, let  $Q_i$  and  $q_{ij}$  be the aggregate flows of all commodities, i.e.,

$$Q_i = \sum_{c \in C} Q_i^{(c)},$$

And

$$q_{ij} = \sum_{c \in C} q_{ij}^{(c)}$$

For each commodity  $c$ , we have a set of origin nodes  $O(c)$  where the information is generated, i.e.,

$$O^{(c)} = \{i \mid Q_i^{(c)} > 0, i \in N\}$$

and a set of destination nodes  $D^{(c)}$  among which any node can be reached in order for the information transfer of commodity  $c$  be considered done.

At each iteration, each origin node  $o \in O^{(c)}$  of commodity  $c$  calculates the shortest cost path to its destination nodes in  $D^{(c)}$ . Then the flow is augmented by an amount of  $\lambda Q^{(c)}$ , on the shortest cost path, where  $\lambda$  is the augmentation step size. After the flow augmentation, the shortest cost paths are recalculated and the procedures are repeated until any node  $i \in N$  runs out of its initial total energy  $E_i$ . The algorithm, should obtain the flow which will be used at each node to properly split incoming traffic. There are three parameters to consider in calculating the link cost  $c_{ij}$  for link  $(i, j)$ . One is the energy expenditure for unit flow transmission over the link,  $e_{ij}$ , the second is the initial energy  $E_i$ , and the third is the residual energy at the transmitting node  $i$  which is denoted by  $\underline{E}_i$ . A good candidate for the flow augmenting path should consume less energy and should avoid nodes with small residual energy since we would like to maximize the minimum lifetime of all nodes.

Obviously, both of these can't be optimized at the same time, which means there is a tradeoff between the two. In the beginning when all the nodes have plenty of energy, the minimum total consumed energy path is better off, whereas towards the end avoiding the small residual energy node becomes more important. Therefore, the link cost function should be such that when the nodes have plenty of residual energy, the energy expenditure term is emphasized, while if the residual energy of a node becomes small the residual energy term should be more emphasized.

With the above in mind, the link cost  $c_{ij}$  is proposed to be

$$c_{ij} = e_{ij}^{x_1} \underline{E}_i^{-x_2} E_i^{x_3}$$

where  $x_1$ ,  $x_2$ , and  $x_3$  are nonnegative weighting factors for each item. Note that if  $\{x_1, x_2, x_3\} = \{0, 0, 0\}$  then the shortest cost path is the minimum hop path, and if it is  $\{1, 0, 0\}$  then the shortest cost path is the minimum transmitted energy path. If  $x_2 = x_3$  then normalized residual energy is used, while if  $x_3 = 0$  then the absolute residual energy is used. The path cost is computed by the summation of the link costs on the path, and the algorithm can be implemented with any existing shortest path algorithms including the distributed Bellman-Ford algorithm.

MTE uses the distance update step of the bellman-Ford algorithm with the link cost given by

$$c_{ij} = e_{ij}.$$

## **5 DESIGN AND IMPLEMENTATION**

### **5.1 ASSUMPTIONS**

- Nodes are randomly distributed in a defined region.
- Each node has a uniform initial energy level.
- Energy used in sensing, processing, idle periods is not a bottleneck and thus constant.
- Nodes route information towards specified gateway nodes.
- Mobile targets are randomly generated and move randomly across the plane of the defined region.

### **5.2 DESIGN DESCRIPTION**

The network is deployed on a plane which represents an area under surveillance by the sensor network.

The nodes are deployed in a random manner on the plane with each deployment resulting in each distinct network topology.

There is a designated upstream zone where packets are directed to, in this case the right side of the plane. The communication is thus directed from left to right and the nodes in the upstream zone are presumed to be in direct contact with the data collector.

Moving targets are randomly generated and moves on the plane. A packet is generated whenever the target trips a sensor node by passing within the nodes node coverage.

Each node is depleted by sending/receiving packets and by detecting vectors. The nodes will eventually power down and drop out of network, the time of the first node dropping out is the lifetime measure of the network.

### **5.3 INPUT VARIABLES**

- Network size – The number of nodes in the network.
- Node coverage – Size of an area in which a sensor can detect movement.
- Sensing duration – how long a tripped sensor waits before sending a subsequent event.
- Sensor cost – Energy consumed by sensor activation.
- Transmission range – The maximum distance between two connected nodes.
- Transmission duration – The amount of time required to send a packet.

- Cost transmitting – The energy consumed by sending a packet.
- Cost receiving – The energy consumed by receiving a packet

#### **5.4 NODES AND TARGET DEPLOYMENT ALGORITHM**

Deployment of sensor nodes is random which means network topology will be different with each simulation. Some research has been done on the deployment algorithms in wireless sensor network. Morteza Maleki and Massoud Pedram in their paper Quality of monitoring and lifetime-constrained random deployment of sensor networks for minimum energy consumption [36] concluded that in practice it is usually infeasible to devise a deployment strategy whereby each sensor is placed precisely at some location. Practical deployment in large sensor networks is usually random, or at best, can be controlled with coarse granularity. As a result, adopting a random deployment model with slowly varying node densities is more realistic. Jiming Chen et al [37] studied various sensor deployment algorithms and classified them as random deployment, incremental deployment and movement-assisted deployment. They concluded that random deployment is the most practical way in placing the sensor nodes. When the target region is subject to severe change in condition or no priori knowledge is available, random deployment is often desirable to achieve a relatively satisfactory coverage. Random deployment is also practical in military application where wireless sensor networks are initially established by dropping or throwing.



## 5.6 DEVELOPMENT TOOLS

The development tool for the simulator is visual c# 2008 express edition and Microsoft .NET framework.

C# is Microsoft's latest object-oriented programming language developed for .NET platform and .NET is Microsoft's latest platform technology for creating web services. C# is a C++ based language and was developed to provide portability for distributed applications over network and internet. Application development in .NET platform can be done in multiple languages including C#, C++, and Visual Basic. Programs developed in all of these languages are compiled to Microsoft's Intermediate Language (IL) and executed within Common Language Runtime (CLR).

.NET is not a programming language; it's a virtual-machine technology (similar to Java virtual machine technology) with a framework that provides capability to run a variety of web applications. The .NET framework class library provides a set of classes that provide essential functionality for applications build within the .NET environment. Web functionality, XML support, database support, threading and distributed computing support is provided by the .NET framework class library. All .NET code is translated to Microsoft Intermediate Language (MSIL) and run within CLR; CLR is similar to Java Virtual Machine (JVM). The IL code is language-independent and similar to the Java byte code. A single .NET application may consist of several different languages. Two very important features of CLR are language interoperability and language independence.

The C# language was built with the observation of many languages, but mostly Java and C++. C# boasts type-safety, garbage collection, simplified type declarations, versioning among other features that make developing solutions faster and easier, especially for COM+ and Web services.

### 5.6.1 ADVANTAGES OF C# OVER JAVA, C AND C++

**MORE PRIMITIVE DATA TYPES:** Java has quite a few primitive data types: byte, char, int, long, floats, double. This is not the case in C#. C# uses the .NET object/type system so that C# programs can communicate with other .NET languages without having type confusion. This enables the primitive, or simple, types in C# function just like any other object.

**STATEMENT COMPARISON:** Statements in C# and Java are very similar, since both languages descend primarily from C and C++. The difference between keywords “import” in Java and “using” in C# is that Java has a packages concept, while C# uses namespaces similar to those of C++. The keyword “using” makes all names in the given namespace accessible to a program.

**CONDITIONAL STATEMENTS:** C# has both of the structures "if-then-else" and "switch". Both are similar except for one difference in the C# "switch" statement syntax. Java allows for control flow to implicitly fall between different cases in the switch statement, whereas the C# compiler explicitly does not allow this, the C# compiler will mark this as a syntactical error.

**DEFINITION OF CLASS:** Definition of Class and inheritance in C# is similar to Java except that in C# classes are inherited from the System.Object class, just as all classes in Java are inherited from the java.lang.Object class. Defining a class which is inherited from another class would be written using “:” instead of the “extends” keyword.

**INDEXERS:** A class's indexer lets you access any instance of that class as if it were an array. A class may define multiple indexers, each of which differs by the number and type of its arguments. Indexers are very similar to properties, especially in the syntax for defining them.

**STRUCTS:** A struct is similar to struct in C++, except that a C# struct can have any kind of class member, including constructors and methods; and the default accessibility for struct members in C# is private, rather than public as in C++. Similar to C++ structs, C# structs always copy by value and are therefore both mutable and exempt from dynamic memory management (but, memory link).

**ENUMS:** Another class member type Java does not provide is the enum. While similar to enums in C++, C# enums are based on an "underlying type," which can be any signed or unsigned integer type. Enumerations are derived from the built-in class System.Enum, and therefore every enum inherits all of that class's members.



**PROPERTIES:** The concept of using “getter” methods to encapsulate internal object properties is a design pattern that spans object-oriented languages. It isn't limited to C# or Java. C# has taken the concept of properties a step further by actually building “getter” methods into the language semantics. An object property has a type, a set method, and a get method. The set and get methods of the property determine how the property's value is set and retrieved.

**DELEGATES:** Delegates are C#'s answer to C++ function pointers; except that delegates are safe, and function pointers are dangerous. Both Java and C# have removed the function pointers, finding safer ways to maintain references to behavior that is determined at runtime.

**EVENTS AND EVENT NOTIFICATION:** C# events operate very much like Java events, except that C#'s are integrated into the language. For a class to receive an event, it must have a field or property that is a delegate marked with the event keyword. From inside the class, this delegate member is just like any other delegate; it can be called, checked to see if it's null, and so forth. From outside the class, there are only two things you can do with it; add or remove a method to the delegate, using the operator+= or the operator-=, respectively.

**OPERATOR OVERLOADING:** C# permits operator overloading, using syntax almost identical to that of C++. Some operators that can be overloaded in C++, such as operator=, cannot be overloaded in C#. Java's creators decided to leave operator overloading out of Java.

**METHODS:** C# methods have some features that Java does not. In particular, C# provides several modifiers on method parameters and has keywords for virtual methods and method overriding. As in C++, C# method parameters are value parameters. The ref modifier on a method parameter makes that parameter a reference. The out modifier indicates that the parameter is an output parameter, which is identical to a reference parameter, except that the parameter must be assigned before the method's returns statement. C# methods are, by default, no virtual but can be made virtual by explicit use of the keyword virtual.

## 6 SIMULATION, RESULTS AND CONCLUSIONS

The simulator has a simulation area of 1017 by 514. The following parameters are preset, are same for all simulations and cannot be changed simulation.

- a) Initial energy            1000.
- b) Number of targets        4.

The following are the inputs that can be varied for any simulation, the values indicated are the preset or default values:

- a. Transmission cost            200
- b. Network size                 50
- c. Sensor delay                 15
- d. Transmission delay         10
- e. Sensor cost                 20
- f. Transmission radius         300
- g. Node coverage               45
- h. Cost receiving                15

The platform used for simulation is a HP 550 laptop with Intel processor core 2 duo 1.8 MHz and 2GB RAM on a Microsoft windows XP professional operating system.

The simulator interface is divided into several functional areas:

- Routing Parameters:** Includes a 'Routing Method' dropdown menu with 'MREP Algorithm' selected and 'MTE Algorithm' as an alternative option.
- Network Setup:** A row of eight sliders for configuring network parameters: 'No. of nodes' (50), 'Node Coverage' (45), 'Sensing Duration' (15), 'Sensor Cost' (20), 'Transmission Rang' (300), 'Transmit duration' (10), 'Cost Transmitting' (200), and 'Cost Receiving' (15).
- Simulation Control:** A central 'START/STOP Simulation' section with a 'Start Simulation' button, and a 'NEW Simulation' button to the right.
- Buttons:** A 'Create WSN' button is located below the sliders, and an 'EXIT SIMULATION' button is on the far right.
- LIFETIME STATISTICS:** A panel on the bottom left showing '1st dead sensor at: 00:00:000' and a field for 'Average lifetime'.
- SIMULATION PROGRESS STATUS:** A panel on the bottom right displaying 'Status: Ready', 'Time: 00:00:000', 'Power: 0', 'Sensors: 0', and 'Live Packets: 0'.

Figure 6. 1 : Simulator at a glance.

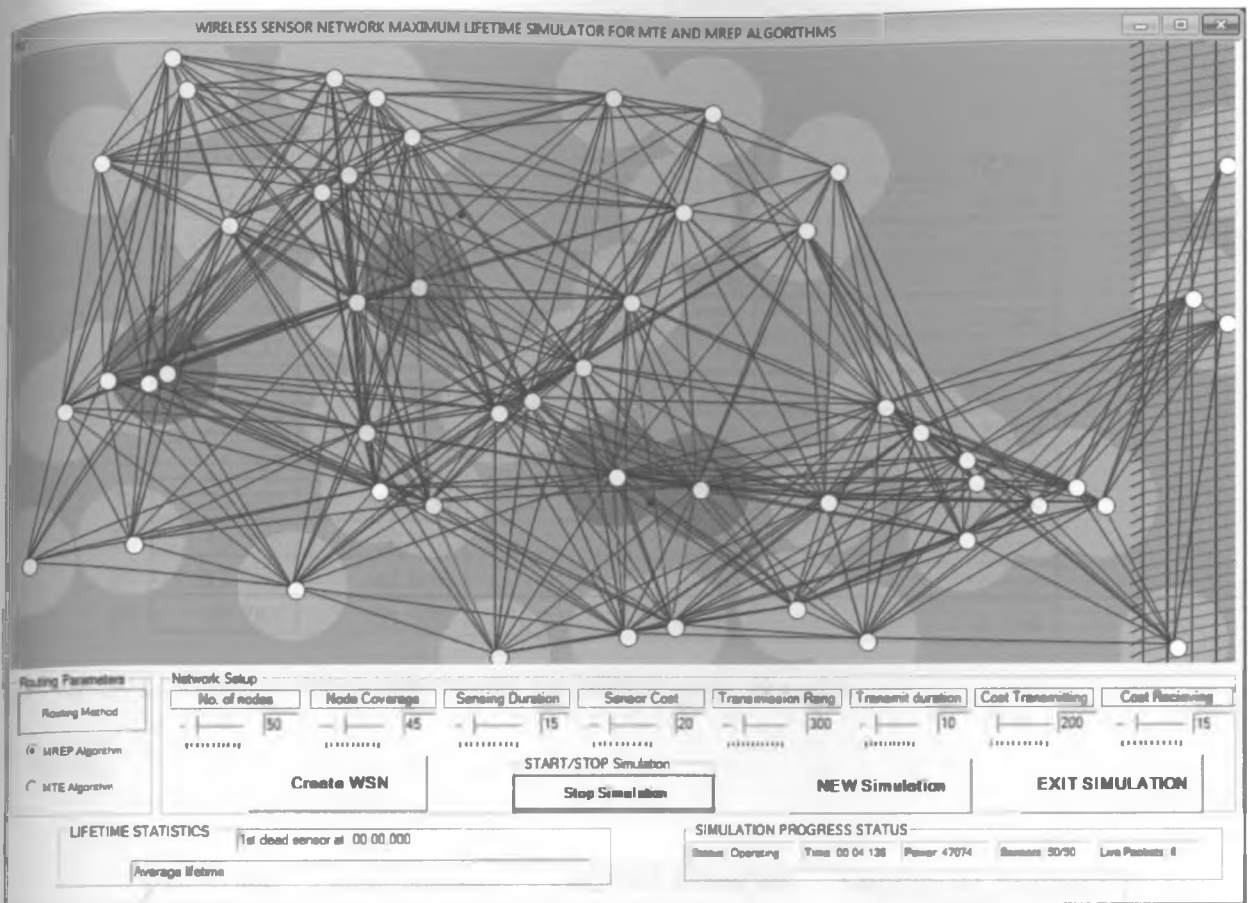


Figure 6. 2: A view of simulation in progress

**6.1 SIMULATION WITH 50 SENSOR NODES**

**MREP**

	LIFETIME	LIFETIME x 2	ACTIVE SENSORS AT LIFETIME x 2	RESIDUAL ENERGY AT LIFETIME x 2
1	23.781	47.562	17	9,446
2	20.671	41.342	15	7,535
3	22.593	45.186	21	9,952
4	22.140	44.280	14	8,331
5	18.862	37.724	26	10,876
6	24.171	48.342	13	8,709
7	22.281	44.562	18	11,162
8	21.984	43.968	26	13,472
9	22.140	44.280	12	4,706

10	19.609	39.218	25	12,928
11	20.078	40.156	16	8,295
12	21.953	43.906	16	7,476
13	23.156	46.312	18	8,152
14	21.390	42.780	23	12,252
15	23.437	46.874	13	4,702
16	20.859	41.718	27	15,430
17	25.937	51.874	16	7,629
18	23.390	46.780	18	8,000
19	24.250	48.500	18	9,175
20	19.359	38.718	23	9,512
21	21.968	43.936	16	10,120
22	25.781	51.562	20	13,154

Table 6. 1 : Table of simulations results for MREP using 50 nodes

**MTE**

	LIFETIME	LIFETIME x 2	ACTIVE SENSORS AT LIFETIME x 2	RESIDUAL ENERGY AT LIFETIME x 2
1	19.328	38.656	38	23314
2	15.234	30.468	32	21358
3	15.140	30.280	35	22876
4	17.187	34.374	25	12566
5	16.152	32.304	30	14782
6	14.421	28.842	39	25297
7	15.097	30.194	39	21645
8	15.734	31.468	40	25469
9	17.156	34.312	42	25320
10	17.406	34.812	36	22676
11	18.515	37.030	32	22369
12	16.578	33.156	36	21778
13	15.734	31.468	41	29152
14	15.390	30.780	38	19237
15	20.406	40.812	35	20482
16	17.812	35.624	37	21758
17	19.562	39.124	31	20009
18	16.015	32.030	39	25060
19	17.140	34.280	37	25297
20	17.718	35.436	32	20503
21	17.875	35.750	30	18275
22	18.453	36.906	32	19214

Table 6. 2 : Table of simulation results for MTE using 50 nodes

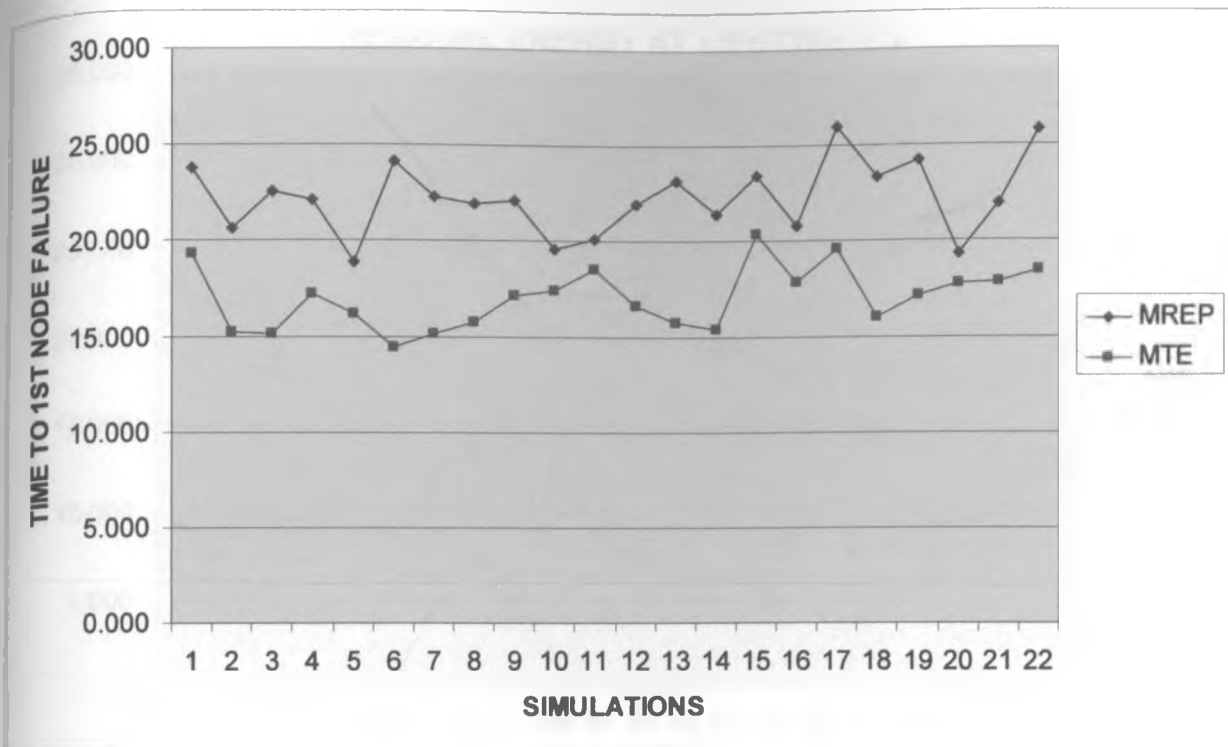


Figure 6. 3: Graph of lifetime with 50 nodes

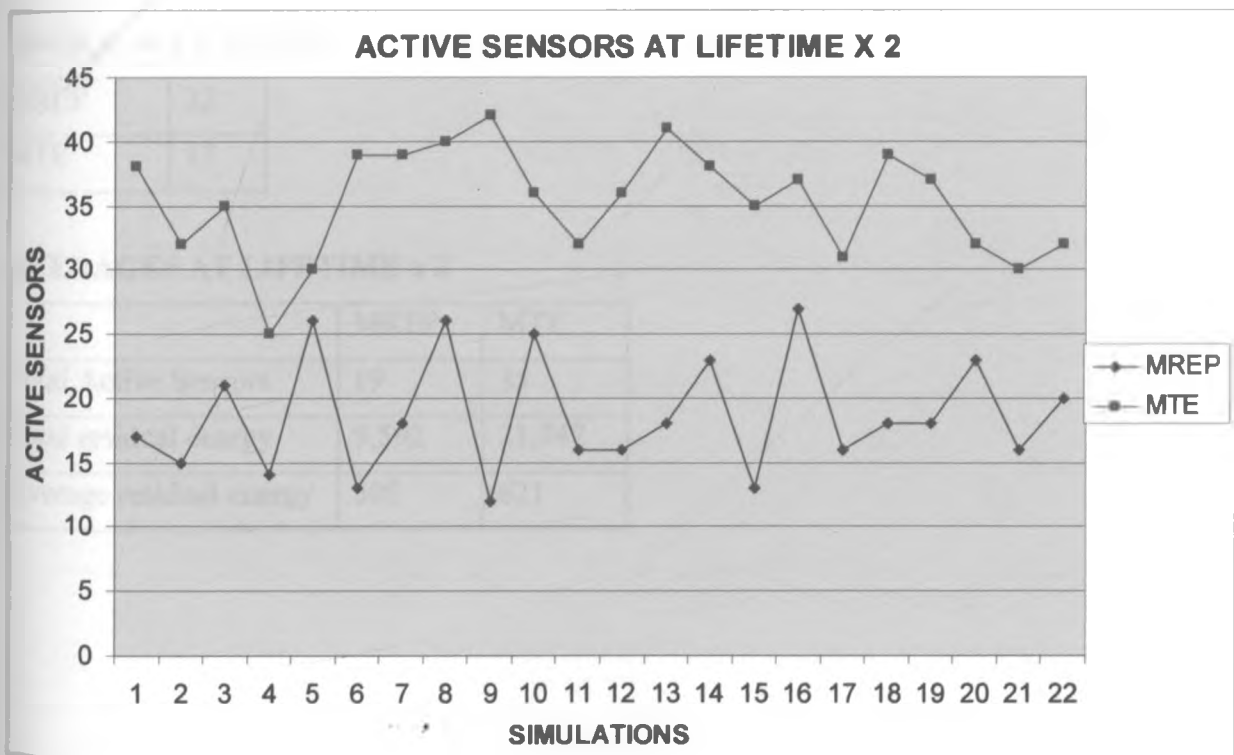


Figure 6. 4: Graph of live sensors at Lifetime x 2 with 50 nodes

### RESIDUAL ENERGY AT LIFETIME X 2

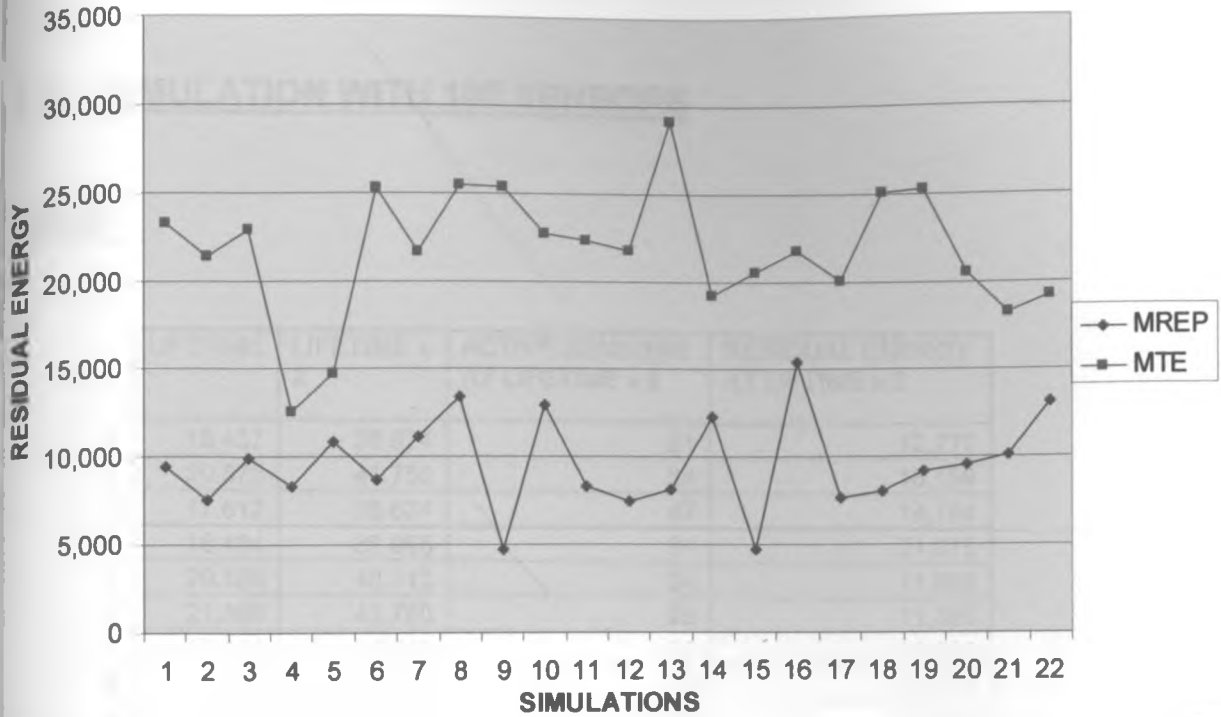


Figure 6. 5 : Graph of residual energy at lifetime x 2 with 50 nodes

### AVERAGE LIFETIME

MREP	22
MTE	17

### AVERAGES AT LIFETIME x 2

	MREP	MTE
Total Active Sensors	19	35
Total residual energy	9,592	21,747
Average residual energy	505	621

## 6.2 SIMULATION WITH 100 SENSORS

### MREP

	LIFETIME	LIFETIME x 2	ACTIVE SENSORS AT LIFETIME x 2	RESIDUAL ENERGY AT LIFETIME x 2
1	19.437	38.874	21	12,270
2	20.875	41.750	24	16,159
3	17.812	35.624	47	14,164
4	18.484	36.968	34	21,875
5	20.156	40.312	24	11,659
6	21.390	42.780	25	11,395
7	17.421	34.842	35	16,555
8	21.937	43.874	19	9,218
9	17.390	34.780	38	21,191
10	21.687	43.374	33	13,107
11	17.203	34.406	28	17,286
12	19.828	39.656	22	12,647
13	22.359	44.718	23	13,453
14	15.718	31.436	43	20,345
15	23.703	47.406	49	22,258
16	19.125	38.250	30	14,109
17	19.187	38.374	27	16,894
18	20.812	41.624	32	19,229
19	20.828	41.656	32	16,760
20	16.562	33.124	35	19,701
21	19.140	38.280	27	14,431

Table 6. 3 : Simulation results for MREP with 100 nodes.

### MTE

	LIFETIME	LIFETIME x 2	ACTIVE SENSORS AT LIFETIME x 2	RESIDUAL ENERGY AT LIFETIME x 2
1	15.750	31.500	78	49914
2	15.406	30.812	73	49013
3	14.750	29.500	88	54266
4	16.937	33.874	75	44278
5	16.796	33.592	73	42273
6	14.312	28.624	85	52144
7	16.875	33.750	71	43419
8	15.343	30.686	70	40375



9	15.062	30.124	71	45473
10	14.328	28.656	82	52170
11	12.375	24.750	85	53893
12	14.906	29.812	82	47925
13	16.718	33.436	79	50230
14	14.390	28.780	82	51702
15	15.593	31.186	71	45306
16	13.125	26.250	76	52386
17	18.421	36.842	69	41974
18	17.109	34.218	79	40081
19	14.609	29.218	76	46169
20	14.953	29.906	81	52592
21	17.593	35.186	86	50113

Table 6. 4 : Simulation results for MTE with 100 nodes.

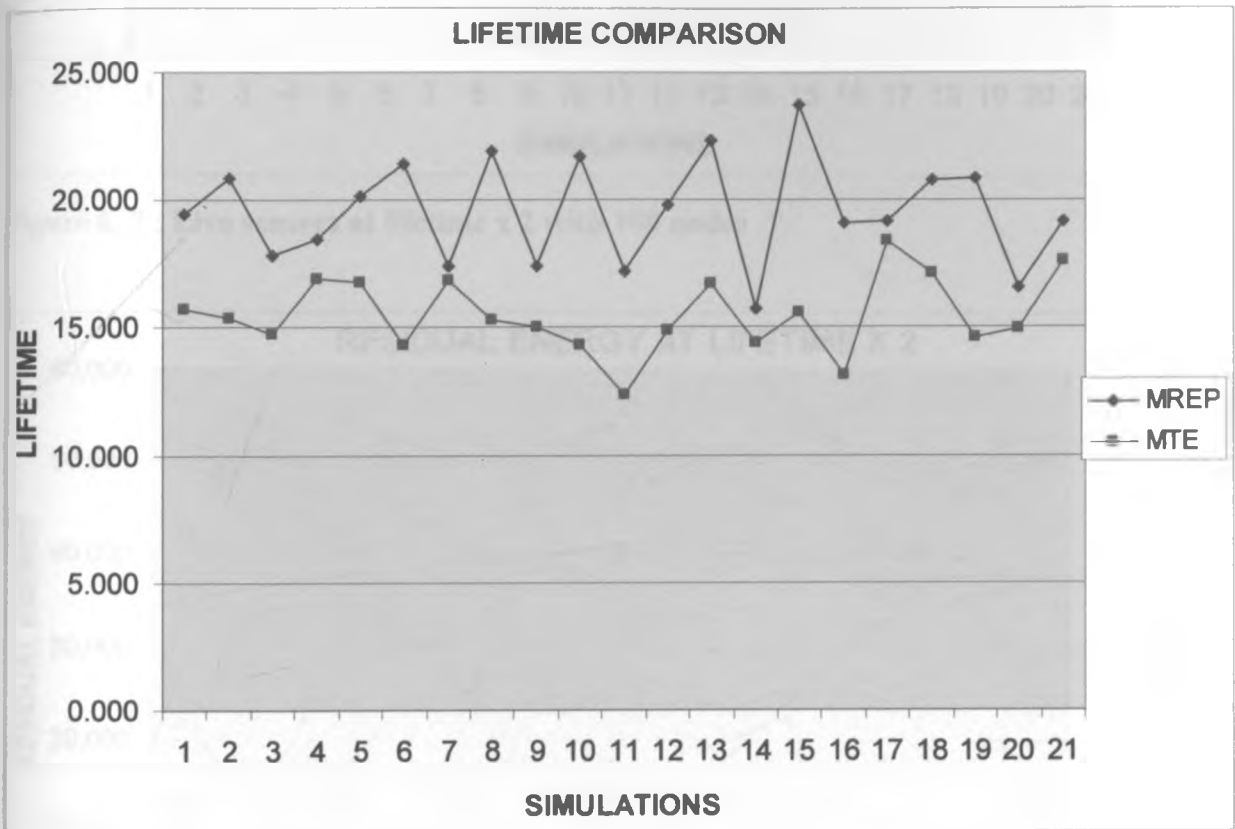


Figure 6. 6 : Graph of lifetime with 100 nodes

ACTIVE SENSORS AT LIFETIME X 2

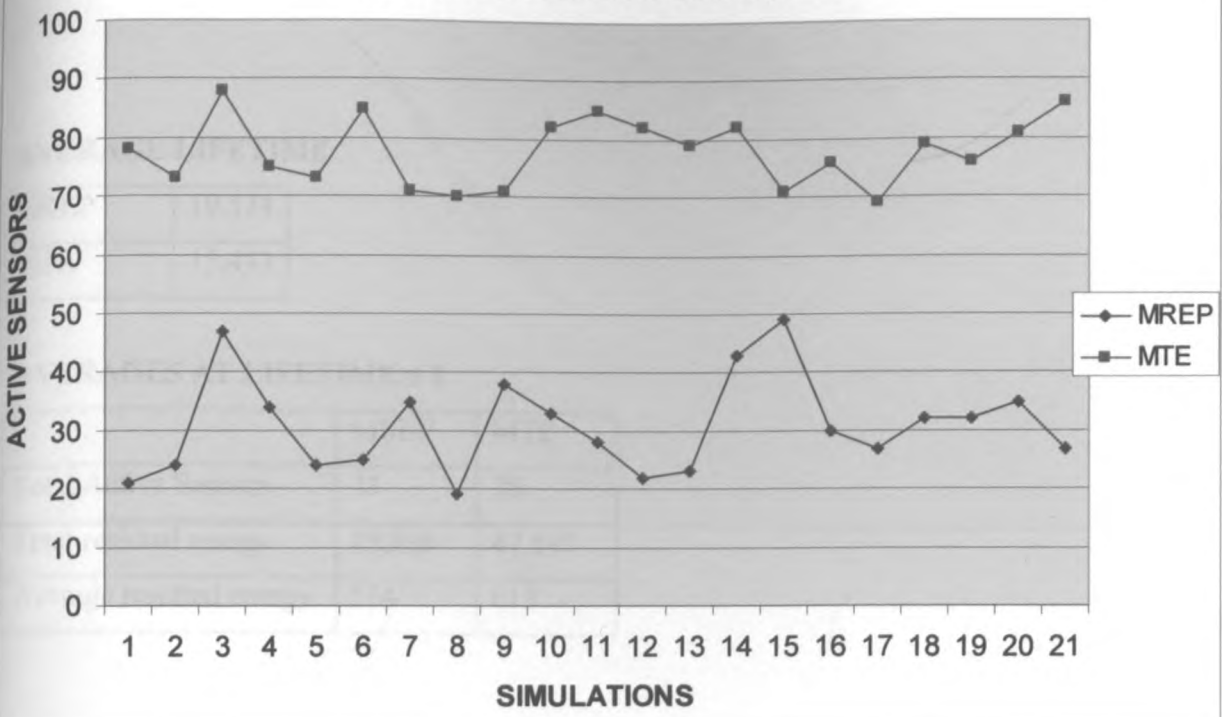


Figure 6. 7 : Live sensors at lifetime x 2 with 100 nodes

RESIDUAL ENERGY AT LIFETIME X 2

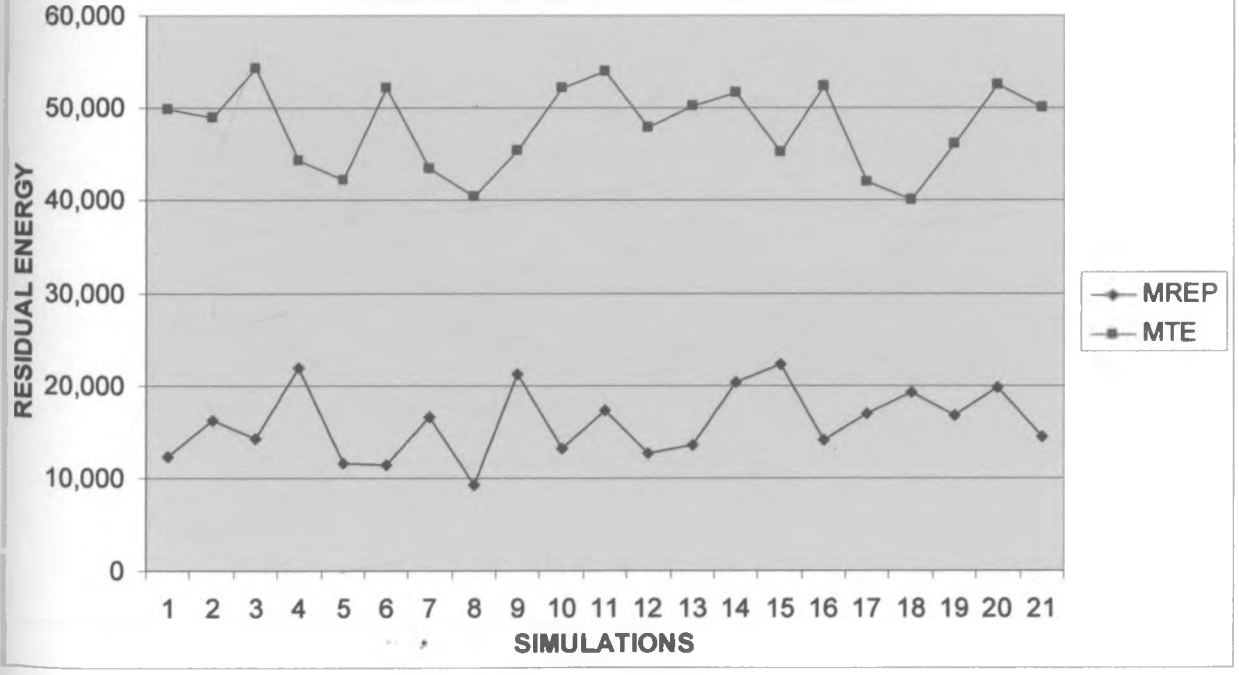


Figure 6. 8 : Graph of residual energy at lifetime x 2 with 100 nodes

**AVERAGE LIFETIME**

MREP	19.574
MTE	15.493

**AVERAGES AT LIFETIME x 2**

	MREP	MTE
Total Active Sensors	31	78
Total residual energy	15,938	47,890
Average residual energy	514	613

### 6.3 SIMULATION WITH VARIABLE NUMBER OF SENSORS

The simulator can have a minimum of 10 sensor nodes and a maximum of 410 sensor nodes. Instead of fixed number of nodes, I going to use variable number of nodes i.e. each simulation will have a different number of nodes for the MREP and MTE algorithm pair. It will also be important to take note of the node density as it will be an additional variant in the performance of the algorithm.

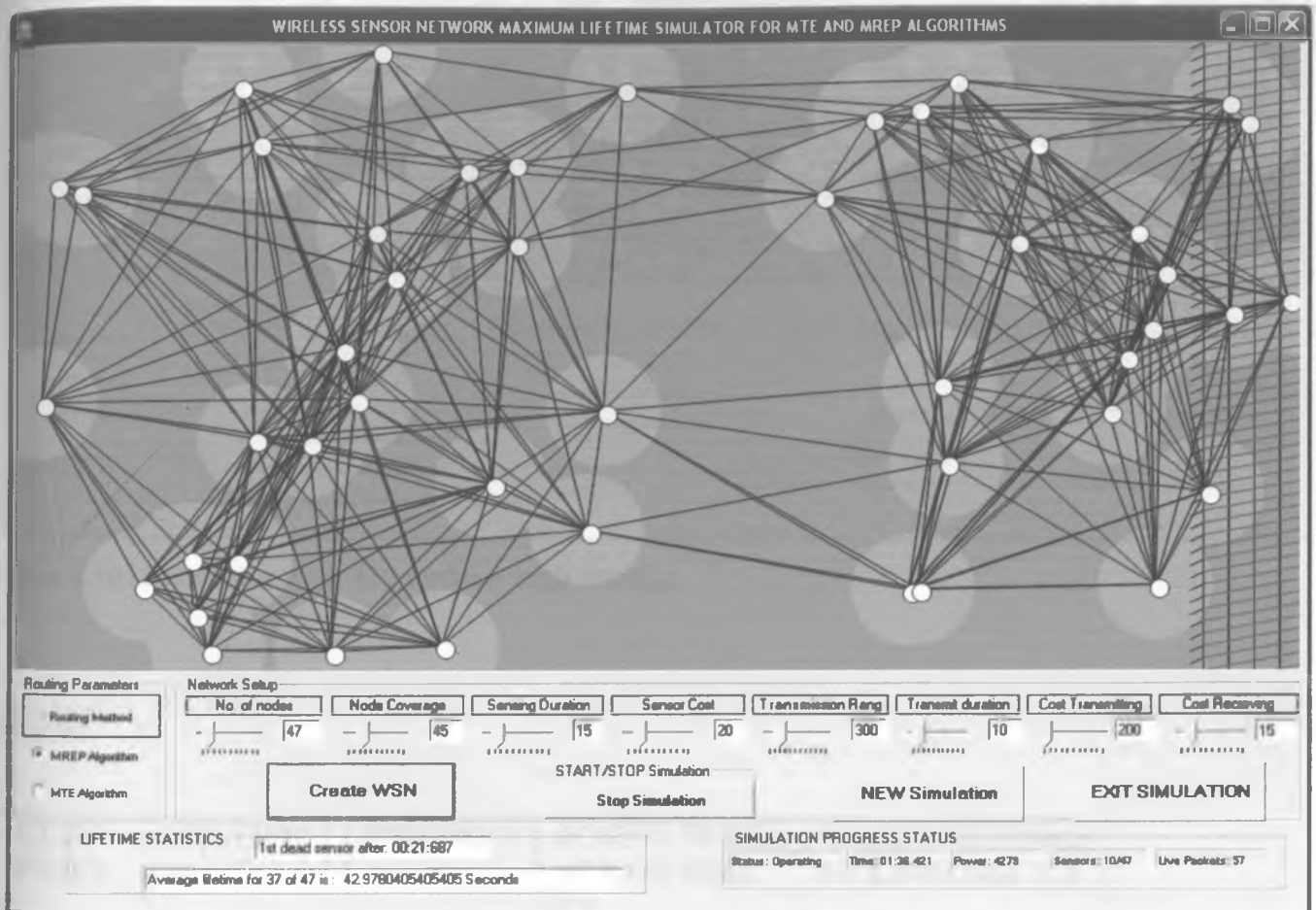


Figure 6.9 : Simulator with 47 nodes – Low density

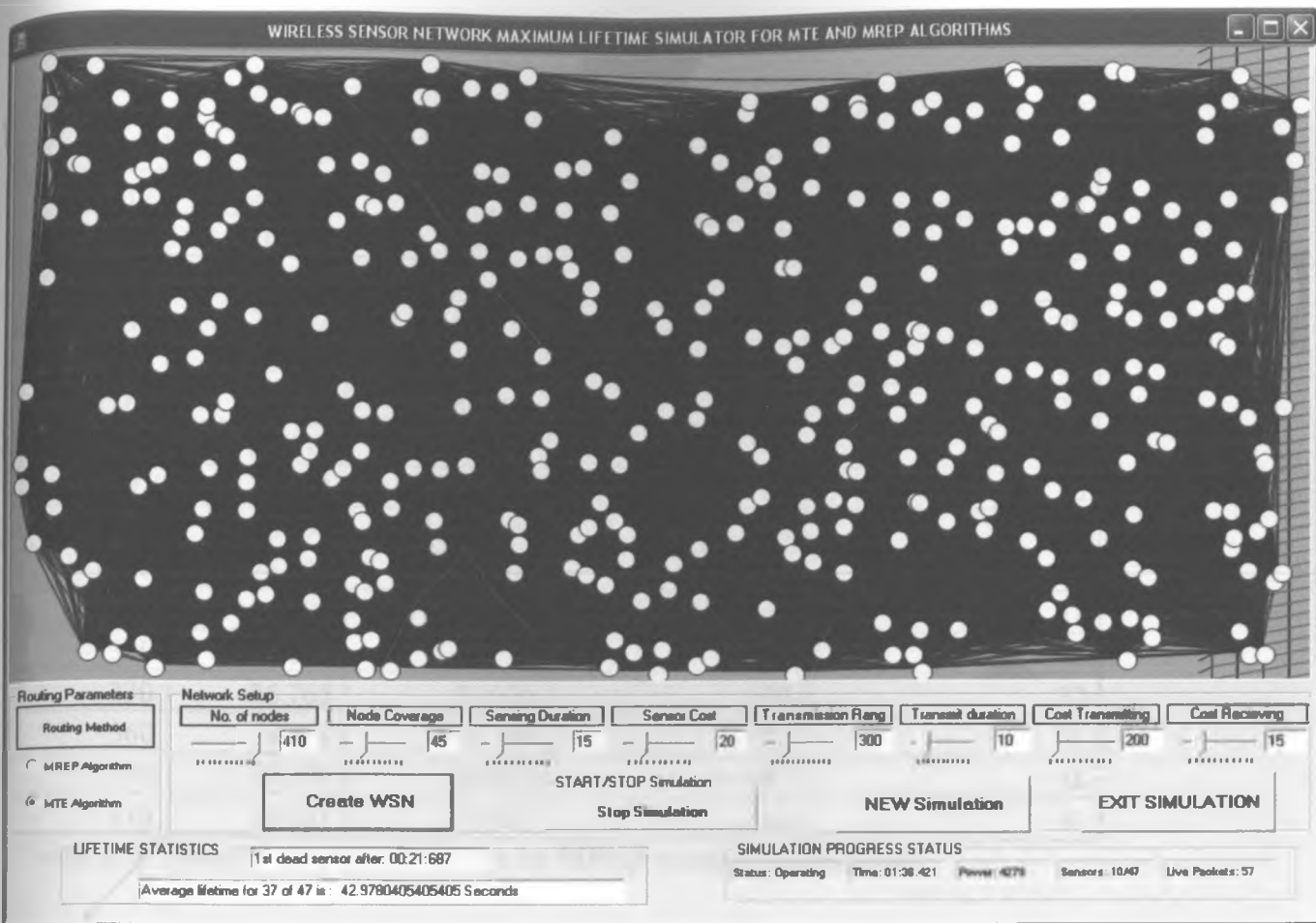


Figure 6. 10 : Simulator with 410 nodes – high density.

**MREP**

NO. OF NODES	LIFETIME	LIFETIME * 2	ACTIVE NODES AT LIFETIME X 2	% ACTIVE NODES AT LIFETIME X 2
47	29.750	59.500	21	45
52	28.968	57.936	18	35
58	27.656	55.312	23	40
65	31.171	62.342	17	26
71	30.671	61.342	23	32
87	28.968	57.936	20	23
95	27.468	54.936	30	32
99	29.875	59.750	28	28

105	30.593	61.186	18	17
112	28.656	57.312	32	29
120	26.062	52.124	31	26
132	25.562	51.124	38	29
145	28.812	57.624	29	20
152	30.000	60.000	28	18
165	26.515	53.030	33	20
180	25.171	50.342	43	24
192	25.484	50.968	50	26
220	26.390	52.780	50	23
250	26.500	53.000	70	28
280	28.093	56.186	59	21
300	24.875	49.750	55	18
330	26.765	53.530	104	32
360	32.578	65.156	113	31
390	29.500	59.000	187	48
410	36.031	72.062	80	20

**Table 6. 5: Table of simulation results for MREP using variable nodes.**

**MTE**

<b>NO. OF NODES</b>	<b>LIFETIME</b>	<b>LIFETIME * 2</b>	<b>ACTIVE NODES AT LIFETIME X 2</b>	<b>% ACTIVE NODES AT LIFETIME X 2</b>
47	18.750	47.500	35	74
52	17.000	34.000	32	62
58	17.234	34.468	41	71
65	15.390	30.780	47	72
71	15.734	31.468	59	83
87	13.734	27.468	66	76
95	13.437	26.874	78	82
99	13.515	27.030	81	82
105	15.000	30.000	70	67
112	16.031	32.062	86	77
120	15.687	31.374	83	69
132	16.140	32.280	89	67
145	15.453	30.906	115	79
152	14.859	29.718	132	87

165	15.421	30.842	131	79
180	18.687	37.374	124	69
192	17.781	35.562	144	75
220	20.625	41.250	177	80
250	21.140	42.280	180	72
280	17.265	34.530	243	87
300	24.625	49.250	248	83
330	23.750	47.500	278	84
360	28.593	57.186	280	78
390	22.750	45.500	355	91
410	23.125	46.250	359	88

Table 6. 6 : Table of simulation results for MTE using variable nodes.

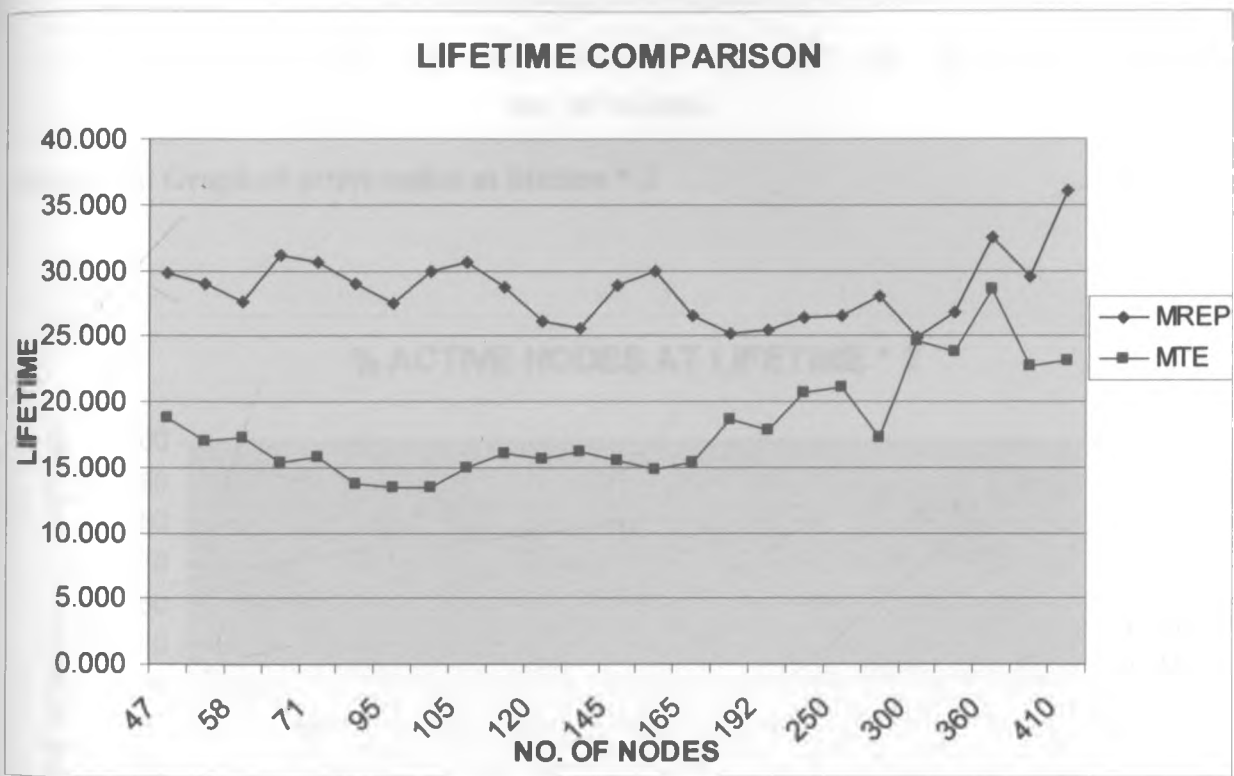


Figure 6. 11 : Graph of lifetime with variable nodes.

### ACTIVE NODES AT LIFETIME \* 2

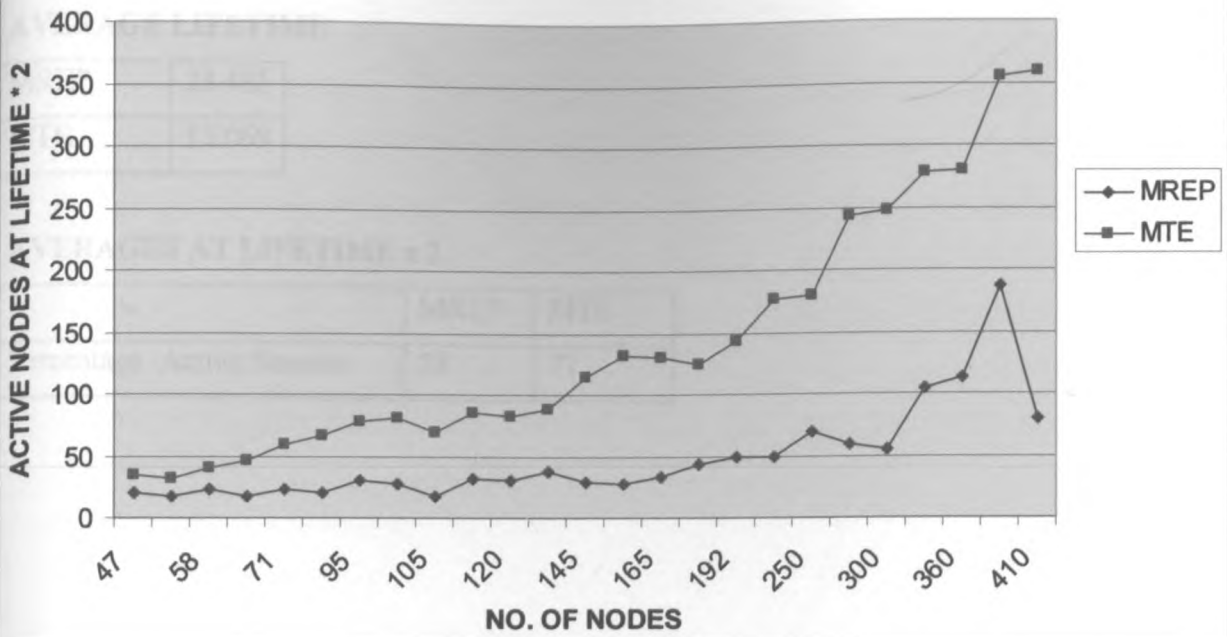


Figure 6. 12: Graph of active nodes at lifetime \* 2

### % ACTIVE NODES AT LIFETIME \* 2

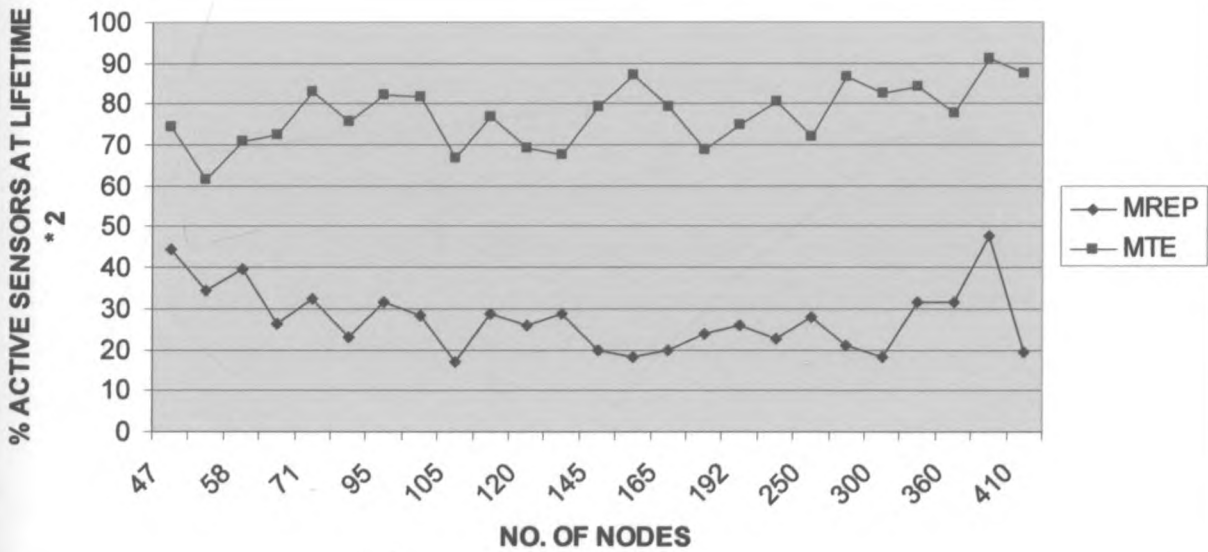


Figure 6. 13: Graph of % active nodes at lifetime \* 2



**AVERAGE LIFETIME**

MREP	28.485
MTE	18.069

**AVERAGES AT LIFETIME x 2**

	MREP	MTE
Percentage Active Sensors	28	77

## **6.4 DISCUSSIONS AND CONCLUSIONS**

For the simulation done using 50 sensor nodes, the lifetime of MREP algorithms is 29% longer than those of MTE algorithms. If we take a time which is double the lifetime of both MREP and MTE algorithms then the number of active sensors for MTE algorithms is more than double the number of active sensors for MREP algorithms; for those active sensors the average residual energy for MTE is 22.9% more than that of MREP. But it should be pointed out that at the time which is double the lifetime; MREP algorithm would have run 32% longer than MTE algorithm, so if taken at the same point in time for both MREP and MTE the average residual energy difference may be insignificant. Given any time which is double the lifetime MTE will have 84% more active sensors than MREP.

For the simulation done using 100 sensor nodes, the lifetime of MREP algorithms is 26.3% longer than those of MTE algorithms. If we take a time which is double the lifetime of both MREP and MTE algorithms then the number of active sensors for MTE algorithms is more than double the number of active sensors for MREP algorithms; for those active sensors the average residual energy for MTE is 19.26% more than that of MREP. But it should be pointed out that at the time which is double the lifetime MREP algorithm would have run 26.3% longer than MTE algorithm, so if taken at the same point in time for both MREP and MTE the average residual energy difference may be insignificant. Given any time which is double the lifetime MTE will have 151% more active sensors than MREP. The low percentages posted by MREP in simulations using 100 nodes compared to 50 nodes is due to the fact that when nodes are many the possibility of a target at a single location being detected by more than one node is very high since node's sensing radius will overlap. There will be many duplicate packets of the same information as a result. The percentages can be improved by reducing the node coverage radius and the transmission radius.

For simulations done with variable number of nodes, the lifetime of MREP algorithms is 57.6% longer than those of MTE algorithms. At a time which is double the lifetime of both MREP and MTE algorithms, MTE has more number of active sensors than MREP, on average MTE has 77% of the sensors still active while MREP only has 28% on average.

Based on the metric of the project, time until the first node runs out of energy, MREP performs better but the number of active nodes for MTE may still make the network useful especially when the purpose of the network can tolerate depletion of some nodes and if some of the gateway nodes are still active. Since almost in all sensor networks data are routed towards some sink (gateway) nodes, hops close to those nodes become heavily involved in packet forwarding and thus their batteries get depleted rather quickly. For us to gain more useful network lifetime from MTE then we have to augment the algorithms with some gateway nodes repositioning algorithms or/and nodes repositioning algorithms based on prevailing node coverage. Some work done by Kemal Akkaya et al [38] and Suganyavinodhini et al [39] have shown that such repositioning of the gateway increases the average lifetime of the nodes by decreasing the average energy consumed per packet and impacted positively on the network throughput.

From the results it can also be seen that the lifetime increases with the increase in sensor node density for both MREP and MTE algorithms, this shows that the average amount of energy required to report an event to sink nodes decreases when the wireless sensor node density increases i.e high node density impacts positively on the lifetime of a wireless sensor network. The same is also confirmed by the work done by Moez et al [40].

The simulation results show that MREP algorithms perform better than MTE algorithms in prolonging the lifetime of a sensor network. The results therefore show that it is more important to route the traffic such that the energy consumption is balanced among the nodes in proportion to their available energy instead of minimizing the absolute consumed power.

We can also conclude that if the sensor network is operable only while every sensor is alive then MREP will be the ideal algorithm, but if the operation of the network is able to tolerate the depletion of some sensors then MTE will be the ideal.

By augmenting MTE algorithms with some node repositioning algorithms we can achieve much higher lifetime performances considering that the percentage active sensors remains much higher compared to MREP algorithms at a given time twice the lifetime reported.

The results also show that the average amount of energy required to report an event to sink nodes decreases when the wireless sensor node density increases, such that higher lifetimes are achieved with very high density nodes for both MREP and MTE.

## **APPENDIX A: REFERENCES AND BIBLIOGRAPHY**

### **REFERENCES**

1. Culler, David E., Estrin, Deborah and Srivastava, Mani B, "Guest editors' introduction: overview of sensor networks," IEEE Computer, vol. 37, issue 8, pp. 41- 49, Aug. 2004.
2. V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. IEEE Signal Processing Magazine, 19(2):40–50, 2002.
3. Muhammad U. Ilyas and Hayder Radha, "Increasing Network Lifetime Of An IEEE 802.15.4 Wireless Sensor Network By Energy Efficient Routing," Department of Electrical & Computer Engineering, Michigan State University.
4. D. Vass, Z. Vincze, R. Vida and A. Vidács, "Energy Efficiency in Wireless Sensor Networks Using Mobile Base Station", Proc. of 11th Open European Summer School and IFIP WG6.6, WG6.4, WG6.9 Workshop (EUNICE 2005), Colmenarejo, Spain, 6-8 July, 2005.
5. Vass, A. Vidács, "Positioning Mobile Base Station to Prolong Wireless Sensor Network Lifetime", CoNEXT 2005 Student Workshop, pp. 300-301, Toulouse, France, 24-27 October, 2005.
6. Jun Luo, Jean-Pierre Hubaux, "Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks," In Proc. of INFOCOM 2005, March 2005.
7. Wei Ding and S. Sitharama Iyengar and Rajgopal Kannan and William Ruml, "Energy equivalence routing in wireless sensor networks," Elsevier Microprocessors and Microsystems 28 (2004) 467–475.
8. Chipara, O. Zhimin He Guoliang Xing Qin Chen Xiaorui Wang Chenyang Lu Stankovic, J. Abdelzaher, T., "Real-time Power-Aware Routing in Sensor Networks Quality of Service," 2006. IWQoS 2006. 14th IEEE International Workshop on, June 2006, 83-92.
9. D. Tian and N. Georganas. "Energy efficient routing with guaranteed delivery in wireless sensor networks". In Proceedings IEEE Wireless Communications and Networking Conference, vol. 3, pp. 1923–1929. 16–20 March 2003.
10. P. Floréen, P. Kaski, J. Kohonen, P. Orponen: Lifetime maximization for multicasting in energy-constrained wireless networks. IEEE J. on Selected Areas in Communications 23 (2005), 117-126.

11. Shah R., C., and Rabaey J., M., "Energy Aware Routing for Low Energy Ad Hoc Sensor Network", In Proc. of IEEE Wireless Communications and Networking conference (WCNC), Volume 1, pp. 17-21, March 2002.
12. André Schumacher, Pekka Orponen, Thorn Thaler and Harri Haanpää, "Lifetime Maximization in Wireless Sensor Networks by Distributed Binary Search," Wireless Sensor Networks, 5th European Conference, EWSN 2008, Bologna, Italy, January 30-February 1, 2008.
13. A. Alfieri, A. Bianco, P. Brandimarte and C. F. Chiasserini, "Maximizing system lifetime in wireless sensor network", European Journal of Operational Research, 181, 2007, pp. 390-402.
14. Taewook kang, jangkyu yun, hoseung lee, icksoo lee, hyunsook kim, byunghwa lee, byeongjik lee, kijun han, "A Clustering Method for Energy Efficient Routing in Wireless Sensor Networks," Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Corfu Island, Greece, February 16-19, 2007.
15. Y.-F. Huang, W.-H. Luo, J. Sum, L.-H. Chang, C.-W. hang and R.-C. Chen, "Lifetime Performance of an energy efficient clustering algorithm for cluster-based wireless sensor networks," LNCS 4743, pringer-Verlag Berlin Heidelberg, pp. 455---464, Aug. 2007.
16. Ioan Raicu, Loren Schwiebert, Scott Fowler and Sandeep K.S. Gupta, "e3D: An Energy-Efficient Routing Algorithm for Wireless Sensor Networks," IEEE ISSNIP 2004 (The International Conference on Intelligent Sensors, Sensor Networks and Information Processing), Melbourne, Australia, December 2004.
17. C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in Proceedings of IEEE Military Communications Conference on Communications for Network-Centric Operations: Creating the Information Force (MILCOM '01), vol. 1, pp. 357-361, McLean, Va, USA, October 2001.
18. Mehdi Kalantari, Mark Shayman, "Energy Efficient Routing in Wireless Sensor Networks," In Proc. of Conference on Information Sciences and Systems, Princeton University, March 2004.
19. D. Ranganathan, P. K. Pothuri, V. Sarangan, and S. Radhakrishnan, "Energy-efficient routing in wireless sensor networks for delay sensitive applications," Oklahoma State University, 2006.
20. Stojmenovic, I., Lin, X., "Power-Aware Localized Routing in Wireless Networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 11, November 2001, 1122-1133.

21. Qun Li and Javed Aslam and Daniela Rus. Distributed Energy-Conserving Routing Protocols for Sensor Network. In Proceedings of the 37th Hawaii International Conference on System Science, January, 2003.
22. Qun Li and Javed Aslam and Daniela Rus. Hierarchical Power-aware Routing in Sensor Networks. In Proceedings of the DIMACS Workshop on Pervasive Networking, May, 2001.
23. Vahid Shah-Mansouri and Vincent W.S. Wong, "Distributed Maximum Lifetime Routing in Wireless Sensor Networks Based on Regularization," in Proc. of IEEE Global Telecommunications Conference (Globecom), Washington, DC, November 2007.
24. Qunfeng Dong, "Maximizing System Lifetime in Wireless Sensor Networks," Fourth International Conference on Information Processing in Sensor Networks (IPSN '05), April 2005.
25. Vasu Jolly, Shahram Latifi and Naoto Kimura, "Energy-Efficient Routing in Wireless Sensor Networks Based on Data Reduction," The 2006 World Congress in Computer Science Computer Engineering, and Applied Computing Las Vegas, Nevada, USA (June 26-29, 2006).
26. K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks," in the Elsevier Ad Hoc Network Journal, Vol. 3/3 pp. 325-349, 2005.
27. N. NARASIMHA DATTA AND K. GOPINATH "A survey of routing algorithms for wireless sensor networks," Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India. May 5, 2006.
28. Chansu Yu, Ben Lee and Hee Yong Youn "Energy efficient routing protocols for mobile ad hoc networks," wireless communication mobile computing 2003; Volume 3: issue 8 pages 959–973 (doi: 10.1002/wcm.119).
29. Jamal N. Al-Karaki Ahmed E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey", IEEE Wireless Communications, Volume: 11, Issue: 6, 26- 28, Dec. 2004.
30. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and Fabio Silva, "Directed Diffusion for Wireless Sensor Networking," IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 2–16, February 2002.
31. Carlyle, W.M. and Wood, R.K., (2003), "Near-Shortest and K-Shortest Simple Paths," Networks, 46, pp. 98–109.
32. John Hershberger, Matthew Maxel, Subhash Suri: Finding the k shortest simple paths: A new algorithm and its implementation. ACM Transactions on Algorithms 3(4): (2007).

33. Yung-Fa Huang, Neng-Chung Wang, Ming-Che Chen: Performance of a Hierarchical Cluster-Based Wireless Sensor Network. 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing SUTC 2008: 349-354
34. Andrew C. Hoffman: Multiple Approaches for Distributed Routing Algorithms, May 6, 2004
35. J.-H. Chang and L. Tassiulas, "Routing for maximum system lifetime in wireless ad-hoc networks," in Proceedings of 37-th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, Sept. 1999.
36. Morteza Maleki and Massoud Pedram, "QoM and lifetime-constrained random deployment of sensor networks for minimum energy consumption", in the IEEE proceedings of Information processing in sensor networks, April 2005.
37. Jiming Chen, Entong Shen and Youxian Sun, "The deployment algorithms in wireless sensor networks: A survey", Information Technology Journal 8 (3): 293-301, 2009.
38. K. Akkaya, M. Younis, and M. Bangad, "Sink repositioning for enhanced performance in wireless sensor networks," *Computer Networks*, vol. 49, no. 4, pp. 512–534, 2005.
39. Suganyavinodhini.I , Ganadhipathy Tulsi's "MAXIMIZING WIRELESS SENSOR NETWORK'S LIFETIME THROUGH NODE REPOSITIONING" Proceedings of the National Conference on Emerging Trends in Computing Science "NCETCS 2011."
40. M. Esseghir and N. Bouabdallah, "Node density control for maximizing wireless sensor network lifetime," *Int'l Journal of Network Management*, vol. 18, no. 2, pp. 159–170, March 2008.

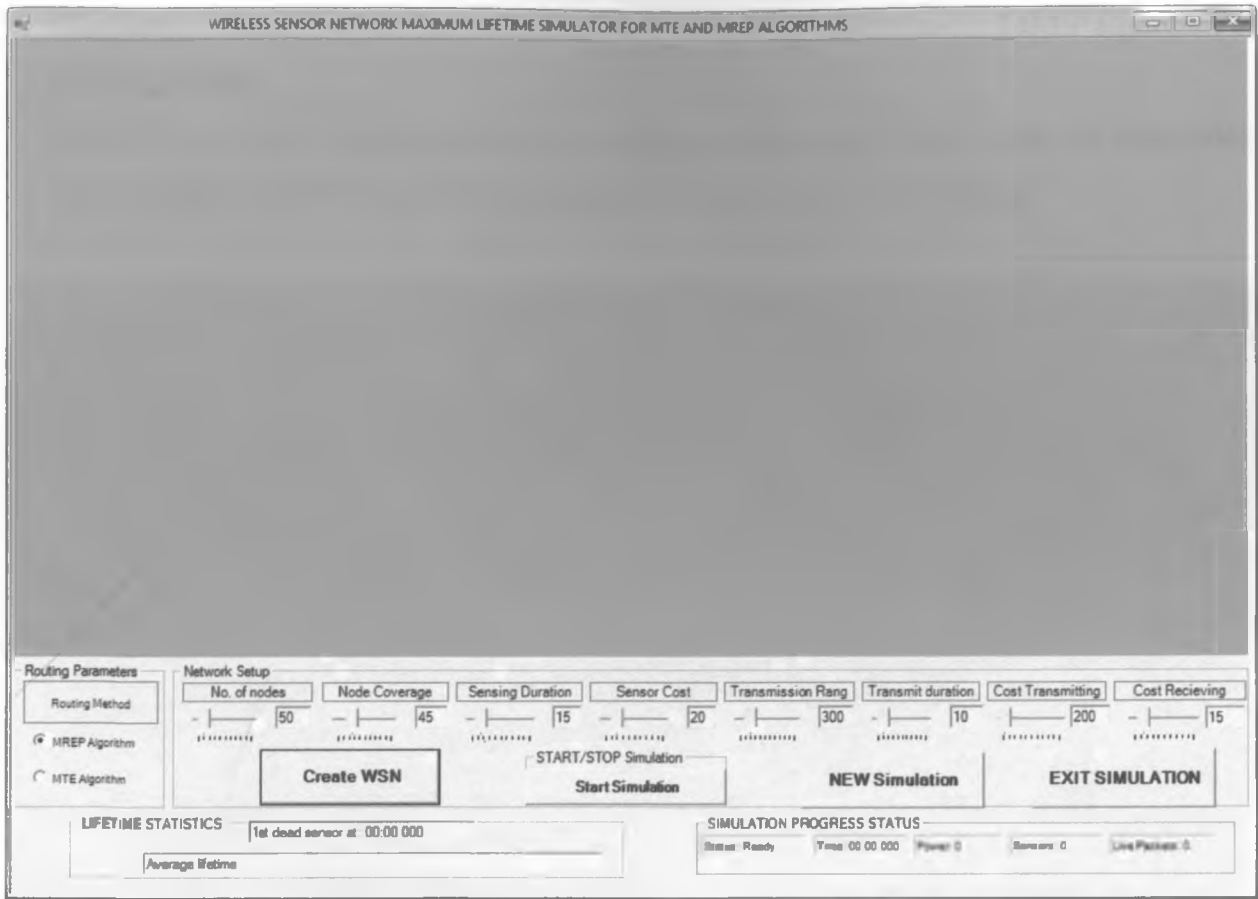
## **BIBLIOGRAPHY**

1. <http://en.wikipedia.org/wiki/Bellman-Ford>
2. The bellman-ford algorithm and “distributed bellman-ford” by David Walden.
3. Adegboyega Ojo and Elsa Estevez. Object-Oriented Analysis and Design with UML - Training Course. e-Macao Report 19, Center for Electronic Governance at UNU-IIST, UNU-IIST, P.O. Box 3058, Macau, October 2005.
4. [http://en.wikipedia.org/wiki/Unified\\_Modeling\\_Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language).
5. Gill Zussman, Adrian Segall, “Energy. Efficient routing in ad hoc disaster recovery networks” ELSEVIER, 2003
6. A. Sobeih, J. C. Hou, L.-C. Kung, N. Li, H. Zhang, W.-P. Chen, H.-Y. Tyan, and H. Lim, “J-Sim: a simulation and emulation environment for wireless sensor networks,” IEEE Wireless Communications, vol. 13, pp. 104-19, Aug. 2006
7. Using the Georgia Tech Network Simulator, Dr. George F. Riley Georgia Institute of Technology
8. B. Tatiana, H. Wen, K. Salil, R. Branko, G. Neil, B. Travis, R. Mark, and J. Sanjay, “Wireless sensor networks for battlefield surveillance,” Land Warfare Conference, 2006.
9. Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. IEEE Internet Computing, 10(2):18–25, 2006.
10. <http://www.ieee.org>
11. <http://www.tinyos.net/>



## APPENDIX B: USER MANUAL

This application is developed to provide lifetime statistics for MREP and MTE routing algorithms for wireless sensor networks. The user must be conversant with sensor networks especially energy aware routing algorithms. When the application opens the screen is as follows:

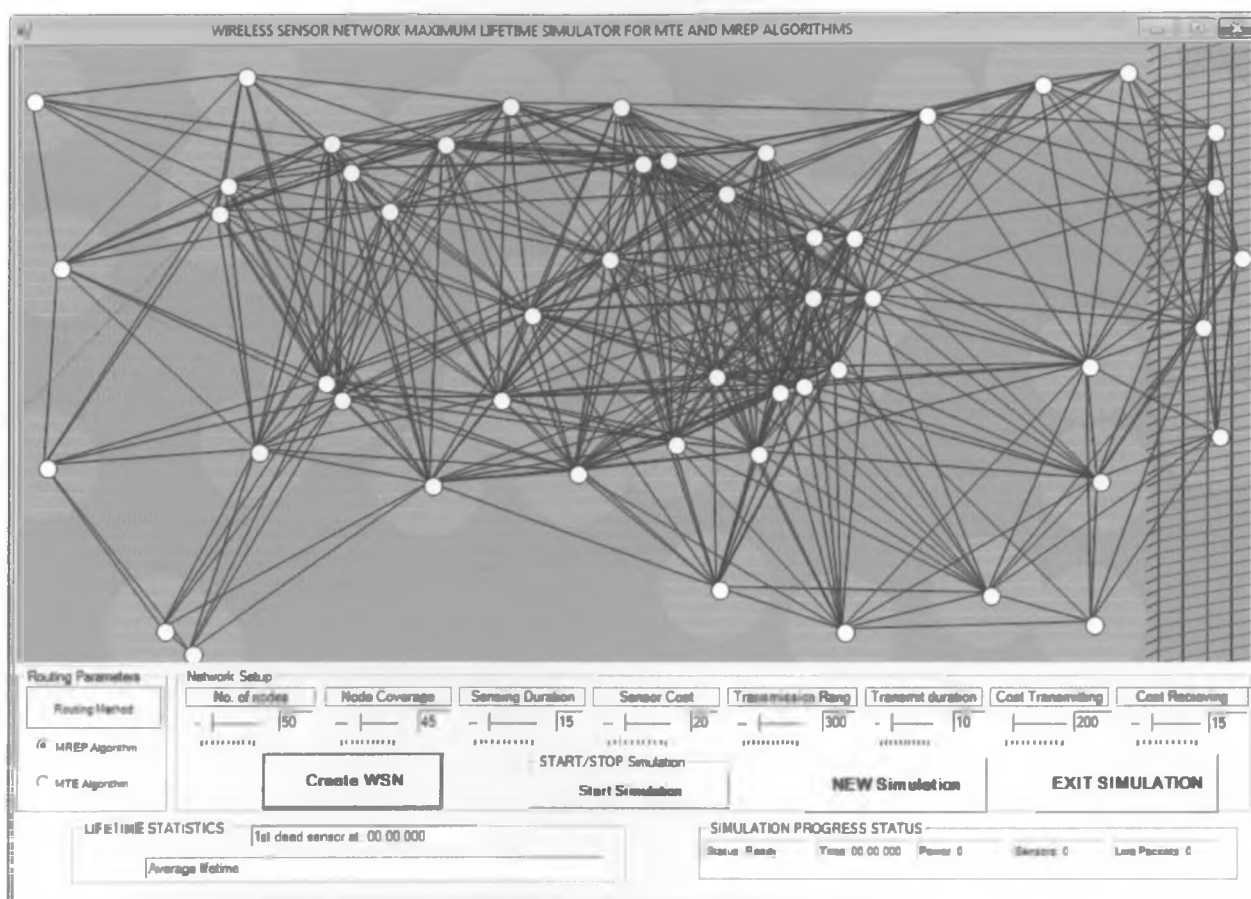


1. The first step is to provide values for the input variables by dragging the trackbars provided or use the default values. The input variables are as follows:
  - a. **No. of nodes** : The number of nodes in the network.
  - b. **Node coverage**: The size of the area within which a sensor can detect movement.
  - c. **Sensing duration**: How long a tripped sensor waits before sending a subsequent event.
  - d. **Sensor cost**: The energy consumed by sensor activation.
  - e. **Transmission range**: The maximum distance between two connected nodes.

- f. **Transmit duration:** The amount of time required to send a packet.
- g. **Cost transmitting:** Energy consumed by sending a packet.
- h. **Cost receiving:** The energy consumed by receiving a packet.

Additionally the objects on the screen are help context sensitive and putting the cursor on them pops up a description message.

2. Choose the routing method for the simulation by checking the button for MREP algorithm or MTE algorithm.
3. Deploy the network with the settings by clicking on the Create WSN button, the application should display as follows (note the topology will vary with each simulation):



4. Click on **start simulation** to start the simulation, as the simulation runs the button changes to **stop simulation** which you may click to stop the simulation incase one needs to capture an

observation during simulation. During simulation the **LIFETIME STATISTICS** region will display the time to the first node failure, the **SIMULATION PROGRESS STATUS** region will display information on the time the simulation has been running, the residual power, active sensors and packets on transit.

5. **New simulation** button will relaunch the application for a new fresh simulation.
6. **Exit simulation** button will exit the simulation and close the application.

## APPENDIX C: SAMPLE PROGRAMS

### 1. BUILDING THE NETWORK OF NODES

```
public void BuildNetwork() {
    aSensors = new ArrayList();
    dSensors = new ArrayList();
    deadsensor = new ArrayList();
    deathtime = new ArrayList();
    while (aSensors.Count < iNetworkSize) {
        // add random sensors to the network
        while (aSensors.Count < iNetworkSize) {
            WSNSensor sensor = new
WSNSensor(r.Next(iMaxX - 10) + 5, r.Next(iMaxY - 10) + 5,
iSensorRadius);

            int i = 0;
            // add to list
            for (; i < aSensors.Count; i++) {
                if (((WSNSensor) aSensors[i]).x >
sensor.x) {

                    aSensors.Insert(i, sensor);
                    break;

                }
                if
((Math.Abs(((WSNSensor) aSensors[i]).x - sensor.x) < 14) &&
(Math.Abs(((WSNSensor) aSensors[i]).y - sensor.y) < 14))
                    break;
            }
            if (i == aSensors.Count) // not added to
list - add at the end
                aSensors.Add(sensor);
        }
        // establish connections
        for (int i = 0; i < aSensors.Count; i++) {
            WSNSensor iSensor = (WSNSensor)
aSensors[i];

            iSensor.aConnections = new ArrayList();
            if (iSensor.x > iDestinationX)
                iSensor.aConnections.Add(new
WSNSensorConnection(iSensor, null, (int) fTransmissionCost, 0,
iTransmitterDelay));

            else {
                for (int j = i + 1; j <
aSensors.Count; j++) {
                    WSNSensor jSensor = (WSNSensor)
aSensors[j];

                    int iRadius = (int)
Math.Sqrt(Math.Pow(iSensor.x - jSensor.x, 2) + Math.Pow(iSensor.y -
jSensor.y, 2));
```

```

        if (iRadius <=
iTransmissionRadius)

        iSensor.aConnections.Add(new WSNSensorConnection(iSensor,
jSensor, (int) (fTransmissionCost * iRadius /iTransmissionRadius),
iReceiveCost, iTransmitterDelay));
    }
}
// weed out all sensors with no downstream
connections that aren't in the destination zone

ArrayList aRemoveSensors = new ArrayList();
for (int i = aSensors.Count - 1; i >= 0; i--) {
    WSNSensor sensor = (WSNSensor) aSensors[i];
    if ((sensor.x < iDestinationX) &&
(sensor.aConnections.Count == 0)) { // dead end - eliminate.
        aRemoveSensors.Add(sensor);
        // scan all upstream nodes, to see if they were
connected to this removed node, and delete the connections
        for (int j = i - 1; j >= 0; j--) {
            WSNSensor sensor2 = (WSNSensor)
aSensors[j];
            ArrayList aRemoveConnections =
new ArrayList();
            foreach (WSNSensorConnection
connection in sensor2.aConnections) {
                if (connection.sReceiver ==
sensor)
                    aRemoveConnections.Add(connection);
            }
            foreach (WSNSensorConnection
connection in aRemoveConnections)
                sensor2.aConnections.Remove(connection);
        }
    }
    foreach (WSNSensor sensor in aRemoveSensors)
        aSensors.Remove(sensor);
}
}

```

## 2. MREP ROUTING

```

public void SetRoutingInformationMREP() {
    // this function updates directed-routing selections
    foreach (WSNSensor sensor in aSensors) {
        if (sensor.iResidualEnergy > 0) {

```





```

    public int iInitialEnergy;
    // the initial power of the node
    public int iResidualEnergy;
    // the current power of the node
    public int iSensorDelay = 0;
    // the timer until the sensor is ready to be tripped again
    public int iSensorRadius;
    // the radius of this sensor
    public TimeSpan iLifeTime;

    public WSNSensor(int x, int y, int iSensorRadius) {
        this.x = x;
        this.y = y;
        this.iSensorRadius = iSensorRadius;
        aConnections = new ArrayList();
        connectionCurrent = null;
        iInitialEnergy = iResidualEnergy =
WSNNetwork.iMaxEnergy;
    }

    #endregion
}

```

## 5. COMMUNICATION LINK

```

public class WSNSensorConnection {
    // This class represents a communications link between two
    wireless sensors.

    #region Variables and initialization code

    public WSNSensor sSender;
    // the upstream sensor
    public WSNSensor sReceiver;
    // the downstream sensor - every node in the data
collector/uplink zone will have a connection with a NULL sReceiver.
    public Packet packet = null;
    // the packet currently being transmitted on this connection
(only one at a time, of course)
    public int iTransmitCost, iReceiveCost;
    // the energy costs of transmitting and receiving the packet
    public int iTransmitterDelay;
    // the total time this node would normally wait to complete
delivery of a packet
    public int iTransmitting;
    // the timer for completing delivery of a packet

```



```

public WSNSensorConnection(WSNSensor sSender, WSNSensor
sReceiver, int iTransmitCost, int iReceiveCost, int iTransmitterDelay)
{
    this.sSender = sSender;
    this.sReceiver = sReceiver;
    this.iTransmitCost = iTransmitCost;
    this.iReceiveCost = iReceiveCost;
    this.iTransmitting = 0;
    this.iTransmitterDelay = iTransmitterDelay;
    //this.iMaxX = iMaxX;
}

#endregion

```

## 6. SIMULATION EVENT HANDLERS CODE

```

private void btnDeploy_Click(object sender, System.EventArgs e) {
    network = new WSNNetwork((int) trackNetworkSize.Value,
(int) trackSensorRadius.Value, (int) trackSensorDelay.Value, (int)
trackTransmissionRadius.Value, (int) trackTransmitterDelay.Value, (float)
trackTransmissionCost.Value / 100.0f, (int) trackReceiveCost.Value, (int)
trackSensorCost.Value, radioMREP.Checked, trackEnergyCost.Value,
trackResidualEnergy.Value, trackInitialEnergy.Value, 1, picNetwork.Width - 5,
picNetwork.Height - 5, picNetwork.Width - 15);
    iSetupDisplay = 0; // initiate the "deploying network"
display
}

private void btnStart_Click(object sender, System.EventArgs e) {
    if (network != null) {
        if (network.tSimulation == null) { // no simulation
running - start a new simulation (by spawning a new thread)
            network.Reset(true);
            network.tSimulation = new Thread(new
ThreadStart(network.RunSimulation));
            network.tSimulation.Start();
            btnStart.Text = "Stop Simulation";
            btnStart.Refresh();
            lblStatus.Text = "Status: Operating";
            lblStatus.Refresh();
        }
        else { // simulation running - tell the thread to
stop running and relabel buttons
            network.bAbort = true;
            network.tSimulation = null;
            btnStart.Text = "Start Simulation";
            btnStart.Refresh();
            lblStatus.Text = "Status: Ready";
            lblStatus.Refresh();
        }
    }
}
}

```

```

private void btnReplay_Click(object sender, System.EventArgs e) {
//FindAndKillProcess("WSNSimulator");
Application.ExitThread();
Application.Restart();
//Application.Run();

#endregion

private void picNetwork_Paint(object sender,
System.Windows.Forms.PaintEventArgs e) {
// painting
if (network != null) {
network.bPainting = true;
network.bPaint = false;
}
// create drawing tools
Font font = new Font("Times New Roman", 7.0f);
StringFormat format = new StringFormat();
format.Alignment = StringAlignment.Center;
Graphics g = e.Graphics;
g.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
// draw background
g.FillRectangle(System.Drawing.Brushes.Turquoise,
e.ClipRectangle);
//g.FillRectangle(System.Drawing.Brushes.Coral,
e.ClipRectangle);
// draw network objects (if network has been deployed)
if (network != null) {
// draw sensor background
if (iSetupDisplay == -1) {
ArrayList activatedSensors = new ArrayList();
foreach (WSNSensor sensor in network.aSensors)
{
if (sensor.iSensorDelay <= 0)
g.FillEllipse(new SolidBrush(Color.FromArgb(196,
196, 100)), sensor.x - sensor.iSensorRadius, sensor.y - sensor.iSensorRadius,
sensor.iSensorRadius * 2, sensor.iSensorRadius * 2);
else
activatedSensors.Add(sensor);
}
foreach (WSNSensor sensor in activatedSensors)
//g.FillEllipse(new
SolidBrush(Color.FromArgb(196, 196, 196 + 48 * sensor.iSensorDelay /
network.iSensorDelay)), sensor.x - sensor.iSensorRadius, sensor.y -
sensor.iSensorRadius, sensor.iSensorRadius * 2, sensor.iSensorRadius * 2);
g.FillEllipse(new SolidBrush(Color.FromArgb(250, 150, 1 +
4 * sensor.iSensorDelay / network.iSensorDelay)), sensor.x -
sensor.iSensorRadius, sensor.y - sensor.iSensorRadius, sensor.iSensorRadius *
2, sensor.iSensorRadius * 2);
}
// draw end zone
g.DrawLine(Pens.Red, picNetwork.Width - 80, 0,
picNetwork.Width - 80, picNetwork.Height);
}
}

```

```

        g.DrawLine(Pens.Black, picNetwork.Width - 60, 0,
picNetwork.Width - 60, picNetwork.Height);
        g.DrawLine(Pens.Blue, picNetwork.Width - 40, 0,
picNetwork.Width - 40, picNetwork.Height);
        g.DrawLine(Pens.Black, picNetwork.Width - 20, 0,
picNetwork.Width - 20, picNetwork.Height);
        for (int i = 0; i < picNetwork.Height; i += 10)
        {
            g.DrawLine(Pens.Black, picNetwork.Width - 90, i + 5,
picNetwork.Width - 80, i);
            g.DrawLine(Pens.Green, picNetwork.Width - 80, i + 10,
picNetwork.Width, i - 5);
        }
        // draw connections
        foreach (WSNSensor sensor in network.aSensors) {
            foreach (WSNSensorConnection connection in
sensor.aConnections) {
                if ((connection.sReceiver != null) &&
((iSetupDisplay == -1) || (iSetupDisplay > connection.sReceiver.x)) &&
(connection.sSender.iResidualEnergy > 0) &&
(connection.sReceiver.iResidualEnergy > 0))
                    g.DrawLine(connection.iTransmitting
> 0 ? Pens.Red : connection == sensor.connectionCurrent ? Pens.Blue :
Pens.Black, connection.sSender.x, connection.sSender.y,
connection.sReceiver.x, connection.sReceiver.y);
            }
            // draw sensors
            Brush sensorBrush = Brushes.DarkGray;
            Pen sensorPen = Pens.Red;
            foreach (WSNSensor sensor in network.aSensors) {
                if ((iSetupDisplay == -1) || (iSetupDisplay >
sensor.x)) {
                    int color = sensor.iResidualEnergy <= 0 ?
0 : (int) (255 * sensor.iResidualEnergy / WSNNetwork.iMaxEnergy);
                    g.FillEllipse(new
SolidBrush(Color.FromArgb(color, color, color)), sensor.x - 4, sensor.y - 4,
15, 15);
                    if (sensor.iResidualEnergy <= 0) {
                        if (sensor.iSensorRadius > 0)
                            sensor.iSensorRadius--;
                    }
                    g.DrawEllipse(sensor.iResidualEnergy <= 0
? Pens.Black : Pens.Red, sensor.x - 4, sensor.y - 4, 15, 15);
                }
            }
            // draw vectors
            if ((network.bRunningSimulation == true) &&
(network.vectors != null)) {
                network.vectors.mutexVector.WaitOne();
                foreach (Vector vector in
network.vectors.aVectors) {
                    //g.FillRectangle(Brushes.Red, vector.x - 1,
vector.y - 1, 6, 6);
                    g.FillRectangle(Brushes.Red, vector.x - 3,
vector.y - 3, 6, 6);
                }
            }
        }
    }
}

```

```

        network.vectors.mutexVector.ReleaseMutex();
    }
    // draw the aeroplane to deploy the network
    if (iSetupDisplay != -1)
        //g.DrawLine(Pens.Coral, iSetupDisplay, 0,
iSetupDisplay, picNetwork.Height);
        g.DrawImageUnscaled(Resources.Resource1.aerol, iSetupDisplay,
(picNetwork.Height / 2) - 100);

        //g.DrawLine(Pens.Coral, 0, iSetupDisplay,
picNetwork.Width, iSetupDisplay);
    }
    // finish painting
    if (network != null)
        network.bPainting = false;
}

private void timerUpdate_Elapsed(object sender,
System.Timers.ElapsedEventArgs e) {
    // check to see if the network is being deployed
    if (iSetupDisplay != -1) {
        iSetupDisplay += 10;
        if (iSetupDisplay >= picNetwork.Width)
            iSetupDisplay = -1;
        picNetwork.Refresh();
    }
    // check to see if network simulation is running
    else if ((network != null) && (network.tSimulation != null)
&& (network.bPaint == true) && (network.bPainting == false)) {
        // refresh objects
        picNetwork.Refresh();
        picRadar.Refresh();
        // display information in textboxes
        if (network.bRunningSimulation == true) {
            TimeSpan counter = new
TimeSpan(System.DateTime.Now.Ticks - network.timeStart.Ticks);
            lblTime.Text = "Time: " +
counter.Minutes.ToString("d2") + ":" + counter.Seconds.ToString("d2") + "." +
counter.Milliseconds.ToString("d3");
            lblTime.Refresh();
            lblRecdPackets.Text = "Rec'd Packets: " +
network.iPacketsDelivered;
            lblRecdPackets.Refresh();
            int iPower = 0;int iSensors = 0;
            int iLivePackets = 0;
            foreach (WSNSensor sensor in network.aSensors)
            {
                iPower += sensor.iResidualEnergy;
                if (sensor.iResidualEnergy > 0) {
                    iSensors++;
                    iLivePackets +=
sensor.aPackets.Count;
                }
            }
            lblSensors.Text = "Sensors: " + iSensors + "/"
+ network.aSensors.Count;
            lblSensors.Refresh();
        }
    }
}

```

```

        lblPower.Text = "Power: " + iPower;
        lblPower.Refresh();
        lblLivePackets.Text = "Live Packets: " +
iLivePackets;
        lblLivePackets.Refresh();
        foreach (WSNSensor sensor in network.aSensors)
        {
            if ((sensor.iResidualEnergy <= 0) &&
(sensor.iLifeTime.TotalSeconds.Equals(0)))
            {
                network.tt++;
                sensor.iLifeTime = new
TimeSpan(System.DateTime.Now.Ticks - network.timeStart.Ticks);
                TimeSpan deadt = sensor.iLifeTime;
                network.dSensors.Add(sensor);
                network.totlifetime = network.totlifetime +
sensor.iLifeTime;
                WSNSensor fsensor = (WSNSensor)
network.dSensors[0];
                TimeSpan timinterval = fsensor.iLifeTime;
                textBox9.Refresh();
                textBox9.Text = "1st dead sensor after: " +
timinterval.Minutes.ToString("d2") + ":" + timinterval.Seconds.ToString("d2")
+ ":" + timinterval.Milliseconds.ToString("d3");
                network.avglifetime =
(network.totlifetime.TotalSeconds / (network.dSensors.Count));
                textBox10.Refresh();
                textBox10.Text = "Average lifetime for " +
(network.tt + 1) + " of " + network.aSensors.Count + " is : " +
network.avglifetime.ToString() + " Seconds";
            }
        }
    }
    else { // network has stopped running - throw away
the completed thread and relabel buttons
        network.tSimulation = null;
        btnStart.Text = "Start Simulation";
        btnStart.Refresh();
        lblStatus.Text = "Status: Ready";
        lblStatus.Refresh();
    }
}
}
#endregion

```