THE GRID SCAN HEURISTIC FOR EXTENDING LIFETIME IN WIRELESS SENSOR NETWORKS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Srikar Pachva

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

October 2011

Fargo, North Dakota

North Dakota State University
Graduate School

Title

A HEURISTIC FOR EXTENDING LIFETIME IN

WIRELESS SENSOR NETWORKS

By

SRIKAR PACHVA

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Pachva, Srikar, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, October 2011. The Grid Scan Heuristic for Extending Lifetime in Wireless Sensor Networks. Major Professor: Dr. Kendall E Nygard.

Wireless Sensor Networks, consisting of sensing devices which sense the environment and communicate information among each other, is an emergent field in wireless networking with many potential applications. Efficiency of energy consumption is an important objective in these wireless sensor networks. All these sensor nodes have limited battery supplies and limited sensing capabilities. Network Lifetime of wireless sensor networks has a strong dependence on sensors' battery power. Since these sensor networks have large numbers of nodes, allowing some nodes to sleep for particular intervals of time can result in an increase of network lifetime.

In this paper, a sleep scheduling heuristic called the Grid Scan heuristic is proposed and implemented. The method extends network lifetime significantly by scheduling sensors such that it maintains a threshold level of coverage. Scheduling sensors to sleep state and active state intelligently extends network lifetime significantly. The Grid Scan Heuristic intelligently decides when a sensor should be turned on and off such that the wireless network maintains a level of coverage. The performance of the Grid Scan approach is compared with another sleep scheduling algorithm called Randomized Scheduling, which is an already existing algorithm in the literature.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor, Dr. Kendall Nygard, for his continued support throughout this paper. I appreciate his time, assistance and continuous guidance. I would also like to thank Dr. Simone Ludwig, Dr. Saeed Salem, and Dr. Chao You for being a part of my graduate supervisory committee. Special thanks to the faculty and staff of the Computer Science Department for their unconditional support throughout my master's program. I am highly obligated to my friend Rajiv. Finally, I am grateful to my parents who are the reason for all my achievements and have supported and motivated me throughout my life.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

1. INTRODUCTION 1

2. LITERATURE REVIEW 4

2.1. Randomized Scheduling 5

2.2. Distance Based Scheduling 5

2.3. Coverage Aware Sleep Scheduling 6

2.4. Hybrid Genetic Algorithm to Improve Network Lifetime 7

2.5. PEAS Algorithm 7

2.6. Circle Perimeter Coverage Theory Algorithm 8

3. PROBLEM DESCRIPTION IN SENSOR NETWORKS 9

3.1. Problem Definition with Respect to Coverage Area 9

3.2. Assumptions 10

3.3. Coverage Calculation 12

3.4. Network Lifetime Calculation 14

3.5. The Randomized Scheduling Approach 15

3.6. The Grid Scan Heuristic Scheduling Algorithm 17

4. METHODOLOGY AND CLASSES 25

4.1. AdhocWireless Class 25

4.2. GraphicDrawer Class 26

4.3. NodesDistributor Class 27

4.4. NodeCoverage Class 28

4.5. GridScan Class 30

4.6. CoverageAlgorithm Class 31

5. RESULTS 32

5.1. Test Case 1: Changing Grid Size and Comparing Both Algorithms 32

5.2. Test Case 2: Changing Threshold Coverage 34

5.3. Comparison of Two Algorithms Based on Number of Sensors 36

6. CONCLUSION AND FUTURE WORK 46

REFERENCES 48

LIST OF TABLES

Table | Page

4.1: AdhocWireless Class: Variables 26

4.2: GraphicDrawer Class: Methods 27

4.3: NodesDistributor Class: Methods 28

4.4: NodeCoverage Class: Methods 29

4.5: GridScan Class: Methods 30

4.6: CoverageAlgorithm Class: Methods 31

5.1: Test Results for Changing Grid Size and Calculating Network Lifetime 32

5.2: Results for Test Cases Varying Threshold 35

5.3: Test Results for Both Algorithms by Varying Number of Sensors 37

5.4: Test Results for Various Inputs and Comparing Network Lifetime 39

5.5: 10 Test Cases for Keeping Grid Size = 10 40

5.6: 10 Test Cases for Keeping Grid Size = 40 41

5.7: Test Cases for Grid Size = 80 42

5.8: Test Results for Number of Sensors = 100 43

5.9: Test Results for Number of Sensors = 250 44

5.10: Test Results for Number of Sensors = 500 44

LIST OF FIGURES

Figure Page

2.1: Blind Point Problem Description in Coverage Aware Algorithm........................6

3.1: Sample Input Configurations for Grid Scan Approach.....................................11

3.2: Example Showing Randomized Scheduling 16

3.3: Example Showing Grid Division and Allocation of Sensors............................18

3.4: Example Showing Allocation of Sensors in Grid Borders................................20

3.5: Example Grid Scan Heuristic Implementation...22

5.1: Network Lifetime Using Both the Algorithms with a Varying Grid Size.........33

5.2: Network Lifetime Calculation by Changing Threshold Coverage....................36

5.3: Test Results for Varying Number of Sensors...38

1. INTRODUCTION

Advancement in the micro-electro mechanical systems technology have made it possible to deploy sensors in a particular geographical area and sense the environment by calculating physical parameters such as heat, sound, temperature, light, etc. [1]. A sensor network consists of group of sensor nodes which are able to sense, observe and transmit data. Sensor networks have many potential applications which include monitoring battlefields and detecting enemy, sense the environmental changes in oceans, forests, and atmosphere, providing security to the buildings and also monitoring human body which is very useful in bio-medical applications [1].

In many applications, wireless sensor networks face some drawbacks such as: limited power, limited communication capabilities, limited computation capabilities, huge deployment area, count of nodes in a network, etc. [2]. Because of all these drawbacks in wireless sensor networks, there are many challenging problems. Of all these, the most important issue regarding the design of sensor networks is power consumption. These sensor networks consist of large number of nodes and are deployed randomly or deterministically based on some criteria. Generally, nodes are spread in to geographical area using aircraft where ground access is not possible resulting in redundant deployment. This redundant deployment makes the sensors' sensing capabilities to overlap such that a part of geographical area is covered by many sensors.

In a given geographical area, if a point p in the area is covered by at least k sensors, then proper coverage of wireless networks is still ensured even if some sensors fail or sleep for particular time intervals that are covering point p. Since these networks consists of large number of nodes, scheduling some of the sensors to sleep periodically result in significant improvement in the lifetime of wireless sensor network.

For every wireless network to remain active, it has to maintain a threshold level of coverage for the area where sensor network is deployed ensuring balance between high coverage and longer network lifetime [3]. Most of the applications do not require 100% of the network coverage and only partial coverage is necessary to maintain the network active. These partial coverage requirements of the wireless networks also improves network lifetime further [1].

In this paper, we propose a scheduling algorithm called the Grid Scan Heuristic which determines the sensors sleep times and active times intelligently and resulting in substantial improvement in the network lifetime. This paper starts out by forming a sensor network with sensors placed randomly in a given area. After forming a sensor network by redundant deployment of sensors, entire network is monitored for the threshold level of coverage *C* and lifetime of wireless sensor network is calculated. Because of the limited power capabilities of sensor nodes, some of the sensors may fail after some time which nullifies the sensing capabilities of particular node and thus decreasing the coverage of the network. At some point of time, more number of nodes failure results in getting coverage less than threshold level where the total network lifetime is calculated i.e. total time when a network is active and maintains a threshold level of coverage.

In order to improve network lifetime in such cases where battery life of node is a major concern, scheduling sensors effectively like when a sensor should be turned ON and when it should be turned OFF in order to maintain desired coverage is very much necessary. So, the goal for this paper is to develop a sleep scheduling heuristic which automatically improves coverage as well as network lifetime.

The application developed to implement the proposed algorithm is a java applet consisting of input parameters like: number of sensor nodes, radius of sensor nodes, threshold coverage limit, sensor lifetime, grid size. These parameters we get from the user input are used to generate a wireless sensor network with randomly deployed sensors in a rectangular area whose length and breadth are also defined by user inputs. In this application, the sensors are represented by various colors for easy identification, such as green for sensors which are active and red for sensors which are dead i.e. sensors battery life has ended. When user inputs are given, firstly, Randomized Scheduling algorithm is implemented and network lifetime is calculated and then the proposed Grid Scan algorithm is implemented and network lifetime is calculated and lifetimes for both algorithms are compared.

The paper is organized as follows: Chapter 2 has a brief discussion about sensor coverage issues and common scheduling algorithms. Chapter 3 describes the problem definition along with the application runtime configurations, and also the description of two algorithms. Chapter 4 explains different classes and controls used in the application along with the sample output for both algorithms. Chapter 5 deals with results and experiments. Chapter 6 addresses conclusions and future work.

2. LITERATURE REVIEW

In various applications using wireless sensor networks the main design objective is to minimize the power consumption and maximize the network lifetime. Generally, many sensor networks are deployed randomly which is the convenient way where geographical areas are very large and uncontrollable. When nodes are deployed randomly, the deployment may be redundant sometimes and results in overlapping of sensing regions. In a dense deployment scenario of wireless sensor networks, if all the sensors operate at same time then ultimately the energy consumption will be excessive and thus networks' reliability is minimized [1].

To improve networks' reliability and extend network lifetime, scheduling of sensors effectively is necessary. One of the main problems in sensor networks is monitoring of coverage attained by the network. The coverage problem in wireless sensor networks is termed as a decision problem where, main goal is to detect whether every point in the service area of the sensor network is covered by at least k sensors where k is any constant [3]. Sensor network remains active until the network maintains a minimum coverage. If a sensor network is not able to maintain its minimum coverage then the quality of service provided by the network is degraded.

In sensor networks, there are two viewpoints of coverage exist: worst case coverage and best case coverage. Worst case coverage deals with the areas that are less observed and left out by sensing regions and whereas the best case coverage deals with the areas that are highly observed and identifying best support regions. There has been a considerable research in the area of sensor coverage and many algorithms have been proposed for effective scheduling of sensors to improve networks efficiency. Some of the approaches are:

2.1. Randomized Scheduling

When sensor nodes are deployed in a geographical area, the minimum threshold level of coverage is given by the user input parameters and by assuming the battery life to be constant throughout the network. Randomized Scheduling schedules a sensor's sleep probability randomly using random number generator function. In this scheduling mechanism, only a few numbers of sensors are turned ON to account for the minimum threshold level of coverage thus improving network lifetime by a considerable amount. Though this algorithm improves network lifetime, coverage decision making is random and is not intelligent so the results we get are not appropriate [3].

2.2. Distance Based Scheduling

In this type of scheduling, the wireless sensor networks considered are cluster based sensor networks. In this approach, authors use a linear distance based scheme which selects the sensors to sleep when the node center is farther from the cluster head to compensate for the higher energy they use in communication [3]. Though distance base scheduling improves network lifetime substantially the coverage guaranteed by this type of scheduling is very low [4]. This type of scheduling considers only cluster based networks and every cluster has one cluster head and distance is calculated between these cluster heads in order to determine the sleep scheduling probability of the sensor nodes that are present in the clusters. With this scheduling scheme the coverage obtained is very less because of the allocation of sensors to different clusters based on the distance and also the complexity in calculating the distance are the main drawbacks for this approach.

2.3. Coverage Aware Sleep Scheduling

Sensor's sleep probability is determined by the sensing area of the node covered by its neighboring sensors. In this scheduling scheme, the sensor *k* detects how many active sensors are there in its neighborhood and calculate the total sensing area that it's active neighbor's covers and probabilistically determines when it should be active or sleep. The algorithm runs for different time cycles and in each cycle only particular set of sensors are turned on and thus improving network lifetime [3].

Though this type of scheduling significantly improves network lifetime, this approach also has some drawbacks like: blind point problem and complexity in overlap calculation. In Figure 2.1 (a) sensors f and g are totally covered by sensors a, b, c, d, e. So, in coverage aware scheduling cycle when sensors f and g are turned in to sleep mode we can see the blind point in figure 2.1 (b).

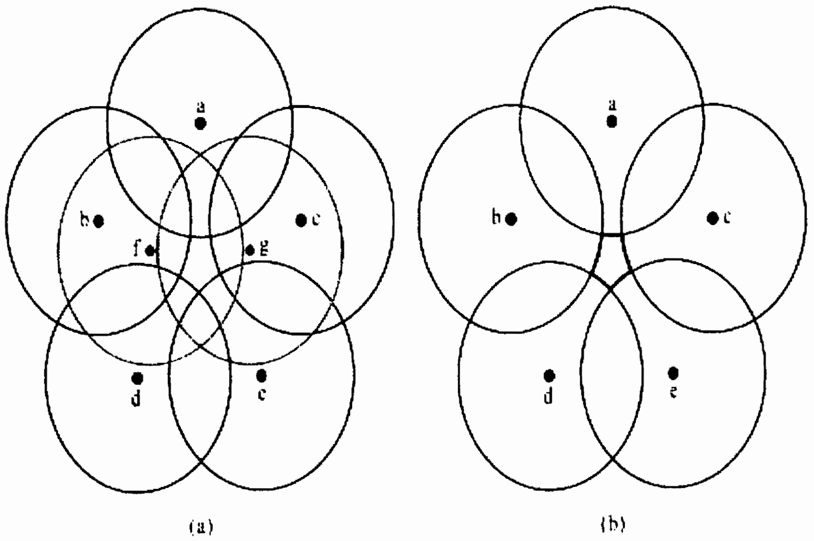


Figure 2.1: Blind Point Problem Description in Coverage Aware Algorithm [5].

Also when the numbers of overlap nodes are higher in number this approach becomes very complex to determine the sensors active and sleep states.

2.4. Hybrid Genetic Algorithm to Improve Network Lifetime

This scheduling scheme comprises of generic operations with fitness improving local search strategy where wireless sensor network is divided in to disjoint set covers, where each disjoint set can cover all the targets. Fitness function is evaluated for each disjoint set covers and a local search is implemented on these sets to determine the sets which guarantee minimum threshold level of coverage. The coverage is preserved and lifetime of the network is extended with this algorithm but, the number of iterations it takes and the processing time to check all the combinations to determine number of disjoint covers is main drawback with this approach [6].

2.5. PEAS Algorithm

In this scheduling algorithm, all nodes are in sleep state at the beginning then node cycle starts and a detection message is sent to all nodes. Distance between the active nodes within the radio radius and a node that sent detection message is calculated. Then the distance is compared with fixed threshold and checks whether it is smaller or not to determine the active state. Though this algorithm is fault tolerant and enhances network lifetime, it cannot ensure fully covered area and it has some coverage holes [7]. The coverage guaranteed by this algorithm is always less and is not appropriate for the networks which require pinpoint coverage such as sensor networks deployed in the enemy territories. The failure probability for this type of scheduling is high because of the dependence of the control messages that are carried by the sensors. If the sensors carrying control message fails then the dependency injection is totally lost and the distance calculation becomes complex. PEAS Algorithm is mainly used in habitat monitoring systems where coverage is not an essential factor.

2.6. Circle Perimeter Coverage Theory Algorithm

Considering nodes are deployed according to circles, if a circle perimeter can be covered by neighbor circle perimeter then the circle is neighbor covered and thus rendering active and sleep states to nodes based on perimeter coverage basis [8].

The above mentioned scheme can be judged by computational geometry according to the circle coverage method and active nodes are selected randomly. In all the scheduling approaches discussed above the number of active nodes present in each cycle are more than required to effectively maintain greater network lifetimes.

So, in order to effectively calculate the approximate number of active nodes required for each cycle to maintain a minimum threshold level of coverage and maximizing network lifetime, the Grid Scan heuristic approach is proposed.

3. PROBLEM DESCRIPTION IN SENSOR NETWORKS

Randomized Scheduling Algorithms and the Grid Scan Heuristic Algorithms described in this paper are used to extend network lifetime by maintaining threshold level of coverage. This chapter deals with the problem definition and detailed description of the two algorithms functionalities. Subsections 3.1, 3.2, 3.3, and 3.4 deals with problem description, assumptions made coverage calculation and network lifetime calculation. The Randomized Scheduling and the Grid Scan Heuristic Algorithm are explained in Subsections 3.4 and 3.5 respectively.

3.1. Problem Definition with Respect to Coverage Area

When sensors are deployed randomly, the main problem that encounter in potential wireless sensor networks is effective area coverage by those randomly deployed sensors. Overlapping of the service area results when sensors are randomly deployed which in turn results in the less area covered by those sensors than their actual sensing capability. In order to effectively utilize the sensors which are deployed in a particular geographical area, we have to schedule the sensors whose sensing areas are overlapped. By taking turns like which sensors should be in active state and which should be in sleep state, the networks efficiency is improved and we get a substantial amount of increase in networks lifetime. The problem with this scheduling is to determine intelligently the set of sensors that should be turned in to active state in a particular scheduling cycle to maintain a minimum threshold level of coverage throughout the network. The Grid Scan Heuristic proposed in this paper try to address this problem [9]. Grid Scan explains the intelligent scheduling of sensors in order to improve network lifetime.

In this paper, two algorithms Randomized Scheduling and the Grid Scan Heuristic approach is tested with a java applet application which explains their functionality and also calculate the coverage and overlap percentage. The application then generates the network lifetimes for both the approaches for comparison. The java application is developed using java SDK 1.6, and the entire application is a java applet application. The input parameters are given at the run time configurations window and the IDE used to implement both algorithms is Eclipse Helios.

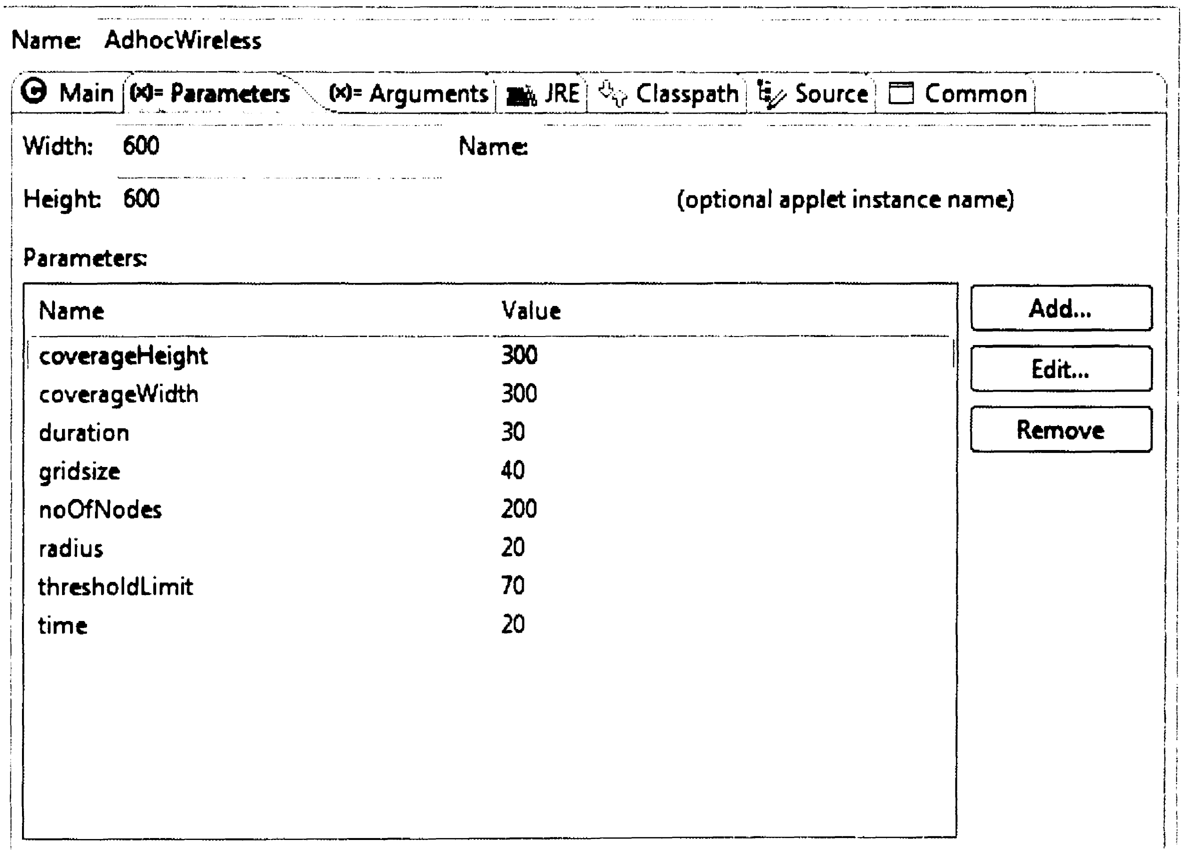
3.2. Assumptions

The first assumption for this algorithm is that the sensor's sensing region is circular in area and all the sensors that are randomly deployed have same sensing radius. This means that sensor can cover the area which is equal to area of the circle with radius r. So, with this assumption we can calculate the area covered by a sensor and also we can calculate the overlap area percentage.

The second assumption is that the time a sensor can live is constant for all the sensors. For every sensor there is a lifetime, when a sensor's battery drains it will go to dead state where it cannot process or sense anything.

The number of sensors, threshold coverage limits and grid size parameters are given as input and the algorithm operates based on these values and calculates network lifetime using both algorithms. Different combination of values for these input parameters results in different network lifetime for both the algorithms. The three parameters result in significant changes in network lifetime when each of them are varied by keeping other two parameters constant. Sensors lifetime also plays a crucial role in determining the lifetime because of the time they remain active.

For example, Figure 3.1 shows a set of input parameters given to the applet application in runtime configurations.



Name: AdhocWireless

Width: 600 Name:

Height: 600 (optional applet instance name)

Parameters:

Name	Value
coverageHeight	300
coverageWidth	300
duration	30
gridsize	40
noOfNodes	200
radius	20
thresholdLimit	70
time	20

Figure 3.1: Sample Input Configurations for Grid Scan Approach

In figure 3.1, the top line consisting of width and height indicates the applet size. The parameters coverageHeight and coverageWidth indicates the rectangular height and width inside the applet where sensors are deployed. The user can add additional inputs using the Add button to the right and also can edit the existing input values using the Edit button. Thus user has the total control over the network to change input parameters and calculate network lifetime using both algorithms. Also for convenience the sensors which are active are colored in green and the sensors which are dead are colored in red. The coverage percentage and the overlap percentage of the network are calculated for every two seconds. The units employed for network lifetime calculation is seconds for convenience purpose but generally those units may be days or months.

3.3. Coverage Calculation

For the sensor network, the generated rectangular area is like a constrained space and sensors are randomly deployed in that area only. The rectangular grid here represents the geographical area where sensors are deployed. In order to calculate the coverage area every unit in the rectangular area should be monitored. So, the total rectangular area is divided in to 1×1 pixels of the total area. For example, in the Figure 3.1, the length and breadth of the rectangular area where sensors are deployed are 300×300. So, by dividing this area in to small grids of size 1×1 pixels we get 90000 pixels. So, each pixel is monitored and checked whether it is covered by a deployed sensor or not by calculating the distance d between the pixel and the node's center c. If the distance d is less than the radius of the sensor r then the pixel lies within the node and thus it is covered by sensor s.

Let's say a set of sensor nodes $S = \{s_1, s_2, s_3, s_4, \ldots, s_n\}$ are spread in to a two dimensional rectangular area A where rectangular length is L and breadth is B and each sensor node can be self-located in the area A with the coordinate (x,y) . The sensing range of node s_i is defined as a circle with the center of node s_i and radius r_i.

Define A_{si} as the area covered by sensor s_i . The area A is divided in to $L\times W$ number of 1×1 pixels and each has a coordinate (a_p , b_p) . From this definition the total rectangular area A in which sensors are deployed is divided into small rectangular units of size 1×1 pixels with which our algorithm can calculate the total coverage of the sensor network. The division of area is required because of the raster scan scheme applied to calculate the coverage percentage.

Coverage Definition: A pixel p (a,b) is said to be covered by sensor s_i , if the distance d_{ip} between center of s_i and the pixel coordinate (a_p , b_p) is less than or equal to radius r_i..

$p_{(a,b)} \in A_{si} \Leftrightarrow d_{ip} = \sqrt{(x_i - a_p)2 + (y_i - b_p)2} \leq r_i$. [10].

For an n-sequence i.e. for n number of pixels we calculate whether the pixel belongs to any sensing area or not. Say for example, A n-sequence $\mu_{p(a,b)} = (\mu_{p(a,b)}^1, \mu_{p(a,b)}^2, \mu_{p(a,b)}^3, \ldots\ldots\ldots \mu_{p(a,b)}^n)$ takes information whether pixel $p_{(a,b)}$ is covered by sensors in set S or not. $\mu_{p(a,b)}^i = 1$ indicates that pixel $p_{(a,b)}$ is covered by sensor node s_i. $\mu_{p(a,b)}^i = 0$ means that pixel $p_{(a,b)}$ is not covered by sensor node s_i [10].

A pixel $p_{(a,b)}$ in area A is said to be *k* covered if the distance between the pixel and any k sensor nodes in S is less than or equal to corresponding radius. So *k* covered rate can be calculated by Grid Scan Scheme using:

$$\text{Coverage Percentage } C = \frac{\sum_{a=1}^{L}\sum_{b=1}^{W} \alpha_{p(a,b)}(k)}{(L\times W)} \qquad (1) \qquad [10]$$

$$\text{s.t. } \alpha_{p(a,b)}(k) = \begin{cases} 1 \text{ if } \sum_{i=1}^{n} (\mu_{p(a,b)}^i) = k \\ 0 \text{ otherwise} \end{cases}$$

Thus the coverage percentage of the wireless sensor network can be obtained by using the equation (1).

Overlap Definition: A pixel $p_{(a,b)}$ in area A is said to be k-covered when the distance between the pixel $p_{(a,b)}$ and any k sensor nodes in S are less than or equal to corresponding radius. Then each pixel is covered by k sensors which results in overlap and overlap percentage is calculated according to the definition.

3.4. Network Lifetime Calculation

Network Lifetime is the most important metric to evaluate sensor networks. The lifetime of sensor networks strongly depends on the individual sensor nodes lifetime because they constitute the whole network. Most energy is consumed when sensing, transmitting and processing some data. The main parameters that effect the network lifetime in wireless sensor networks are coverage, connectivity and node availability [11]. In this paper, we deal with the calculation of network lifetime based on the coverage.

In order to remain network active, the region of interest with some finite target points have to covered say for example α percentage of network points. If at least one sensor covers such point then the target coverage is said to be achieved. Then the network is said to be α covered. Thus the lifetime of sensor network is defined as the time that the region of interest is completely within the sensing range of at least one sensor node [11]. So, the time that network maintains a threshold level of coverage is defined as network lifetime.

Defining Network Lifetime solely based on the sensor coverage don't yield good results in most of the cases because the sensed information should be processed and transmitted in order to achieve good practical results. But in this scenario, for our convenience we are considering the calculation of network lifetime only based on sensor coverage. In the subsections 3.5 and 3.6 the calculation of network lifetime by using two algorithms Randomized Scheduling and the Grid Scan approach is determined and compared based on the assumptions which are described in section 3.2. Pseudo Codes for both the algorithms are explained in subsections 3.5 and 3.6 and also the figures representing both the outputs are shown in these sections.

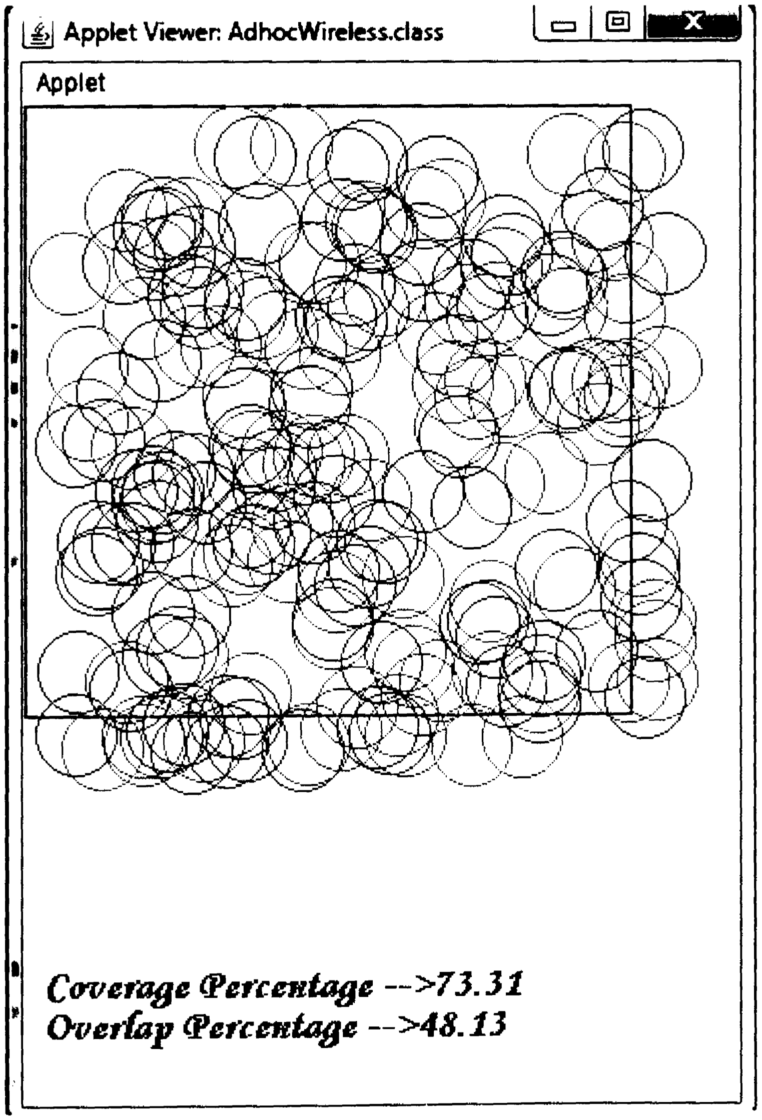
3.5. The Randomized Scheduling Approach

The objective of Scheduling scheme is to maximize the network lifetime while covering most of the sensor field at all time. Using Randomized Scheduling, network lifetime is increased considerably. Two of the main reasons to employ randomized scheduling are: (i) Scheduling sensors in a network is NP-complete problem and there is no deterministic approach to prove that results are accurate. With randomized scheduling approach we can increase the lifetime considerably. (ii) Randomized Scheduling Algorithms are easily implemented in a distributed network environments as there is no need to store state information.

In our random scheduling approach, sensors are deployed randomly to form a network and a minimum threshold coverage limit is given as an input parameter which tells, the network should maintain threshold level of coverage in order to be active. Firstly, only some random count of sensors are made active and coverage percentage of the total network using active sensors is calculated by using the formulae described in section 3.3 and also the overlap percentage is calculated. If the calculated coverage percentage of the sensor network is greater than the threshold level of coverage then the timer is made active to calculate the network lifetime. The ON times of the sensors in random scheduling depends on the input maximum start time i.e. the interval of time in which all sensors should start. In this way, the sensors ON times depend on the maximum start time and all sensors are not turned ON at the same time.

We continue this process until the calculated coverage percentage is less than the threshold coverage percentage. Once the coverage percentage is less than the threshold coverage percentage then the sensor network lifetime is calculated as how much time the network is active maintaining threshold level of coverage after deployment of sensors.

Figure 3.2 explains the Randomized Scheduling approach with green color nodes represents the active sensors and red color nodes represent the inactive and dead sensors.



Coverage Percentage -->73.31
Overlap Percentage -->48.13

Figure 3.2: Example Showing Randomized Scheduling

But, as we described in the section 3.2 assumptions, every node has a time to live which is constant throughout our approaches and this also reflects the calculation of network lifetime. In both approaches proposed in this paper, lifetime of a sensor is constant and the value is inputted as 20 units.

Pseudo Code for Randomized Scheduling:

```
Set Sensor Time to Live = 15 Units;
Set Maximum Start Time Interval = 50 Units;
Set Input Start Times of Sensors = Maximum Start Time * Math. Random ();
    For Active Nodes
        Calculate Coverage Percentage;
         If Coverage Percentage > Threshold Coverage Then
        {
          Start Timer to Calculate Network Lifetime;
        }
        Else
       Exit Loop;
   Repeat until All Nodes are turned ON
Output Network Lifetime;
```

3.6. The Grid Scan Heuristic Scheduling Algorithm

In this algorithm, the individual sensor inputs like radius of the sensor, time to live and the length and breadth of the rectangular area remains same as the randomized scheduling approach. According to this algorithm, sensors are randomly deployed and the rectangular area is divided into *n* number of grid cells where grid size is given as input parameter. After dividing the whole area in to n number of grid cells our algorithm intelligently finds the set of sensors which belong to particular grid cell. So, for *n* number of grid cells we get *n* disjoint sets of sensors. The allocation of sensors to a particular grid cell is calculated based on coverage mapping i.e. if the center of a sensor is in a specific grid cell then it belongs to that grid cell and cover maximum area in that particular grid cell. In this way, deployed sensors are divided in to disjoint sets. In Figure 3.3, the sensor network consisting of 9 sensor nodes and the rectangular area divided in to 4 square grids.

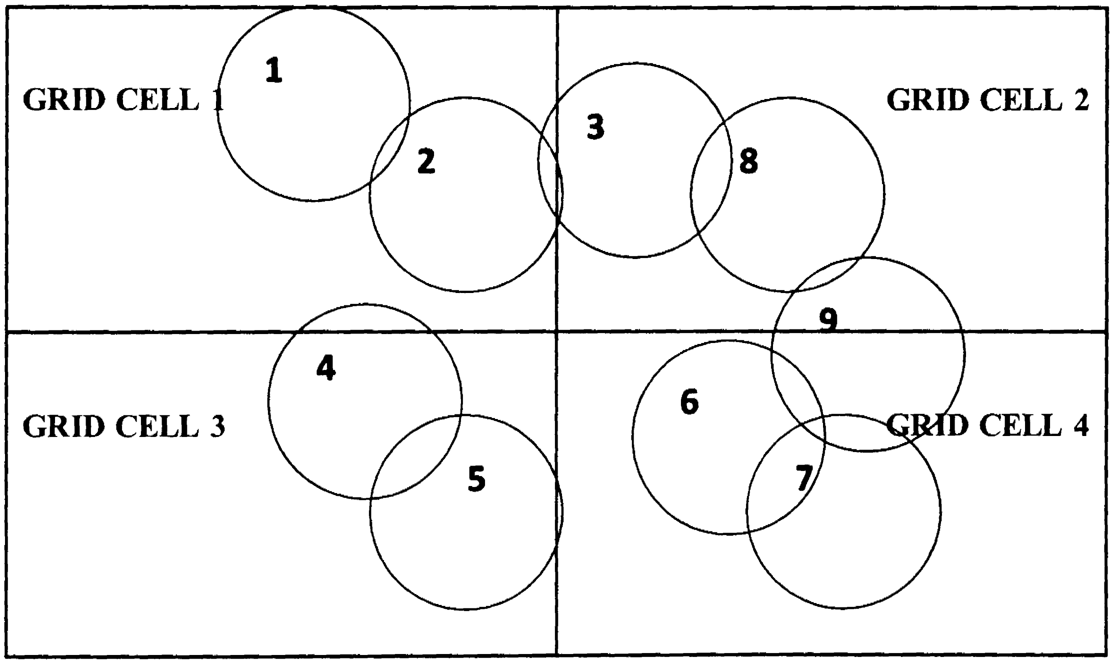


Figure 3.3: Example Showing Grid Division and Allocation of Sensors

In this example based on the coverage mapping scheme we allocate all those sensors into 4 disjoint sets i.e. each grid cell consisting of set of sensors belonging to that grid cell. In this example, the disjoint sets are:

Grid cell 1: {Sensor1, Sensor2}

Grid cell 2: {Sensor3, Sensor8}

Grid cell 3: {Sensor4, Sensor5}

Grid cell 4: {Sensor6, Sensor7, Sensor9}

After allocating all the sensors in to different disjoint sets for each grid cell, sort each disjoint set consisting of sensors whose center co-ordinates (x, y) is known based on the attribute (x^2+y^2). By sorting the sensors based on condition (x^2+y^2) we arrange sensors in each set positioning from left to right.

Now, the sensors in the disjoint sets for each grid cell are in sorted order. After this, select the left most small square grid cell and turn on the left most sensors in the set of sensors for that grid cell. Calculate how much area it is covering in that grid cell, if the sensor that is turned active's coverage is less than the threshold level of coverage in that grid cell then turn the right most sensor active in that grid and again calculate for the coverage, if the calculated coverage is still less than threshold level of coverage then turn second left most sensor active. Repeat this process in each grid cell until we get all the grid cells threshold covered.

In some grid cells, we may not have nodes in the grid's disjoint set to ensure threshold level of coverage. In such cases our approach automatically activates the sensors which are in the neighboring grids. In this way, the whole sensor network is threshold coverage ensured. Also, if the center of nodes position is exactly on the center of the 4 grid cells then our scheduling scheme automatically assigns that to any of the grid cells. Similarly, when the center of the sensor is in between the border of two square grid cells then the sensor may be assigned to any of the grid because it is covering the same area in both the grid cells.

In Figure 3.4, Sensor 1 center is located on the center of the whole rectangular region and sensor covers equal area in all 4 grid cells so our scheduling scheme has the ability to assign this sensor to any of the 4 grid cells. Similarly, Sensor 2 and Sensor 3 centers are located exactly between the middle of the two grid cells so they can be a part of any grid cells because they cover equal area in both grid cells.

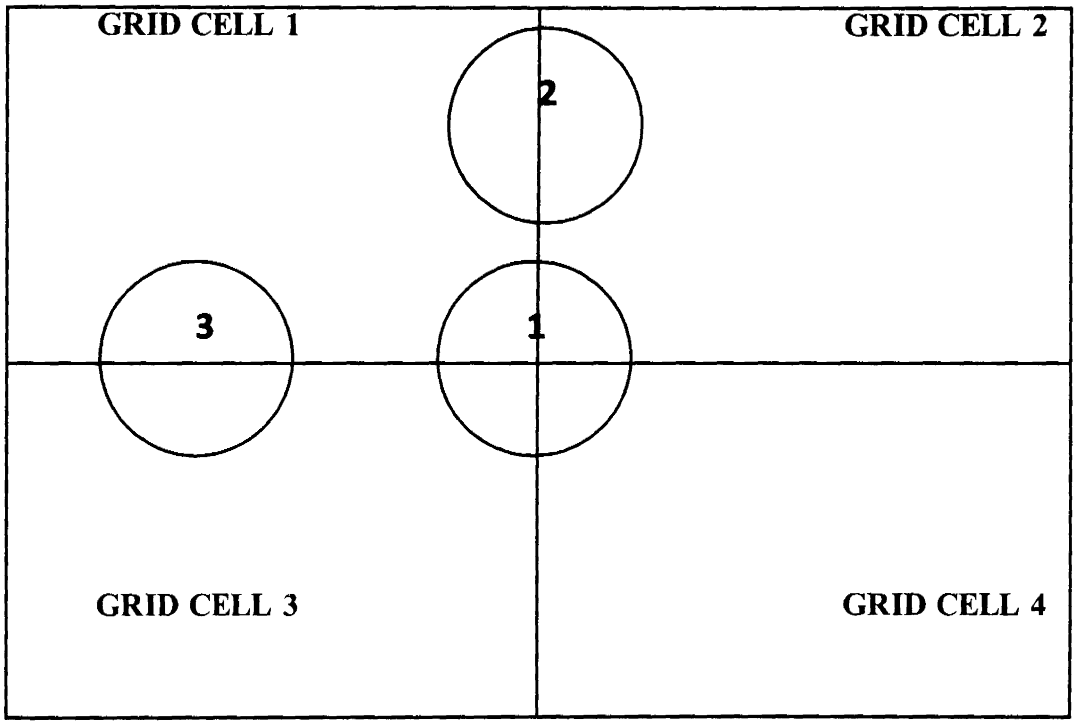


Figure 3.4: Example Showing Allocation of Sensors in Grid Borders

For time cycle t1, the above scheduling process takes place and each grid cell turn on some of its sensors to make sure that it is threshold covered. The total coverage of the network is then calculated and it should be definitely greater than threshold level of coverage. But, due to lifetime constraint of the sensors every sensors end times varies and expires after some time resulting in the decrease of coverage. The coverage percentage of the whole network is calculated every 2 seconds. Whenever the calculated coverage percentage drops below threshold then another time cycle t2 begins.

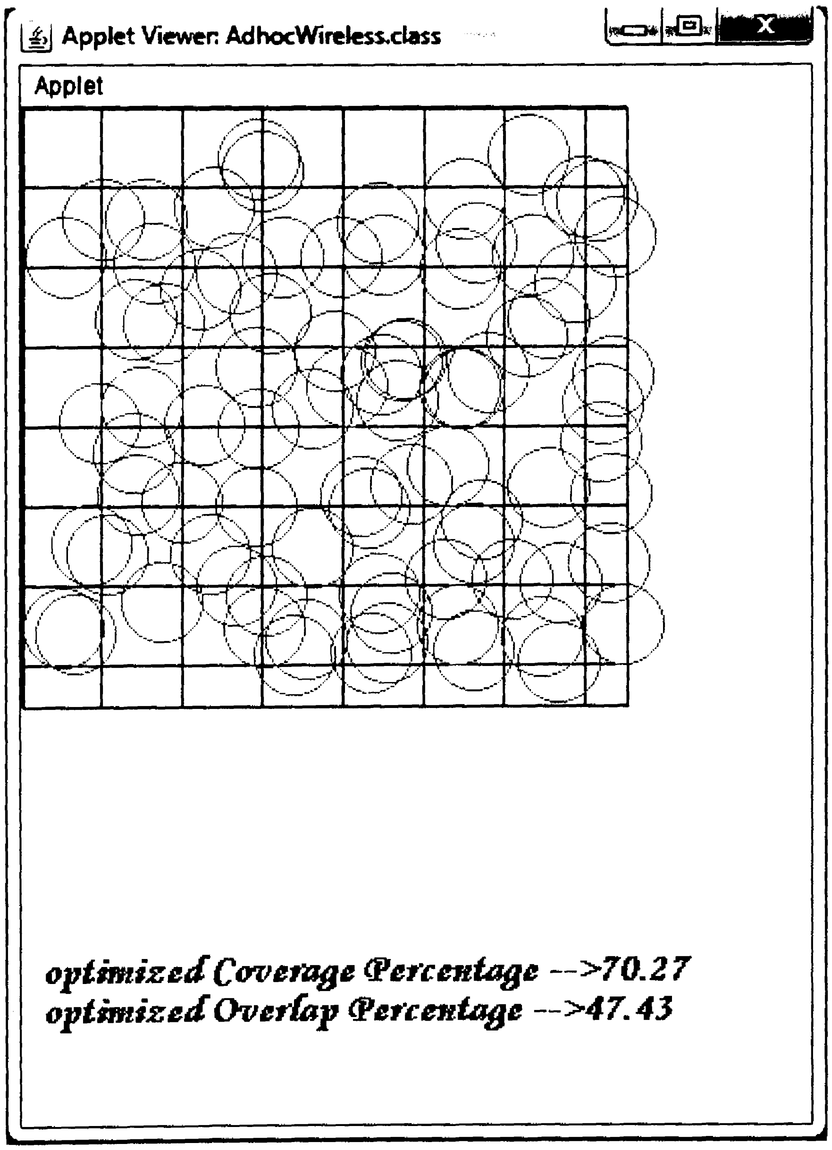
In time cycle t2, the scheduling is done as the same way discussed above but with the sensors that are inactive in each grid cell, those sensors that are not used in the time cycle t1 and are not expired.

So, when the calculated coverage is less than threshold level of coverage then another time cycle t3 starts and the algorithm runs continuously and stops when there are no nodes to schedule and the calculated coverage is less than threshold coverage. When the algorithm stops, timer is stopped and the lifetime of the sensor network is calculated. Since, the sensors are scheduled intelligently and are used in different time cycles the network lifetime is increased substantially and the overall network reliability and the energy consumption by individual sensors is effectively managed.

Consider a sensor network having number of sensors = 200, threshold coverage limit = 70% and the Grid Size = 40 units (double the size of the sensor radius). The Grid scan approach is graphically implemented as a java applet application in Figure 3.4

From Figure 3.5, The Grid Scan approach time cycle t1 implementation is shown and number of nodes that are required for optimized coverage percentage is very less, also the overlap percentage of the sensor used in time cycle t1 is also very less when compared to randomized scheduling approach. By placing the sensors in grids and sorting them according to their position, the overlap percentage is decreased by using the sensors which are less redundant in each time cycle. By calculating overlap percentage the sensors sleep probability can be calculated.

Figure 3.5, also shows the division of grids based on the input parameter grid size given by the user and displays the equal and unequal sized grids based on the input parameter. The coverage percentage and the overlap percentage are calculated for every 2 seconds. The allocation of sensors in to different grids using coverage mapping by taking the number of sensors = 200 is also shown in this figure. The optimized coverage percentage is the coverage percentage of the grid scan approach.



optimized Coverage Percentage -->70.27
optimized Overlap Percentage -->47.43

Figure 3.5: Example Grid Scan Heuristic Implementation

Every Grid employs following steps in each time cycle while scheduling sensors and pseudo code for calculating network lifetime using Grid Scan methodology is explained below:

Pseudo Code for Grid Scan Methodology:

```
Loop
  Loop
   Select Grid Cell;
```

```
Sort and Store Positions of Sensors in Grid Cell Using Condition ($x^2+y^2$);
    Loop
            If (Coverage Percentage < Threshold Coverage)
            {
              Turn Left Sensor Active in Selected Grid Cell;
            }
             Calculate Coverage Percentage Again;
            If (Coverage Percentage < Threshold Coverage)
            {
              Turn Right Sensor Active in Selected Grid Cell;
            }
    Repeat-Until
   (Coverage Percentage > Threshold Coverage in selected Grid Cell OR (All Sensors
are used in selected Grid Cell)
    Select Next Grid Cell;
  Repeat-Until (All Grid Cells are Traversed)
  Calculate Coverage Percentage of Entire Grid Area;
   Loop
    Start Timer to Calculate Network Lifetime for Time Cycle T1;
   Repeat-Until (Coverage Percentage < Threshold Coverage)
Start Another Time Cycle Using Remaining Sensors that are inactive and not used in Time
Cycle T1;
Repeat the whole algorithm (Until all the sensors deployed are used)
```

This algorithm explains the total Grid Scan Methodology in first time cycle t1 and then same method is employed for different time cycles until the entire sensor nodes are used. In each time cycle we turn ON the dominating set of nodes which constitute threshold coverage.

4. METHODOLOGY AND CLASSES

The application developed as a part of this project is a java applet application consisting of an applet displaying both the Randomized Scheduling and the Grid Scan Heuristic approach. Both the algorithms have the same input parameters required to generate a sensor network along with some addition information like coverage width and coverage height of the applet, sensing radius and sensor lifetime information. All these inputs are given as parameters in the Eclipse IDE. Since all the inputs are given at run time, user has the option to change the parameters according to his requirements. The two algorithms which are discussed in this research paper have a combination of various different classes and methods which will be discussed in this chapter briefly.

4.1. AdhocWireless Class

This class is the base class for all functionality in this paper. This class extends applet and establishes applet functionality to our application by calling the method paint (). This class has variables which hold values for sensor radius, sensor lifetime, applet width, applet height, threshold coverage, number of sensor nodes. All objects of the classes that are used in this application are created in this class. The network lifetime calculation is all done in this class by allocating maximum start times and time to live in this class. Randomized Scheduling is implemented in this class by populating the nodes based on the maximum start time and end times. Table 4.1 shows the variables that are initialized in this class. The attributes described in Table 4.1 holds different values like sensor lifetime, threshold coverage limit for calculating network lifetime and number of sensors present in the network. These values are used throughout the algorithm for calculating coverage and overlap percentage.

Table 4.1: AdhocWireless Class: Variables

Variable	Type	Purpose
nodeCount	Int	Stores the number of nodes of the sensor network
nodeRadius	Int	Stores the radius of node which gives the sensing range of the node
Width	Int	Stores the width of the Grid
Height	Int	Stores the height of the Grid
timeToLive	Int	Holds the sensor lifetime i.e. time sensor can live using all the battery power
thresholdLimit	Int	Value of the threshold coverage is stored in this variable

4.2. GraphicDrawer Class

This class is responsible for the user interface generation of the application. The functionality of the methods in this class is to paint all the list of sensors to the screen in specified color. This class use graphics object, radius of the sensor and color as parameters for projecting the sensor network. Table 4.2 shows the method used in this class and its functionality. Table 4.2 describes a method called drawCircleNodes() which is used to generate circular nodes and color the nodes accordingly. When sensors are in active state the color object paints green color to sensor nodes and red color is painted for inactive sensor nodes. This method also draws the grids for Grid Scan scheduling by taking the grid size as input parameter.

Table 4.2: GraphicDrawer Class: Methods

Return Type	Method	Purpose
Void	drawCircleNodes()	This method draws nodes to the screen in specific color.

4.3. NodesDistributor Class

This class is called to generate sensor network randomly by using the input parameters given in the AdhocWireless class. Table 4.3 shows the methods that are implemented in the NodesDistributor class. The main functionality of this class is to distribute nodes randomly and efficiently store the list of active and inactive nodes.

This class generates two lists of sensor nodes. First list of nodes for randomized distribution with random start and end times. Second list of nodes for the Grid Scan approach with start time is equal to 0 seconds and end time equal to time to live. By taking the current time interval as an input the method of this class returns the list of sensors that are active and inactive in that particular interval of time.

The method getDistributedActiveNodes() described in Table 4.3 gets the current active nodes in time cycle t1 and sleep scheduling is done based on the active nodes in particular time interval. This method determines the active nodes in time cycle t1 and returns this list to Grid Scan class.

Table 4.3: NodesDistributor Class: Methods

Return Type	Method	Purpose
List	distributeNodesRandomly()	This method distributes nodes randomly by using all the input parameters and assign start and random times randomly based on the maximum start time and sensor lifetime.
List	getDistributedActiveNodes()	This method takes randomly distributed sensor list and current time interval and returns the list of active and inactive sensors in that particular time interval

4.4. NodeCoverage Class

This class is called from the AdhocWireless in order to check the coverage of the sensor network generated. Table 4.4 shows the methods that are implemented in the NodeCoverage class. The main functionality of this class is to calculate the coverage of network whether each pixel is covered by a sensor node or not. This class also returns a map containing pixel-node mapping i.e. the set of nodes covered for each pixel in the sensor network and another map containing node-pixel mapping i.e. the list of pixels which are covered by a node. Coverage percentage and the overlap percentage of the sensor nodes which are active in Grid Scan Methodology is returned by the nodeCoverage() class described in Table 4.4. The getOverlapbyNodes() method returns the overlap percentage of the entire network.

Table 4.4: NodeCoverage Class: Methods

Return Type	Method	Purpose
List	getNodeCoverage()	This method takes the list of active sensors and returns coverage percentage, overlap percentage and list of pixels covered in the grid.
Hashmap	getOverlapByNodes()	This method returns two maps, the first map containing pixel-node mapping i.e. the set of sensors that cover each pixel and the second map containing node-pixel mapping i.e. the list of pixels that are covered by a node.

4.5. GridScan Class

This method populates all the pixels in to the grid and returns all the grids in the sensor network. The method populatePixelMap() takes grid length and grid breadth as the inputs and determines the pixels belonging to each grid. The method getNodesByGrid() checks the presence of sensor center in a grid and populates the grid with sensors. The main functionality of this GridScan class is to divide the total sensors into *n* disjoint sets of sensors where *n* is the number of grids the whole sensor network is divided. Table 4.5 explains the method getNodesByGrid(), which is used to get all the sensor nodes that are present in each grid and this method return the number of sensors for GridCoverage class which sorts the sensors in each grid.

Table 4.5: GridScan Class: Methods

Return Type	Method	Purpose
Map	populatePixelMap()	For each grid populate the pixels that belong to that grid and then remove the pixels that are covered.
Map	getNodesByGrid()	Checks the presence of sensor center in a grid and populates the grid with the sensor whose center is present in that. This method returns all the sensors in each grid.

4.6. CoverageAlgorithm Class

The main functionality of this class is to sort the sensors in a grid from left to right considering both x-axis and y-axis. By effectively sorting the sensors based on the attribute (x^2+y^2) they are arranged in a queue from left to right in each grid. Table 4.6 shows the only method used in this class and its purpose.

Table 4.6: CoverageAlgorithm Class: Methods

Return Type	Method	Purpose
Map	sortGridNodes()	This method sorts all the sensors in a grid from left to right considering both x-axis and y-axis.

5. RESULTS

This section describes various test cases that are performed with the two algorithms. By changing the three main parameters i.e. Grid Size, Threshold Coverage, Number of Sensor Nodes we determine various results and the performance of the algorithms are compared based on then Network Lifetime calculation.

5.1. Test Case 1: Changing Grid Size and Comparing Both Algorithms

In this test case tests are performed by changing the grid size of the sensor network and keeping number of sensors = 500 and threshold coverage = 70%. In Table 5.1 network lifetimes are calculated using both algorithms that are discussed in this paper. The grid size of the network is varied for all the test cases and lifetime is calculated.

Table 5.1: Test Results for Changing Grid Size and Calculating Network Lifetime

Grid Size	Network Lifetime Using Random Scheduling	Network Lifetime Using Grid Scan Scheduling
10	45	50
20	43	51
30	42	53
40	45	73
50	44	64
60	47	49
70	45	49
80	46	47

Throughout this test case, the number of sensors and the threshold coverage are kept constant and the total area where sensors are deployed is equal to 300×300 i.e. 90000 pixels and sensor radius is equal to 20 units. In order to get the test results in Table 5.1 grid size is changed and network lifetime is calculated.

From Figure 5.1, we can conclude that network lifetime obtained by using Randomized Scheduling is always less than the Grid Scan Approach. We can also see that the Grid Size increase results in increase of lifetime up to some threshold limit and then gradually decreases network lifetime for the Grid Scan approach.

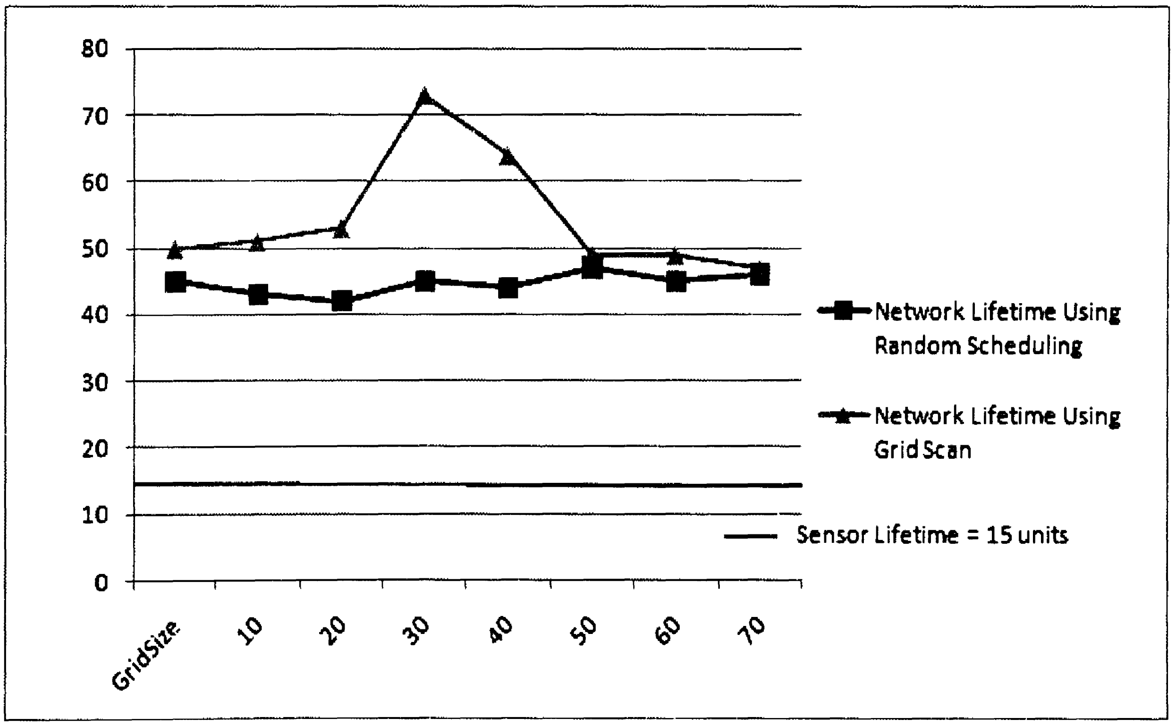


Figure 5.1: Network Lifetime Using Both the Algorithms with a Varying Grid Size

In the Grid Scan Heuristic, when the grid size is equal to diameter of the sensor sensing region then test results experience maximum network lifetime for the Grid Scan approach. For above test case the sensors radius is 20 units and we can see network lifetime increase for the Grid Scan approach when grid size is 40 pixels. The decrease in lifetime after particular increase in grid size is because of the use of more sensors to cover the grids and utilizing more battery power at a time.

5.2. Test Case 2: Changing Threshold Coverage

By changing the threshold coverage of the sensor network i.e. the minimum coverage network should maintain in order to be active, and keeping the number of sensors and grid size constant we compare both the scheduling approaches and compare network lifetimes for different test cases. Some of the constant values used in those test cases are as follows:

Number of test cases = 8,

Total Area = 90000 pixels,

Number of Sensors = 500,

Radius of Sensor = 20 pixels,

Grid Size = 40 pixels

Time to Live = 15 units

These values remain constant throughout the tests for both the algorithms and results are generated. For each test case we get general coverage percentage and overlap percentage. For all these values network lifetime is calculated using randomized scheduling and grid scan methodology and a comparison graph is presented to show how much network lifetime is varying for both randomized scheduling and grid scan methodology in time units.

Table 5.2 shows the calculation of Network Lifetime for Randomized Scheduling and the Grid Scan Heuristic. The lifetime of the sensor network always depends on the threshold coverage because each time network should ensure to reach that coverage level.

Table 5.2: Results for Test Cases Varying Threshold

Threshold Coverage	Network Lifetime Using Randomized Scheduling	Network Lifetime Using Grid Scan
95%	42	59
90%	45	61
85%	44	68
80%	45	74
75%	50	76
70%	53	82
65%	54	83
60%	54	85

From Table 5.2 we can conclude that network lifetime increases for the Grid Scan approach for different values of threshold coverage. We can also see that for more threshold coverage we get less lifetime for both approaches because to maintain more coverage we have to use more sensors to schedule which ultimately results in more usage of power for each time cycle.

By considering the values in Table 5.2 a graph is plotted for comparing the network lifetimes for both approaches. This finding gives how much our heuristic approach improves the lifetime when threshold level of coverage changes. In Randomized Scheduling approach the start times vary and there is no effective scheduling, so more numbers of sensors are used to maintain threshold coverage. The Grid Scan approach improves the lifetime when compared with all the other approaches.

From Figure 5.2, we can see the improvement in network lifetime for the Grid Scan approach compared to Randomized Scheduling and conclude that this heuristic approach is effective and reliable for all test cases.

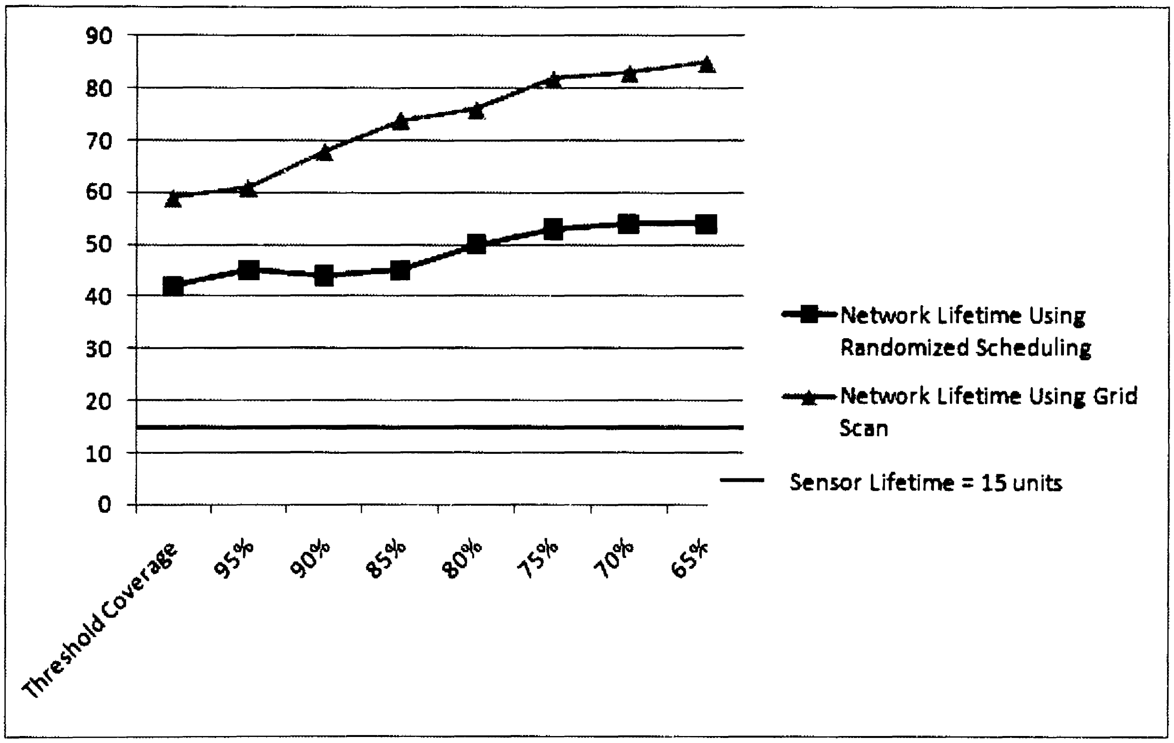


Figure 5.2: Network Lifetime Calculation by Changing Threshold Coverage

5.3. Comparison of Two Algorithms Based on Number of Sensors

In these test cases number of sensor nodes are varied by keeping grid size and threshold coverage constant. We can see from the data in Table 5.3 that, when sensors count is increased gradually, both the algorithms increase network lifetime accordingly. As the sensors count increase the coverage of the total area increases and the network lifetime can be increased. The Grid Scan approach used here shows substantial improvement in the lifetime when compared with other approaches.

Table 5.3: Test Results for Both Algorithms by Varying Number of Sensors

Number Of Nodes	**Network Lifetime Using Randomized Scheduling**	**Network Lifetime Using Grid Scan**
100	31	48
200	41	55
250	44	67
300	45	70
350	47	76
400	49	80
450	53	82
500	54	87

From Table 5.3 we can verify that the network lifetimes are increasing for increasing the number of sensor nodes for both approaches. But we can also state that our Grid Scan approach improves the results by approximately 60% of increase compared to Randomized Scheduling.

The increase in the network lifetime with the increase in number of sensor nodes is due to the availability of nodes for each time cycle. When more the number of nodes available for the sensor network to operate we can schedule them accordingly and can improve network lifetime. Figure 5.3, gives a clear picture of the increase in network lifetimes for both the approaches and compare their output values. The obtained values in Table 5.3 are by changing number of nodes and keeping threshold coverage = 70% and Grid Size = 40.

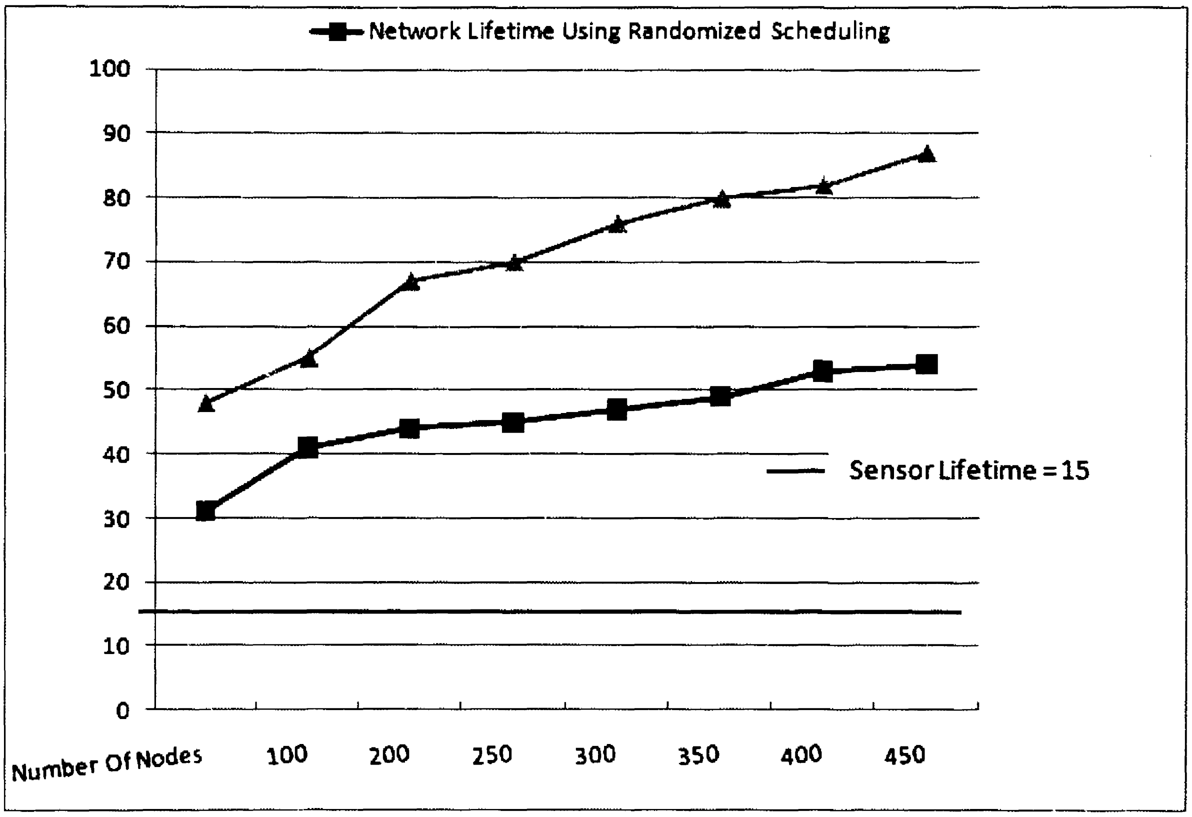


Figure 5.3: Test Results for Varying Number of Sensors

In Figure 5.3, graph is plotted by considering Number of Nodes on x-axis and Network Lifetime on y-axis. This finding shows that, for any given number of sensors, the Grid Scan approach improves the network lifetime approximately up to 70%, a number which is very high for large sensor networks. Figure 5.3 shows there is a considerable amount of increase in network lifetime when compared to randomize scheduling for increasing the number of nodes. Also it can be seen that lifetime increases with increase in number of sensor nodes.

From all the results obtained from subsections 5.1, 5.2, 5.3 we can conclude that the Grid Scan approach of scheduling sensors yields very promising results because of the intelligent scheduling of sensors for every time cycle and increasing the throughput.

In order to test for the consistency of the results, different inputs are given to the two algorithms that are discussed and the results are tabulated in Table 5.4. From the results obtained from various test inputs by varying all the three variables we get different results and the efficiency of the algorithms are calculated.

Table 5.4: Test Results for Various Inputs and Comparing Network Lifetime

Number Of Sensors	Grid Size	Threshold Coverage %	Network Lifetime Using Randomized Scheduling	Network Lifetime Using Grid Scan Scheduling
100	40	70	31	51
200	30	80	39	54
250	10	75	42	73
300	40	85	40	61
350	60	65	44	66
400	50	90	41	61
450	40	70	53	82
500	20	80	48	79
150	30	60	34	56
400	40	70	49	80

All these values are obtained by using some constants throughout the network such as:

Sensing Radius = 20 pixels

Time to Live = 15 units

Maximum Start Time = 100 units (All sensors start within these interval in Randomized Scheduling)

The time units considered in our approach may be days or weeks or months depending on the type of sensor network used.

Our approach for scheduling the sensors is a heuristic approach, so the results obtained by the set of input values are not final values because they are based on the random time estimate which depends on random method generator. In order to get approximate heuristic values the test cases are run many times and the average, standard deviation and variance are calculated for all the iterations considered and then approximate values for the network lifetime is generated. Test results were generated by keeping Grid Size = 10 and the values are taken and tabulated in Table 5.5.

Table 5.5: 10 Test Cases for Keeping Grid Size = 10

Iterations	**Grid Size**	**Network Lifetime Using Randomized Scheduling**	**Network Lifetime Using Grid Scan**
1	10	43	50
2	10	45	52
3	10	42	54
4	10	45	52
5	10	43	49
6	10	44	53
7	10	45	56
8	10	41	52
9	10	43	50
10	10	42	53
Mean	**10**	**43.3**	**52.1**
Standard Deviation	**0**	**1.41814**	**2.079**
Variance	**0**	**2.01111**	**4.32222**

Average and Standard deviation are calculated for those values. 10 iterations for the same input values i.e. Grid Size = 10, Threshold Coverage = 70% and Number of Sensors

= 500 are taken and from the standard deviation calculation network lifetime varies not more than 2 units. For Grid Size = 40, Threshold Coverage = 70% and Number of Sensors = 500 and standard deviation is calculated for all these values. Test results for Grid Size = 40 heuristic is tabulated in Table 5.6.

Table 5.6: 10 Test Cases for Keeping Grid Size = 40

Iterations	**Grid Size**	**Network Lifetime Using Randomized Scheduling**	**Network Lifetime Using Grid Scan**
1	40	46	73
2	40	48	75
3	40	44	76
4	40	44	72
5	40	45	72
6	40	46	70
7	40	46	78
8	40	46	81
9	40	43	72
10	40	44	74
Mean	**40**	**45.2**	**74.3**
Standard Deviation	**0**	**1.47573**	**3.30151**
Variance	**0**	**2.17778**	**10.9**

From Table 5.6, it can be concluded that the heuristics employed in this paper are showing approximately the same results throughout all iterations. Standard deviation calculation shows that there is only 2 seconds difference for 10 iterations that are performed by keeping Grid Size = 40, Threshold Coverage = 70% and Number of Sensors = 500.

Similarly, results for Grid Size = 80 are tabulated in Table 5.7. These results indicate that lifetime of the network decreases when compared to lower Grid Sizes because of the number of sensors that can cover Grid Cell.

Table 5.7: Test Cases for Grid Size = 80

Iterations	**Grid Size**	**Network Lifetime Using [illegible] Scheduling**	**Network Lifetime Using Grid Scan**
1	80	42	49
2	80	42	47
3	80	44	48
4	80	44	48
5	80	43	49
6	80	42	46
7	80	41	47
8	80	44	46
9	80	43	49
10	80	43	48
Mean	**80**	**43**	**48.7**
Standard Deviation	**0**	**1.82574**	**1.25167**
Variance	**0**	**3.33333**	**1.56667**

From all these test results in subsections 5.5, 5.6, 5.7 conclude that the standard deviation value is not greater than 3 units and thus the efficiency of our heuristic approach is estimated.

The test results by changing number of sensors are calculated and 10 test iterations are run for each set of input parameters like Number of Sensors = 100, Threshold Coverage = 70%, Grid Size = 40. Mean, Standard Deviation and Variance are calculated for all those results. From Table 5.7 Standard Deviation calculation we can see that the network lifetime varies less than 2 units for all iterations.

Test results for Number of Sensors = 100 and keeping Threshold Coverage = 70% and Grid Size = 40 is tabulated in Table 5.8. The two heuristic approaches discussed in this paper give approximately constant results for all the iterations. Standard Deviation calculation by varying number of sensors shows that network lifetime varies only 1 time unit from the mean.

Table 5.8: Test Results for Number of Sensors = 100

Iterations	**Number Of Sensors**	**Network Lifetime Using Randomized Scheduling**	**Network Lifetime Using Grid Scan**
1	100	32	46
2	100	34	47
3	100	33	46
4	100	33	45
5	100	32	45
6	100	31	46
7	100	34	47
8	100	33	44
9	100	32	45
10	100	31	47
Mean	**100**	**32.5**	**45.8**
Standard Deviation	**0**	**1.08012**	**1.0328**

From the standard deviation calculation we can see that for 10 iterations the network lifetime deviation is not more than 1 unit. Similarly for same set of inputs and changing number of sensors to 250 the results are tabulated in Table 5.9. Standard Deviation and Variance is calculated for those results.

When numbers of sensors are less, then the percentage of the coverage they accomplish on the sensor network is also less. So, the results for calculating the network lifetime with 100 sensors obviously displays a minimum network lifetime when compared lifetime with more number of sensors.

Table 5.9: Test Results for Number of Sensors = 250

Iterations	Number Of Sensors	Network Lifetime Using Randomized Scheduling	Network Lifetime Using Grid Scan
1	250	44	67
2	250	42	66
3	250	41	67
4	250	43	66
5	250	44	66
6	250	41	65
7	250	42	64
8	250	43	66
9	250	43	64
10	250	44	65
Mean	**250**	**42.7**	**65.6**
Standard Deviation	**0**	**1.1595**	**1.07497**
Variance	**0**	**1.34444**	**1.15556**

Similarly, Test results for Number of Sensors = 500 are tabulated in Table 5.10. This table shows the lifetimes when sensors are very high in number and both the standard deviation and variance is calculated.

Table 5.10: Test Results for Number of Sensors = 500

Iterations	Number Of Sensors	Network Lifetime Using Randomized Scheduling	Network Lifetime Using Grid Scan
1	500	55	87
2	500	57	86
3	500	54	88
4	500	56	90
5	500	57	90
Mean	**500**	**55.3**	**88.5**
Standard Deviation	**0**	**1.159501809**	**1.58113883**
Variance	**0**	**1.344444444**	**2.5**

From all these results in subsections 5.8, 5.9, 5.10 different values of Number of nodes are given as input parameters and 10 test iterations are generated. These values show there is very minimum difference in each case from the calculation of standard deviation. The experimental and statistical analyses' results are summarized as follows:

1) The Grid Scan Heuristic performs relatively better when compared with Randomized Scheduling

2) Greater the Number of Sensors yields increase in Network Lifetime for both approaches.

3) Increase in Grid Size increases the Network Lifetime only up to a certain extent and then decreases later that threshold.

4) Decrease in Threshold Coverage limit increases the Network Lifetime for both Algorithms.

6. CONCLUSION AND FUTURE WORK

In this paper the problem of scheduling sensors is addressed effectively in order to increase the networks efficiency and extend network lifetime. Scheduling of sensors only activates a subset of sensors to maintain some user defined constraints regarding the coverage and increases network lifetime. An approach called the Grid Scan Heuristic is proposed in this paper to schedule the sensors intelligently in the sensor network. In this paper, we analyzed our experimental results with another scheduling algorithm called Randomized Scheduling.

The proposed approach in this paper tries to cover the threshold percentage of region of interest by scheduling the sensors. The main objective of the algorithm discussed is to find the minimum number of subset of nodes which satisfies the desired level of coverage. The Grid Scan Heuristic approach and Randomized Scheduling approach have been designed and developed using a java applet application which gives the coverage percentage and overlap percentage for the set of active nodes that form the entire network. The sleep scheduling schemes are employed in many applications of wireless sensor networks to improve energy consumption and for very long lifetime of sensor network.

Some of the major issues encountered while designing these algorithms is coverage calculation which is done by pixel-pixel mapping and division of total sensor nodes in to subset of sensors for each grid which is done by coverage mapping i.e. checking the center of each sensor with the pixels in the grid and assigning them to the grid.

Further studies can be done on this issue by considering the communication capability as the energy consuming parameter. In this paper, we addressed the problem by considering only sensing capability of the sensor. In order to account for the optimality in our solutions, we plan to compare our approach with the linear integer programming techniques and improve the solution considerably.

REFERENCES

[1] Yardibi, T. (2006). Sleep scheduling for energy conservation in wireless sensor networks with partial coverage. Master's thesis, Bilkent University, Ankara, [Online]. Retrieved on: http://www.ee.bilkent.edu.tr/ytarik/thesis.pdf.

[2] C.T. Vu, S. Gao, W.P. Deshmukh, and Yingshu Li , "Distributed Energy-Efficient Scheduling Approach for K-Coverage in Wireless Sensor Networks," *Military Communications Conference, 2006. MILCOM 2006. IEEE* , vol., no., pp.1-7, 23-25 Oct. 2006.

[3] Fangyang Shen, Chunlei Liu, and Jun Zhang, "A Distributed Coverage-Aware Sleep Scheduling Algorithm for Wireless Sensor Networks," *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, vol., no., pp. 524-527, Apr. 2009.

[4] J. Deng, Y. S. Han, W. B. Heinzelman, and P. K. Varshney, "Scheduling Sleeping Nodes in High Density Cluster-based Sensor Networks," *ACM/Kluwer Mobile Networks and Applications (MONET) Special Issue on "Energy Constraints and Lifetime Performance in Wireless Sensor Networks"*, vol. 10, no. 6, pp. 825–835, Dec. 2005.

[5] Sayed Mostafa Torabi , Mohammad Ali Samadian, "Covering of problem in wireless sensor networks", *Proceedings of the 8th Wseas international conference on Telecommunications and informatics, Istanbul, Turkey*, vol., no., pp.88-94, May. 2009.

[6] Chia-Pang Chen, Cheng-Long Chuang, Tzu-Shiang Lin, Chia-Yen Lee, Joe-Air Jiang, "A coverage-guaranteed algorithm to improve network lifetime of wireless sensor networks", *Procedia Engineering*, vol. 5, no. 8, 2010, pp. 192-195, Jun. 2008.

[7] Ye Fan, G. Zhang, J. Cheng, et al, "PEAS: a robust energy conserving protocol for long-lived sensor networks", *Proc. the 23rd International Conference on Distributed Computing Systems, Rhode Island, USA.* vol., no., pp. 28-37, Mar. 2003.

[8] Chi-Fu Huang , Yu-Chee Tseng, "The coverage problem in a wireless sensor network, Mobile Networks and Applications", vol.10, no.4, pp.519-528, Aug 2005 .

[9] K.E. Nygard, Sunil Reddy Maddi (Mar 2011). "Algorithms for Coverage Improvement in Wireless Sensor Networks", Master's Thesis Dept. of Computer Science, North Dakota State University, unpublished notes.

[10] Xingfa Shen; Jiming Chen; Youxian Sun; , "Grid Scan: A Simple and Effective Approach for Coverage Issue in Wireless Sensor Networks," *Communications, 2006. ICC '06. IEEE International Conference on*, vol.8, no., pp.3480-3484, Jun. 2006.

[11] Isabel Dietrich , Falko Dressler, "On the lifetime of wireless sensor networks", *ACM Transactions on Sensor Networks (TOSN)*, vol.5, no.1, pp.1-39, Feb. 2009.