ISLAMIC UNIVERSITY OF TECHNOLOGY

UNDER GRADUATE THESIS

CELLULAR AUTOMATON BASED NODE SCHEDULING IN WIRELESS SENSOR NETWORK

Authors:

Salman Hossain (114412)

Tousif Islam (114413)

Supervisor:

Dr. Muhammad Mahbub Alam

Professor

Department of Computer Science and Engineering

A THESIS SUBMITTED TO THE DEPARTMENT OF CSE IN ACCORDANCE WITH THE REQUIRMENTS FOR DEGREE OF BSC ENGINEERING IN CSE

ACADEMIC YEAR: 2014-15

OCTOBER 2015

DECLARATION OF AUTHORSHIP

We hereby certify that the work presented here is, to the best of our knowledge and belief, original and the result of our own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

This thesis was not previously presented to another examination board and has not been published.

AUTHORS:

Salman Hossain

Student ID:-114412

Tousif Islam

Student ID:-114413

SUPERVISOR:

Dr. Muhammad Mahbub Alam
Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)

ISLAMIC UNIVERSITY OF TECHNOLOGY

ABSTRACT

Introduction of cellular automata in wireless sensor network (WSN) has a great impact. In fact now a days, cellular automata and its inclusion to WSN is one of the most discussed topic. The main objective of applying cellular automata in WSN is optimization of the coverage and the lifetime in WSN. By Wireless Sensor Network we understand special kind of ad-hoc network where there are good number of sensors with the ability to sense, actuate, compute and communicate with other sensors. Sensor network has a wide range of applicability such as environmental monitoring, energy monitoring, structural health monitoring, machine condition monitoring, transportation monitoring, industrial monitoring etc. In spite of having diversified application, there exist some limitations in WSN such as low computational power, limited power source, reduced bandwidth, complex configuration, reduced network lifetime etc. In this paper, we have tried to identify the problems of existing algorithms, adjust the rules in order to optimize the coverage and the lifetime of WSN furthermore.

ACKNOWLEDGEMENTS

We would like to express our gratitude and indebtedness to those who assisted us, particularly Dr. Muhammad Mahbub Alam, Professor, CSE, IUT - without whose outstanding and generous help in obtaining and providing material, this book would have been hardly possible—A fact which cannot be highly emphasized enough. His patience, motivation and vast knowledge inspired us a lot.

We would also like to give an additional thanks to Md. Sakhawat Hossen, Assistant Professor, CSE, IUT for his encouragement and many useful observation and recommendations.

Last but not the least, we would like to thank our families who supported us wholeheartedly throughout the writing of this thesis and our life in general.

With Regards,

Salman Hossain (114412)

Tousif Islam (114413)

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

A wireless sensor network or WSN is the distribution of autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure etc. and to cooperatively pass their data through the network to a main location. Due to its wide application WSN has attracted much more interest .Usually a WSN consist of nodes - from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. WSN has a wide range of applicability. Depending on the application sensors are deployed randomly. Normally, a large number of sensors are deployed in the environment to sense the data. Since the Sensors are densely deployed, multiple sensors can sense or cover the same region. The sensor spends most of its energy to sense the environment and dies as soon as it loses all its energy. Different techniques have been applied to preserve the energy as long as possible. One of these techniques is to let the sensors sleep for a certain amount of time and let them awake only when necessary. Keeping that in mind we have introduced cellular automaton in WSN to optimize the lifetime and the coverage of a WSN. Different physical systems have been developed using cellular automation. Cellular automaton can be used to design one dimensional or multi-dimensional grid.

1.2 THESIS CONTRIBUTION

Our contributions towards the thesis are as follows:-

We were able to

- Increase the Lifetime of the network.
- Increase the area under the curve

- Synchronization of the utilized energy was done in EEBP/EBP.
- We were able to create room for adjustment according to application needs. That is, by changing the input variable of the simulation environment, it was possible to create different range of coverage, area under the curve and network lifetime.

1.3 THESIS OUTLINE

The remainder of the book is structured as follows. Section 2 presents an overview on wireless sensor networks. Chapter 3 presents the main concepts about cellular automata. Formal definition of coverage, network lifetime and network scheduling and application of CA in WSN is given in Chapter 4 and Chapter 5 shows various kinds of algorithms proposed and their simulations and results. Chapter 6 includes the conclusion and future plans.

Chapter 2

Background

2 WIRELESS SENSOR NETWORK (WSN)

WSNs were initially designed to facilitate military operations but its application has since been extended to health, traffic and many other consumer and industrial areas. A WSN consists of anywhere from a few hundreds to thousands of sensor nodes. The size of the sensor nodes can also range from the size of a shoe box to as small as the size of a grain of dust. As such, their prices also vary from a few pennies to hundreds of dollars depending on the functionality parameters of a sensor like energy consumption, computational speed rate, bandwidth, and memory.

A wireless sensor network consists of three main components: nodes, gateways and software. The spatially distributed measurement nodes interface with sensors to monitor assets or their environment. The acquired data wirelessly transmits to the gateway, which can operate independently or connect to a host system where you can collect, process, analyze, and present your measurement data using software.

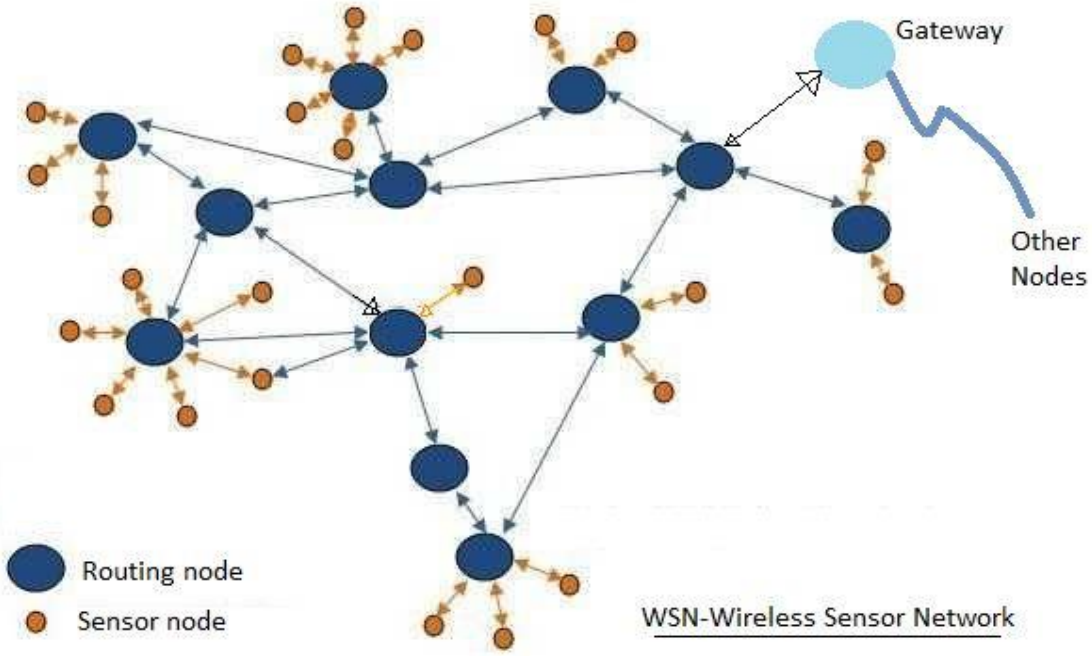


FIGURE 2.1: Wireless Sensor Network

Chapter 3

Background

3 CELLULAR AUTOMATON (CA)

A cellular automaton is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells.

A cellular automation is a model of a system of cell objects with the following characteristics:

- Cellular automaton contains one or multi-dimensional grids while under each grid there are cells.
- The cells live on a grid.
- Each cell has some states. The number of states must be finite for each cell. For example in a 1-Dimensional array a cell may have two states ON and OFF or we can simply say '1' will represent ON state while '0' will represent OFF state.
- Each cell has a neighborhood. The neighborhood can be defined in many ways but usually the adjacent cells are called the neighboring cells.
- The state of a cell is determined by observing the states of its neighboring cells.

A cellular automata can be formally defined as a 4 tuple such that A=(C, S, F, N) Where

- C represents the cells of the cellular grid,
- S denotes the states of the cells,
- T represents the transition rule of the automaton and
- N represents the neighborhood of a cell.

At time t, each cell of C is assigned a state of S. The state of a cell c at time i+1, is determined via the transition function F depending on the current state of c and the states of cells in the neighborhood of c at time t. If all the adjacent cells (maximum 8) of a cell are considered as the neighbors, then this is called a Moore Neighborhood. In a typical cellular automaton, all the cells synchronously verify the states of their neighbors and change their states accordingly.

Chapter 4

Background

4.1 COVERAGE

In wireless sensor network coverage is defined as the area where the nodes can communicate. We can also define coverage as the total area which can be monitored by all the sensor nodes of a WSN. Efficient resource management and providing reliable QOS are two of the most important requirements in sensor networks. However, due to severe resource constraints and hostile environmental conditions, it is non-trivial to design efficient node deployment strategies that would minimize cost, reduce computation and communication overhead, provide a high degree of coverage, and maintain a globally-connected-network simultaneously. Challenges also arise because of the fact that most often topographical information about the monitoring region is unavailable, and that such information may change over time due to the presence of obstacles. Many WSN applications are required to perform certain tasks whose efficiency can be measured in terms of coverage.

4.2 NETWORK LIFETIME

Network lifetime is the time until the first sensor node or group of sensor nodes in the network runs out of energy. It can be also defined as the amount of time that a Wireless Sensor Network would be fully operative. One of the most used definitions of network lifetime is the time at which the first network node runs out of energy to send a packet, because to lose a node could mean that the network could lose some functionalities. But, is also possible to use a different definition, in which some nodes could die or run out of battery power, whenever other network

nodes could be used to capture desired information or to route information messages to their destination.

4.3 NODE SCHEDULING

From the definition of WSN we came to know that WSN contains dedicated sensors to monitor and evaluate surrounding environment. It is often found that two or more sensor nodes are monitoring the same area which is not needed or expected. That is redundancy occurs often in case of WSN where multiple nodes perform the same function .That is not beneficial to the performance of WSN in terms of coverage, efficiency etc. This problem of redundancy could be solved by the introduction of node scheduling. Node Scheduling is the scheduling of the jobs of the sensor nodes in WSN. That is all the nodes will not perform their action at the same time. Jobs will be distributed among the nodes. Some nodes will remain active while other nodes will wait till the active nodes perform their action. Suppose multiple nodes are covering the same area of a WSN. Now if we want to reduce redundancy in that area we can implement node scheduling in that area. If a node finds that its neighbor is sensing and covering the same area as the node then it will go to a stand by state for a particular period. When the neighboring nodes will become inactive or stand by or dead then the node can return back to active state and starts sensing the environment. In this way redundancy will be reduced in an area covered by multiple nodes. Now a question may come how can we design or establish node scheduling in WSN. Introduction of cellular automata can answer this question. By studying cellular automata we came to know that it is a mathematical model that contains grid of cells where each cell is associated with some finite states. Usually the states could be referred as sleep state, awake state or dead state. In a cellular automata state of each cell may change after a definite or random time interval depending on the states of the neighboring cells. State of a cell changes on the basis of different algorithm.

We can consider the example of "Game of Life":
The "Game of Life" is executed in a bi-dimensional grid with cells that can assume two values: 1 (one) indicating

That the cell is alive, or 0 (zero) indicating that the cell is dead. The game's creator, John Horton Conway, established the following rules:

- A cell that is dead on time t comes back to life on time t + 1 if exactly three of the eight other adjacent cells are alive at the same time.
- A cell that is alive on time t dies on time t + 1 if, on time t, less than two or more than three adjacent cells are alive.

The "Game of Life" follows a specific algorithm. We can also define other algorithm for determining the states of cells. For Example:

- In a square grid systems, a cell that is in sleep mode at time t will go to awake mode at time t+1 if less than 1 neighboring cells are in sleep state.
- A cell that is in awake mode at time t will go to sleep mode at time t+1 if 1 or more than 1 neighboring cells are in awake.

Now if we apply Cellular Automata in WSN then redundancy of nodes will be reduced or completely vanished. That is two or more nodes will not monitor the same area at the same time. This is because if a node finds that its neighboring cell is monitoring the same area it is supposed to monitor then it will stop its functionality and let its neighbor to monitor the area. In this way there will be no redundancy and there will be maximum coverage.

4.4 RADIUS 1 NEIGHBORHOOD:

Radius 1 Neighborhood is the system in which each sensor can have a maximum of one node at its top, bottom, left, right and corners. That is in case of a 3*3 square grid system the central node will have maximum 8 neighbors at its surroundings when we will consider neighborhood of radius 1.

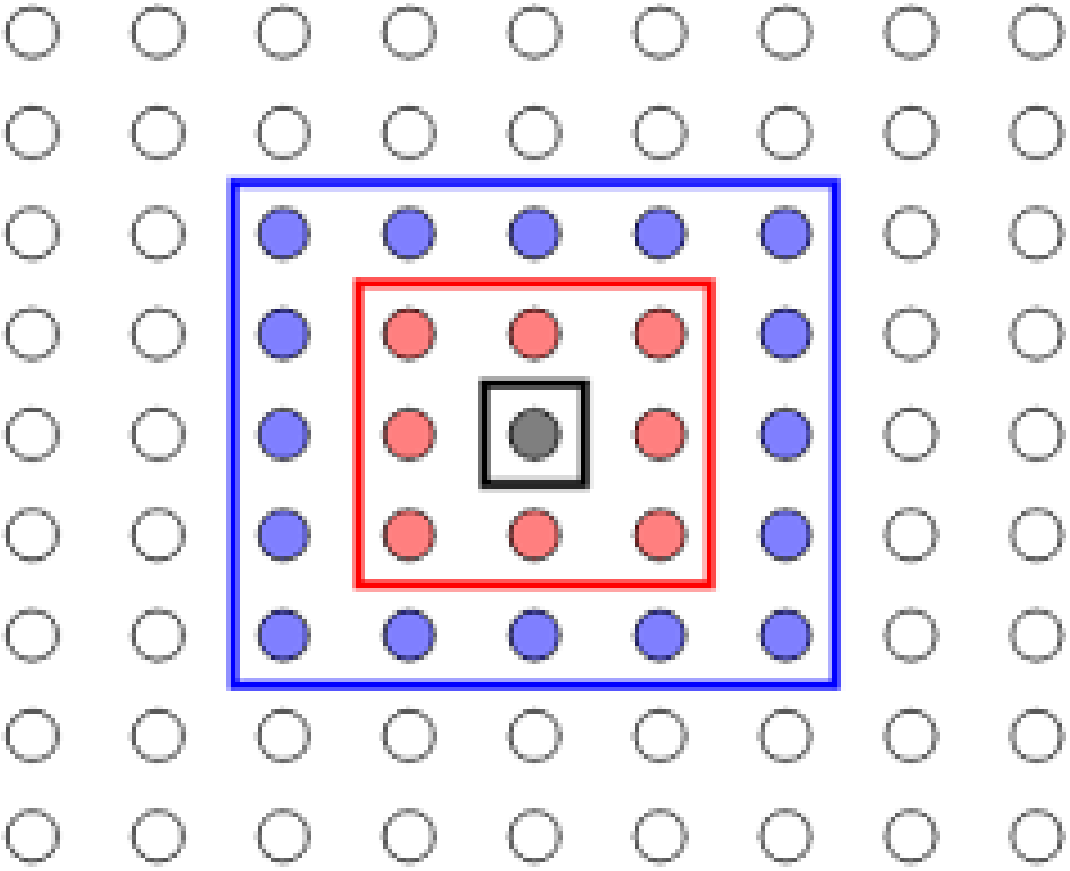


FIGURE 4.1: Square under red line is representing a 3*3 square grid and grey node is representing the central node

Chapter 5

SIMULATION AND RESULTS:

We have developed a simulator in JAVASCRIPT to compare the different rules of cellular automata of our algorithms. Initially, we have considered a 2-D grid of size 150 (150 X 150) and consider a network of 22500 sensors. Afterwards we have continued our work with hexagonal grid of size 150.

In our algorithm, we have used 0.8 J energy as the initial energy of each sensor. We have used 0.0165 J as the awake value (the amount of energy spent by a sensor in the awake state) and 0.00006 J as the asleep value (the amount of energy spent by a sensor in the asleep state). Afterward we have simulated our system applying different algorithm. Detailed descriptions of those algorithms are sectioned below.

5.1 CUNHA ET.AL ALGORITHM

Initially we have simulated our system using Cunha's Algorithm. Cunha's Algorithm was applied in square grid and follows following steps:

- A cell that is in sleep mode at time t will go to awake mode at time t+1 if less than 1 neighboring cells are in sleep state.
- A cell that is in awake mode at time t will go to sleep mode at time t+1 if 1 or more than 1 neighboring cells are in awake.
- Initially each sensor will be given a random amount of time t and this value of t decrements. Next status of a sensor will be determined when the value of t is 0.

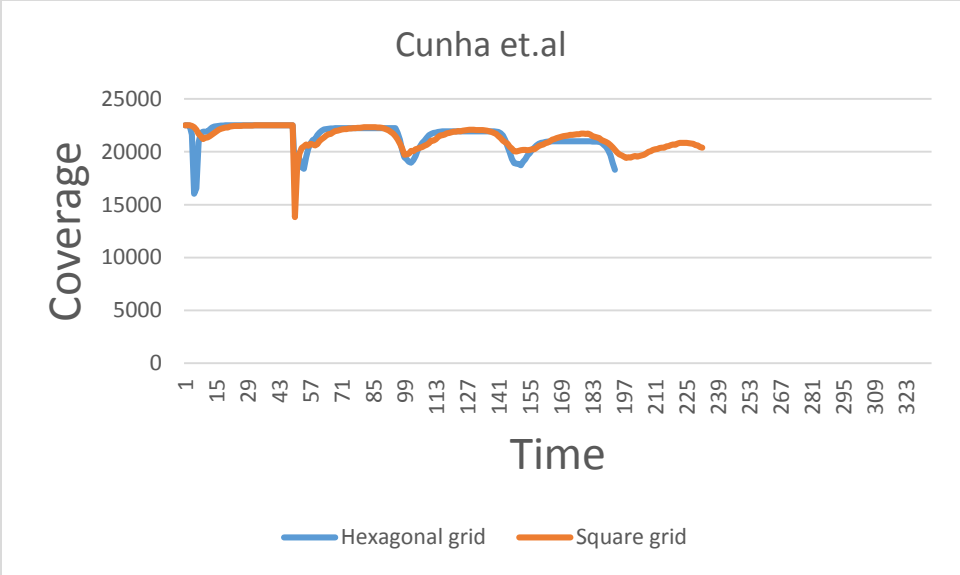


FIGURE 5.1: Lifetime vs Coverage graph for Cunha et.al Algorithm (Until 75% sensor nodes death)

OBSERVATION:

Initially lots of unnecessary energy consumption occurs due to making all the sensors awake. Produces odd spikes time to time. Spikes generate as cunhas' rule allows some sensor node to be awake for rest of their lifetime, allowing so, significant amount of sensor nodes die nearly altogether in turns after few iterations. Required to modify the algorithm to remove those spikes and ensure an overall consistent performance.

5.2 CHOUDHURY ET.AL ALGORITHM

The sensors can deplete their energy too early if they communicate with their neighbors in every time period. For this reason, sensors were selected randomly for communication with their neighbors .A number was drawn between 0 and 1 for every sensor and if the number is smaller or equal to 0.2, then that sensor was allowed to communicate with its neighbors and determine its next status.

The Algorithm follows following procedure:

- Initially, a small number of sensors were allowed to be awake. We pick a random number between 0 and 1 for each sensor. If the number is less than 0. 1 then the status of the sensor will be awake otherwise asleep. If we start with more than 10% sensor awake then initially we may obtain a good coverage but after a certain time there will be a sudden fall of coverage. Selecting a small number of sensors, does not solve this problem entirely. That's why we have applied a probabilistic technique to solve the problem afterwards.

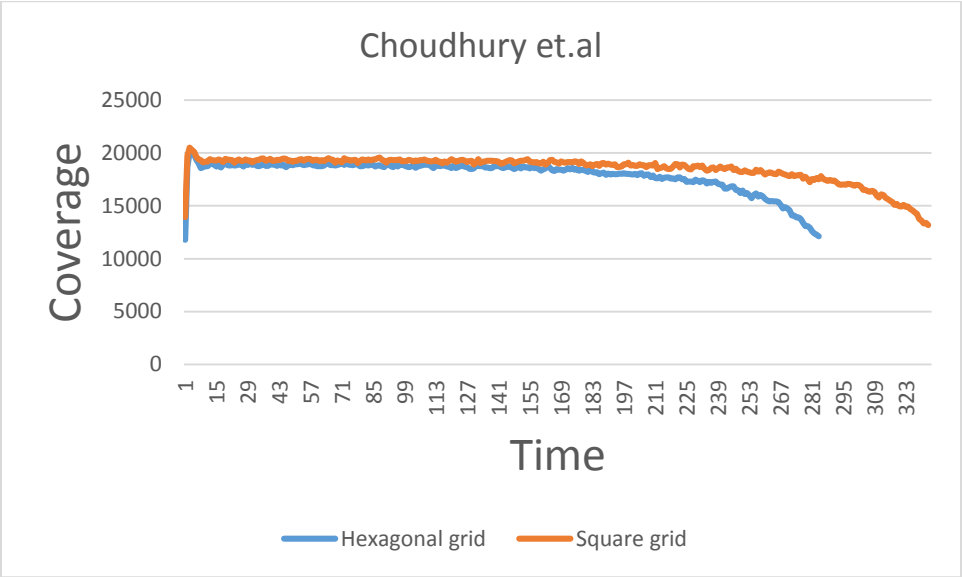


FIGURE 5.2: Lifetime vs Coverage for Choudhury et.al Algorithm (Until 75% sensor nodes death)

- Instead of timers used in Cunhas' algorithm, sensors are selected probabilistically with 20% probability to verify their neighbors.
- In the next step we have forced some sensors to change its status form awake to asleep if the sensors were awake for longer periods. Sensors were selected probabilistically to change their status from awake to asleep state. We set a threshold value 0 to each sensor when it is in the asleep state. We call this as the probability of going to the asleep state. Whenever any sensor changes its state to the awake state this value starts to increase and each time period (as long as it remains in the awake state) it increases by 0.05. So at each time period, we randomly select any number between 0 and 1 for each

awake sensor and if the number is smaller or equal to this threshold then we forcefully set the status of the sensor to the asleep state.

- A cell that is in sleep mode at time t will go to awake mode at time t+1 if less than 1 neighboring cells are in awake state.
- A cell that is in awake mode at time t will go to sleep mode at time t+1 if 1 or more than 1 neighboring cells are in awake state.

OBSERVATION:

This algorithm produces better area under the curve along with network lifetime, not to mention less spikes which is an important factor - as all service providers thrive for consistent performance.

5.3 SEPARATED NEIGHBOR RULE

In the next step, we explain our attempts at producing better algorithms for flexible and efficient performance. All our upcoming experiments concern square grid structure. We start with separated neighbor rule.

We divided the neighbors of a sensor into two groups each containing 3 or 4 members for the case of hexagonal or square grid structure respectively. Here different combinations of group members have been taken. Afterwards the algorithm will work as follows:

- If a sensor is awake and at least one sensor of each neighbor group remains awake then the sensor will go to asleep state otherwise not.
- If a sensor is asleep and no sensor among its neighbor is awake then the sensor will go to awake state.

It is possible to take one of several different combinations to create two groups with each consisting of three/four neighboring cells. It doesn't actually create that much of a significant

difference, so we are providing one graph which will approximately represent as a general, all the combinations.

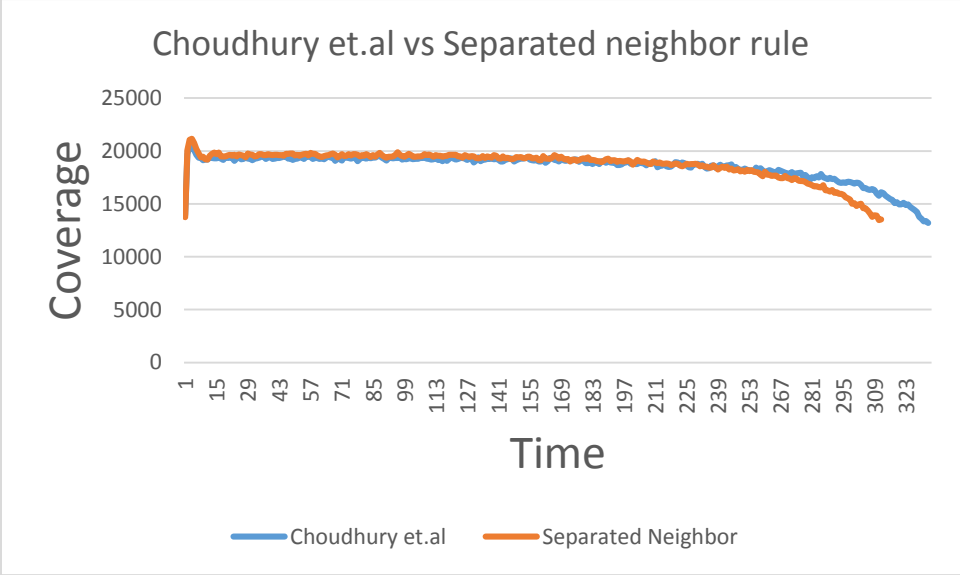


FIGURE 5.3: Performance of separated neighbor rule in comparison to Choudhury et.al algorithm (75% network lifetime)

OBSERVATION:

Choudhury et.al algorithm provides a bit more network lifetime but to compensate for trade-off, it has to let go better average coverage. Separated neighbor rule competes well with Choudhury et.al. algorithm, but still not good enough to be distinctively significant.

5.4 ENERGY LEVEL RULE

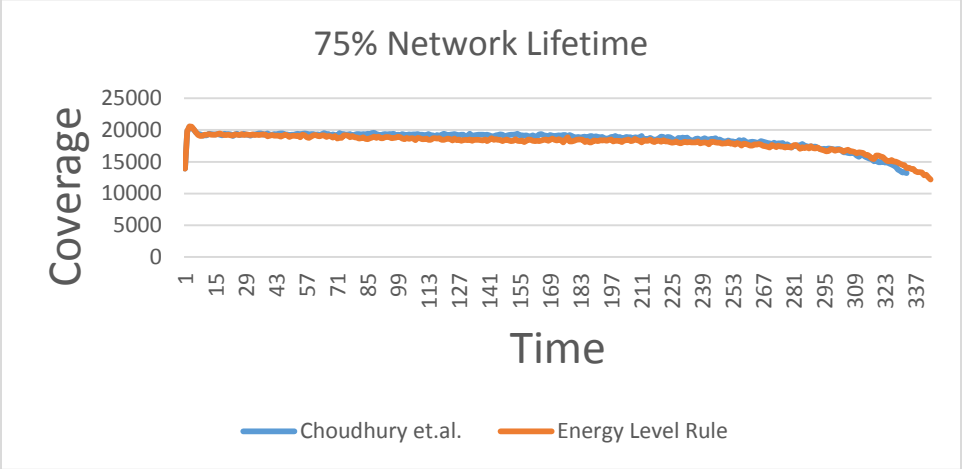
While applying the energy level rule, we applied all the rules mentioned in Choudhury et.al. algorithm. Additionally, we differentiated 3 different energy levels where the sensor node can reside in according to their residual energy. Assuming, initially all the sensors are charged with .8J, energy level division would be like following -

1. Level 1 : 0 to <0.266

2. Level 2 : .266 to <0.5322
3. Level 3 : .5322 up to 0.8

Additional Rule:

- Along with other constraints, an awake state sensor can't stay awake if its energy level is lower than any of his neighboring cells sensor. Although, one thing is left loose in this consideration, sensor nodes can't always be aware of their neighbors' exact energy level, as their neighbor wouldn't be available for communicating all the time. So, we are assuming a hypothetical scenario where all the sensors are able to capture the information of each neighbors' energy level.



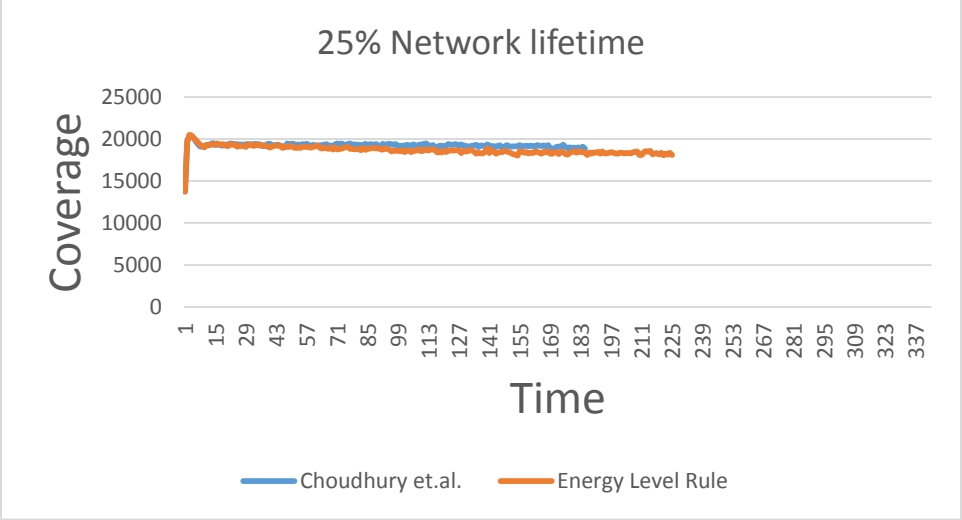


Figure 5.4: Choudhury et.al. vs Energy level rule (75% and 25% network lifetime respectively)

OBSERVATION:

Energy level rule implementation achieves success in gaining a longer lasting lifetime, although certainly after trading its average coverage in each iteration. Performance in 25% network lifetime provides us with greater hope for energy level rules' feasibility, as it is gaining significant amount of additional lifetime than Choudhury et.al. which contributes in better performance in overall area under the curve evaluation. This result shows promise that energy level rule implementation will bring more balance in the utilization of energy in the sensor nodes – thus a greater lifetime.

5.5 FOUR (4) GROUP ALGORITHM

Below there is a figure of a regular node surrounded by eight neighbor nodes.

a	b	c
d	**x**	e
f	g	h

FIGURE 5.6: Regular node consisting of eight neighbor nodes surrounding it.

Now for each sensor (i.e. each cell) divides its neighborhood in four groups. For example, the sensor X in the figure has the following groups

- Group 1: a, b
- Group 2: c, d
- Group 3: e, f
- Group 4: g, h

The algorithm works as follows:

- Initially 10% sensors are randomly chosen as awake sensors. All others remain asleep.
- Each sensor probabilistically (with p=0.2) checks all its four groups.
    1. If the sensor itself is awake and finds that at least one sensor from each group is awake, it will become asleep, otherwise remains awake.
    2. If the sensors itself is asleep and finds that at least one sensor from each group is awake, then it remains asleep otherwise wakes up.
- Forced timer option is also included, that is to say every node that is turned on will start increasing a threshold value by .05 in each iteration. While increasing the value, we check whether it is greater than a random value (random value is generated at that instance, range being from 0 to 1). If it is, it is forced to sleep, else not.

FIGURE 5.7: Network lifetime until 1 dead node

FIGURE 5.8: Network lifetime until 10% dead node

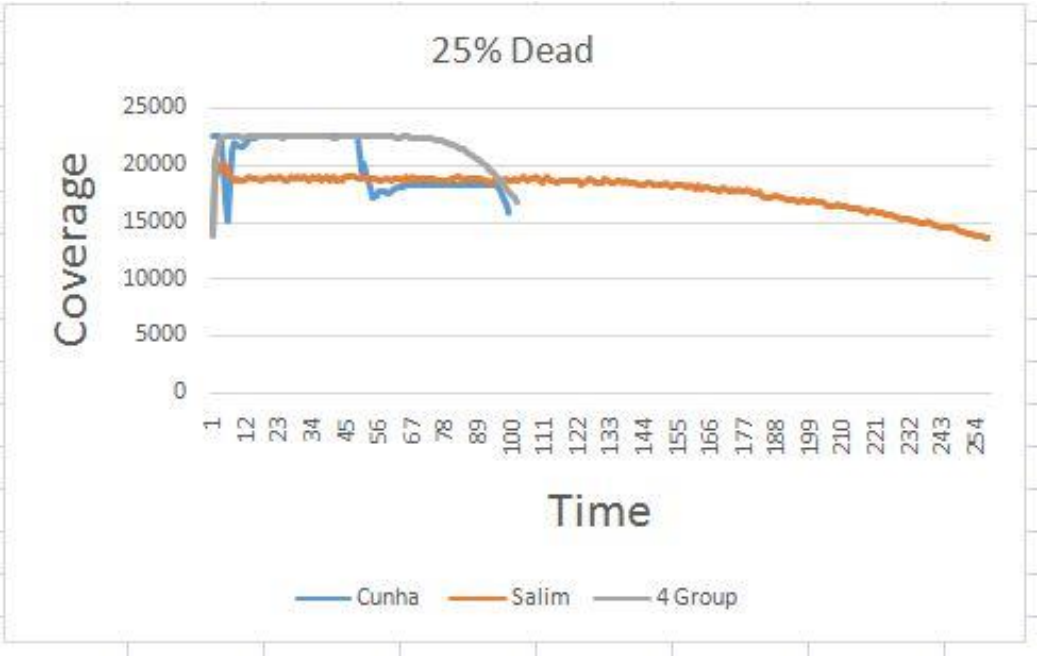


FIGURE 5.9: Network lifetime until 25% dead node

FIGURE 5.10: Network lifetime until 50% dead node

FIGURE 5.11: Network lifetime until 100% dead node

OBSERVATION:

When the network is active until 1 sensor is dead the coverage is pretty high and network lifetime stays alive for a considerable amount of time.

On the other hand, when the network lifetime is considered to be alive until 10% nodes are dead, though the average coverage is high, network lifetime is reduced. Similarly for network lifetime until 25%, 50% and 100% sensor nodes are dead, network lifetime is reduced respectively - the coverage remaining high in all the cases.

If the network lifetime is considered to be active until 1 sensor node is dead, we can observe that the 4 group algorithm works like a charm. Although lifetime is compromised a bit, it provides more average coverage. So for the above mentioned definition of network lifetime, Four (4) group algorithm could easily be preferred over cunha and Choudhury et.al. algorithm. As the definition of network lifetime evolves and allows more sensors to be dead to reach its end state, 4 group algorithms' performance doesn't continue its previously set promise, as the lifetime is compromised drastically – so, not a suitable option for a lengthy amount of network lifetime definition.

5.6 ENERGY BALANCING PROTOCOL

Algorithm Procedure:

- Around 10% of the sensors are awakened in the beginning, choosing this 10% is a random process.

Further Discussion:

Instead of awakening all the sensor nodes in the beginning, awakening 10% of the nodes provides around 60% coverage - that is 60% of all the node are covered at start. Which means that each node is covering around 6 sensor node areas out of possible 9. If we increase this number, we get more coverage in the beginning, but each node covers less areas in average, and the case is not also satisfactory in terms coverage when we decrease this number.

- Each node keeps a record of energy level values of its neighbors. In each iteration, every active/awake node checks their recorded values with other active/awake neighbor nodes' kept record and update their memory. Additionally, 20% of the nodes are selected probabilistically to check their neighbor, if such a node is selected and is currently at the sleeping state, it still communicates with the awake sensors and updates its memory.

Further Discussion:

Energy values can't possibly increase, so if a sensor node has kept a record of its neighbor node, then communicates with an active node, then finds that the latter one has kept a record of that same neighbor node, but the value is lower, then the former node accepts that value as the latest update, and changes its kept record to new value.

- A sleeping sensor - after being probabilistically selected for checking, checks the number of active sensors in the neighborhood. If the count is less than one, it suffices the initial condition for waking up. It then checks its memory of neighbors' energy level whether any of the values are greater than one plus its own energy level. If not, it wakes up. Otherwise, the transition procedure is blocked.

Further Discussion:

A sleeping sensor wakes up if none of its neighbor node is awake. As a result, if it wakes up, it generally covers nine node areas. Such instances assure complete utilization of energy and overall a balanced coverage versus lifetime scenario. Although, instead of node, allowing up to 1 or 2 sensor nodes in the neighborhood to be alive as a condition for a sleeping sensor node to be alive ensures more average coverage per iteration, but through the cost of network lifetime. On other note, this is just the initial primary condition that is mandatory for the transition. It also needs to ensure that no other neighbor node has more than a certain amount of energy compared to primary sensor nodes energy. If any of the neighbor node has extra energy, it means that primary sensor node considers that it has used his energy enough and it's better if

that node/one of those nodes wakes up. In a hope of waking up that specific neighbor node, it decides to stay asleep, which gives the neighbor node staying in higher energy level a chance to wake up.

- An awake sensor, if chosen probabilistically, tries to communicate with neighbor nodes to establish the number of awake neighbor node, if the number is greater than or equal to 1, it goes to sleep. Before going to sleep, it checks in his memory the number of neighbor nodes that is in higher energy level than its energy level. After summing up those differences, it multiplies the value with .05, and sets it as its threshold value. It starts with this threshold value the next time it awakens.

Further Discussion:

Use of higher level energy's influence in threshold value provides more balance in the synchronization of energy utilization. If a primary sensor node starts with an increased threshold value than zero in the beginning, it will have more probability to be enforced an early sleep. Higher number of higher level neighbor nodes existence increases this threshold value accordingly, thus reduces primary sensor nodes probability of being in active period and using energy. Energy utilization is synchronized even more.

- Forced timer rule: Sensors start with a threshold value of 0 (>0 in some other case as mentioned above) when they wake up. It increases .05 in each iteration. A random value is taken each iteration, from 0 to 1, if the threshold value measured in that moment exceeds the randomly generated value, it is forcible made asleep:

Further Discussion:

In general case, where there are no other concerns, sensor nodes wake up with a threshold value of 0.It increases .05 each iteration - through which it gains more probability to go to sleep. A random value - from 0 to 1 is taken and compared with that threshold value each iteration, the higher the threshold value - it is more likely to be greater than that random value - generated at that instance. And as explained in the previous point (d) - a sensor node, before

going to sleep after fulfilling the initial condition, accumulates the higher level sensor nodes energy level differences from its own, and starts with a threshold value next time with a .05 multiplied value of that.

SIMULATION:

Algorithm variations:

- Variation 1 - Regular algorithm as mentioned above.
- Variation 2 - Sleeping sensors initial condition to awake is to find that less than 2 of its neighbor nodes are awake.

Experiment 1:

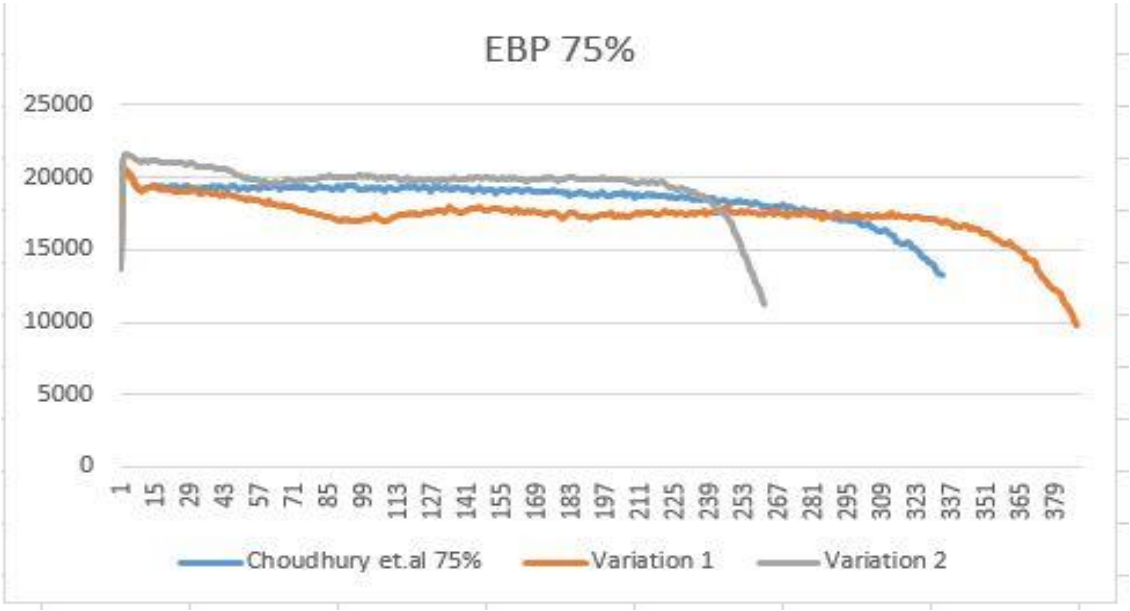


FIGURE 5.12: Energy Balancing Protocol (Until 75% sensors are dead)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	18509	333	6163723
Variation 1	17297	388	6711359
Variation 2	19609	261	5118025

FIGURE 5.13: Output Data for Energy Balancing Protocol (Until 75% sensors are dead)

OBSERVATION:

Comparing with Choudhury et.al algorithm, we propose 2 alternatives. Variation 1 provides balanced coverage and for a longer period of run. As we can observe the figure in the middle, it provides less coverage and continues to provide more or less the same even after the period when Choudhury et.al algorithm runs out. Applications, where longer runtime is essential, this can be beneficial, as the coverage remains balanced. For even coverage service, we propose Variation 2 Where average coverage increases a lot, compromising some of the lifetime period. All these algorithms provide a better average coverage along with the Variation 1. Variation 2 is for the purpose of providing vast and enhanced average coverage to keep the security system more intact.

Experiment 2:

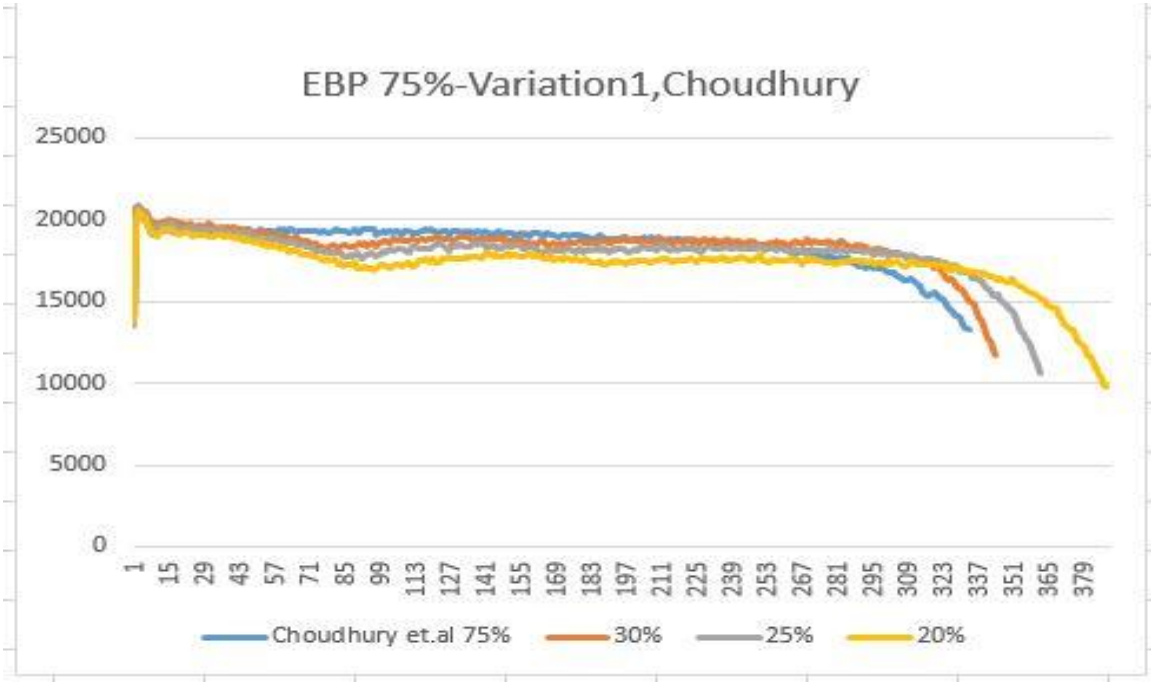


FIGURE 5.14: Variation 1 Alternatives (Until 75% sensors are dead)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	18509	333	6163723
20%	17221	388	6705188
25%	17979	362	6508749
30%	18449	344	6346596

FIGURE 5.15: Output Data for Variation 1 Alternatives (Until 75% sensors are dead)

OBSERVATION:

As the variation 1 keeps the curve below the Choudhury et.al algorithm in the ongoing middle stage, we propose more checking. Instead of the regular 20% probability, we increase that

value to 25% and 30%, so that the coverage difference in the middle stage overcomes along with enough lifetime period.

Experiment 3:

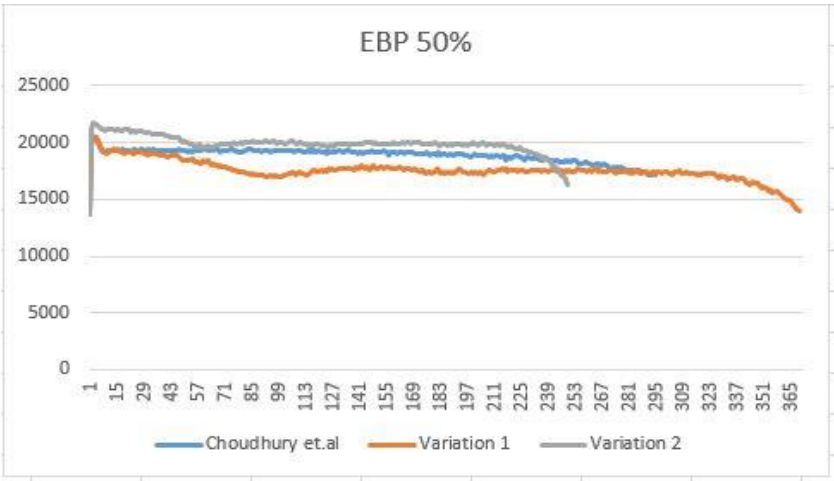


FIGURE 5.16: Energy Balancing Protocol (Until 50% sensors are dead)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	18895	295	5574302
Variation 1	17542	370	6490839
Variation 2	19897	249	5954462

FIGURE 5.17: Output Data for Energy Balancing Protocol (Until 50% sensors are dead)

OBSERVATION:

50% death denoting the end of lifetime - lets us observe even greater difference with Choudhury et.al algorithm.

All the impacts are achieved like in the 75% lifetimes' observation, with more success as the average coverage differentiating even more. Lifetime period is also getting in touch with

Choudhury et.al algorithm, with Variation 1 providing still more iterations and variation 2 nearly the same.

Experiment 4:

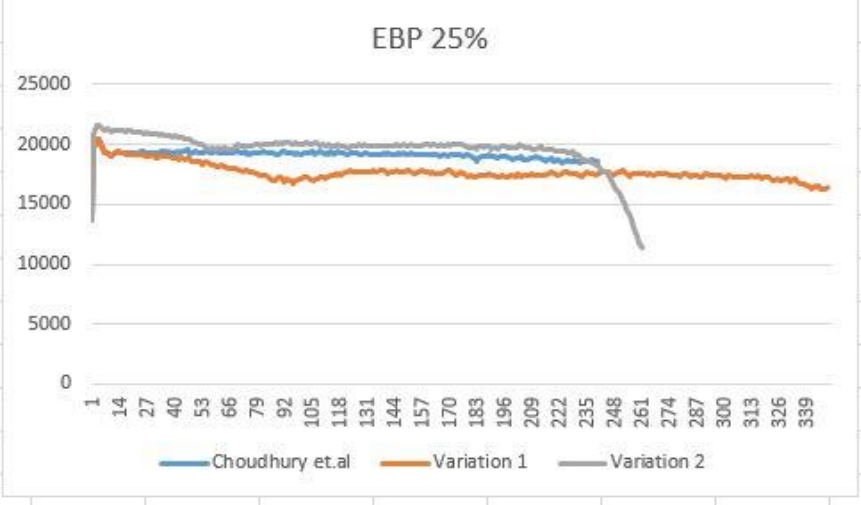


FIGURE 5.18: Energy Balancing Protocol (Until 25% sensors are dead)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	19131	240	4591454
Variation 1	17686	349	6172533
Variation 2	19613	261	5119238

FIGURE 5.19: Output Data for Energy Balancing Protocol (Until 25% sensors are dead)

Experiment 5:

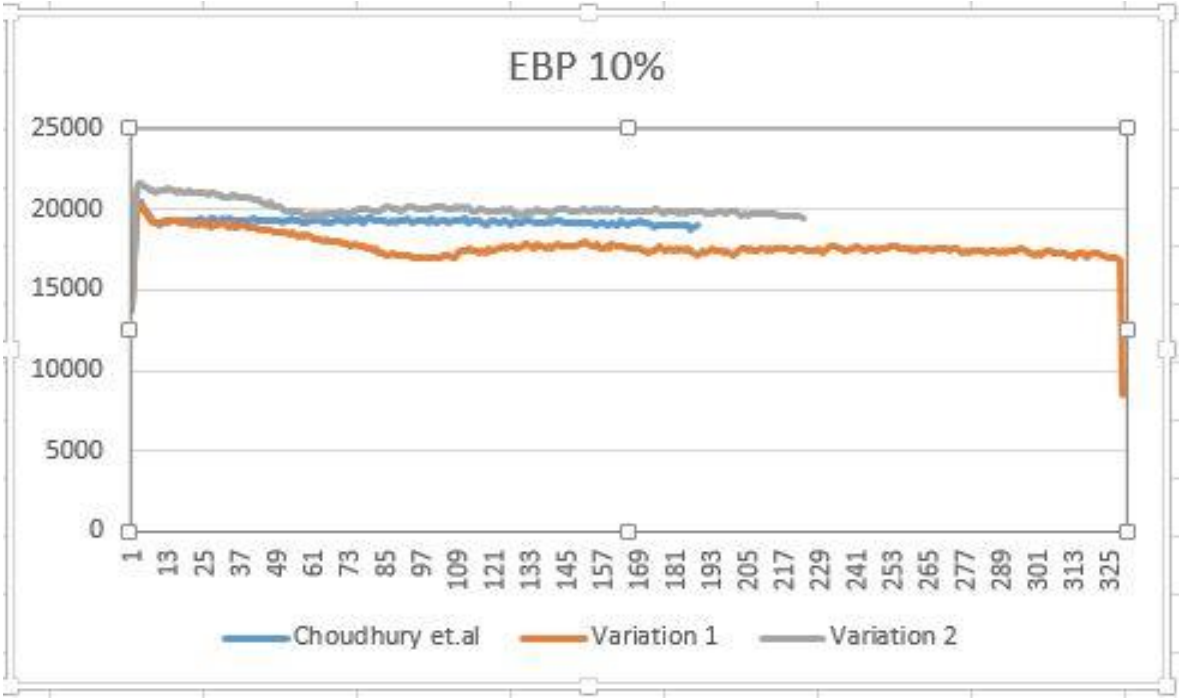


FIGURE 5.20: Energy Balancing Protocol (Until 25% sensors are dead)

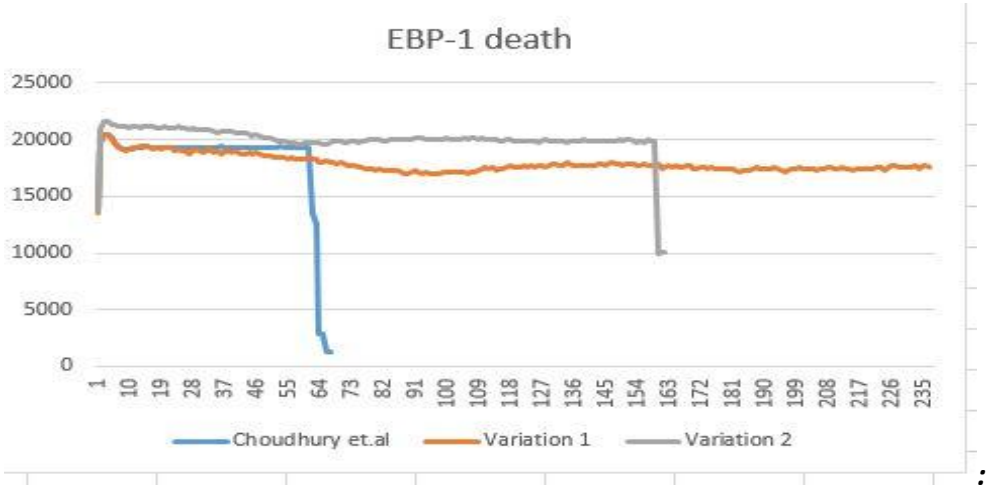
	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	19225	188	3614302
Variation 1	17712	329	5827455
Variation 2	20071	223	4475883

FIGURE 5.21: Output Data for Energy Balancing Protocol (Until 10% sensors are dead)

OBSERVATION (Experiment 4 and 5):

For the case of 25% and 10% - we observe that, the difference in lifetime period between proposed algorithms and Choudhury et.al one is growing larger. Good average coverage is not compromised as expected.

Experiment 6:



:

FIGURE 5.22: Energy Balancing Protocol (Until 1 sensor is dead)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	18064	67	1210304
Variation 1	17879	237	4237392
Variation 2	19993	162	3238890

FIGURE 5.23: Output Data for Energy Balancing Protocol (Until 1 sensors is dead)

OBSERVATION:

We have averaged around 20 to 30 simulations and result is incorporated in the graph. Since 1 death marks the network lifetimes' end, different simulation end in different iteration. The variation is comparatively large for Variation 1, Variation 2 and Variation 3 algorithms rather than Choudhury et.al one. Averaging the result provides us with the spikes, as some of the simulations don't contribute in the graph. Nevertheless, all the proposed algorithms run with solid average coverage and end (each of the simulation individually) after around completing 3 more cycles than Choudhury et.al one. Although averaging them causes the end iteration provide a lot less coverage, but if we simulate any of the algorithm individually, it will still provide greatly enhanced lifetime not to mention the fact - a solid average coverage.

Experiment 7-9:

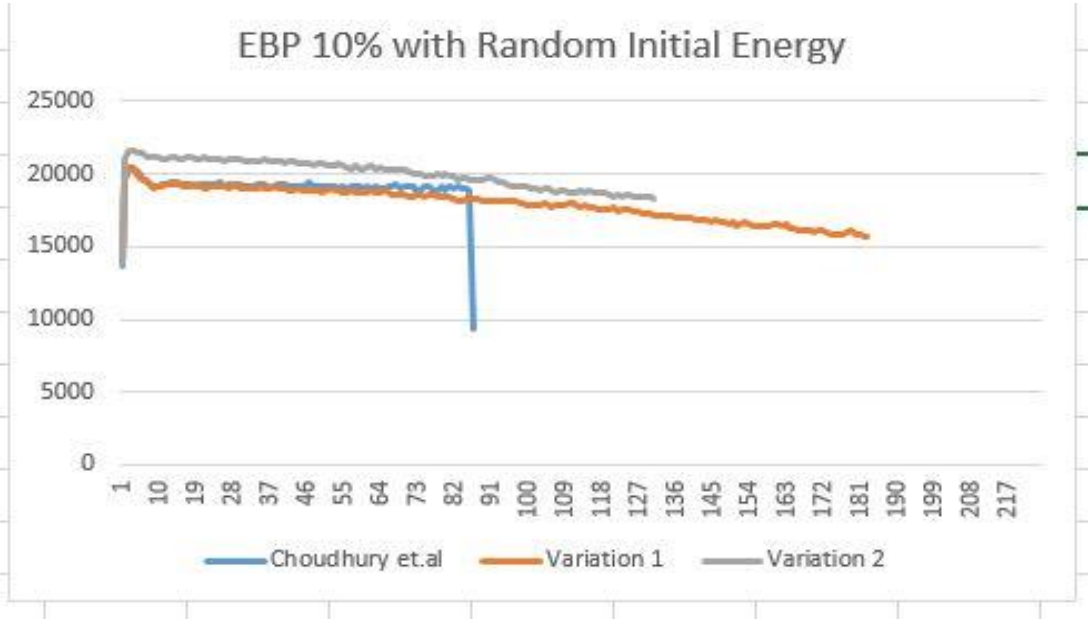


FIGURE 5.24: Energy Balancing Protocol with Random Initial Energy (10% death)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	19045	87	1656928
Variation 1	17933	183	3281890
Variation 2	20014	131	2621954

FIGURE 5.25: Output data for Energy Balancing Protocol with Random Initial Energy (10% death)

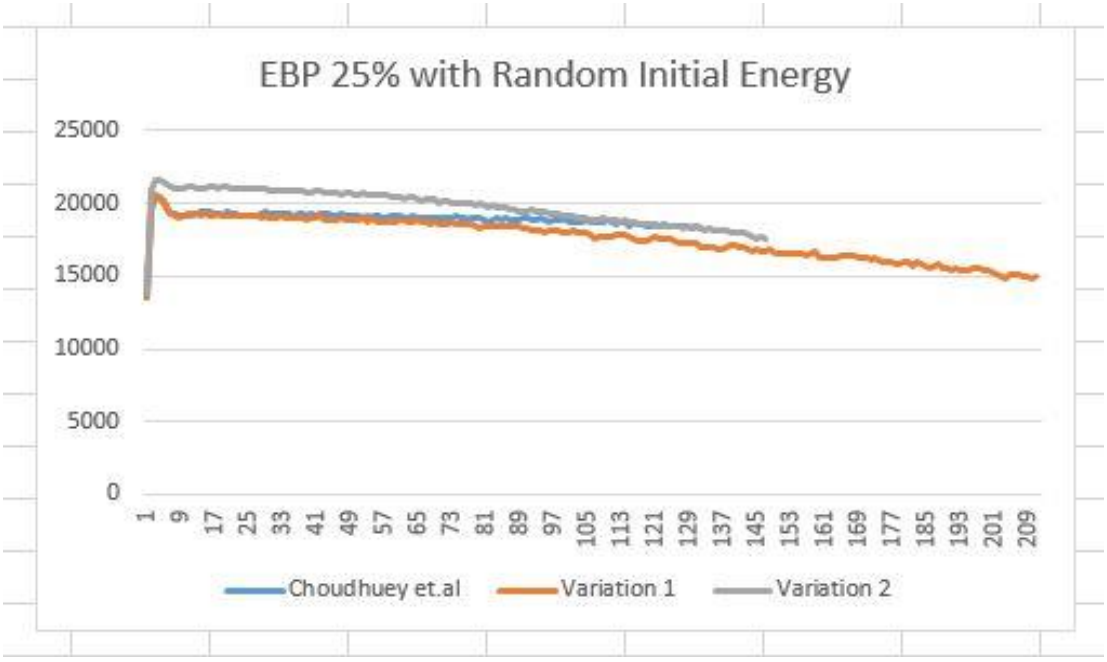


FIGURE 5.26: Energy Balancing Protocol with Random Initial Energy (25% death)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	18982	132	2505653
Variation 1	17603	211	3714362
Variation 2	19791	147	2909345

FIGURE 5.27: Output data for Energy Balancing Protocol with Random Initial Energy (25% death)

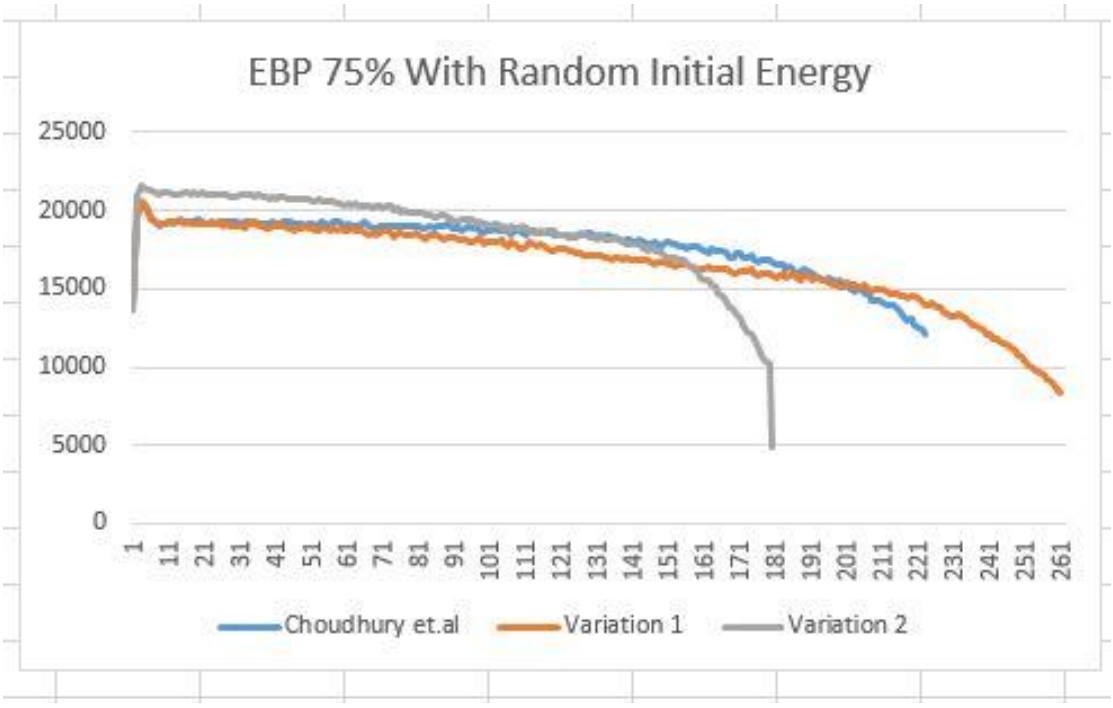


FIGURE 5.28: Energy Balancing Protocol with Random Initial Energy (75% death)

	Average Coverage	Network Lifetime	Area under the curve
Choudhury et.al	17886	223	2988623
Variation 1	16566	261	4323889
Variation 2	18777	180	3380003

FIGURE 5.29: Output data for Energy Balancing Protocol with Random Initial Energy (75% death)

Observation (Experiment 7-9):

Random initial energy influence works in a similar way as it does in the fixed initial energy scenario, where the differences are proportional. These observation increases Energy Balancing Protocols' feasibility to a greater extent – that, it can work just as fine in various environments.

FINAL OBSERVATION:

Overall observation suggests that, two of the proposed algorithms, Variation 1, Variation 2 - all delivers better average coverage and area under the curve. Variation 2 is for ensuring strict average coverage service. As the network lifetime's definition changes through decreasing death allowance, we find out both algorithm variations tend to beat Choudhury et.al algorithms output in all the sections - network lifetime, area under the curve, average coverage. Since Variation 1 provides lower coverage in the middle period, we also offer more alternatives to reduce that gap through increasing the parameter of checking.

5.7 EXTENDED ENERGY BALANCING PROTOCOL

Algorithm Procedure:

Algorithm follows the Energy Balancing Protocol described in the earlier sub-section. So all the rules are same as before except the following two, which are slightly modified from the previous algorithm.

- A sleeping sensor - after being probabilistically selected for checking, figures out-
  1. Number of active sensors in the neighborhood.
     - If the count is less than one, it suffices the initial condition for waking up.
  2. Number of neighbor nodes in energy level greater than 1 comparing to primary nodes level according to latter nodes memory.
     - If the count is zero, it continues to check the last rule.
  3. Number of neighbor nodes probabilistically selected for checking who are also asleep.
     - If the count is zero, it continues to check the last rule.
  4. If, (III)*Random value (0, 1) <1, it wakes up.

Further Discussion:

A sleeping sensor can wake up if none of its neighbor node is awake. As a result, if it wakes up, it generally covers nine node areas, if we consider a basic scenario avoiding complexities. Such instances assure complete utilization of energy and overall a balanced coverage versus lifetime scenario. Although, instead of zero node, allowing up to 1 or 2 sensor nodes in the neighborhood to be alive as a condition for a sleeping sensor node to be alive ensures more average coverage per iteration, but through the cost of network lifetime. On other note, this is just the initial primary condition that is mandatory for the transition. It also needs to ensure that no other neighbor node has more than a certain amount of energy compared to primary sensor nodes energy, parameter (II) deals with this case. If any of the neighbor node has extra energy, it means that primary sensor node considers that it has used his energy enough and it's better if that node/one of those nodes (which has excess energy) wake up. In a hope of waking

up those specific neighbor nodes, it decides to stay asleep, which gives the neighbor node staying in higher energy level a chance to wake up. In case of utilizing parameter (III), it reduces the probability of neighbor nodes in each other's range to wake up all at once - which wastes lots of energy, given the situation occurs where nodes residing in the same neighborhood of each other are probabilistically selected and asleep also. Being a concurrent process, these nodes might all think that they are eligible to wake up, not considering the probability of other sensors waking up. We don't want more than one nodes to cover a single cell, so neglecting the mentioned fact, will hurt this purpose.

- An awake sensor, if chosen probabilistically, checks neighbor nodes to figure out –
    1. Number of awake sensors in the neighborhood
    2. Number of probabilistically selected sensors in the neighborhood which are also awake .If (I) is greater than or equal to 1, it collects (II). It generates a random value between 0 and 1 and multiplies by (II), if this value is less than .5, it goes to sleep.

Further Discussion:

If no other neighbor is awake, it has the best possibility to utilize his energy while monitoring its neighborhood. Few other things though come into possibility, that the neighbor cells are already being covered by other sensors, which is beyond the sight of our primary sensor. There might occur some other scenarios where several neighbor nodes have been selected probabilistically and are also awake along with the primary sensors. As it is a concurrent process, all of them will see that at least one of the neighbor nodes are awake, thus they will all go to sleep if no further rule is associated. Such occurrences will create a gap in the coverage, sensor nodes will idly waste their energy being asleep altogether.

SIMULATION:

Experiment 1:

EEBP's performance in different network lifetime definition.

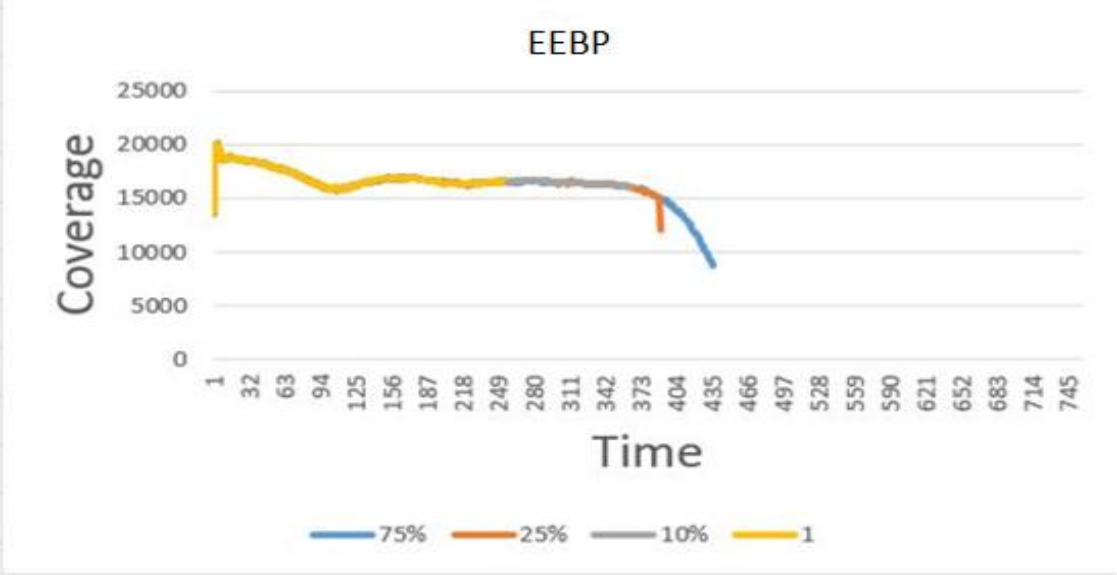


FIGURE 5.30: Extended Energy Balancing Protocol (for different limit of sensor nodes death)

EBP	Average Coverage	Network Lifetime	Area under the curve
75%	16311	436	7111620
25%	16746	390	6531127
10%	16849	365	6150210
1	17026	252	4290790

FIGURE 5.31: Data for Extended Energy Balancing Protocol (for different limit of sensor nodes death)

OBSERVATION:

Healthy average coverage and extended lifetime - subsequently a great amount of area under the curve. Since it is balanced where these three mentioned outputs, namely - average coverage, lifetime and area under the curve are considered having equal significance, we have the luxury of changing variable input parameters to any direction, either upward or downward to adopt our application requirement. For instance, we can consider of increasing the selection probability from .20 to .25, .3, .35 - and get more average coverage, where we focus on that very output. We can also adjust other input parameters to increase network lifetime also. All of them will be discussed in the further reports. Since energy usage is fairly balanced, around 250

iterations pass before one single node die, which can be counted as a terrific amount of achievement for the system.

Experiment 2:

We will now change some of the variable inputs. Instead of a fixed initial energy of .8J, we will be using varying energy, where initial energy threshold is determined as 30% of highest energy, which is to say 30% of .8 = .24. So, energy will be distributed uniformly and randomly between .24 and .8. Sensor nodes initialize its memory according to its own energy level, which means that it will assume that all the neighbor nodes are in the same level as it is.

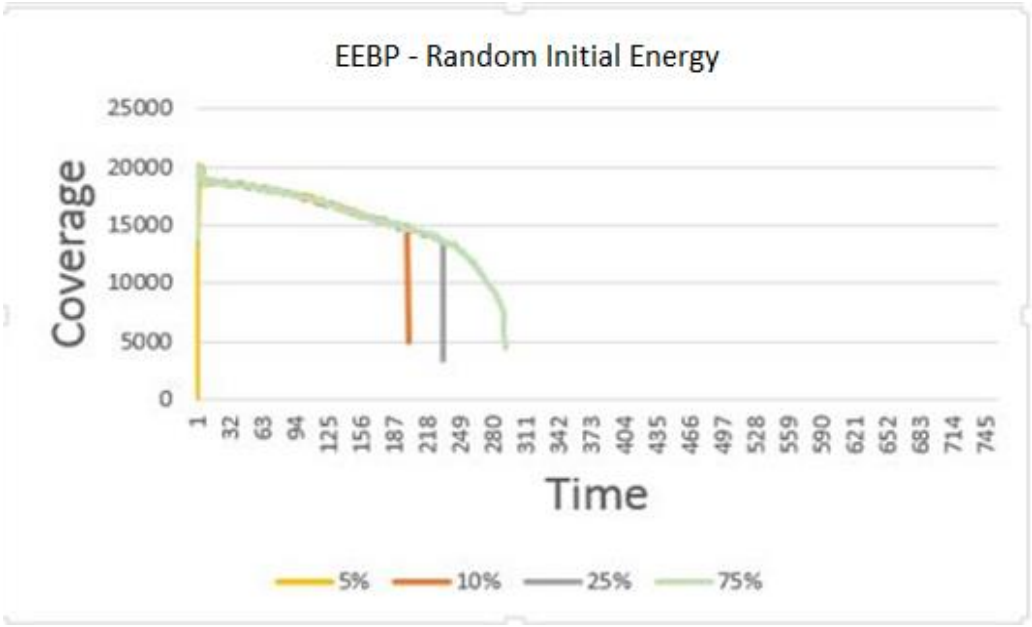


FIGURE 5.32: Extended Energy Balancing Protocol with Random Initial Energy (for different limit of sensor nodes death)

EBP – Random Energy	Average Coverage	Network Lifetime	Area under the curve
75%	15634	292	4565165
25%	16709	234	3909996
10%	17119	201	3441037
5%	17519	172	3013330

FIGURE 5.33: Data for Extended Energy Balancing Protocol with Random Initial Energy (for different limit of sensor nodes death)

OBSERVATION:

Since the energy is randomly distributed in between .24 and .8, we begin with much less cumulative energy to begin with, so performance differs from the experiment where initial energy is fixed. Another important fact is that all the sensor nodes initializes memory with its own energy level. That apparently will provide slight wrong information to others, but it helps them think that neighbor nodes are not residing in extra higher level, which is consequently allowing them to be awake and ensure that a large portion of the whole network is under cover. We have avoided the experiment where network lifetime finishes when only one sensor node is dead, because of the reason that some nodes start with a very low energy and they die too soon - and the apparent result will fail to depict the potential the system holds to deliver. Still, the four scenarios of different lifetime have competitive outputs, comparing them with the existing algorithms will further clear its position. Comparison will be demonstrated in the upcoming segments

Experiment 3:

Experiment 3 identifies comparison between algorithms when network lifetime definition is relatively high.

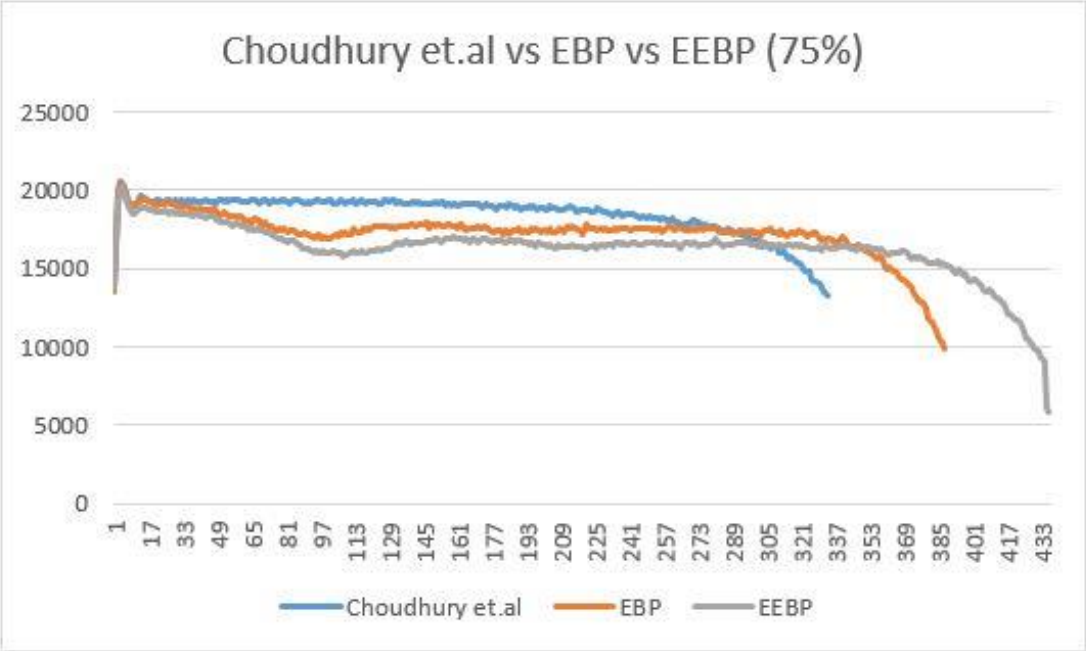


FIGURE 5.34: Comparison among EEBP and other algorithms (until 75% death)

Algorithms	Average Coverage	Network Lifetime	Area under the curve
EEBP	16311	436	7111620
EBP	17282	388	6705594
Choudhury et.al	18517	333	6166464

OBSERVATION:

EBP offers more area under the cover than BCEP and Salim. Granted that is has lower average coverage, but it has to sacrifice bit of its average coverage to provide more lifetime and a humongous total of area under the curve. And with the luxury in hand of trading off a large

amount of extra lifetime, it can alter various inputs to reduce those gaps of coverage compared to other algorithms. Of the top of head, we can consider increasing selection probability, less blocking due to extra higher level neighbor nodes existence, increasing the probability of waking up in the case of finding neighbor nodes also probabilistically selected - all of which can increase average coverage and still be better than others.

Experiment 4:

Comparison between algorithms when network lifetime definition is the lowest possible one, which is to be exact, one sensor nodes' death defines end of network lifetime.

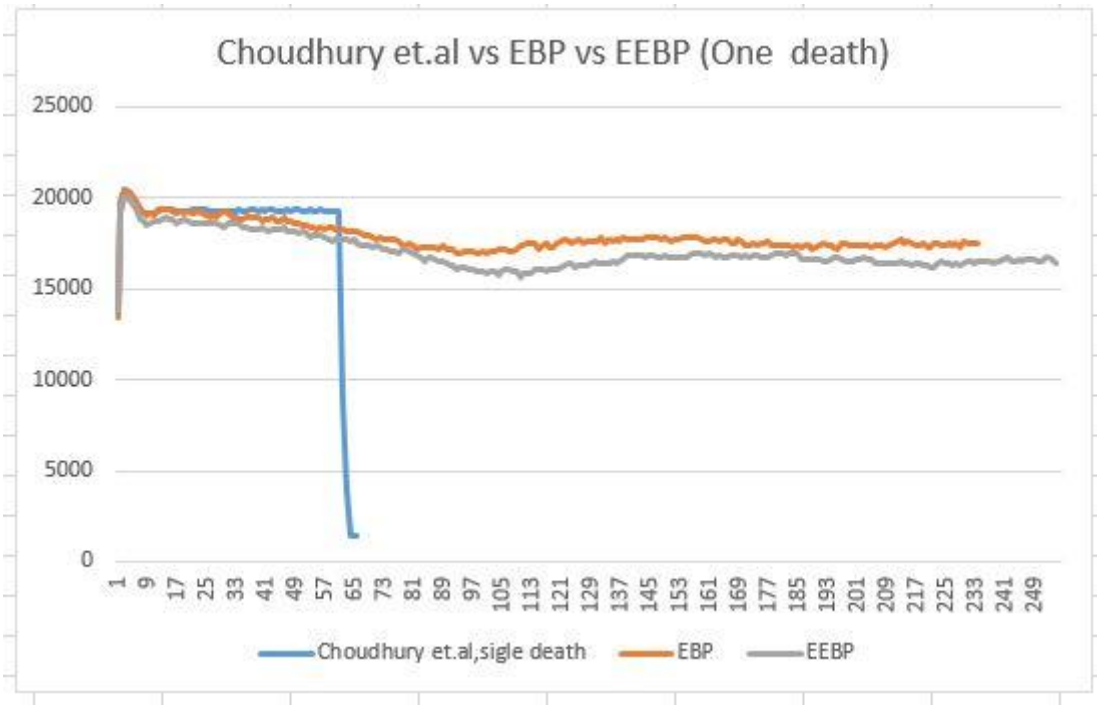


FIGURE 5.36: Comparison among EEBP and other algorithms (one death)

Algorithms	Average Coverage	Network Lifetime	Area under the curve
EEBP	17026	252	4290790
EBP	17475	237	4141800
Choudhury et.al	18468	64	1181965

FIGURE 5.37: Data from comparison among EEBP and other algorithms (one death)

OBSERVATION:

As the lifetime length decreases, the difference between EBP and EEBP does so. But Choudhury starts to fall way behind in terms of area under the curve in this process. Though the EBP and EEBP are becoming quite competitive in this manner, EBP still has the edge over EEBP with higher area under the curve value.

5.8 RECHARGEABLE SENSOR

In this section, we implement characteristics of rechargeable sensors. We are leaving its practical implementation of what should be the strategy to recharge this sensors, how it is possible and procedures' feasibility to more in-depth analysts. For now, we focus on how it would perform, if an existing algorithm was to be enforced on them.

We used Choudhury et.al. & hexagonal grid structure for the time being to test its performance. The system adjusts few of the following staffs:

- Each sensor can be recharged 3 more times.
- Sensors take 20 time periods to be fully recharged and transit from dead state to alive.

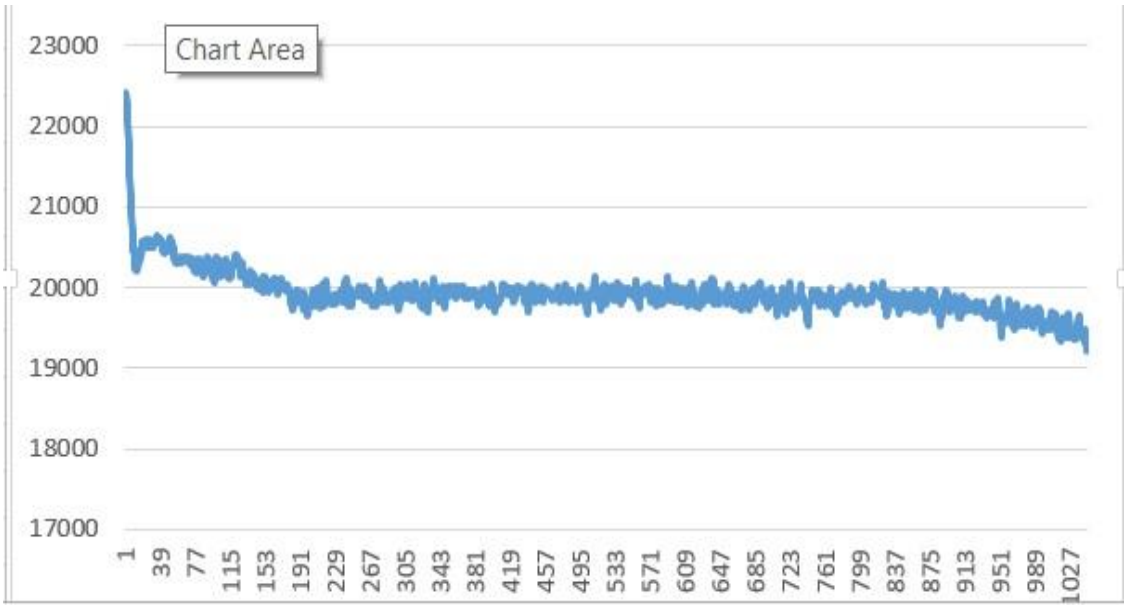


FIGURE 5.5: Lifetime vs Coverage graph for Rechargeable sensor

Network lifetime is considered to be alive until 25% of the sensors become dead.

OBSERVATION:

Network coverage stays high, around 90% of all the sensor nodes are monitored all the time. Current adjustment allows more aggressive approach from the sensor nodes, as they can be alive again, which reduces the systems' suffering from loss of average coverage due to sensor nodes death.

CHAPTER 6:

Future Work & Conclusion:

6.1 CONCLUSION:

After analysis and comparison of the existing algorithm with our proposed algorithm we came to following conclusions:-

- Lifetime and area under the curve performance declares supremacy over the existing algorithms

- Extended lifetime offers more flexibility in the system, can be traded-off for better average coverage
- Energy utilization of the sensors are fairly balanced, sudden occurrence of mass sensor death disappears
- Potential of the algorithms for variety of structures and advanced techniques show promise

6.2 FUTURE WORKS:

In future we would like to work in the following areas:-

- Implementation of extended rules
    - Neighborhood of radius 2, 3 etc.
    - Hexagonal grid structure
    - Loss of energy due to sensor nodes transition and its optimization
- Verification of rechargeable sensor application and its efficiency
- Concentration on the connectivity between sensor nodes
- Modification of current algorithms to explore utmost utilization
- Generating specific algorithms, for all kinds of systems, which require
    - Optimum coverage versus network lifetime
    - Focus on network lifetime
    - Focus on coverage

REFERENCES

[1] Renan O. Cunha, Aloizio P. Silva, Antonio A. F. Loreiro and Linnyer B. Ruiz Simulating Large Wireless Sensor Networks Using Cellular Automata(2005).In Proceedings 38th Annual Simulation Symposium (ANSS-38 2005),Pages 1-8.

[2] M. Cardei and J. Wu.Energy-effcient coverage problems in wireless ad-hoc sensor networks (2006).Computer communications 29(4):413-420.

[3] Salimur Choudhury, Kai Salomaa AND Selim G. Akl - A Cellular Automaton Model for Wireless Sensor Networks (2011) Journal of Cellular Automata, Volume-7, pages 223-241.
[4] G. Huang, Casting the wireless sensor net, Technol. Rev. (Manchester, NH) 106 (2003), pp. 50–57.
[5] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, Mobility improves coverage of sensor networks, in Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05), University of Illinois at Urbana–Champaign, Urbana, IL, ACM, IL, USA, 2005, pp. 300–308.
[6] S.A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, and J. Ma, Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing, in Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07), IEEE, Niagara Falls, Ontario, Canada, Vol. 2, 2007, pp. 113–120.
[7] Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, Mobility improves coverage of sensor networks, in Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05), University of Illinois at Urbana–Champaign, Urbana, IL, ACM, IL, USA, 2005, pp. 300–308.
[8] Y. Baryshnikov, E. Coffman, and K. Kwak, High performance sleep–wake sensor systems based on cyclic cellular automata, in International Conference on Information Processing in Sensor Networks, 2008 (IPSN'08), IEEE, St. Louis, Missouri, USA, 2008, pp. 517–526.
[9] S. Choudhury, K. Salomaa, and S.G. Akl, A cellular automaton model for wireless sensor networks, J. Cell. Autom 7 (2012), pp. 223–241.
[10] R.O. Cunha, A.P. Silva, A.A.F. Loreiro, and L.B. Ruiz, Simulating large wireless sensor networks using cellular automata, in Proceedings of the 38th Annual Simulation Symposium, 2005, IEEE, Washington, DC, USA, 2005, pp. 323–330.
[11] M. Esnaashari and M. Meybodi, A cellular learning automata based clustering algorithm for wireless sensor networks, Sens. Lett. 6 (2008), pp. 723–735.
[12] M. Garzon, Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks, Texts in Theoretical Computer Science, Springer-Verlag, London, UK, 1995.

[13] Sami Torbey, Selim G. Akl. Reliable Node Placement in Wireless Sensor Networks Using Cellular Automata, 11th International Conference, WCNC 2012, Orléan, France, September 3-7, 2012.
[14] Salimur Choudhury, Kai Salomaa and Selim G. Akl (2014). Cellular Automaton Based Algorithms for the Dispersion of Mobile Wireless Sensor Networks, International Journal of Parallel, Emergent and Distributed Systems 2014, 29:2, 147-177