ENERGY-EFFICIENT DATA AGGREGATION IN WIRELESS SENSOR NETWORKS USING PROBABILISTIC SLEEP SCHEDULING AND COMPRESSED SENSING

By

ARAM ALMUHANA

Bachelor of Science in electrical engineering

University of Baghdad

Baghdad, Iraq

2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2013

ENERGY-EFFICIENT DATA AGGREGATION IN WIRELESS SENSOR NETWORKS USING PROBABILISTIC SLEEP SCHEDULING AND COMPRESSED SENSING

Thesis Approved:

Dr. Nazanin Rahnavard

Thesis Adviser

Dr. Damon Chandler

Dr. Weihua Sheng

Name: ARAM ALMUHANA

Date of Degree: MAY, 2013

Title of Study: ENERGY-EFFICIENT DATA AGGREGATION IN WIRELESS SENSOR NETWORKS USING PROBABILISTIC SLEEP SCHEDULING AND COMPRESSED SENSING

Major Field: ELECTRICAL AND COMPUTER ENGINEERING

Each node in a wireless sensor network (WSN) is an inexpensive and small device with a limited source of energy. In many applications related to monitoring of physical phenomenon and natural signals, there is spatio-temporal correlation among the readings from different sensors and different times. This work takes advantage of the correlation and integrates compressed sensing and sleep scheduling to significantly reduce the energy consumption of data aggregation in WSNs. The proposed method is based on probabilistic models, allowing it to be simple, fast, flexible, reliable, and suitable for randomly deployed networks with no defined topology, known location, or time synchronization.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

II. PROBABILISTIC SLEEP SCHEDULING IN WSN.....4

2.1 CONNECTIVITY OF NODES 4
2.2 DUTY CYCLE 9

III. COMPRESSED SENSING 12

3.1 WHY COMPRESSED SENSING? 12
3.2 COMPRESSED SENSING METHODOLOGY 13
3.3 COMPRESSED SENSING MATRICES 14
3.4 SAMPLING RATE.....15

IV. COMPRESSED SENSING BASED PROBABILISTIC AGGREGATION 17

4.1 CS-PAGG CONCEPT 17
4.2 NETWORK ROUTING IN CS-PAGG 19
4.2.1 Background 19
4.2.2 Directed diffusion 19
4.2.3 Neighbors discovery algorithm.....24
4.2.4 Aggregation initiation 26
4.2.5 Aggregation path.....28
4.3 TRANSMITTED PACKETS IN WSN 30
4.3.1 Background 30
4.3.2 Hop distance in shortest path algorithm.....30
4.3.3 Dead measurements 41
4.4 GOBACK ALGORITHM.....46
4.4.1 GoBack concept 46
4.4.2 GoBack forwarding map 48
4.4.3 GoBack aggregation length.....50
4.4.4 GoBack dead measurements 51
4.4.5 GoBack speed and complexity.....53

Chapter Page

4.4.6 GoBack memory 53
4.4.7 GoBack communications cost 3
4.5 HOP DISTANCE IN CS-PAGG 62

V. POWER CONSUMPTION IN CS-PAGG 65

5.1 BACKGROUND 65
5.2 POWER IN CS-PAGG 66
5.2.1 Power modes 66
5.2.2 Activity power 69
5.2.3 Operation power 70
5.2.4 WSN power consumption 72
5.3 OPTIMUM ACTIVE PROBABILITY 76
5.3.1 Definition of optimum active probability 76
5.3.2 Simulation optimum active probability 77
5.3.3 Theoretical optimum active probability 79
5.4 REGIONS OF OPERATION IN CS-PAGG 80

VI. PERFORMANCE OF CS-PAGG 83

6.1 SIGNAL RECONSTRUCTION 83
6.2 SIMULATION RESULTS 84
6.3 MINIMUM ACTIVE PROBABILITY 85
6.4 EFFICIENCY EVALUATION IN CS-PAGG 87

VII. CONCLUSION AND FUTURE WORK 90

REFERENCES 94

LIST OF TABLES

Table Page

Table 1. Routing table of all nodes in the network 23
Table 2. Aggregation and Measurement paths. 50
Table 3. Percentage of lost measurements. 51
Table 4. Power model for Mica2. 67

LIST OF FIGURES

Figure Page

Figure 1. Randomly deployed WSN with B.S in the middle. 6
Figure 2. Connectivity links when all nodes are active 7
Figure 3. Connectivity links when 56.8% of nodes are active. Active and sleeping nodes are represented as black and white circles respectively. 8
Figure 4. Different screenshots of the same network with $P^{awake} = P^{th} = 0.568$. 9
Figure 5. Sleeping schedules for 4 nodes using random duty cycle adjustment. 10
Figure 6. Sleeping schedules for 4 nodes using synchronized duty cycles. 10
Figure 7. Sleeping schedules for 4 nodes using random refresh rate 11
Figure 8. Initiation of directed diffusion in a small part of a network. The base station is represented with a star 21
Figure 9. Second stage of introductory diffusion. 21
Figure 10. Third stage of introductory diffusion. 22
Figure 11. Inquiry hello broadcasting. Black nodes are awake and white nodes are sleeping. 25
Figure 12. (a) shows duty cycle of a node and (b) shows the instances at which the node makes decisions regarding measurement initiation. 27
Figure 13. A network with active nodes (Black), sleeping nodes (White), and some aggregation paths (red lines). 28
Figure 14 Four random aggregation paths. Each set of arrows represents one aggregation 29
Figure 15. Measuring the change in probability of reaching a distance of Z using only n hops when R changes by dR. 32
Figure 16. Summation Sectors in normal distribution circular networks with base station in the middle 35
Figure 17. Infinite number of sectors in Integration method for the same network from the previous figure. 36
Figure 18 . Wireless network with two sensors. Sensor A has the best orientation while sensor B has the worst orientation. 37
Figure 19. Large sector radius increment value. Sensor B orientation may cause some calculation error 38
Figure 20. Best sector radius increment value. Both sensors A and B are accurately assigned to their corresponding sectors. 39
Figure 21. Shortest path from node 1 to the base station with all nodes active. 41
Figure 22. Best available path when node 3 is sleeping 43
Figure 23. Dead measurement path 43

Figure Page

Figure 24. Active nodes (Black) in a network with a dead end (Red) located one hop away 46
Figure 25. Active nodes (Black) in a network with a dead end (Red) located three hops away.... 47
Figure 26. Active nodes (Black) in a network with two dead ends (Red) located on hop away ... 47
Figure 27. Forwarding maps for (a) Best available path, and (b) GoBack..... 49
Figure 28. Change in percentage of dead measurements with P^{awake} increment for a circular network with a radius of 280m and 3000 deployed nodes. P^{th} is 0.46 with shortest available path method used. 52
Figure 29. Comparison between GoBack and best available path in terms of number of transmitted packets per measurement for a network with a radius of 280m and 3000 randomly deployed nodes. $P^{awake} = P^{th}$ 54
Figure 30. Average number of hops per measurement for a network with a radius of 280m and 3000 randomly deployed nodes 55
Figure 31. Normalized reconstruction error for a 50-sparse signal using DCT sparsifying matrix and norm 1 minimization 55
Figure 32. Probability of back-to-back GoBack chains when $P^{awake} = P^{th}$..... 57
Figure 33. Relationship between average number of hops per measurement and the maximum allowed GoBack degree 58
Figure 34. Normalized error for each GoBack threshold using DCT as sparsifying matrix 58
Figure 35. Total average number of transmitted packets needed from different GoBack thresholds to achieve the same signal reconstruction accuracy 61
Figure 36. Best available path equal to shortest path..... 62
Figure 37. Difference between real best path and chosen best path..... 63
Figure 38. Total WSN power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits, δ_2=16 bits, and μ=700 mes/sec..... 72
Figure 39. Activity power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits. 73
Figure 40. Operation power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits. 74
Figure 41. Power consumption components for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits. 75
Figure 42. WSN total power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits. 76
Figure 43. Optimum P^{awake} for a WSN. Region shown by vertical lines shows possible P^{awake} values within 5% of minimum power consumption. 77
Figure 44. P^{awake} region within 5% of minimum power. 78
Figure 45. Accuracy of theoretical analysis of optimum P^{awake} 80
Figure 46 Regions of optimum probability in a WSN. 80
Figure 47. Normalized reconstruction error for 700 received measurements and 200-sparse signal. 84
Figure 48. New CS-PAGG regions of operation after P^{min} consideration. 86
Figure 49. Total WSN power consumption with and without CS-PAGG for a network with a radius of 280m and 3000 randomly deployed Mica2 sensors..... 87

Figure Page

Figure 50. Signal reconstruction error with and without CS-PAGG for a 200-sparse signal and 700 measurements at P^{min} of 0.525. 88
Figure 51. Power saving under CS-PAGG. 89

CHAPTER I

INTRODUCTION

A wireless sensor network (WSN) is a network of very small and relatively cheap sensors that can be deployed over a large area for monitoring purposes. The advanced technology today allows us to use such networks for commercial and scientific purposes widely. Each node in a WSN is a small device that consists of the desired sensors, a radio, a small processor, a memory, and a limited source of energy (battery, solar cells, etc). Since the energy is very limited, conserving it has been a big part of the research in WSN over the years. Many methods already exist in reducing the energy consumption on both hardware and software levels. In many applications of WSN such as monitoring temperature, soil moisture, wind speed, and other natural signals, the signal is correlated among nodes. Such correlation allows the usage of compressed sensing methods (CS) [1], to sample below the Nyquest rate and save energy.

Natural signals change slowly with time. Therefore, in most cases, natural signals get over-sampled by the nodes. When monitoring such signals, all nodes are usually active all the time, which can consume a lot of unnecessary energy. Therefore, in addition to saving energy using compressed sensing techniques, more energy can be saved if some nodes are turned on and off regularly. If nodes are turned on and off fast enough according to a pre-defined duty cycle, power can be saved for the same level of signal reconstruction accuracy.

While CS can reduce the energy consumption related to the data transmission in the network and sleep scheduling reduces the energy wasted in the idle mode (radio is on but no transmission or reception happens) and there are well-researched method that consider each of them separately [2] [3] [4] [5] [6]. To the best of our knowledge, there is no study that considers both techniques for maximizing the energy efficiency. Some of the proposed work on CS for WSNs may even lose their performance if there is an underlying sleep scheduling. We came to realize that not only a CS-based technique should not be harmed by the sleep-schedule of nodes, but also a well-designed CS scheme can provide the opportunity for putting a simple sleep scheduling techniques in to the place. This work is the first to integrate Compressive sensing and sleep scheduling to provide a very energy efficient framework for data acquisition in WSNs. Interestingly, in addition to providing enhanced energy efficiency, our proposed technique that relies on a simple probabilistic sleep scheduling removes the need for sleep scheduling methods that rely of complex and sometimes infeasible operations such as time synchronization among the nodes. Therefore, the gain of employing CS in WSNs is actually two fold.

The work described in this thesis identifies the problems with existing methods, presents practical solutions, and describes the energy model of the network using both theoretical analysis and practical simulation. The main idea of the method is to reduce the active time for each node in the network to the minimum possible, saving a lot of energy that is otherwise wasted on idling. Moreover, using CS enables saving on communication cost. Another aspect of this research was to remove the limitations that other methods have such as time synchronization, localization, and node position restrictions.

In the next few chapters, the method will be defined and described. Operation environments will also be setup and discussed. Mathematical formulas will be presented to analyze the scheme rapidly and accurately. Later, the efficiency of the method will be compared

to other traditional techniques and the performance of the method will be evaluated, followed by a performance conclusion and future improvement ideas.

CHAPTER II

PROBABILISTIC SLEEP SCHEDULING IN WSNs

2.1 CONNECTIVITY OF NODES

In order to achieve a fully functional network when some nodes are turned off randomly, the network should always be connected and there should always be a path from any node in the network to the base station. The connectivity of a network depends on the density of nodes in it. More node density leads to better connectivity. Increasing the density directly affects the energy consumption of the network because more nodes consume more energy. Therefore, finding a point of density that is the minimum required to keep the network connected can improve the power consumption of WSNs.

For the same number of nodes in the network, the node density can be manipulated by changing the ratio of active nodes. When more nodes are active in a network, node density increases and vice versa. Assuming at each time instance only a fraction P^{awake} of all nodes in a WSN are on, the work in [7] has found P^{th} given for connectivity using the two formulas:

$$r \geq \sqrt{\frac{\ln(N)A}{N\pi}}$$

Equation 1

and,

$$\frac{P^{th}\,\pi r^2(N)}{A} = \frac{\ln(P^{th}N)+\omega(N)}{N} \quad \text{as } N \to \infty$$

Equation 2

where:

r: radius of communication

N: number of nodes in the network

A: area in which nodes are deployed randomly

P^{th}: minimum ratio of active nodes required to maintain node connectivity

ω (N): any slowly growing function such that $\omega(N) \to \infty$ as $N \to \infty$

The conditions mentioned in Equation 1 and Equation 2 are valid for any large number of nodes. Applying P^{th} to a sensor network happens during the designing of the network. After setting the radius of communication and calculating P^{th}, each node in the network gets informed of the minimum probability that it should choose. Nodes have to be active for at least P^{th} of the time in order to have a connected network. When P^{th} is known, the actual active probability of nodes (P^{awake}) should satisfy:

$$P^{awake} \geq P^{th}$$

Equation 3

At any given time, the network has $P^{awake} * N$ nodes that are active and connected to each other. A base station (BS) is a node that collects the monitored signal readings. Base stations are usually more powerful than normal nodes and have larger communication radius, but to consider

worst cases, base stations will be assumed to have the same communication range of normal nodes in the network. Therefore, base stations are connected to at least one active node in their surroundings, giving all active nodes in the network full connectivity and making the WSN fully functional at any time while P^{awake} is in use.

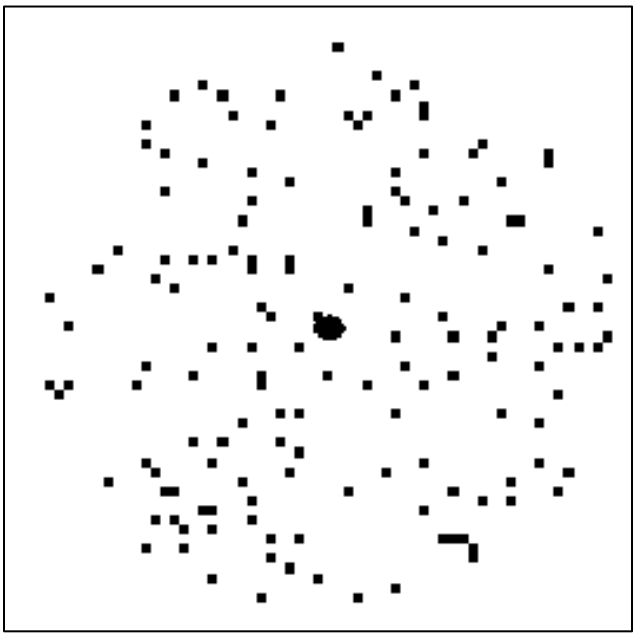


Figure 1. A randomly deployed WSN with BS in the center.

The figure above is an example of a simple randomly deployed wireless sensor network. In this particular network, there are N=150 sensor nodes deployed in a circle with a radius of 30 distance units, covering a total area of $A = 30^2 * \pi = 2.8\, distance\, units^2$. The communication radius for each sensor node is r=6.85 distance units. When all the nodes are awake and active, if the connectivity links between every two connected nodes are drawn simultaneously, the network will have a large number of connections as shown in Figure 2. As can be seen, most of the links provide redundant paths between any two sensor nodes. Such redundancy makes the network connectivity map very dense and contributes to a big part of the energy consumption of the network. Since multi-hop communications are used for most wireless networks with short

transmission radius, all active nodes will participate in message forwarding. Therefore, when the number of active nodes in a network increases, the energy consumption increases too.

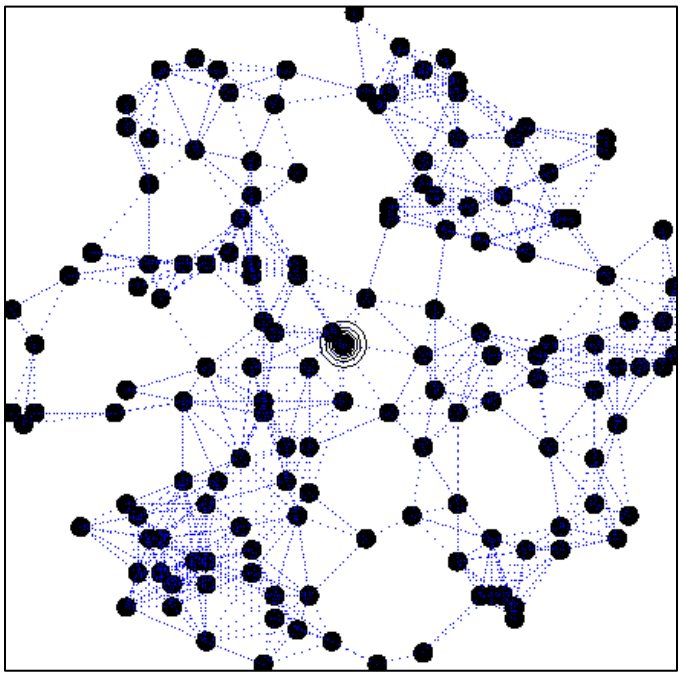


Figure 2. Connectivity links when all nodes are active.

Shortest path algorithm can be used to forward message from any sending nodes to the base station. Hop-based shortest path, as the name suggests, chooses the multi-hop path that requires the least number of hops before reaching the destination. Another type of shortest path depends on the actual distance between the nodes rather than their hop distance. Both schemes can be implemented using one of many available methods and protocols. Since the sensors that the research deals with are randomly deployed, have very basic electronics, and do not have access to their own location, hop based methods were used in the analysis part of the work.

If the network uses shortest path algorithm with all WSN nodes in active mode, messages generated from any certain node will always go through the same path during their journey to the base station, making the routing load for certain nodes very high. In addition, if compressed

sensing is used and random measurement matrices are formed during data aggregation, the data collected will be heavily correlated. Aggregation paths for measurements generated from the same nodes would always go through the same shortest path towards the base station.

Plugging the parameters of the network mentioned previously in Equation 2 yields a P^{th} of 0.568. Since the probability of each node is independent, the total number of active nodes in the network at any time is equals to $P^{th} * N$.

According to the previous results, even after randomly turning off almost half the sensor nodes in the WSN, the total network should remain connected. Any messages generated from any sensor node should have a valid path to the base station as long as the probability does not drop below P^{th}.

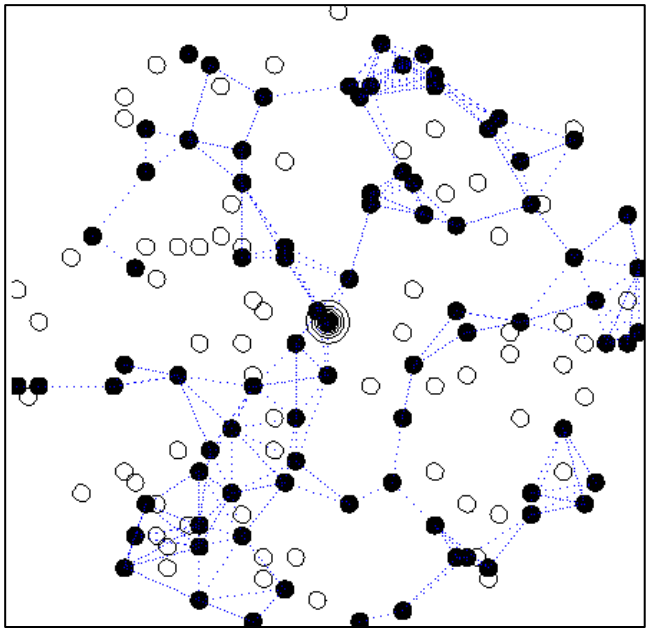


Figure 3. Connectivity links when 56.8% of nodes are active. Active and sleeping nodes are represented as black and white circles, respectively.

In the network screenshots in Figure 3, each sensor node independently chose to be awake and active with a probability of P^{th} = 0.568. Each time a screenshot of the network was

taken, the active sensors changed randomly, changing with them the total configuration of the network.

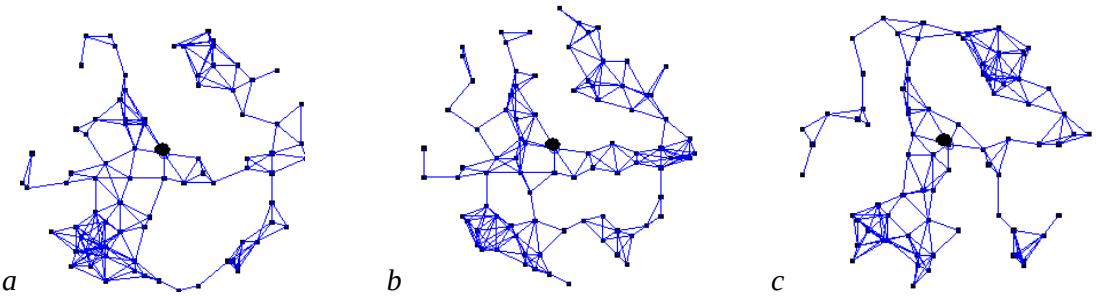


Figure 4. Different screenshots of the same network with $P^{awake} = P^{th} = 0.568$.

Three different instances of a network are shown in Figure 4. The different screenshots clearly show that although almost half the nodes are inactive, the network remains functional and all the active nodes remain connected to each other. If shortest path algorithm is implemented in the network, it will have a different path for each instant. Since the selection of the active nodes happens randomly, the routing load becomes more distributed than the first network shown in Figure 2. Random paths also have a great advantage when implementing compressed sensing since they result in more random measurement matrices, and therefore they enhance the incoherence between measurement and sparsifying matrices. Better incoherence between the two matrices leads to better signal reconstruction for the same number of measurements [1].

2.2 DUTY CYCLE

There are two easy ways to implement a random sleeping schedule with P^{awake}. The first one is to use a duty cycle of P^{awake} for each node. If each node has a duty cycle of P^{awake} the whole network will have a duty cycle of P^{awake} too.

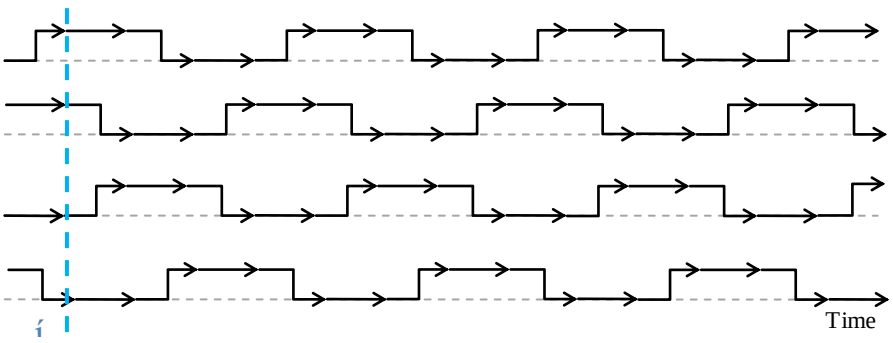


Figure 5. Sleeping schedules for 4 nodes using random duty cycle adjustment.

In order to maintain the condition of having $N\,P^{awake}$ active nodes at any given time, it is crucial that the nodes do not all sleep at the same time and wake up at the same time. In Figure 5, the duty cycle of each node is 0.5. There are only two awake nodes at every instance (i), resulting in a P^{awake} of 0.5. The same method can be up-scaled to higher numbers of sensor nodes as long as starting points of each node's sleeping schedule are kept random. If the network is perfectly synchronized and all nodes be on and off at the same time, having $N\,P^{awake}$ active nodes at each time will be invalid as explained in Figure 6.

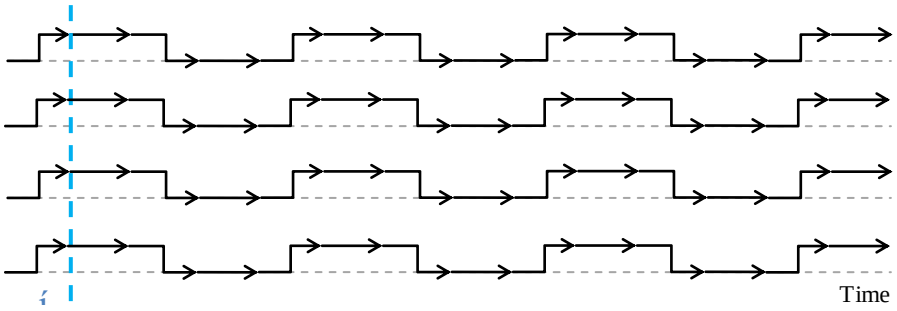


Figure 6. Sleeping schedules for 4 nodes using synchronized duty cycles.

The randomness can be easily achieved by not using any synchronization algorithms when deploying the network. Whenever a sensor is commanded to start operating for the first time, it chooses a random time to start its duty cycle and keeps it until another command that changes the duty cycle is received. The natural drifting in each sensor's internal clock will further

help increase the randomness of the network, and therefore, it will reduce the chances of having few neighboring nodes with the exact duty cycle timing. The second method of achieving a valid sleeping schedule is to set a refresh rate (τ) for the network such that the sensors change their statuses periodically. Figure 7 explains the later method and shows how it works.

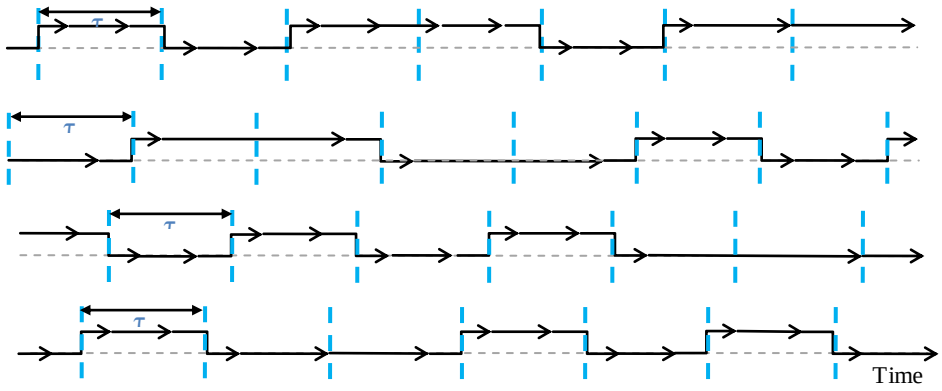


Figure 7. Sleeping schedules for 4 nodes using random refresh rate.

The refresh rate of the sleeping schedule depends on the frequency of the signal monitored. If the frequency of the collected signal is much higher than the sensor status refresh rate, there will be some instants at which a signal is missed because of nodes being inactive for the whole duration of the time when the signal changes. Therefore, choosing a refresh rate that is higher than the frequency of the collected signal reduces the chances of such instants. Like the previous method, the starting point of each sensor's refresh cycle is chosen randomly when the sensor starts operating for the first time. For each node, every τ seconds the node decides to be either active with a probability P^{awake} or inactive with probability of 1- P^{awake}.

The second method does not depend on the level of synchronization of the nodes. Thus, it was selected to perform the simulation of this work. It is worth noting that either of the two methods yields the same results obtained since they both achieve the same goal of having only a certain active portion of the sensor nodes at any time.

CHAPTER III

COMPRESSED SENSING

3.1 WHY COMPRESSED SENSING?

In many applications that have a lot of data to store or transmit, like in wireless sensor networks, there is a need for a large amount of energy and storage to collect the signal from the deployed sensors. Many compression schemes started to appear as the need for more energy-efficient methods increased. A compressed signal requires less energy and storage, but should be decompressed to the original signal when needed.

Lossless compression is one of the most common ways to compress signals. Due to its lossless characteristics, a signal that is compressed using lossless compression scheme can be recovered with 100% accuracy. The problem with lossless compression is that a signal cannot be compressed at a rate below the Nyquest rate without losing accuracy in reconstruction. Also, all of the signal should be available prior to compressing it. Sensor nodes would have to collect the whole signal before they can compress it. Parts of the signal could get compressed separately in different nodes, but in general, the usage of lossless compression in sensor networks is neither efficient nor practical.

Compressed sensing is a type of compression that can be applied to sparse signals at rates below the Nyquest rate. In compressed sensing, the signal can be compressed without losing the uniqueness of the solution when decompressing the compressed signal to the original one. Also, compressed sensing can be applied while aggregating the sensor readings. Thus, there is no need to acquire the whole signal or parts of it prior to applying the compression. Compared to the traditional compression techniques, CS encoding has a very low complexity and can easily be implemented at sensor nodes.

3.2 COMPRESSED SENSING METHODOLOGY

The idea of compressed sensing is to simply map a signal $\underline{X}$ to a shorter signal $\underline{Y}$ through an underdetermined system of linear equations. If $\underline{X}$ is K-sparse and the underdetermined system fulfills certain conditions, a signal $\underline{X}$ of length N can be successfully estimated from $M << N$ set of equations [2]. The projection matrix that projects signal $\underline{X}$ to $\underline{Y}$ is called the measurement matrix $\Phi_{M \times N}$ and we have:

$$\begin{bmatrix} \underline{Y} \end{bmatrix}_{M \times 1} = \begin{pmatrix} \Phi \end{pmatrix}_{M \times N} \begin{bmatrix} \underline{X} \end{bmatrix}_{N \times 1}$$

Equation 4

The original signal can be reconstructed by finding the sparsest solution to the underdetermined system. Such solution can be found using linear programming to find the solution that has the lowest norm1.

$$\hat{\underline{X}} = arg\min ||\underline{X}||_1 \quad subject\ to\ \underline{Y} = \Phi\,\underline{X}$$

and

$$||\underline{X}||_1 = \sum_{i=1}^{N} |X_i|$$

Equation 5

where $\hat{\underline{X}}$ is the signal estimation.

The number of projections (M) needed to reconstruct the signal $\underline{X}$ is correlated to the sparsity rate of the signal. When the measurement matrix satisfies the restricted isometric property RIP conditions, M becomes a function of N and K [8].

$$M \geq C\,K\,log(N/K)$$

Equation 6

where C is a small constant

3.3 COMPRESSED SENSING MATRICES

The idea of compressed sensing for non-sparse signals comes from the concept that some signal might not be sparse in the canonical domain, but they become sparse when projected into another domain [9]. For most of the natural monitored signals that wireless sensor networks deal with, like temperature and moisture, the signal coefficients do not vary much among adjacent nodes. If the signal is projected into any arbitrary orthonormal signal sparsifying basis such as DCT, the resultant signal will have far less distinct coefficient that the original signal. Thus, compressed sensing can be applied to the projected sparse signal. The transforming matrix (ψ) is called sparsifying matrix [10]. When dealing with sparsifying matrices, the L-1 norm minimization method in *Equation 5* changes to:

$$\underline{\hat{\alpha}} = arg\min ||\underline{\alpha}||_1 \quad subject\ to\ \underline{Y} = \Phi\,\psi\,\underline{\alpha}$$

Equation 7

and

$$\underline{\hat{X}} = \psi\,\underline{\hat{\alpha}}$$

Equation 8

where $\underline{\alpha}$ is a sparse projection of signal $\underline{X}$ onto sparsifying matrix ψ.

The measurement matrix Φ should be incoherent with the sparsifying matrix ψ in order to validate Equation 6 and achieve better CS signal reconstruction. Although designing a measurement matrix that is good requires following many design aspects, it has been proven that a good measurement matrix that satisfies RIP and compressed sensing guidelines can simply be a random Gaussian or binary matrix [11]. After *M* measurements are collected, the signal can be recovered using one of many methods including L-1 minimization. Methods such as message passing [12] and iterative hard thresholding [13] use fast localized algorithms to reconstruct the signal in less time than L-1 minimization. Although the mentioned alternative methods result in almost the same reconstructed signal as L-1 minimization for certain network configurations, for the sake of generalization and fair comparison, the reconstruction method of choice in the results chapter of this research was L-1 minimization.

3.4 SAMPLING RATE

We define μ as the number of measurements per second that is collected at the BS. Given that only *M* measurement is needed to reconstruct the signal using CS, it means that we can refresh the signal with the frequency of:

$$f = \mu/M \text{ Hz.}$$

Equation 9

The value of f depends on both the maximum frequency of signal change (f_X) and the minimum response time to signal changes that is required by the WSN. At $f < f_X$, μ is said to be in the under-sampling zone and $\underline{X}$ could change while the WSN is still collecting CS measurements from its sensor nodes. Since the signal changes faster than it is reconstructed, some signal changes might not be monitored by the base station and the reconstruction accuracy becomes low. At $f > f_X$, the situation is reversed and μ switches to the over-sampling zone. When over-sampling, the base station might recover $\underline{X}$ several time before it changes its values, causing unnecessary CS transmissions and wasted energy. Ideally, f should be slightly greater than f_X in order to avoid under-sampling and over-sampling. f_X can be found for natural signals by monitoring the FFT transform of the signal $\underline{X}$ over a period of time, then selecting the maximum frequency it records. Choosing the correct value of f is very important for both energy saving and valid signal recovery. Later in the thesis, the implications of μ on the efficiency of the network and the amount of saved energy will be explained in more detail.

CHAPTER IV

COMPRESSED SENSING BASED PROBABILISTIC AGGREGATION CS-PAGG

4.1 CS-PAGG CONCEPT

CS-PAGG is based on the idea of combining both compressed sensing and probabilistic sleep scheduling in order to achieve improved power efficiency for data collection in WSNs. CS-PAGG deals with natural signals that have many deployed nodes for monitoring purposes. The wireless sensor nodes in CS-PAGG are engineered to have the least activities possible in order to keep the network functional.

When a WSN that runs on CS-PAGG is designed, a probability that is called the active probability P^{awake} is calculated (Equations 2,3) and stored in each node in the network. The nodes then reduce their duty cycles based on the probability, allowing them to decrease their power consumption level. The probability is chosen to be the most effective one in reducing the power consumption of the network, in addition to keeping the network functional and accurate.

When sensor nodes are first distributed over an area, they get deployed in a random fashion for signal monitoring purposes. The nodes usually have one base station that collects the signal from the nodes through multi-hop paths. The base station starts a HELLO message broadcasting (based on a modified version of directed diffusion [14] as we will explain later), based on which each node will know its neighbors and their minimum hop distance to the base station.

The network starts collecting information after forming the routing table and the sleeping schedule begins its effect on the network. Based on P^{awake}, each node decides to either be active or turn off. At every instance there are only NP^{awake} active sensor nodes. The rest of the nodes go into a power saving mode.

Some of the active nodes decide to start compressed sensing measurements randomly and forward them to the base station. Along the multi-hop path of each measurement, each node detects its active neighbors through a neighbor discovery protocol then decides to forward their messages to the neighbor with the smallest hop distance to the base station. The same process happens for M measurements before they reach the base station in the form of aggregations. The base stations then reconstruct the monitored signal using compressed sensing reconstruction techniques. The monitored signal keeps refreshing based on the rate of the measurements μ received by the base station. CS-PAGG can be summarized by the algorithm below. Each part in the algorithm will be discussed in more detail later in the chapter.

1) Upon deployment, sensor determine their hop distance to the BS using directed diffusion
2) P^{th} is calculated and duty cycle of each node in the network is reduced to $P^{awake} \geq P^{th}$
3) Some nodes in the network decide to initiate CS measurements randomly [more explanations will be provided later]. We call those nodes CS initiators (CSI nodes).
4) Each CSI node sends a hello (who is awake?) message to its neighbors to be informed of active nodes within its transmission range. Each CSI node then selects the best active node (node with the smallest hop distance) to forward its data to.
5) Sensors along the aggregation path keep adding their readings and IDs till the measurement reaches the base station.
6) The base station receives transmitted CS measurements and reconstructs the monitored signal

4.2 NETWORK ROUTING IN CS-PAGG

4.2.1 Background

In most large-scale wireless sensor networks, the transmission range of nodes is not big enough to reach the base station directly. Therefore, in order to transmit a message from any node to the base station, the message has to go through many multi-hop stages before reaching the destination. There are many ways to control the path that a message can take in a multi-hop network in order to reach its destination. Some methods focus on fast routing while others seek simplicity. Since the topology of the network changes every time some nodes go to sleep and wake up, it is hard to find a routing method that is able to work with it and still be energy efficient.

Most of the available algorithms deal with networks nodes that are always active. Greedy algorithms in [15] work in a localized way to provide fast routing time. Power consumption does not get involved in greedy algorithms and the method assumes that all nodes are active all the time. Random walk is another routing method that works with nodes that turn on and off randomly [16]. The problem with random walks is that they require a pre-configured network and are not specialized to work with randomly deployed nodes. Shortest path algorithm is the simplest and most energy efficient algorithm. It requires knowledge of the whole network prior to each aggregation, but can be replaced by knowledge of hop distances at each node. Therefore, the shortest path was selected to be the backbone to CS-PAGG's routing.

4.2.2 Directed diffusion

A good aggregation method that deals with random networks is directed diffusion [14]. In directed diffusion, the base station periodically sends commands (interests) to initiate a measurement from certain nodes. While flooding the interests to all nodes in the network, routing tables (gradients) are formed to find the best path from the data source to the sink. Since directed

diffusion refreshes its routing tables based on active nodes each time it is performed, it does not get affected if some nodes become inactive. The problem with directed diffusion is that it uses interest commands for event-activated networks which are not what this work is tackling. Also, the nodes need to be application aware to choose the best path and to prevent aggregation loops. In addition to making a complicated routing algorithm, requiring the nodes to be application aware prevents more application layer communications protocols to be implemented in the network. Therefore, the method that was used in this research is a modified version of directed diffusion with implemented shortest path. The method can work with dynamically changing monitoring networks and does not require the nodes to be application aware.

The new method makes use of the fact that in most monitoring networks, the base station and the nodes do not change their places until they die or get damaged. Any new nodes that are introduced to the network at later instances are also likely to stay in the same position they get deployed at until their lifetimes are over. The main advantage of such property is the ability to construct routing tables only when there are changes in the network that are caused by adding and removing nodes. When a new network is deployed, all the nodes remain active until commanded to start functioning. The base station then propagates introductory message (interest) to all the nodes using the first phase of normal directed diffusion. The introductory packet consists of ID (sensor's physical address) of the base station, a loop flag bit that is reset by default, and a hop distance counter that is set to one. Figure 8 shows the first step after deploying the network. In this case, nodes 4, 5, and 7 are within the range of the base station, therefore they receive the initial message first. Upon receiving the initial message, each node adds the id of the sending node to its routing table along with the hop counter value.

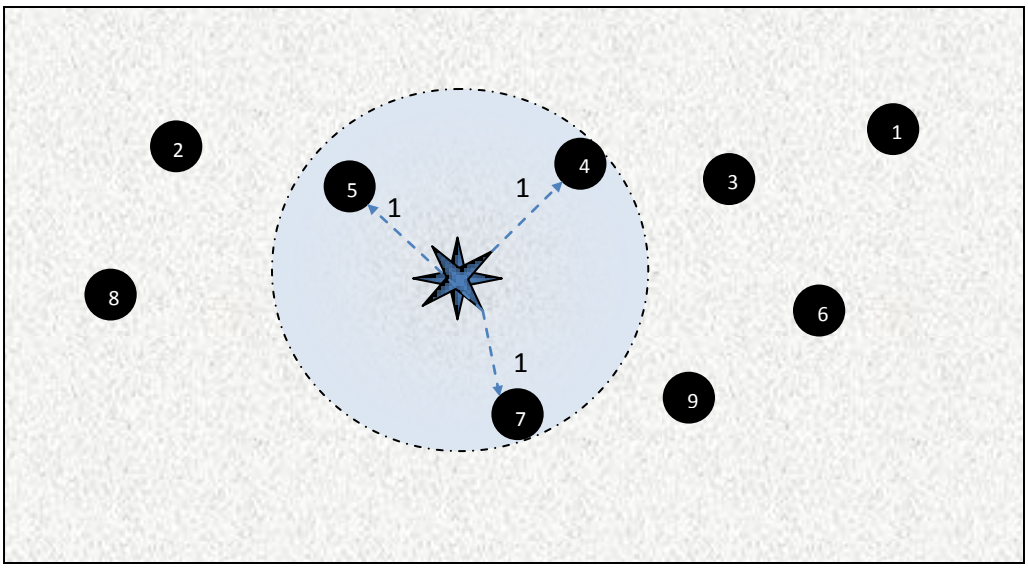


Figure 8. Initiation of directed diffusion in a small part of a network. The base station is represented with a star.

After the first stage (hop), each of the receiving nodes increases the hop counter by one then retransmits the introductory message with their own IDs instead of the base station's. The second stage of diffusion is illustrated in Figure 9.

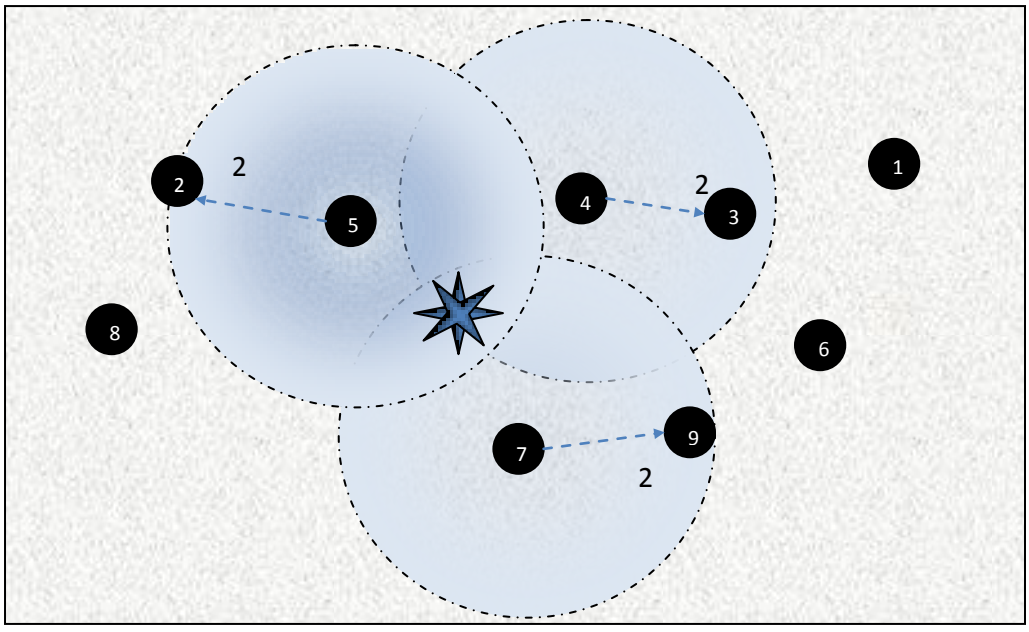


Figure 9. Second stage of introductory diffusion.

The routing tables of nodes 2, 3, and 9 are now updated with the information from the previous stage. The process continues until all nodes in the network receive and broadcast the introductory message, assuming that there exist at least one node in the communication range of each node, and the network is connected when all nodes are active. Some methods deal with nodes that are out of range from other nodes and are not connected to the main network, but that was not within the scope of this research.

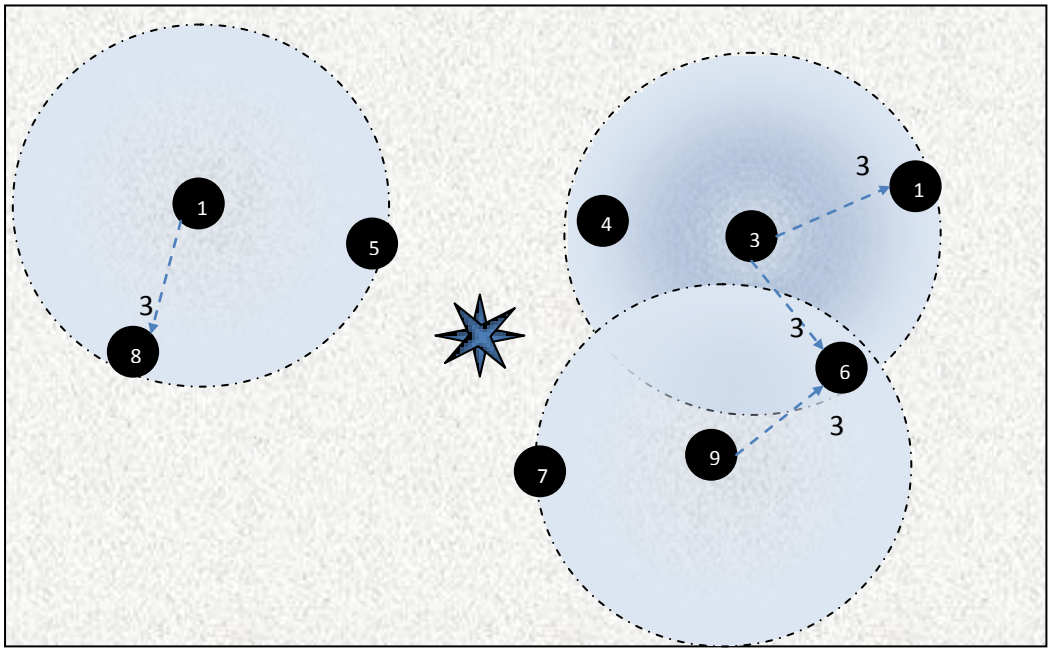


Figure 10. Third stage of introductory diffusion.

At the time instance shown in Figure 10 above, the third stage of introductory messaging is under processing. Node 6 receives two introductory messages with the same number of hops. Therefore, when stage 4 is performed and node 6 transmits its introductory message to nodes 3 and 9, they will both ignore it in order to prevent loops, despite the fact that the message has two different paths. In order to solve such conflicts, the loop flag is set in introductory messages whenever a node receives more than one introductory messages coming from different nodes but

with the same hop distance. When nodes 9 and 3 receive the introductory message from node 6 that has a distance of 4, they check the loop flag. When they find that it is set, they know that there are other possible paths through the same node and they add the incoming message to their routing matrices accordingly.

Since all nodes send only one introductory message when they get their first introductory message, the number of hops forwarded to the neighboring nodes is always the lowest possible. Such method was used in order to prevent infinite loops and to ensure that when all nodes are active, the shortest path to the base station can always be followed.

After all nodes broadcast the introductory message, the diffusion stops and the routing matrix of each node are formed. Each routing matrix consists of a list of all the nodes that sent an introductory message along with the hop distance of each one. The routing table of the network is a table that contains all routing matrices from all nodes. The routing table does not exist in the real network as a whole. Instead, each node saves its own routing matrix. The routing table of the previous network is shown in Table 1 for the sake of illustration.

Node 1 routing matrix	**Node ID**	Node 3	Node 6	
	Distance	3	4	
Node 2 routing matrix	**Node ID**	Node 5		
	Distance	2		
Node 3 routing matrix	**Node ID**	Node 4	Node 6	Node 1
	Distance	2	4	5
Node 4 routing matrix	**Node ID**	Base Station		
	Distance	1		
Node 5 routing matrix	**Node ID**	Base Station		
	Distance	1		
Node 6 routing matrix	**Node ID**	Node 9	Node 3	Node 1
	Distance	3	3	4
Node 7 routing matrix	**Node ID**	Base Station		
	Distance	1		
Node 8 routing matrix	**Node ID**	Node 2		
	Distance	3		
Node 9 routing matrix	**Node ID**	Node 7	Node 6	
	Distance	2	4	

Table 1. Routing table of all nodes in the network.

4.2.3 Neighbor discovery algorithm

Because of having a certain duty cycle, not all the nodes in the network are awake all the time. At any instance, there are only $P^{awake} * N$ awake nodes. When the initial diffusion is performed and the network forms its routing table, all the nodes are awake. Therefore, if a route is chosen based on the routing table, some of the nodes on the route might not be available. A neighbor discovery and efficient routing protocol is necessary for the network to function under such circumstances. This research introduces hello routing algorithm that brings solutions to the previously mentioned issues. The algorithm simply checks for active neighbors before a node plans the data route through small size discovery packets that are called hello messages. Hello messages are divided into two types, inquiry hellos and reply hellos. Inquiry hellos consist of one flag bit. Reply hellos hold the ID of the nodes that send them. The hello routing algorithm works as follows:

1. Node A is trying to forward a measurement to the base station. It broadcasts an inquiry hello message and waits for replies.
2. The awake neighbors of node A (Δ) receive the broadcast and prepare reply hello messages.
3. Each node belonging to Δ generates a random back off time with a maximum of T_{Max} then they send the message after waiting for that time.
4. Node A waits for T_{Max} after the initial inquiry hello message in order to receive all the reply hello messages from all the active neighbors.
5. Node A checks the hop distance of each ID it received from the reply messages and chooses node B with the minimum available distance in its routing matrix.
6. Node A forwards the measurement to node B after adding its own value to it.

7. Node B starts an inquiry hello message again. The process continues until the measurement reaches the base station.

Each node remains awake for at least T_{Max} after it replies to a hello message. That guarantees that nodes will not turn off while the measurement is being sent to them.

Since each reply hello message does not have specific information about the node it is replying to, the information gained can be used by other broadcasting nodes as well without the need to broadcast their initial request. For example, in a case where node A is trying to forward a measurement and node B is the neighbor with minimum hop distance from the base station. If node A listens to a broadcast from B within T_{Max} then it can directly forward the measurement to node B without the need to inquire for available neighbors. Such protocol greatly reduces the traffic and packet collision rate in the network.

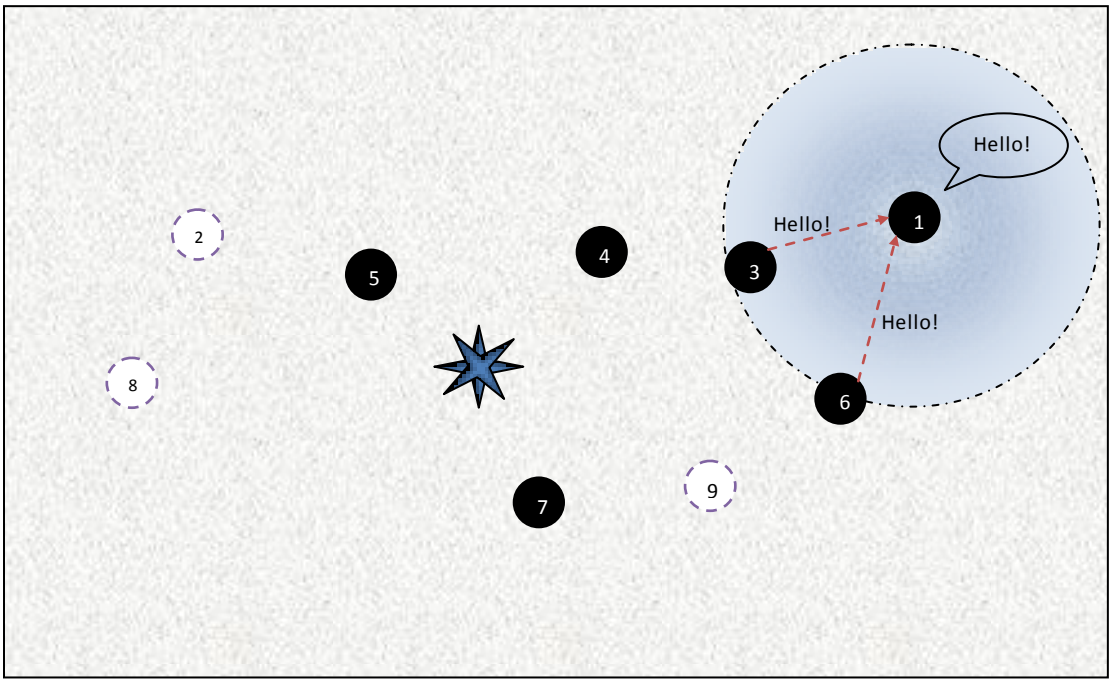


Figure 11. Inquiry hello broadcasting. Black nodes are awake and white nodes are sleeping.

In the network in Figure 8, if node 1 decides to start a measurement, it will broadcast a hello message to all its active neighbors (nodes 6 and 3) as shown in Figure 11. Then, nodes 3 and 6 reply back confirming that they are awake. Node 1 finally looks at its routing matrix shown in Table 1 and decides to forward the measurement to node 3 (3 hops) rather than node 6 (4 hops). The aggregation then continues through nodes 3 and 4 before reaching the base station.

4.2.4 Aggregation initiation

Once all the nodes have their routing matrices formed, the network can start functioning and aggregating data to the base station for signal reconstruction. Each monitored signal requires a certain number of measurements to construct a compressed sensing Φ matrix. Earlier in Equation 6 it was shown that a base station needs M measurements to reconstruct the signal accurately. In order to implement time in the process, each node at multiples of τ_c decides to start a measurement with probability $\frac{M}{N*P^{awake}}$. The value of τ_c is equal to $1/f$ where f is the frequency at which the network is refreshed which is usually higher than the maximum frequency of the monitored signal over time as explained earlier.

The algorithm works as follows for each node:

1. Wait for τ_c seconds then check node's status. If awake proceed to 2. If sleeping stay at 1
2. Generate a random number $Rand$ between 0 and 1
3. If $Rand > \frac{M}{N*P^{Awake}}$, go to 1
4. If $Rand < \frac{M}{N*P^{Awake}}$, start a measurement

The previous algorithm results in M measurements initiated every τ_c seconds. The total number of measurements initiated per second is:

$$\frac{M}{\tau_c} = f * M = \mu$$

Equation 10

Since the measurements are based on probability, there is no need for the nodes to be synchronized. Each node has its own timer and measures its own τ_c according to it. The algorithm is set to work with active node only, therefore when a node is in sleeping mode, it does not need to wake up and check for measurement initiation at τ intervals. The way the method works is that when a node is finally active, it checks how much time has passed since its last τ_c check and waits till the next multiple of τ_c happens before deciding whether to start a measurement or not. It is important that the nodes' activity priority is given to sleep scheduling rather than to τ_c monitoring. Such prioritizing makes sure that the nodes look for multiples of τ_c in their internal clock only when they are awake and active. Figure 12 shows how nodes make decisions about starting a measurement only when they are awake. For each of the instances shown in (b), the node runs the probabilistic algorithm of starting a measurement, giving the node a probability of $\frac{M}{N*P^{Awake}}$ of starting a measurement.

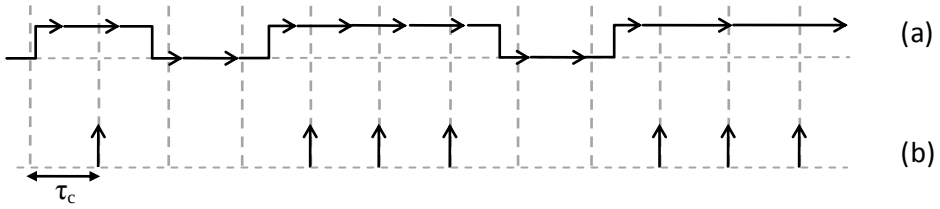


Figure 12. (a) shows duty cycle of a node and (b) shows the instances at which the node makes decisions regarding measurement initiation.

4.2.5 Aggregation path

After a node decides to start a measurement, the aggregation path of compressed sensing begins. The aggregation path that measurements follow has a huge impact on the quality, speed, and accuracy of signal reconstruction. Each node on the way of an aggregation becomes a part of the measurement matrix Φ that is used for signal recovery. Random binary matrices were proven to satisfy compressed sensing requirements with minimum complexity [11], thus, the way the data is aggregated in this method is based on random binary sparse matrices. An aggregated measurement consists of one compressed sensing value along with an aggregation index that has the IDs of all participating nodes along the aggregation path. When a node receives an aggregation packet, it sums its own sensor's value with the measurement's value and adds its own ID to the aggregation index. The node then forwards the aggregated measurement to the next node. Figure 13 visually shows a network with three measurements being aggregated towards the base station in the center.

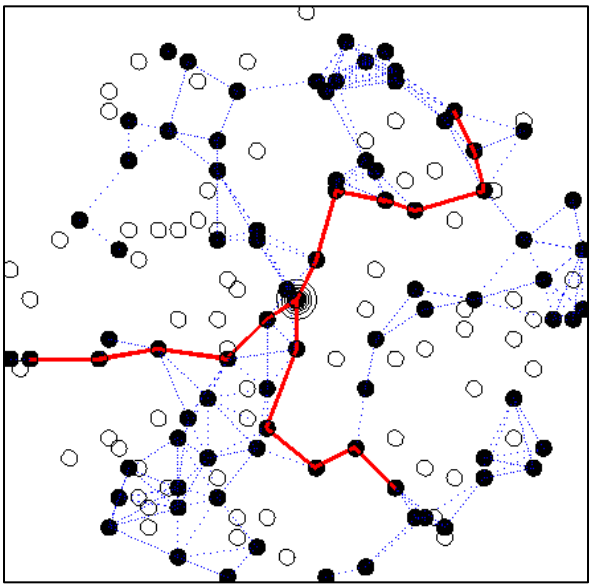


Figure 13. A network with active nodes (Black), sleeping nodes (White), and some aggregation paths (red lines).

Since the binary measurement matrix is constructed by adding the values of all the nodes involved in the aggregation, each row in the measurement matrix is the result of one aggregation. In the previous example mentioned in page 25, the aggregation passes through nodes 1, 3, and 4. Therefore, the measurement value is $X_1 + X_3 + X_4$. If a total of four measurements are taken from the network as shown in Figure 14, Equation 4 will become:

$$\begin{bmatrix} Y_1 = X_1 + X_3 + X_4 \\ Y_2 = X_6 + X_3 + X_4 \\ Y_3 = X_9 + X_7 \\ Y_4 = X_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ X_9 \end{bmatrix}$$

Where Φ is a sparse binary matrix constructed based on the path of each aggregation.

If M aggregations are initiated and gathered at the base station, the signal $\underline{X}$ can be reconstructed accurately.

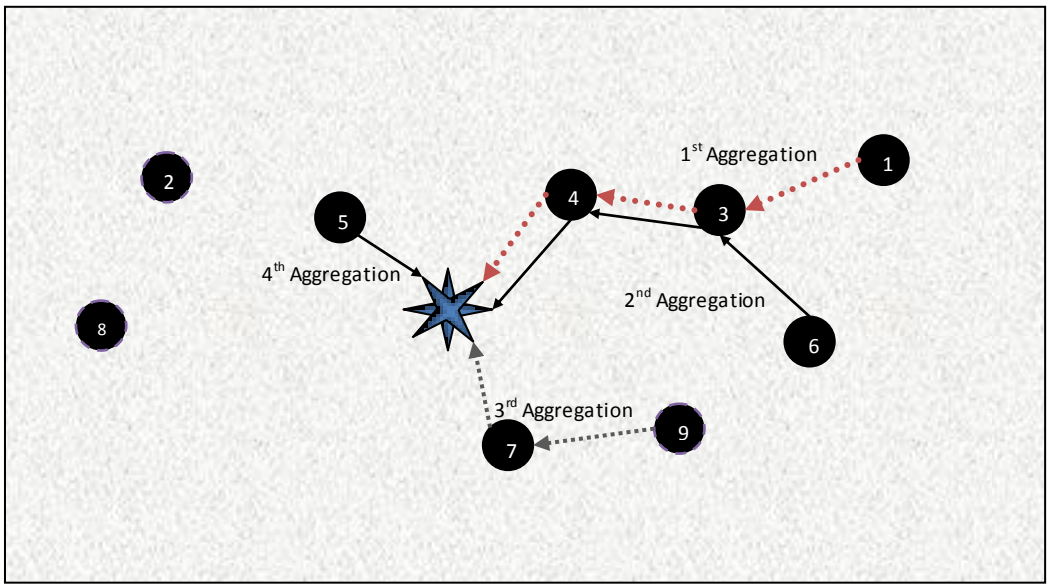


Figure 14 Four random aggregation paths. Each set of arrows represents one aggregation

4.3 TRANSMITTED PACKETS IN WSN

4.3.1 Background

When dealing with wireless networks, transmitted packets play a big role in building the characteristics of the network. The number of transmitted packets and the size of them are linearly correlated to the energy consumption of the network. The average number of hops in multi hop networks has a direct effect on the energy consumption because it affects the number of transmitted packets. Each one hop in the aggregation path needs a certain number of packet transmissions. Therefore, nodes which are far from the base station need more hops than close nodes, and as a result, need more energy. Since this work focuses on energy saving for wireless sensor networks, the number of transmitted packets in the network had to be calculated mathematically and by the use of simulation.

4.3.2 Hop distance in shortest path algorithm

Due to having all possible routes in the routing table of a network, when all nodes are awake and active, the shortest path can easily be chosen for any aggregation initiated from any node. To follow the shortest route for a measurement, each node has to forward its messages to the neighbor that has the least number of hop distances from the base station. Therefore, when P^{awake} is 1, the network acts like a normal shortest path network. When P^{awake} is not equal to 1, the shortest path might not be available and measurements might have to take different paths. The number of hops per measurement is a very important parameter of the network. It affects the number of packet transmissions directly as for each hop; there are hello messages along with actual data readings that are transmitted. Therefore the energy spent increases for every additional hop in the network. On the other hand, since each measurement is a row in the Φ matrix, less hops per measurement means fewer entries in the Φ matrix and leads to a less accurate

reconstruction of the signal. The least number of hops per measurement can be achieved when using the plain shortest path algorithm, or when P^{awake} is 1.

The number of hops for each measurement depends on location of the measurement source, location of the base station, radius of communication, and density of the network. The work in [17] proposes a theoretical way to evaluate the average number of hops of networks that have randomly deployed nodes. The method calculates the probability of reaching a node using a designed number of hops based on the distance and communication range of nodes. Basically when the communication range of each node is r, the probability of reaching a node that is Z distance away from the sensing node using one hop only is:

$$P_1(Z) = \begin{cases} 1 & if\ Z < r \\ 0 & if\ Z > r \end{cases}$$

Equation 11

For higher numbers of hop distances, the method works in reverse by finding the probability of not being able to make a connection using n number of hops, and then subtracting that from 1.

The reverse probability is evaluated iteratively. If the probability of not being able to make a connection through a station within an area A was denoted by q(R), and the area increased by dA by increasing R as shown in , then the probability of successfully being able to reach the destination within dA would be:

$$\frac{Q}{\pi r^2}\left[P_{n-1}(R) - P_{n-2}(R)\right] 2R\theta\, dR$$

Equation 12

And

$$\theta = cos^{-1}\frac{Z^2 + R^2 - r^2}{2ZR}$$

Equation 13

where Q is the number of neighbors for each node.

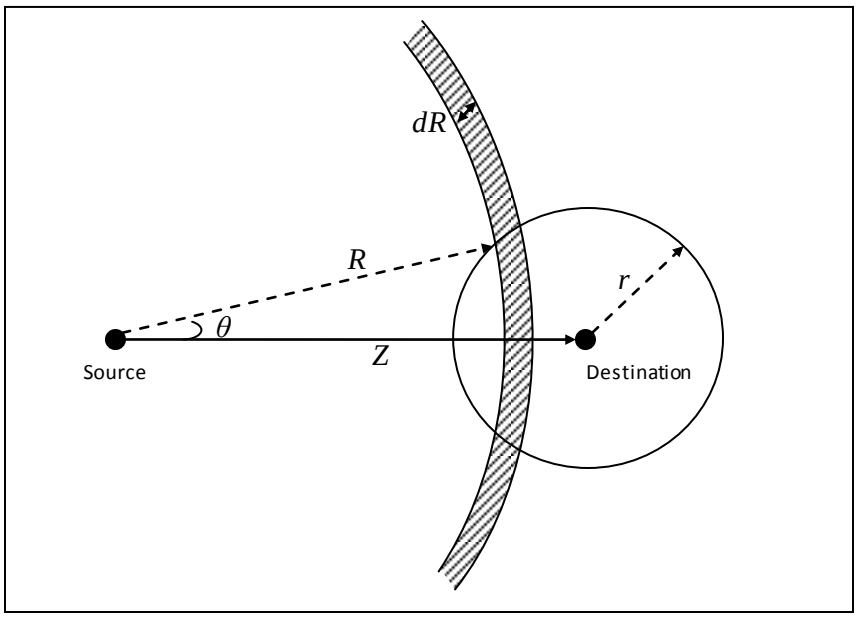


Figure 15. Measuring the change in probability of reaching a distance of Z using only n hops when R changes by dR.

Figure 15 shows all the parameters involved when calculating the distance from the shown source to the shown destination that are Z distance apart. The method works based on circular expansion. Therefore, R represents the integral path and dR is represented by a small change in the radius of the integral R. All nodes that lie within dR have the same average hop distance from the source.

The probability of not reaching any node within $R + dR$ using n hops therefore becomes:

$$q(R) + dq(R) = q(R)\left\{\frac{Q}{\pi r^2}\left[P_{n-1}(R) - P_{n-2}(R)\right] 2R\theta\, dR\right\}$$

Equation 14

In order to find the total probability to make a connection using a certain number of hops, Equation 14 has to be integrated for all values of R from $Z - r$ to $Z + r$ to cover all possible distances that *n* hops can reach.

The final probability of reaching Z distance using n or less hops after integration and substituting Equation 13 is:

$$P_n(Z) = 1 - [1 - P_{n-1}(Z)] * exp\left\{-\frac{2Q}{\pi r^2}\int_{Z-r}^{Z+r}[P_{n-1}(R) - P_{n-2}(R)]\, R\cos^{-1}\frac{Z^2 + R^2 - r^2}{2ZR}\, dR\right\}$$

Equation 15

The previous equation can be evaluated by using numeric integration iteratively until reaching the saturation level for each *Z*. The saturation level happens after the first probability of reaching Z using n hops reaches 1. After the saturation value of n, increasing the number of hops does not change the probability anymore, and the iterations should end. The same work in [17] also provides a formula to calculate the average number of hops required to reach a certain distance ignoring impossible connections that cannot be made using a certain number of hops.

The average number of hops $H(z)$ for nodes with distance Z was proven to be:

$$H(z) = Nh - \sum_{n=1}^{N-1}\frac{P_n(Z)}{P_{Nh}(Z)}$$

Equation 16

where *Nh* is the maximum number of hops allowed and $P_0(z) = 0$.

Nh can be found by monitoring the probability *Pn(Z)* for all *n* values and stopping when saturation in the probability is reached at n_{sat}. *Nh* may then be chosen to be any number of hops beyond the saturation point $Nh \geq n_{sat}$.

Since the previous formula measures the average number of hops required to reach a certain distance, the average number of hops for a network of sensors can be measured by integrating the formula along all possible distances of the nodes. Each distance in the integral should be normalized based on the ratio of nodes at that distance to all nodes in the network. The integral can be shown as:

$$Hop = \int_{Z_0}^{Z_{max}} H(Z) * Ra(Z)dZ$$

Equation 17

and

$$Ra(Z)dZ = \frac{2\pi Z dZ}{A}$$

Equation 18

In order to find the average number of hops for the whole network, the integration can be either solved numerically or by converting it to summation. For standard shaped networks with their base station in the middle and with normal node distribution, the problem can be simplified further and transformed to summation. The summation can be easily implemented by splitting the network into smaller circular sectors that are originated from the base station as shown in Figure 16.

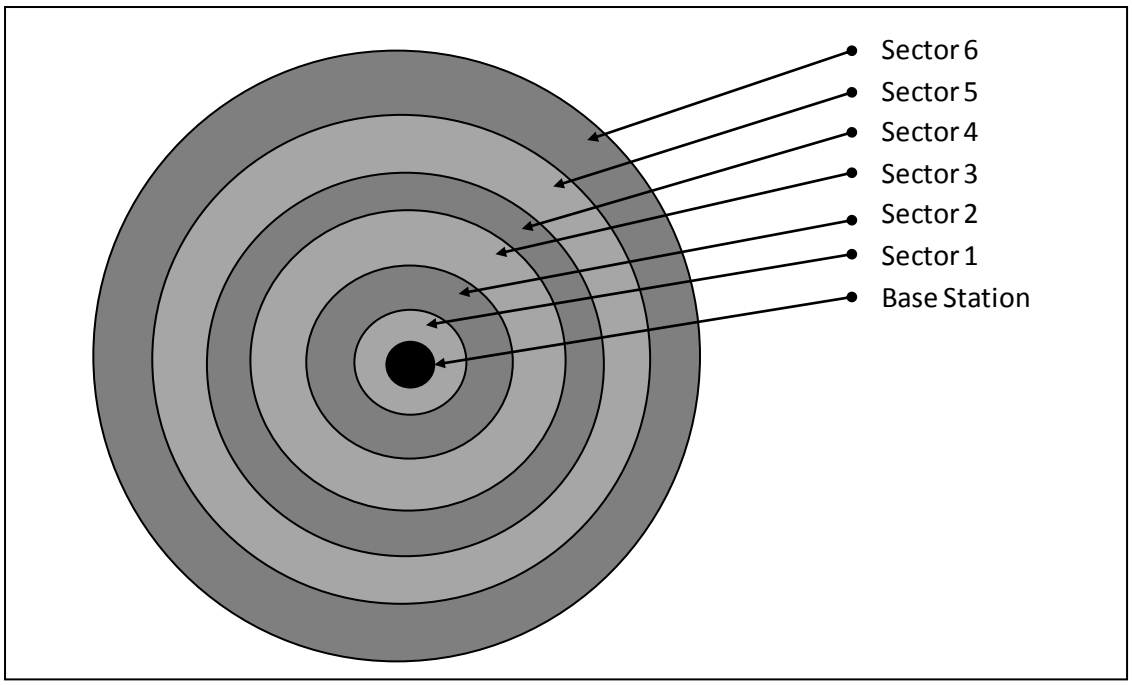


Figure 16. Summation Sectors in normal distribution circular networks with base station in the middle.

The accuracy of the sectoring method depends on the number of sectors the network is divided to. The distance between adjacent sectors decreases when more accuracy is required. The best accuracy can be obtained by applying direct integration because it has infinitely small sectors as shown in Figure 17 where each different gray level represents a sector.

As can be seen by comparing summation to integration methods from the figures, the smaller the sector increment distance gets, the better the resolution becomes, and the closer the results of the two methods are. However, sector increment distance can be chosen to be the smallest realistic size possible in order to make calculations of average hop distance easier.

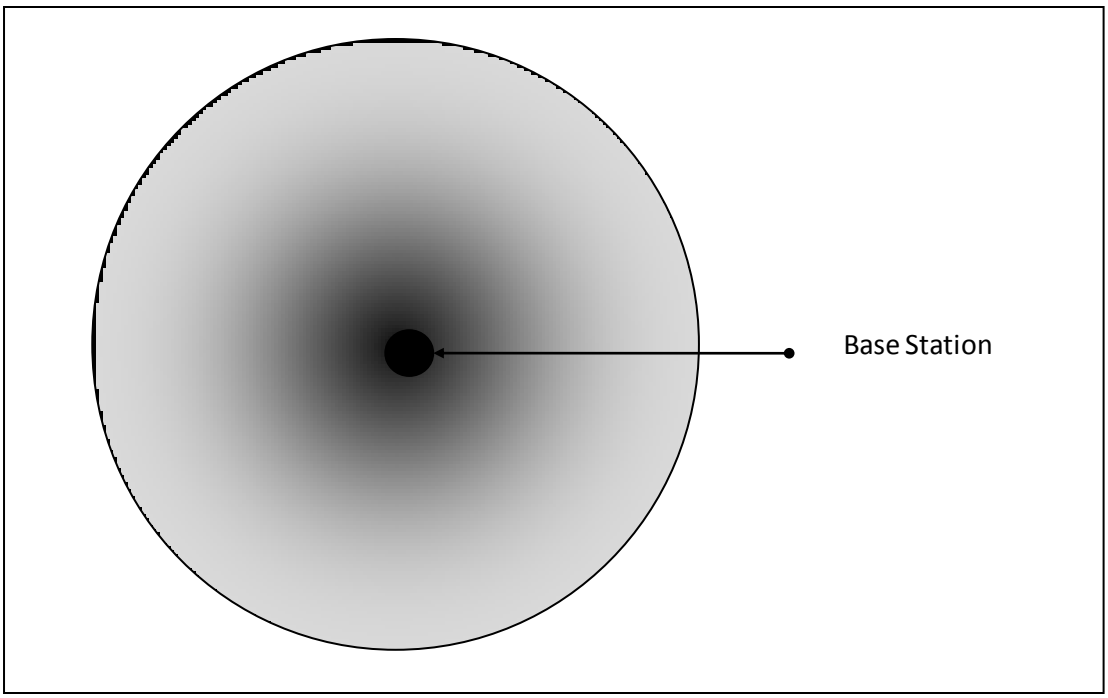


Figure 17. Infinite number of sectors in Integration method for the same network from the previous figure.

The minimum sector increment value that can be chosen should be the minimum dimension of the sensor node type that is used in the network. For instance, if a network uses sensors with the dimensions 3x2x1 Cm, the worst case distribution of the network would be if all the sensors were deployed in such orientation that their smallest dimension, which is 1Cm, lies along an imaginary line from the base station to each node. To explain the situation further, consider the network in Figure 18. There are two sensors in the network. Sensor A has its maximum dimension along an imaginary line that passes through the base station. Such orientation allows the usage of relatively large sectors, and decreases the number of sectors needed to calculate the average number of hops in the network accurately.

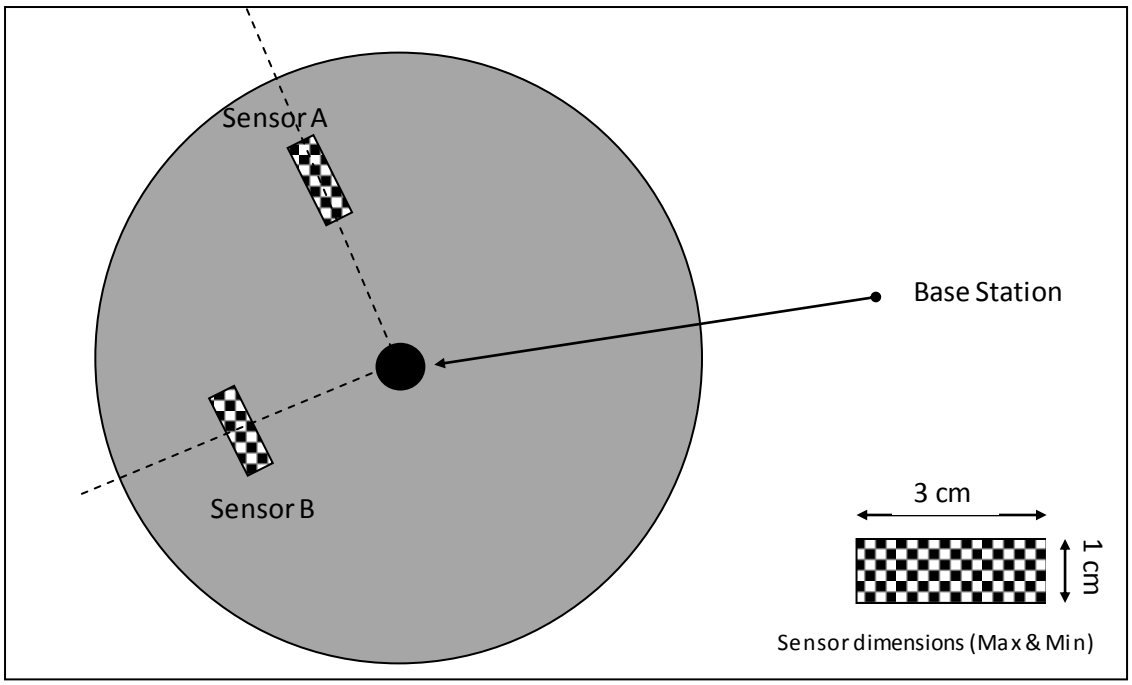


Figure 18 . Wireless network with two sensors. Sensor A has the best orientation while sensor B has the worst orientation.

Although using few sectors is sufficient for sensor A to get accurate results, it might not be the most accurate method unless all sensors are oriented the same way as sensor A. Figure 19 shows the same network divided into few sectors. The figure shows that sensor A is perfectly contained within sector 2, and therefore the distance from sensor A to the base station is the same as the one from the center of sector 2 to the base station. The same cannot be said about sensor B though. While it is still contained within sector 2, the distance from the center of sector 2 to the base station is not the same as the distance from sensor B to the base station.

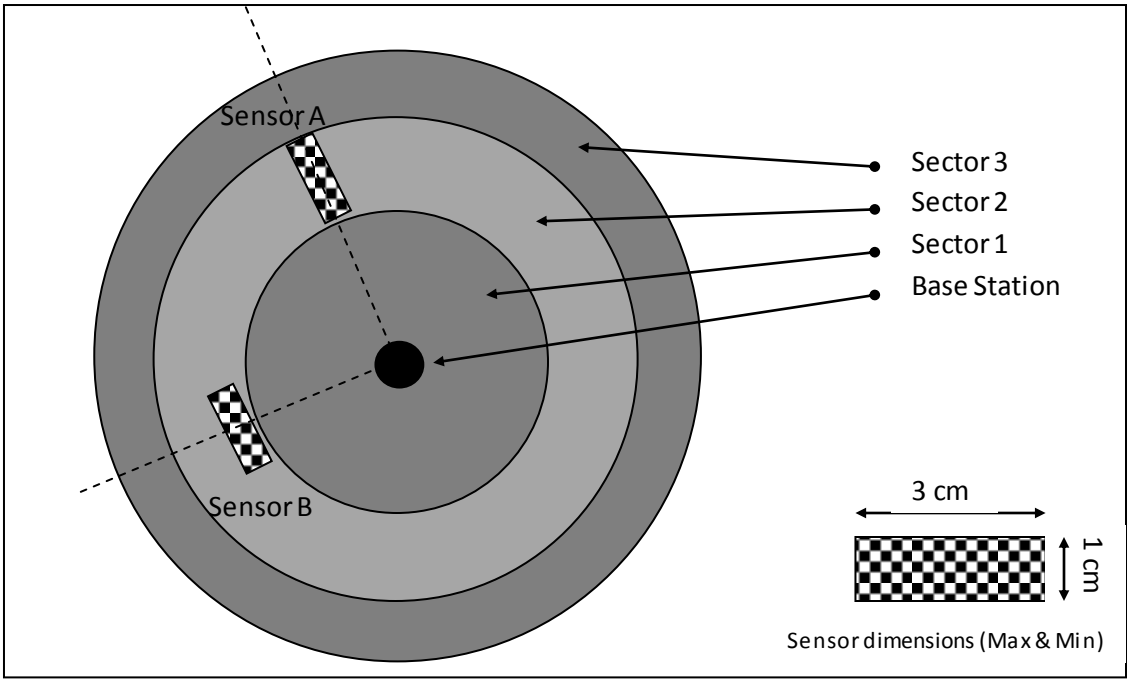


Figure 19. Large sector radius increment value. Sensor B orientation may cause some calculation error.

If the number of sectors was increased to the point that sensor B is prefectly contained within one sector, sensor A will belong to more than one sector as shown in Figure 20. All of sensor B is inside sector 2 while sensor A extends along sectors 2, 3, and 4. In such configuration, it can be said that the distance from sensor A to the base station is the same as any of the distances from the centers of the three sectors to the base station. The exact position of the radio antenna within the sensor itself could have a small effect on the results, but since most sensors are very small in size compared to their transmission range, the total effect on the average number of hops is negligible.

Going back to the example mentioned earlier, when calculating the average number of hops for the network in Figure 18, the minimum sector increment value can be chosen based on the sensor with the worst orientation in the network.

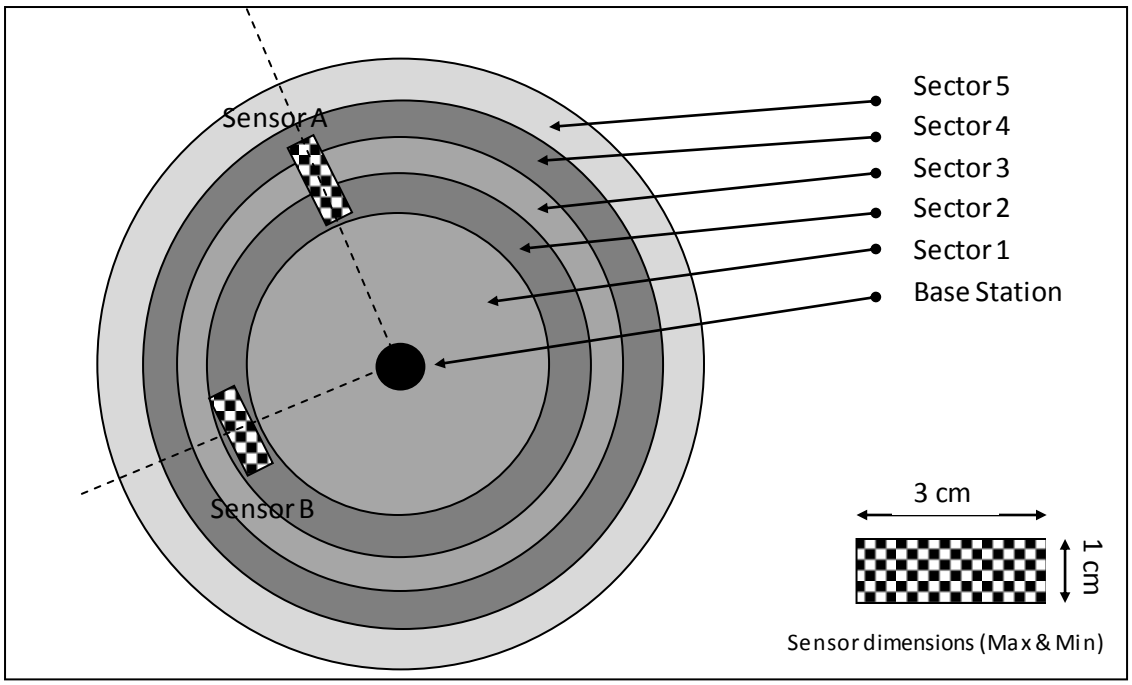


Figure 20. Best sector radius increment value. Both sensors A and B are accurately assigned to their corresponding sectors.

Since knowing the exact orientation for each sensor in the network is impractical, the worst case orientation is considered for every node in the network. Sensors with better orientation do not get affected since they belong to more than one sector. Therefore, for this particular network, taking a sector increment of 1 Cm which is the smallest dimension of the sensor node is realistic and will not cause less accuracy than the integration model.

The number of sectors N_{Sec} in the network therefore becomes:

$$N_{Sec} = \frac{Network\ radius}{Smallest\ dimension\ of\ sensor\ nodes}$$

Equation 19

For circular networks with base station in the middle, the summation model becomes:

$$Hop = \sum_{i=1}^{imax} \left[\frac{Area\ of\ sector\ i}{Area\ of\ network}\right] * Average\ hop\ distance\ for\ nodes\ in\ cluster\ i$$

Equation 20

Substituting for areas:

$$Hop = \sum_{i=1}^{imax} \left[\frac{Z(i)^2 - Z(i-1)^2}{Z_{max}{}^2}\right] * H(Z(i))$$

Equation 21

Plugging in the average hop distance from Equation 16:

$$Hop = \sum_{i=1}^{imax} \left[\frac{Z(i)^2 - Z(i-1)^2}{Z_{max}{}^2} * \left(Nh - \sum_{n=1}^{Nh-1} \frac{P_n(Z(i))}{P_N(Z(i))}\right)\right]$$

Equation 22

Since the results from Equation 16 depend on the density of the network, *Hop* can be described as a function of P^{awake}.

$$Hop\left(P^{Awake}\right) = Hop|_{Q=\frac{P^{awake} * n * r^2}{R^2}}$$

Equation 23

4.3.3 Dead measurements

In the method explained in Figure 14, each node forwards its messages to the neighbor with least hop distance among active nodes. The method might seem exactly similar to the shortest path algorithm, but close inspection reveals differences. The shortest path always has all of the nodes active at all times and therefore any messages sent at any time would go through the shortest path towards their destination and there is no chance of dropping a message. The nodes in the proposed method forward their messages to the best neighbor available at the time of transmission; however, some messages might go reach a dead end, causing the aggregation to get dropped.

Dead ends happen when the shortest path created by the original directed diffusion of the network suddenly changes due to sleeping of some nodes along the path.

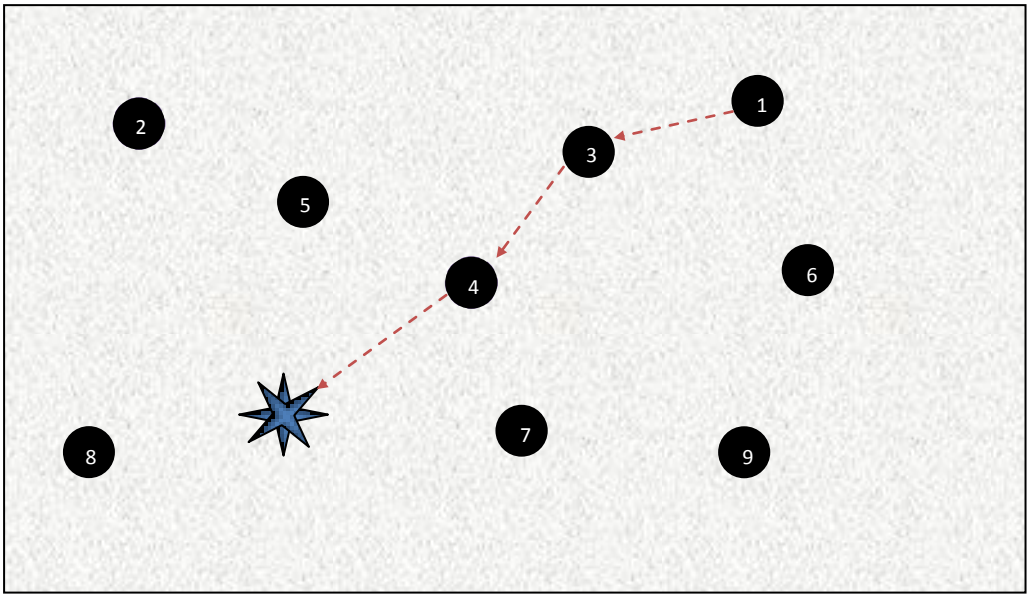


Figure 21. Shortest path from node 1 to the base station with all nodes active.

Figure 21 shows the shortest path from sensor 1 to the base station for a sensor network. The routing tables of each sensor are originally created using the previously explained directed diffusion method when all sensors are active. When sensors that are on the shortest path are

inactive, due to the hello message technique, the sending sensor chooses the shortest path available at the time which is different than the best shortest path.

Sometimes the selected route and the shortest path are similar in their number of hops required before reaching the base station. Other times, the selected path is a little longer. The ratio of length of the selected path to length of the shortest path is a function of the density of the network. More density of the network means that the selected path has a good chance of having the same number of hops as the shortest path and vice versa. That happens because the density of the network has direct impact on the number of neighbors that each sensor has if the network is deployed randomly without biasing. When each node in the network has many neighbors, the routing tables of neighboring nodes become similar because they share many nodes in common. Therefore, the average number of hops required by the shortest path to reach the base station becomes similar among nodes that are close to each other. Calculating the exact increment of average number of hops on the shortest path's average number of hops is complex and is not within the scope of this research.

In the network of Figure 21, the shortest path from node 1 is through nodes 3 and 4. The hop distance of the path is therefore 3 hops. Such path will be called the best path throughout the chapter. For the same network, if node 3 turns inactive as shown in Figure 22, the shortest path becomes a 4-hop path along nodes 6, 9, and 7. This path will be called the best available path. The best available path can be the same as the shortest path when all nodes along the shortest path are active during measurement aggregation.

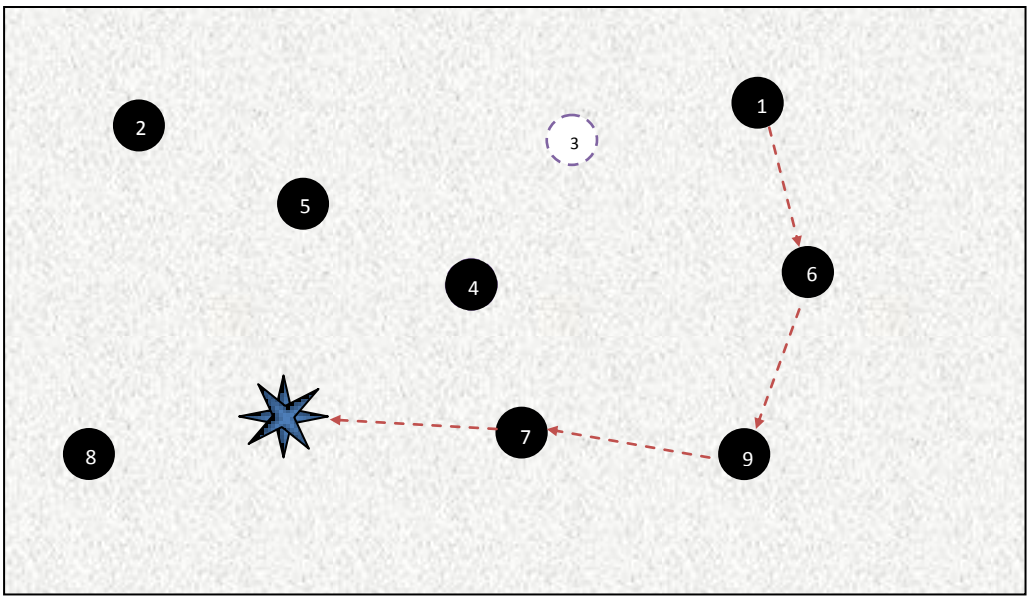


Figure 22. Best available path when node 3 is sleeping

Now consider the case when node 3 is active, but node 4 is sleeping. As shown in Figure 23, node 1 checks for neighbors, and since both nodes 3 and 6 are available, it forwards to node 3 due to having a shorter path.

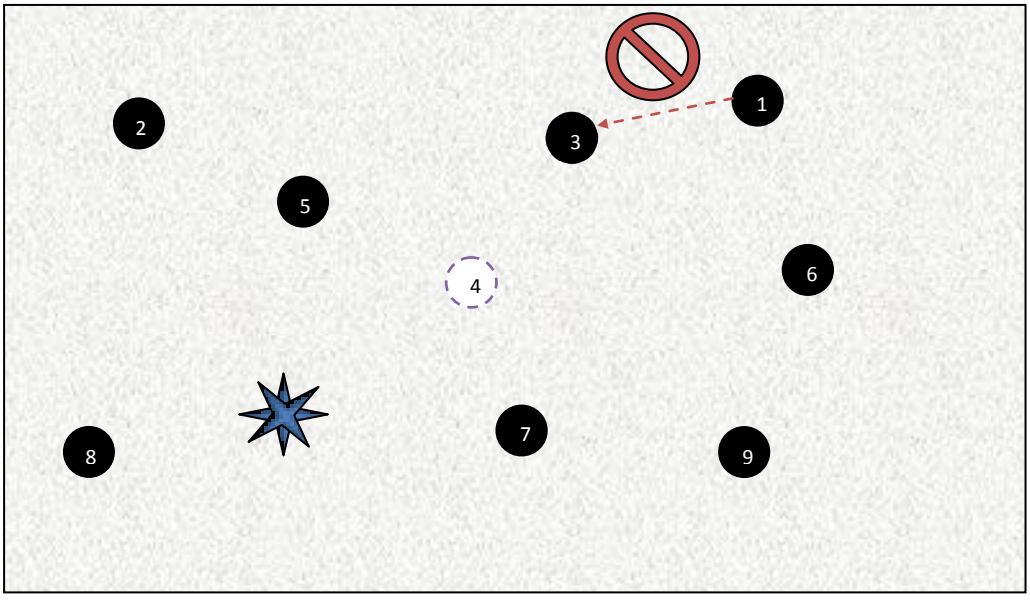


Figure 23. Dead measurement path

Obviously, the best available path is not through node 3, but since node 1 does not know that node 4 is inactive, it thinks that the best available path is through node 3. Node 3 does not have any new neighbors to forward to, so the measurement reaches a dead end and is called a dead measurement.

To reduce the number of dead measurements in the network, several methods were tested theoretically and by the use of simulation. The first group of the tested methods focuses on altering the method by which a node chooses which neighbor to forward to, and therefore prevents dead measurements before they happen. The other group works on recovering the dead measurements after they happen. Both groups will be discussed briefly then the best method will be explained in depth and compared to the original best available path method.

Dead measurements prevention group:

In this group, the nodes alter the way they choose their next node, making it more random than always choosing the shortest path, and eventually reducing the occurrence of dead ends.

1. Random selection:

In this method, nodes select their destination randomly from their list of active nodes every time they transmit. The method resulted in longer paths for measurements with no considerable reduction in the number of dead measurements.

2. Weighted selection:

In this method, nodes choose the next node based on their hop distances from the base station. Each neighbor is weighted based on its distance, then the node selects its destination from a random vector that has every neighbor weighted based on their distances. This method did better than random selection in terms of measurement length, but did not give much improvement over the plain best available path method.

Dead measurements recovery group

In this group, dead measurements are recycled into normal measurements.

1. Time freeze:

In this method, dead measurements are resent after a neighbor awakes within some time limit. The total delay caused by all dead ends along a path should be within a tolerated value in order for the received measurement to represent the measured signal accurately. If the signal changes rapidly, not much delay time is allowed and the method becomes inefficient. For some signals that change very slow, such method can be implemented to reduce most, if not all, dead measurements in the network. Since the method requires specific details and different configurations for each measured signal, it will not be discussed in details in this research, but it remains a part of our future signal-specific work.

2. GoBack:

In this method, dead measurements are simply returned to the nodes that sent them, and then are resent to the next best available node. The method resulted in a much lower dead measurement percentage on the cost of having a slightly longer aggregation path.

Since the GoBack method had the best results among the rest of the methods, it will be investigated deeply in the next section, and it will be compared to the best available path method in all aspects of reconstruction error, energy cost, simplicity, and performance. At the end of the discussion, the best method will be chosen as the default for the rest of the research.

4.4 GOBACK ALGORITHM

4.4.1 GoBack concept

The idea of the method is that the network tries to resolve the problem of having dead ends and hence, dead measurements, by retransmitting the dead packet back to its source. The method works as follows: When a node that does not have any active neighbors other than the one sending to it receives a measurement, it adds itself to the aggregation and then transmits the packet back to where it came from to avoid losing the measurement. Every re-transmitting of a packet along an old path is called one GoBack. The number of GoBacks can be long or short depending on the chain of isolated nodes. The following case in Figure 24 has only one GoBack:

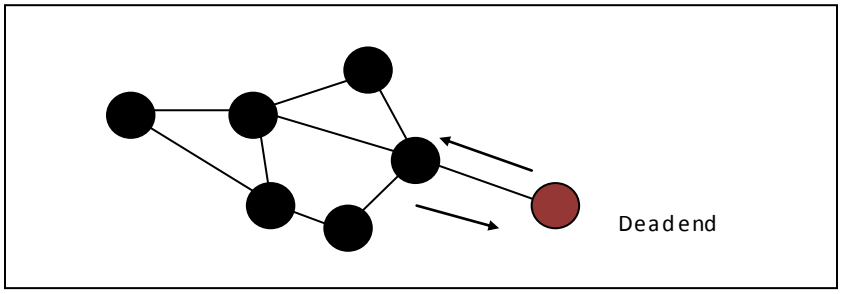


Figure 24. Active nodes (Black) in a network with a dead end (Red) located one hop away

The figure shows only the active nodes of a small sector of a network. The dead measurement happens after one hop only, and therefore needs only one GoBack to recover. Figure 25 shows another dead end that happened after three hops. The measurement in the figure needs three GoBacks in order to recover. Multiple GoBacks in a row are called GoBack chains and the number of GoBack required is called the chain's degree. For the 3-hop dead end network, the network is said to have a GoBack chain of degree 3.

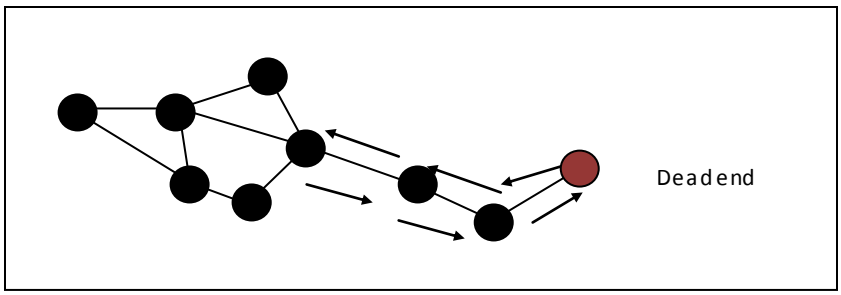


Figure 25. Active nodes (Black) in a network with a dead end (Red) located three hops away

GoBack chains do not necessarily have to happen in a row. The following case in Figure 26 demonstrates a GoBack chain that has three GoBacks in a different situation at which only two of them are in a row:

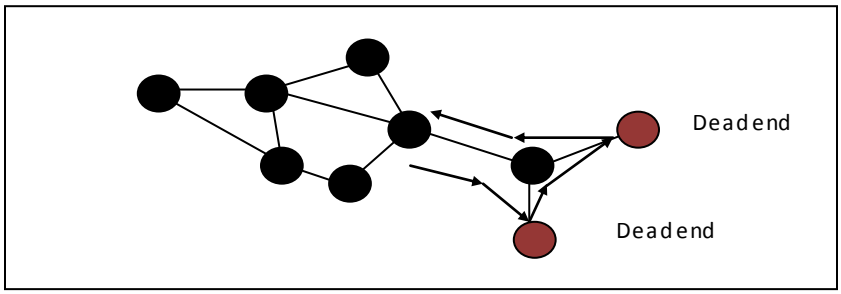


Figure 26. Active nodes (Black) in a network with two dead ends (Red) located on hop away each

GoBack algorithm works as follows:

1. Transmitting node (A) chooses the next node (B) according to best available path algorithm then forwards the measurement to node (B).
2. Node (B) receives the measurement and stores the ID of node (A) as a return address for the measurement.
3. Node (B) discovers its active neighbors using Hello messaging algorithm.
4. If node (B) has active neighbors, go to 1.

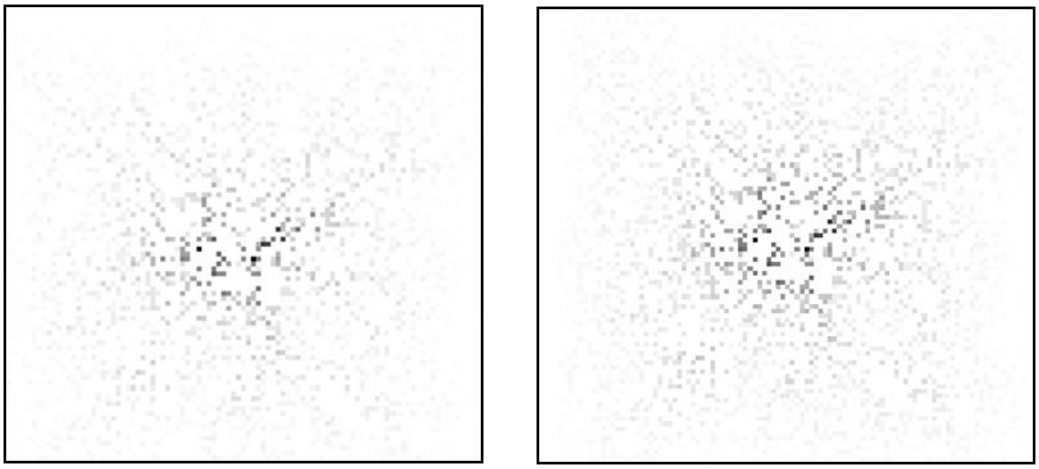
5. If node (B) does not have active neighbors, send the measurement back to its return address (A).
6. Node (A) receives the measurement back, excludes node (B) from the active neighbors, then applies hello messaging algorithm.
7. Go to 1.

Each return address is kept in the memory of the receiving node for a certain period of time depending on the highest degree of continuous GoBack chain allowed in the network. Higher back-to-back GoBack chains require the return address of each measurement to remain in memory longer. Each GoBack chain results in extra packet transmissions. For each degree, two extra transmissions are made. One sent to the dead end, and another one sent back to the source. The communication cost is therefore a function of the degree of GoBack chains in the network. Furthermore, the way that GoBack chains happen does not affect the number of packet transmissions. As a result, the communication cost of GoBack chains of the same degrees is the same regardless of having the chain in a row or not. In the next few pages, a full comparison between best available path and GoBack aggregation algorithms will be performed.

4.4.2 GoBack forwarding map

Forwarding map is a measurement of the distribution of energy consumption among nodes in a network. It is represented by measuring the number of packets that each node transmits over time. The more a node transmits, the darker it is represented on the map. A perfect situation happens when all the nodes in the map have the same gray level, meaning that the transmission energy is well distributed among all nodes. In multi hop networks, most of the energy is concentrated close to the base station since the close nodes have to re-route the traffic from far

nodes to the base station, along with their own traffic. Figure 27 shows the forwarding map of a network that has an area of 10000 distance units^2 and 3000 randomly deployed sensors.



(a) *(b)*

Figure 27. Forwarding maps for (a) Best available path, and (b) GoBack

From comparing the forwarding maps of both methods, GoBack method has a slightly better energy distribution than best available path. The little improvement is due to the re-routing of dead measurements. When a dead measurement is sent back to the sender, the sender will retransmit the measurement to a longer route since the shortest route is not available. Therefore, if many measurements are sent back, the forwarding map becomes more distributed. However, the improvement in energy distribution causes more energy consumption in far nodes and does not reduce the energy consumption of nodes in the center. Therefore, some energy is lost when using GoBack method, but less dead measurements are dropped.

4.4.3 GoBack aggregation length

Aggregation length is the number of hops that belong to one aggregation from the starting point of the measurement to the base station. Average aggregation length in a network is equal to average hop distance of a network. Measurement length represents the actual length of the measurement, i.e. the number of non-zero elements in a row of the measurement matrix Φ. In an ideal case that has no dropped measurements and no GoBack retransmission, the average measurement length of a network is equal to the average aggregation length. The two are similar because each one hop results in one entry in the measurement matrix. When some measurements are dropped or some retransmissions occur, the two lengths become different.

	Avg. aggregation length	Avg. measurement length	Efficiency
GoBack	25.8	25.7	99.6%
Best available path	20.07	17.92	89.39%

Table 2. Aggregation and Measurement paths for a network that has an area of 10000 distance units^2 and 3000 randomly deployed sensors. $P^{awake} = P^{th}$.

The simulation results in Table 2 show that the aggregation and measurement length are different in both methods. In Best available path method, the difference happens because of dead measurements. Each dead measurement uses a certain number of hops; however, dead measurements do not reach the base station, do not contribute to the measurement matrix Φ, and therefore, have a measurement length of zero.

When using GoBack method, the difference between the two lengths is a result of extra hops used for GoBack retransmissions. Whenever a GoBack chain occurs, the involved nodes use extra hops for retransmissions, but contribute to the measurement matrix only once. Therefore,

the average measurement length of GoBack decreases when high degree GoBack chains happen in the network.

The table above indicates that the efficiency of the GoBack algorithm is much higher than the efficiency of the best available path method. It is also noticeable that GoBack aggregations are longer than best available path aggregations on average. The extra length happens when a dead end occurs and the aggregation goes back before continuing along a longer path to the base station. On the other hand, if best available path is selected and a dead measurement happens, the measurement is dropped and no further aggregation length accumulates.

4.4.4 GoBack dead measurements

The number of lost measurement in a network represents the average number of measurements that reach dead ends and get dropped.

	Number of lost measurements
GoBack	0.4%
Best available path	22%

Table 3. Percentage of lost measurements for a network that has an area of 10000 distance units^2 and 3000 randomly deployed sensors. $P^{awake} = P^{th}$.

Table 3 shows that when testing at $P^{awake} = P^{th}$, the number of lost measurements in best available path is much higher than that of GoBack method. The results mean that the actual number of measurements received by the base station is different than the desired number of measurements in order to recover the signal accurately.

$$M_{received} = {M_{desired}}\Big/{(1 - \frac{Percentage\ of\ lost\ measuremets}{100})}$$

Equation 24

For example, when using GoBack and 100 measurements are needed, the actual number that should be sent is 100/ (1-0.04%) = 100.4 ≈ 100.

On the other hand, if the best available path algorithm is used, then the actual number of measurements needed to be sent becomes 128.2.

Dead measurements happen only when the density of the nodes is low. Since dead measurements are affected by the number of neighbors for each node, for the same density of the network, the dead measurement rate is the lowest when $P^{awake} = 1$, meaning that all nodes are active. Decreasing P^{awake} leads to more dead measurements, but the decrement curve is not linear as Figure 28 shows.

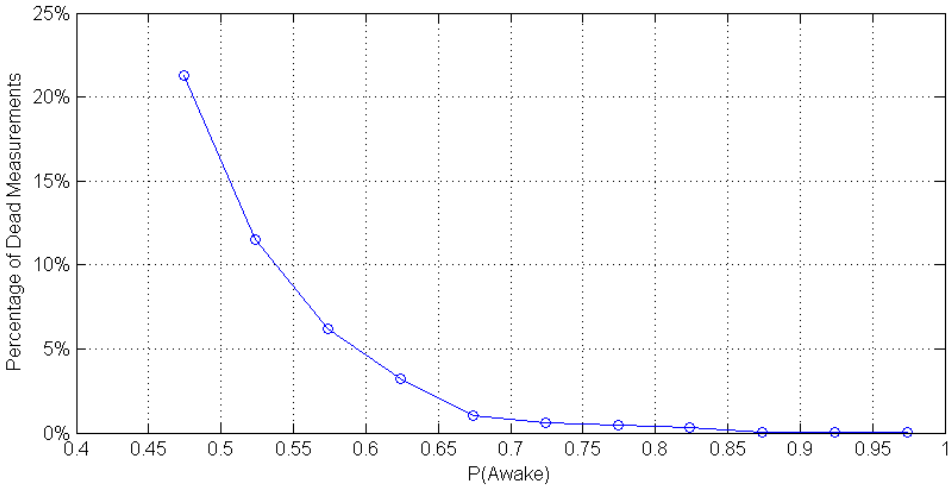


Figure 28. Change in percentage of dead measurements with P^{awake} increment for a circular network with a radius of 280m and 3000 deployed nodes. P^{th} is 0.46 with shortest available path method used.

As can be seen from the graph, the percentage of lost measurements decreases rapidly with the increment of P^{awake}. At P^{awake} = 0.66, the percentage of lost measurements reaches a negligible value of 1% only. It is also noticeable that at P^{awake} = 0.72, the efficiency of both methods becomes similar and they both get only 0.4% lost measurements.

4.4.5 GoBack Speed and complexity

Under the same conditions, both methods were tested to find out the difference in speed and complexity between the two. The GoBack method was 0.03% slower than the best available path algorithm. The speed loss is due to having to store the id of each sending node in case a GoBack occurs. Memory access delay was proven to be too small to be considered for comparison.

4.4.6 GoBack memory

Since the GoBack method keeps track of the sender's address, the nodes need to have a small memory. The best available path does not need that extra memory. More memory could mean more cost; however, since the needed memory to store a node's ID is very small, the extra cost that memory adds to GoBack is negligible.

4.4.7 GoBack communications cost

The communications cost of a network is represented by the number of transmitted packet per measurement matrix (Number of measurements * number of transmitted packets per measurement). The communication cost is significantly higher in the GoBack method since the average aggregation length is higher than the shortest path method as explained earlier. The communication cost for both cases is shown below in Figure 29.

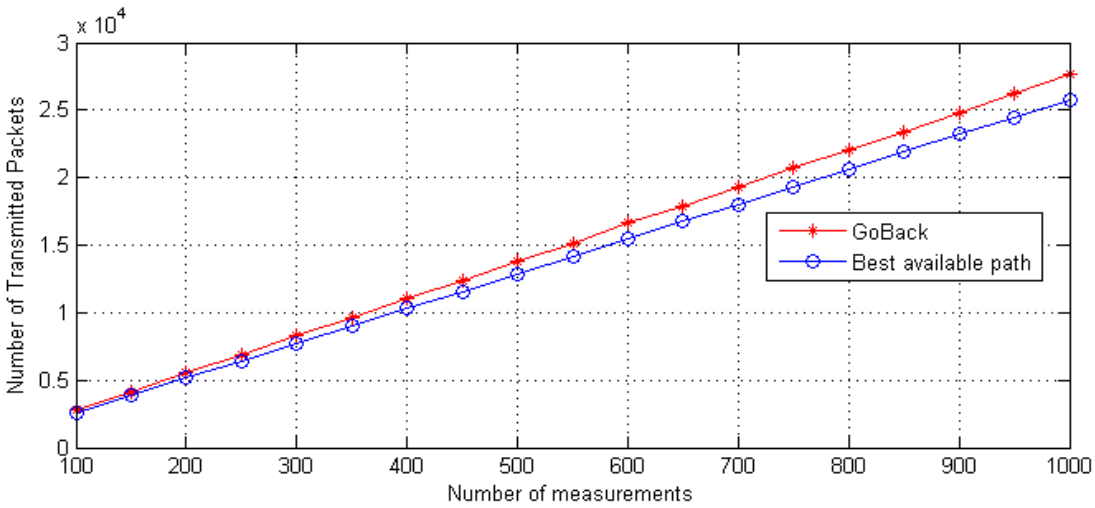


Figure 29. Comparison between GoBack and best available path in terms of number of transmitted packets per measurement for a network with a radius of 280m and 3000 randomly deployed nodes. $P^{awake} = P^{th}$

The relationship between the number of transmitted packets and the number of measurements is linear and the relationship between the two lines is the same at any number of measurements. Since the average aggregation length is a function of the density of the network as Figure 28 shows, the relationship between the two lines in Figure 29 changes when P^{awake} changes. When P^{awake} increases, the number of dead measurements becomes the same for both methods, affecting the difference between the two lines as Figure 30 illustrates.

As can be seen from the figure, GoBack method always needs more number of hops than best available path method in order to aggregate a measurement. The increment in the number of hops needed for aggregation increases the number of packets that need to be transmitted per measurement, and eventually increases energy consumption of the network. Therefore, GoBack performs worse than best available path in terms of average number of transmitted packet per measurement. On the other hand, since best available path method has more dead measurements than GoBack, the energy saved from less number of transmitted packets could get lost due to dead measurements. A comparison between the total energy consumption is therefore needed to find the best operation environment for each method.

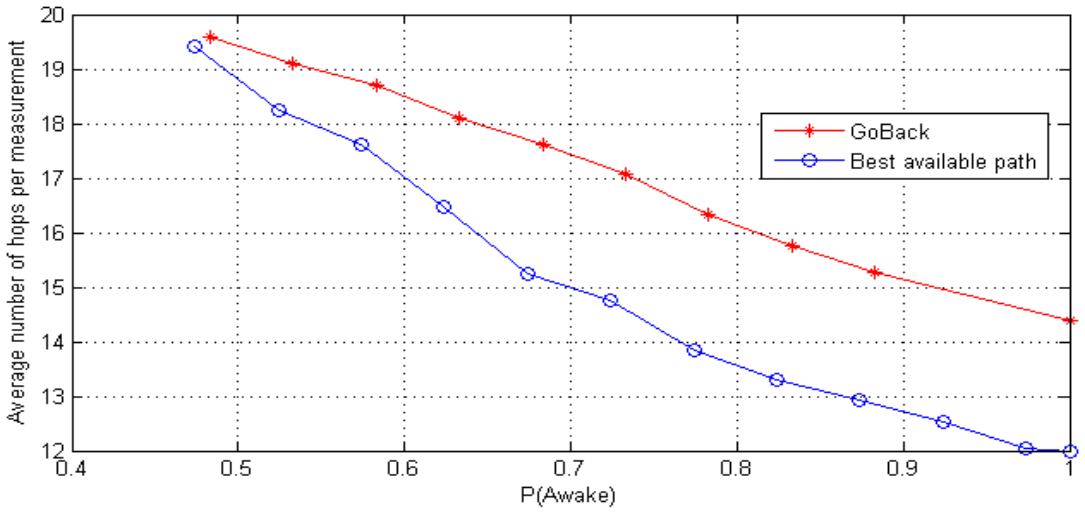


Figure 30. Average number of hops per measurement for a network with a radius of 280m and 3000 randomly deployed nodes

Since the GoBack method has longer aggregations and measurements than best available path, more non-zero values in the measurement matrix are expected. As a result, the expected reconstruction error is affected.

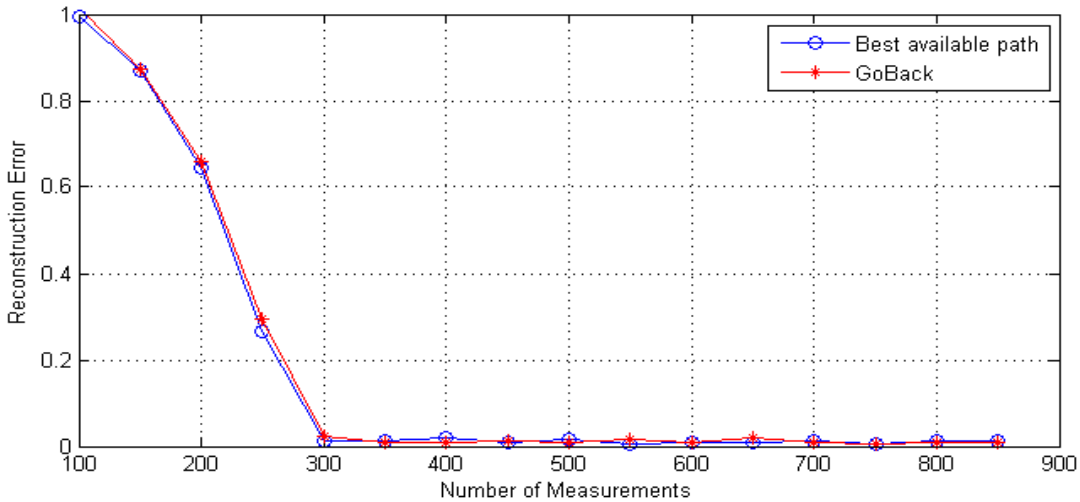


Figure 31. Normalized reconstruction error for a 50-sparse signal using DCT sparsifying matrix and norm 1 minimization

The results in Figure 31 show that although the GoBack method has more non-zeros in its measurement matrix, both methods produce very similar normalized reconstruction error. The

reason behind the results is that GoBack results in more dense measurement matrix; however, the extra entries induced by GoBack are highly correlated to other entries in the matrix since they GoBack packets are retransmitted to close neighbors.

When DCT is used as sparsifying matrix, the measurement matrix should not be dense in order to satisfy RIP conditions. Adding few correlated entries in the measurement matrix while using DCT sparsifying matrix does not contribute to better reconstruction. Therefore, new nodes that are added to the aggregation because of the GoBack algorithm do not noticeably affect the signal reconstruction process.

Other sparsifying matrices like Wavelet are sparse and need dense measurement matrices in order to satisfy RIP conditions. If the monitored signal is compressed better when using such matrices, the reconstruction error of GoBack method might be affected to the positive side. The signals that this research deals with are natural slow-changing signals and work better when DCT sparsifying matrix is used. Therefore, full analysis of GoBack under other sparsifying matrices than DCT is beyond the scope of this thesis.

Before comparing the total energy consumption for the two methods, a more efficient version of the GoBack method will be explained. The efficient version will be test against the best available path to find out if dead best available path is not the most efficient method.

In order to optimize GoBack, the average length of GoBack chains should be defined. The histogram in Figure 32 shows the probability distribution function of having a certain degree of GoBack streak in a network when $P^{awake} = P^{th}$.

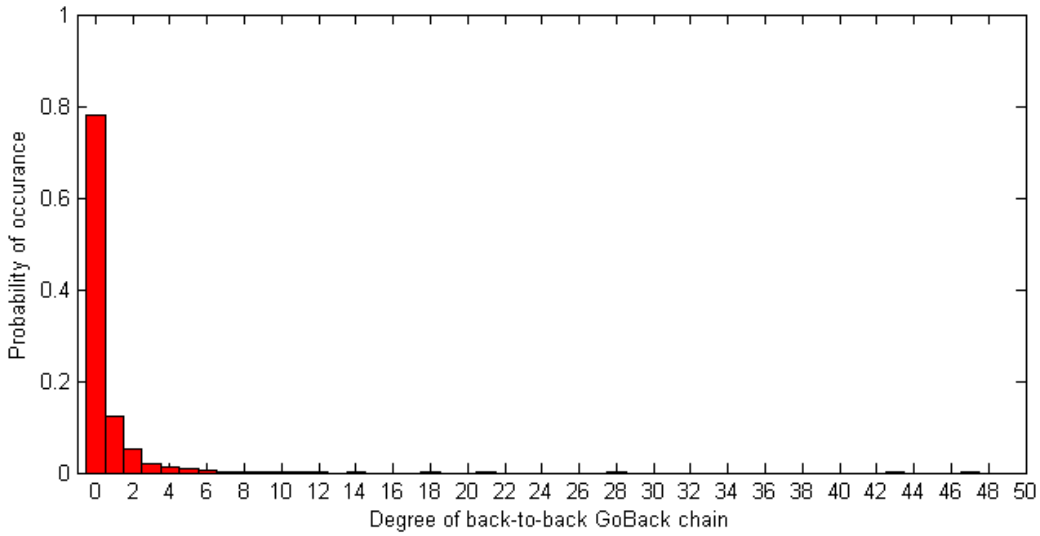


Figure 32. Probability of back-to-back GoBack chains when $P^{awake} = P^{th}$

GoBack chain of degree zero means that dead measurements cannot be retransmitted. Therefore, GoBack chain of length zero represents best available path algorithm. The probability distribution function shows that when GoBack is not allowed, the probability of reaching the base station is 0.78 which is the probability of non-dead measurement from Table 3. After allowing GoBack with one degree, the probability of reaching the base station rises to 0.9049. Raising the GoBack degree threshold to 4 allows the recovery of 99% of the measurements and leaves only 1% dead measurements. It is important to notice that although GoBack degrees of over 4 do not happen frequently, they consume a lot of energy when they do. For instance, one GoBack chain of degree 50 consumes the same energy as 50 GoBack chains of degree 1, but recovers only one measurement instead of 50. Therefore, GoBack degree threshold should be assigned to the network in order to get the most efficiency possible. The average number of packet per measurement increases when GoBack threshold increases as Figure 33 illustrates. It can be seen that although only 1% of the measurements need a GoBack threshold that is larger than 4, the increment they induce on the average number of hops is significant.

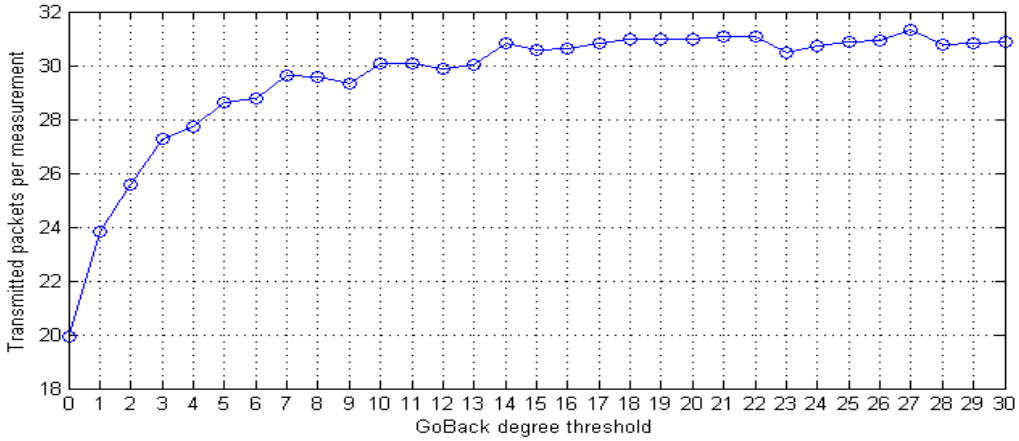


Figure 33. Relationship between average number of hops per measurement and the maximum allowed GoBack degree

Figure 34 further explains the situation by showing the behavior of normalized reconstruction error when GoBack threshold changes. The error decreases when the threshold increases due to having less dead measurements and more entries in the measurement matrix, which leads to a better reconstruction for the signal.

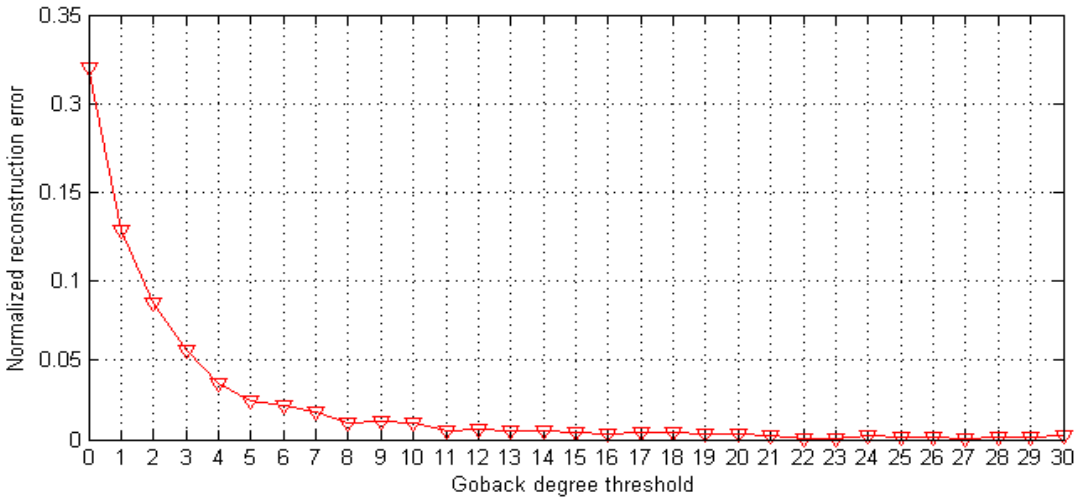


Figure 34. Normalized error for each GoBack threshold using DCT as sparsifying matrix

Supporting the conclusions from Figure 33, the previous figure shows that the majority of the change in reconstruction error happens in the first four GoBack thresholds. Any larger

GoBack threshold increases reconstruction accuracy by less than 5% only, but significantly increases the average number of hops required per measurement which in turn increases energy consumption. Same reconstruction accuracy can be achieved more efficiently by increasing the number of measurements instead of increasing GoBack threshold.

From the previous conclusions and explanations, it can be said that GoBack is most efficient when GoBack thresholding is used. In the particular settings that were used in a network of 3000 sensors deployed randomly over 10000 distance units^2, and using DCT as the sparsifying matrix, the most efficient GoBack threshold was found to be 4. The optimum threshold value does not change when the network settings are changed assuming always having a connected network. The reason behind that is that the value of the optimum threshold depends only on the density of nodes in a network rather than the absolute number of nodes in the network. Since the testing was performed when $P^{awake} = P^{th}$, the density of the network is the least possible density that prevents nodes from being disconnected. Optimum threshold decreases only when node density increases. Eventually, the optimum threshold becomes reaches zero when the network becomes very dense, meaning that the best available path method is the most efficient method between the two when the network is dense enough.

When operating in lower node density and lower P^{awake} than 1, the total energy consumption for both methods had to be tested to find out whether best available path has better efficiency only in dense networks. Since the least density of the network that keeps it connected happens when P^{awake} is equal to P^{th}, the GoBack thresholds that were used for comparison were the most efficient ones, which are zero through 4. Best available path is easily represented by GoBack threshold of zero.

The total number of packets required to get the same accuracy for each method is different than the average number of packets per measurement for each method. The total number

of packets depends on the percentage of dead measurements as well as the average number of packets per measurement. In order to get *M* number of measurements, each threshold required the network to initiate more than M measurements in order to compensate for dead measurements.

By taking into consideration both dead measurements and average transmitted packets per measurement while fixing reconstruction accuracy, the total number of transmitted packets can be found using the following formula:

$$F = MA + LMA$$

Equation 25

$$F = MA(1 + L)$$

Equation 26

where

F: Total average number of transmitted packets per measurement

L: Percentage of lost measurements/100

M: Number of received measurements

A: Average number of packets per measurement

Since *M* represents the number of received measurements and the accuracy is fixed, it is the same for all threshold levels and can be considered as a constant.

When GoBack threshold value decreases, the number of packets per measurement decreases. Decreasing the threshold also leads to increasing the percentage of dead measurements. However, the calculated total number of packets per measurement required for accuracy increases as shown in Figure 35 below.

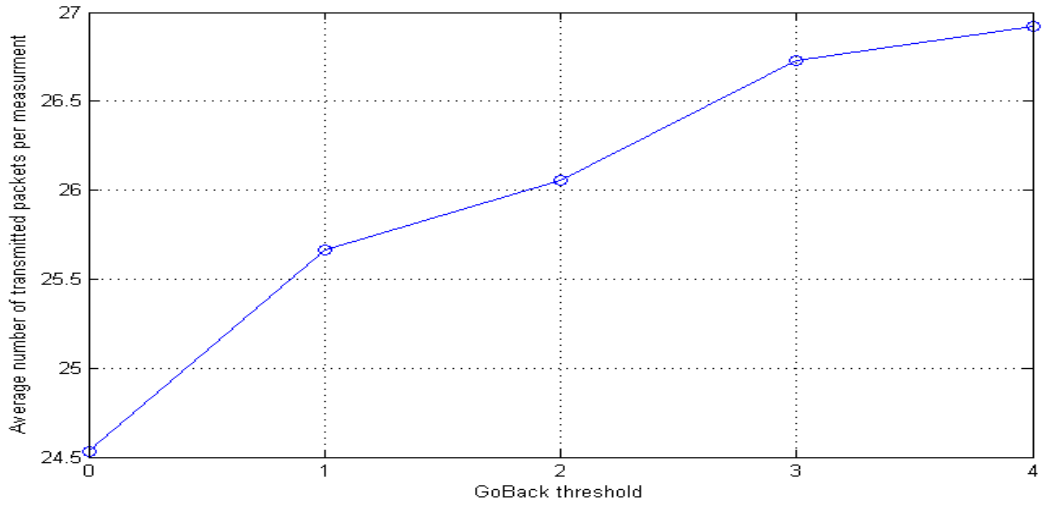


Figure 35. Total average number of transmitted packets needed from different GoBack thresholds to achieve the same signal reconstruction accuracy

The figure demonstrates that even for sparse node distributions, best available path is the most efficient method when considering the accuracy gain for the same amount of transmission energy. Increasing the density of the network will only increase the efficiency of best available path method. Therefore, the best available path is more efficient at all node density levels that maintain connectivity of the network.

As a conclusion, trying to recover dead measurements through re-routing them consumes more energy than to start a new measurement. It is better to initiate more measurements than to recover dead measurement. The best solution to dead measurement problem is to prevent them from happening by increasing the node density of the network. As Figure 28 shows, working with P^{awake} a little more than P^{th} can dramatically reduce the number of dead measurements in the network. Therefore, it is better to operate the network at a higher value than P^{th} in order to increase the efficiency of the network.

4.5 HOP DISTANCE IN CS-PAGG

The average number of hops per measurement for the shortest path algorithm was calculated in Equation 22 earlier. When dealing with the best available path algorithm, the average number of hops becomes larger due to sleeping nodes. In order to calculate the increment in hop distance mathematically, the aggregation difference between best available path and shortest path should be explained. The difference happens when some of the nodes along the shortest path are sleeping. New routes that are formed due to sleeping nodes were named back routs. Back routing happens when nodes use routing information obtained from a certain network to route packets after the network's configuration has been changed. In best available path, the routing information is measured when the initial diffusion is performed. After that, the network keeps routing packets based on the same measured information which might not give the current optimum path.

Consider the same network that was used to create Table 1. Ideally, messages generated from node 1 follow the shortest path through nodes 3 and 4 before reaching the base station in 3 hops as shown in Figure 36. In such case, the best available path is similar to the shortest path.

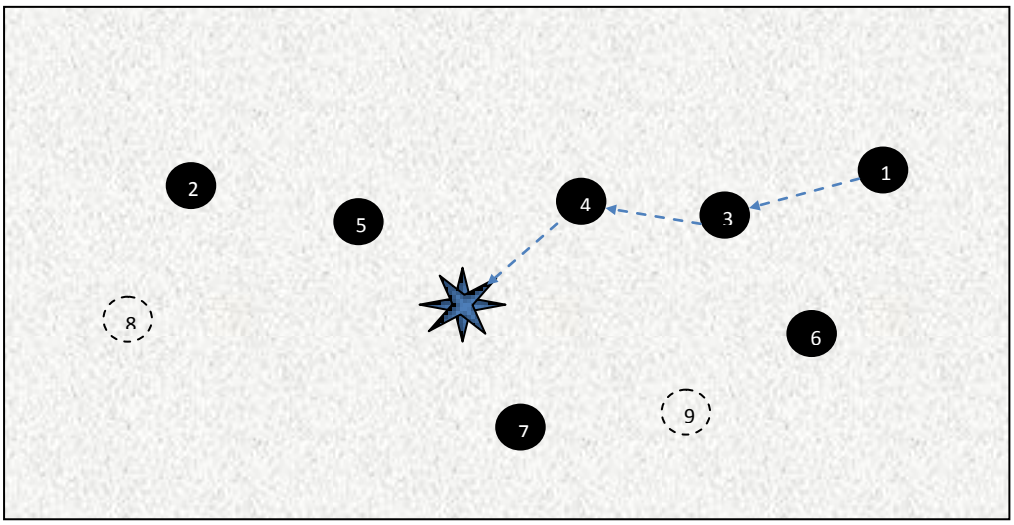


Figure 36. Best available path equal to shortest path

If one of the nodes included in the shortest path is not available, the best available might either have the same number of hops as the shortest path, or slightly higher. When a network chooses the best available path, it chooses it based on the diffusion that originally created all routing information of the network. However, the routing information obtained from the diffusion is specified for a network with all nodes active. When some of the nodes turn off randomly, the network becomes unaware of the real best available path. For the network to be aware of the real optimum path, each node has to know the statuses of every node along the message's path before choosing the next node. Implementing such method could be very energy inefficient since nodes need to talk to each other all the time. To explain the problem, consider the same network discussed earlier. When node 1 tries to forward its message, it will always forward it to node 3 as long as it is available. Node 1 does not know the status of the neighbors of node 3. As a result, node 3 might not be the best node to forward to as Figure 37 shows.

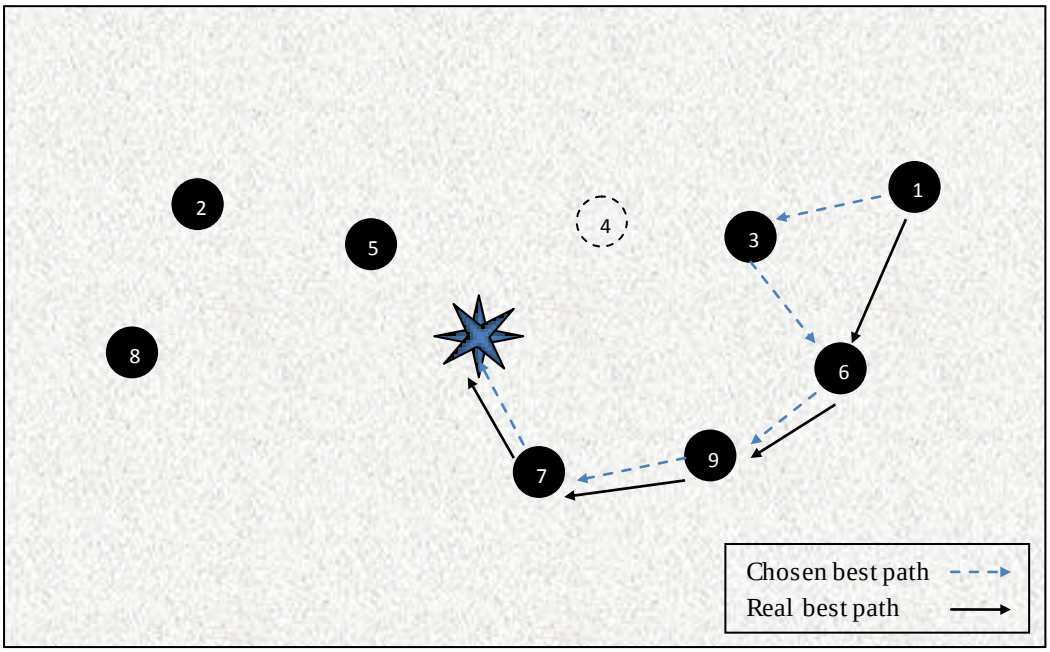


Figure 37. Difference between real best path and chosen best path

In the figure, since node 4 is sleeping, the best available path from node 1 to the base station should be through nodes 6, 9, and 7 with a total distance of 4 hops. When node 1 is looking for the next node, it knows that node 3 is active, and therefore thinks that the best available path is through it. In reality, node 3 receives the message then forwards it to the best available node which is node 6, increasing the total distance travel by the message by one hop.

Back routing can be avoided by implementing complex protocols among the nodes to predict the best path. Due to the nature of random sleep scheduling, the rate of control packets sent should exceed the rate at which the nodes change their statuses. Otherwise, nodes may change their status between the time of making forwarding decision and the time of sending the message. The control packets required for such protocols are far more than the extra packet transmissions caused by the best available path method since each node would have to continuously send a list of its available neighbors to all neighbors. If lists are sent only when a message is waiting to be sent, the message will be delayed while the path is formed, and involved nodes would have to remain active the whole time, removing by that the random property of the method and consuming more energy. Extra hops could also improve the diversity of measurements and increase signal reconstruction accuracy. As a conclusion, having a random sleeping schedule increases the average number of hops per measurement by a small amount, but it helps signal reconstruction accuracy and is more energy efficient than non-random, controlled aggregations.

CHAPTER V

POWER CONSUMPTION IN CS-PAGG

5.1 BACKGROUND

Sensor networks consist of many sensor nodes that run on different sources of energy. Each node consumes energy when performing any activities. Different sensor activities consume different amounts of energy depending on the hardware used per activity. The lowest energy consumption level for a node, which is zero, can only be achieved when the node is turned off. Low-power activities including sleeping, hibernating, and power saving modes consume small amounts of energy in order to operate basic sensor board electronics. In each network, there are different energy modes that nodes use frequently. Knowing the energy consumption level of each energy mode and the number of time it occurs can result in a very accurate measure for energy consumption of the whole network.

A lot of research classifies the energy modes in a network based on the application used by the nodes, and the hardware needed for each operation. Some research depends on the code of sensor nodes to find energy consumption. The work in [18] uses Mica2 platform with TinyOS in order to find the energy consumption for each program that runs on the platform. Such methods, although very accurate, are specific to certain applications that are performed by the network. For different networks that use different methods to sense their data, the energy consumption will be different when measured using application-based methods. Other researchers in [19] found the relationship of energy consumption with different sensor components.

By evaluating the correlation between node components, the total energy consumption of the whole network can be accurately estimated by knowing little information about the application used by each node.

5.2 POWER IN CS-PAGG

5.2.1 Power modes

Since this thesis focuses on low level communication protocol regardless of the application layer of each sensor, the main focus of energy consumption calculations will be on the difference between using and not using sleep scheduling. The comparison was chosen with no consideration for a specific application in order not to lose generality. The aim of the energy calculations is to analyze the performance of best available path method with probabilistic sleeping, and to give a fair evaluation of the energy consumption change.

When nodes use probabilistic sleeping, there are two main activity states that the nodes operate under, low power (sleeping) and high power (active). During sleeping state, nodes need only minimum hardware to trigger the wake up process according to one of the methods described in Figure 5 and Figure 7. When nodes become active, they use one of two power modes, listening or listening/transmitting. Nodes in the network listen to incoming messages as long as they are active, thus, the energy consumption is considerably higher than sleeping mode. When nodes receive messages that they have to forward, or when they start their own messages, the transmission consumes even more power. The work in [20] shows exactly how much power is used for each operation for the Mica2 platform. They measured the current consumption for the platform and noted the readings in Table 4.

Mode	Current	Mode	Current
CPU		**Radio**	
Active	8.0 mA	Rx	7.0 mA
Idle	3.2 mA	Tx (-20 dBm)	3.7 mA
ADC Noise reduction	1.0 mA	Tx (-19 dBm)	5.2 mA
Power-down	103 µA	Tx (-15 dBm)	5.4 mA
Power-save	110 µA	Tx (-8 dBm)	6.5 mA
Standby	216 µA	Tx (-5 dBm)	7.1 mA
Extended standby	223 µA	Tx (-0 dBm)	8.5 mA
Internal Oscillator	0.93 mA	Tx (+4 dBm)	11.6 mA
LEDs	2.2 mA	Tx (+6 dBm)	13.8 mA
Sensor board	0.7 mA	Tx (+8 dBm)	17.4 mA
EEPROM access		Tx (+10 dBm)	21.5 mA
Read	6.2 mA		
Read time	565 µs		
Write	18.4 mA		
Write time	12.9 ms		

Table 4. Power model for Mica2. Operating voltage was 3 volts.

The differences among different power modes are clear as shown in the table above. Nodes consume the least amount of energy when sleeping and consume the most energy when transmitting. In order to evaluate the total power consumption of the scheme, the power consumption of each power mode should be measured. The power categories used in this research were obtained from Table *4* as follows:

1- Sleeping power (W_{Sleep}):

In sleeping mode, the nodes use the minimum energy available in order to keep the node's internal circuit running. The mode is described in [20] as "snooze" state and the current consumption was measured to be 30µA.

2- Idle power (W_{Idle}):

When the nodes are active, they always listen to incoming messages. Therefore they work on standby bases. The components used when idle (listening) are radio Rx, sensor board, and idle CPU state.

3- Transmission power (W_{TR}):

The mode is triggered when information sent from other nodes needs to forward and when the node decides to start a measurement. The components used when transmitting are radio Tx and active CPU state.

To find the power consumption of each mode, the following equation is used:

$$Power\ Consumption = Voltage * \sum Current\ readings\ of\ all\ used\ components$$

Equation 27

It is normal in all known wireless sensor platforms to consume more power when transmitting. The exact ratio of idle power consumption to transmission power consumption may vary slightly with different sensor platforms; however, the general relationship among the three power modes is the same in all platforms. Power consumption is always lowest when sleeping, highest when transmitting, and somewhere in between when idle.

Since the calculations are only used for efficiency evaluation of the method rather than a specific platform, any platform can be chosen to perform the calculations. It was decided to test the method under Mica2 platform In order to evaluate the efficiency of the method using numbers and without losing generality. The reason behind choosing Mica2 is because of its popularity and the extensive amount research that has been done on it.

By employing the information from Table 4 and Equation 27, the power consumption of each mode was found to be:

$$W_{Sleep} = A_{snooze} * voltage = 30\mu A * 3V = 90\mu W$$

Equation 28

$$W_{Idle} = (A_{Idle\,CPU} + A_{Rx} + A_{sensor}) * voltage = (3.2mA + 7mA + 0.7mA) * 3V$$
$$= 32.7mW$$

Equation 29

$$W_{TR} = (A_{active\,CPU} + A_{Tx}) * voltage = (8mA + 21.5mA) * 3V = 88.5mW$$

Equation 30

5.2.2 Activity power:

The number of packets transmitted in a sensor network is a factor of the number of relay hops required for each measurement before reaching the destination. In a simple protocol where all transmitted measurements go through the shortest path, the total number of transmitted packets is equal to the number of measurements multiplied by the average number of hops per measurement for the network.

Transmissions in best available path are divided to two types. Hello packets and data packets. Hello packets hold very little information as explained earlier. Each hello packet has a size of δ_2 bits. Data packets hold signal information and IDs of participating nodes. The size of each data packet is δ_1. The power consumption for each packet depends on its length. Mica2 platform can transmit packets at a transmission rate (γ) of 38.4 Kbps [20].

Using the information from Equation 30, the power consumption for each packet transmission becomes:

$$W_{Data} = W_{TR} * \delta_1/\gamma$$

Equation 31

$$W_{Hello} = W_{TR} * \delta_2/\gamma$$

Equation 32

Each time a data packet is transmitted, number of hello messages that is roughly equal to the number of active neighbors is transmitted. Therefore, if Q (P^{awake}) represents the number of neighbors in a sensor network, the activity power of the whole network ($W_{activity}$) is defined as the power consumed by data communications and can be obtained as follows:

$$W_{activity} = W_{TR} * \frac{\delta_1 + \delta_2 * Q\left(P^{awake}\right)}{\gamma} * \mu * Hop\left(P^{awake}\right)$$

Equation 33

And

$$Q\left(P^{awake}\right) = \frac{PNr^2}{R^2}$$

Equation 34

5.2.3 Operation power:

Wireless sensors consume energy when idle. Although not as large as activity energy consumption, idle energy consumption for the summation of many nodes sometimes exceeds the power spent on transmission activities. When awake and not transmitting, sensors sample data

through their sensor boards, and listen to incoming messages from other nodes. The power consumed by the radio receiver is not small and should be considered when calculating the total energy consumption of networks.

When operating under best available path and probabilistic sleeping schedule methods, non-transmitting nodes are divided into two groups: sleeping and idle. The number of sleeping node is reflected by the network active probability P^{awake}. The remaining nodes are considered idle. The operation power consumption of a network is defined as the power consumed by non-transmitting nodes and can be found by combining both power used by idle nodes and power used by sleeping nodes. Operation power is calculated using the following formula:

$$W_{operation} = [P^{awake} * W_{idle} + (1 - P^{awake}) * W_{sleep}] * N$$

Equation 35

5.2.4 WSN power consumption:

The total power consumption of a WSN is a combination of both operation and activity power consumptions. In general, the total power consumption of a WSN is:

$$W_{WSN} = W_{activity} + W_{operation}$$

Equation 36

after substituting Equation 33 and Equation 35:

$$W_{WSN} = W_{TR} * \frac{\delta_1 + \delta_2 * Q(P^{awake})}{\gamma} * \mu * Hop(P^{awake}) + [P^{awake} * W_{idle} + (1 - P^{awake}) * W_{sleep}] * N$$

Equation 37

The energy consumption of a network can be found by multiplying the power by time of operation (t).

$$E_{WSN}\,(Jouls) = W_{WSN}(Watts) * t$$

Equation 38

The values obtained from theoretical analysis closely match the results from practical simulation as Figure 38 shows. In the figure, a 200-*sparse* signal was measured and reconstructed with 99% accuracy.

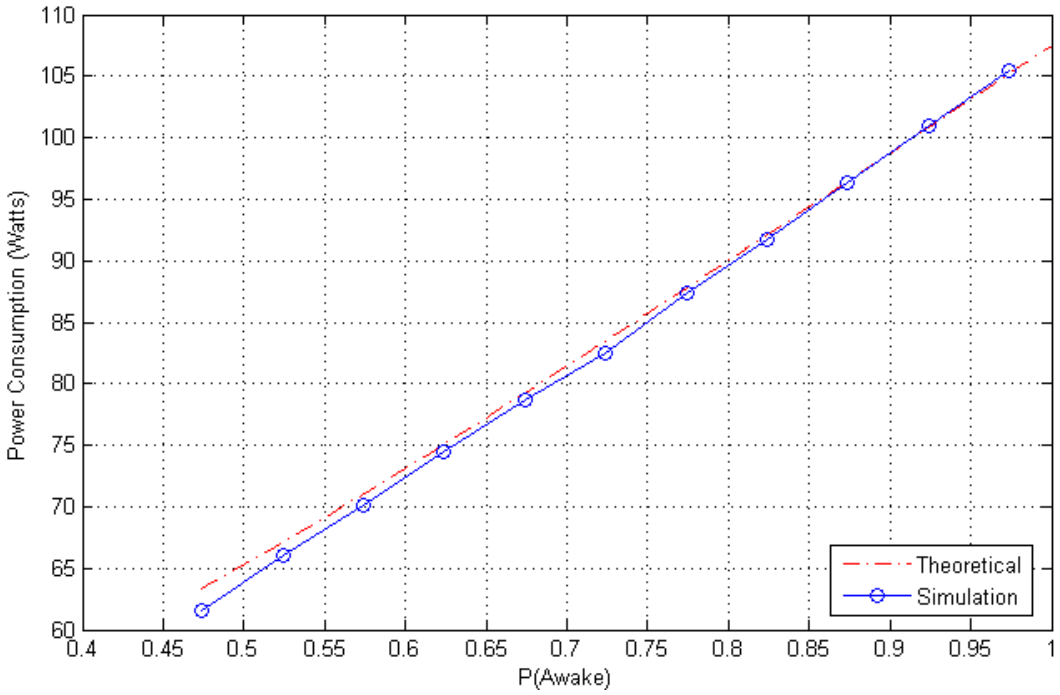


Figure 38. Total WSN power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits, δ_2=16 bits, and μ=700 mes/sec.

The activity and operation power consumption of WSN do not depend on each other. The activity power consumption is caused by packet transmission. Therefore, the main factor that determines the activity power consumption is the number of measurements required by the network per second, or as it was called earlier, the sampling rate μ. Activity power consumption changes with density of the network too. Low density networks increase the average number of hops required, increasing the required number of transmissions, and eventually increasing activity power consumption of the network.

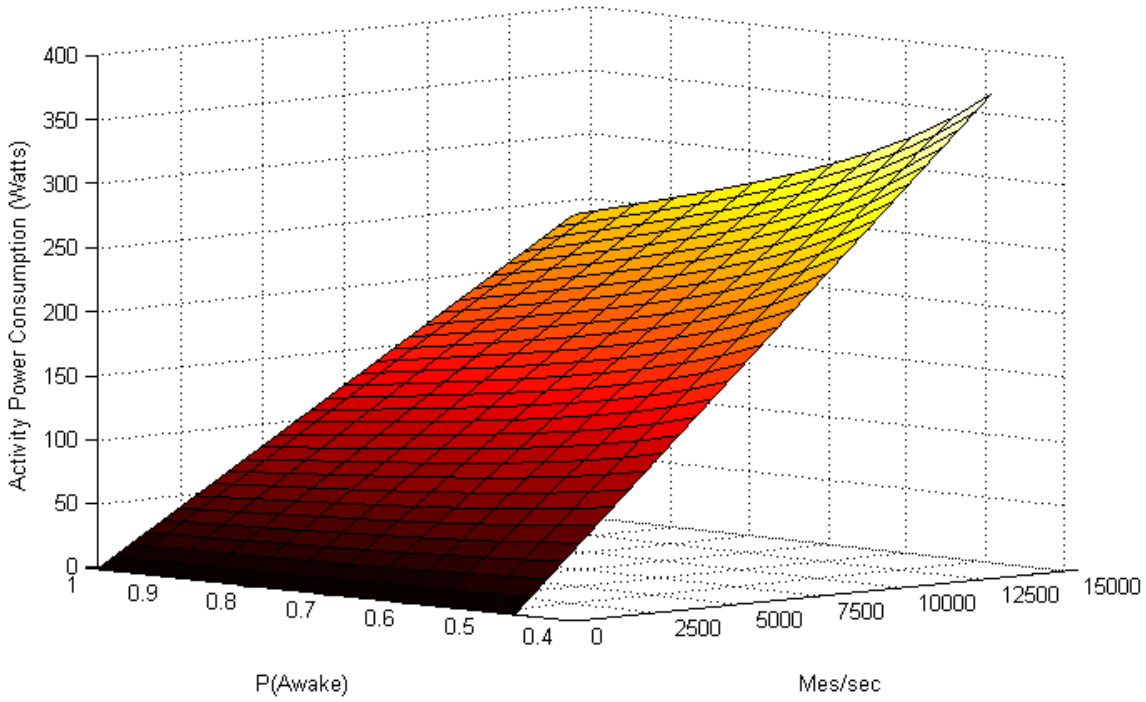


Figure 39. Activity power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits.

The figure above shows the relationship between activity power consumption and both P^{awake} and μ. It can be seen that the power change caused by increasing the number of measurements is far more than the power change caused by reducing the density of nodes.

Node density (ϵ) is controlled by P^{awake} according to the equation:

$$\epsilon = \frac{P^{awake} * N}{Area}$$

Equation 39

Since the number of node N and the Area of networks do not change frequently, P^{awake} can represent the density of nodes. Operation power consumption does not depend on the number of measurements μ. It is a function of density of the network only as shown in Figure 40.

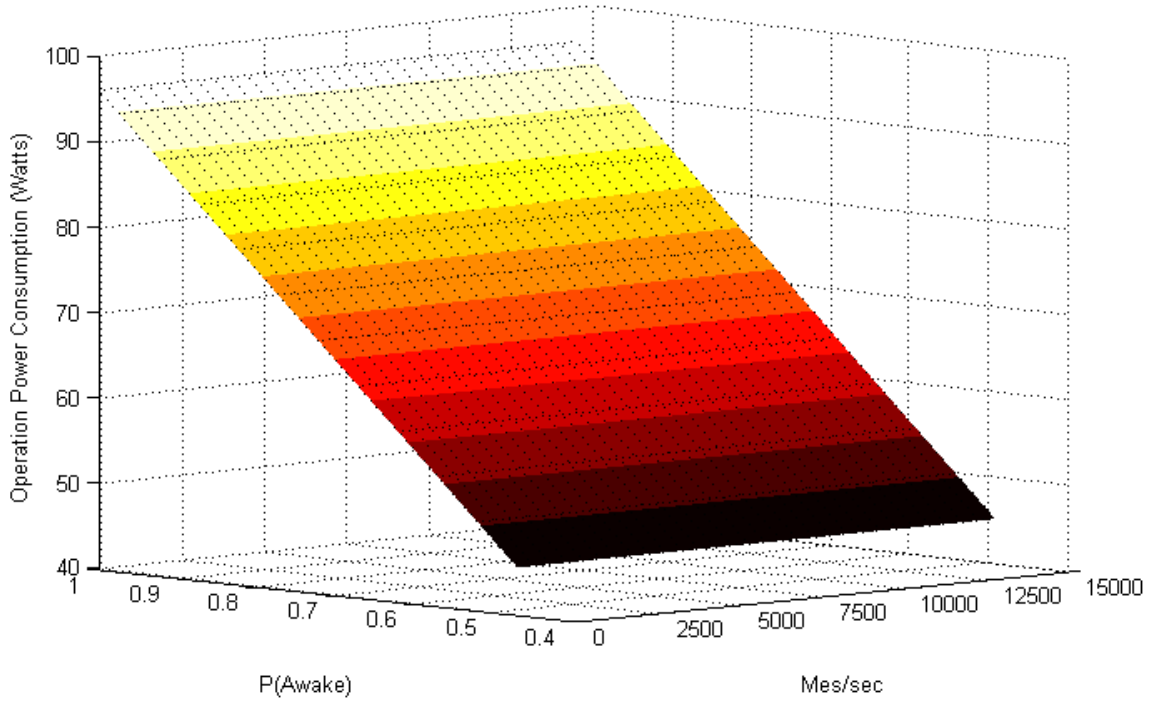


Figure 40. Operation power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits.

As can be seen from the figure, the power consumption increases dramatically when density of the network increases. The increment is caused due to having more active sensors and less sleeping ones. When plotting both power consumption sources, it becomes clear that the relationship with node density is opposite for each power consumption component. Figure 41 shows the intersection between the two power components.

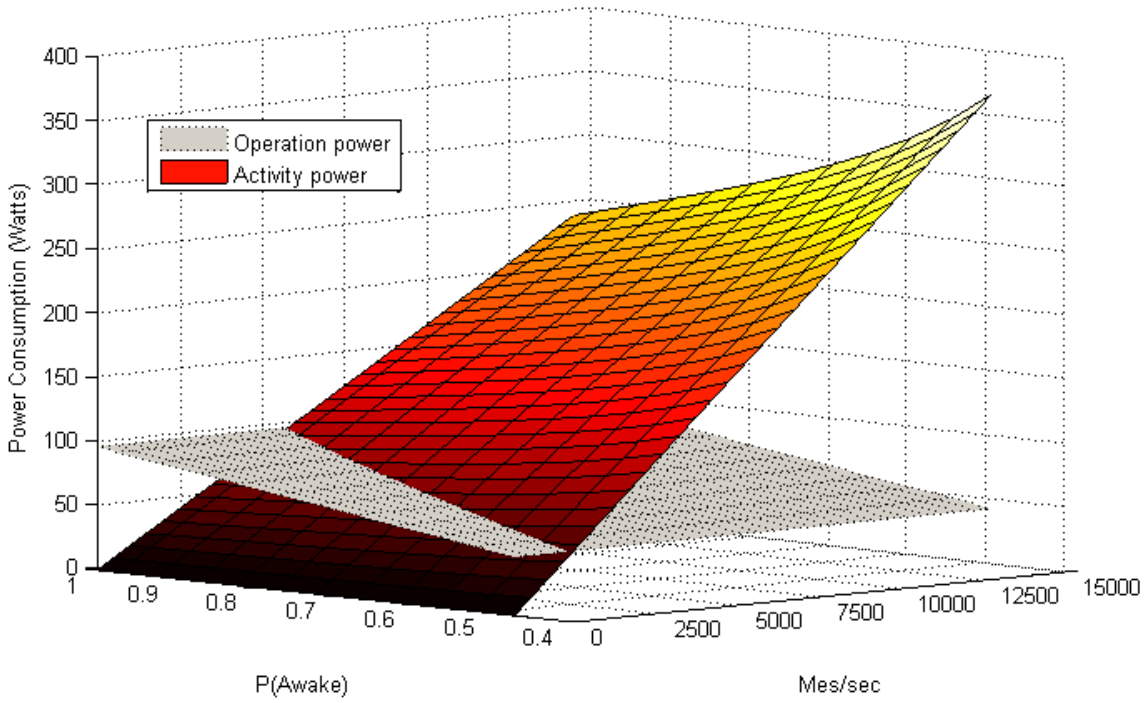


Figure 41. Power consumption components for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits.

5.3 OPTIMUM ACTIVE PROBABILITY

5.3.1 Definition of optimum active probability

When the density increases, hop distance/measurement for best available path algorithm decreases, and less activity power is required. On the other hand, increasing node density results in less sleeping nodes, resulting in increasing the energy. Both changes in energy are not subtle and cannot be ignored. Having opposite correlations with node density, power consumption components indicate that there should be a point at which minimum total WSN power consumption is achieved. The sleep scheduling probability P^{awake} could be engineered for each network, increasing the power efficiency of best algorithm path to its maximum value.

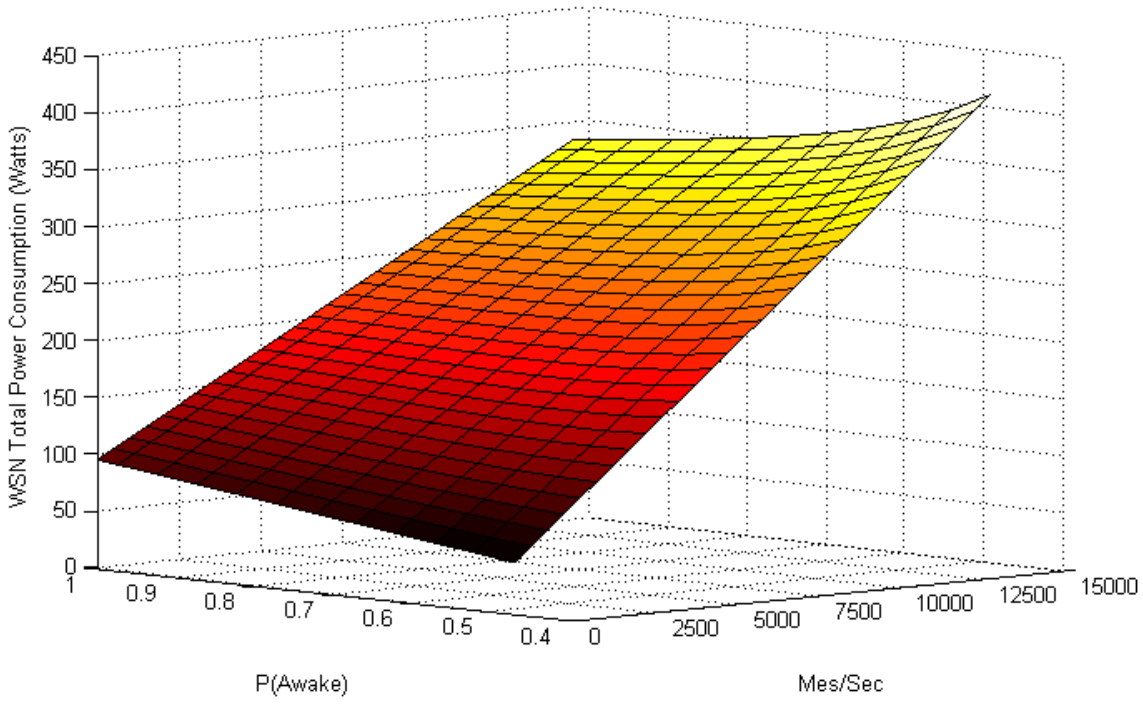


Figure 42. WSN total power consumption for a network with 280m radius and 3000 randomly deployed Mica2 sensors. δ_1=400 bits and δ_2=16 bits.

When looking at the total energy consumption of a WSN in Figure 42, it is clear that the density with minimum power consumption is different for each μ. At low sampling rates, the power consumption is always at its minimum when P^{awake} is the lowest possible or P^{th}. However, for higher sampling rates like 10000 measurement/sec for this specific network, the power consumption is the lowest when P^{awake} is 1, meaning that no nodes are sleeping. For values of μ that are in between the two ranges, P^{awake} that results in the smallest power consumption ranges between P^{th} and 1. Therefore, optimum P^{awake} that results in lowest power consumption is a function of the sampling rate of the network μ.

5.3.2 Simulation optimum active probability

In order to measure the exact optimum P^{awake} for lowest power consumption, a wireless sensor network was simulated using best available path algorithm. Power consumption was then measured using an application-based method that takes packet transmissions and idle network power consumption into consideration.

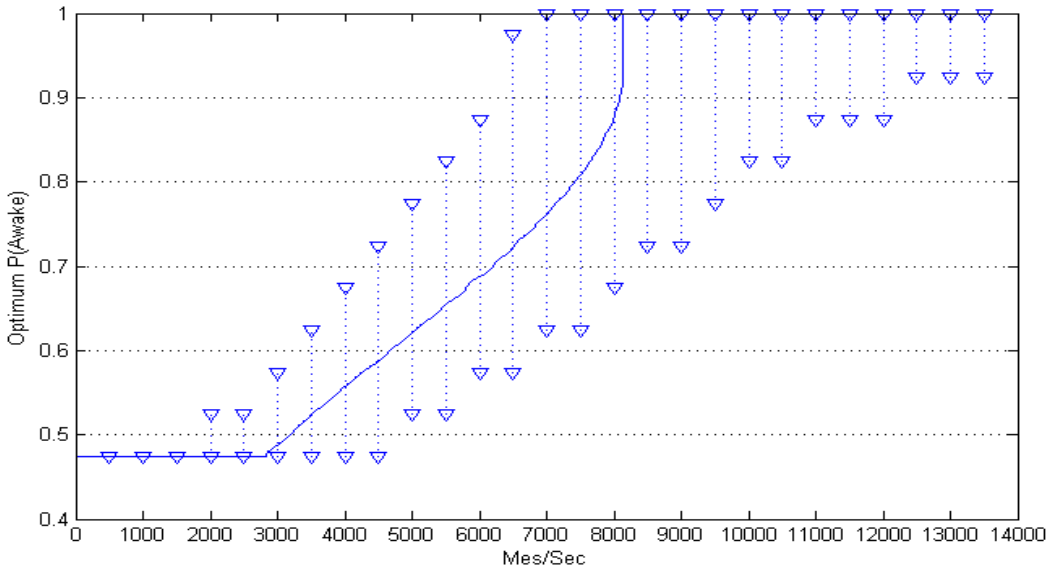


Figure 43. Optimum P^{awake} for a WSN. Region shown by vertical lines shows possible P^{awake} values within 5% of minimum power consumption.

The curve in Figure 43 shows the simulation results for optimum P^{awake}. Each value of μ has a corresponding P^{awake} that keeps total power spent by the network minimum. The figure also shows that optimum P^{awake} is stable and flexible. Each one of the vertical lines shows all values of P^{awake} that can be chosen for the specific μ, and obtain power consumption within 5% of the minimum possible. Therefore, any P^{awake} that is chosen from the region showed in Figure 44 results in a power consumption level that is within 5% of the minimum power consumption level possible.

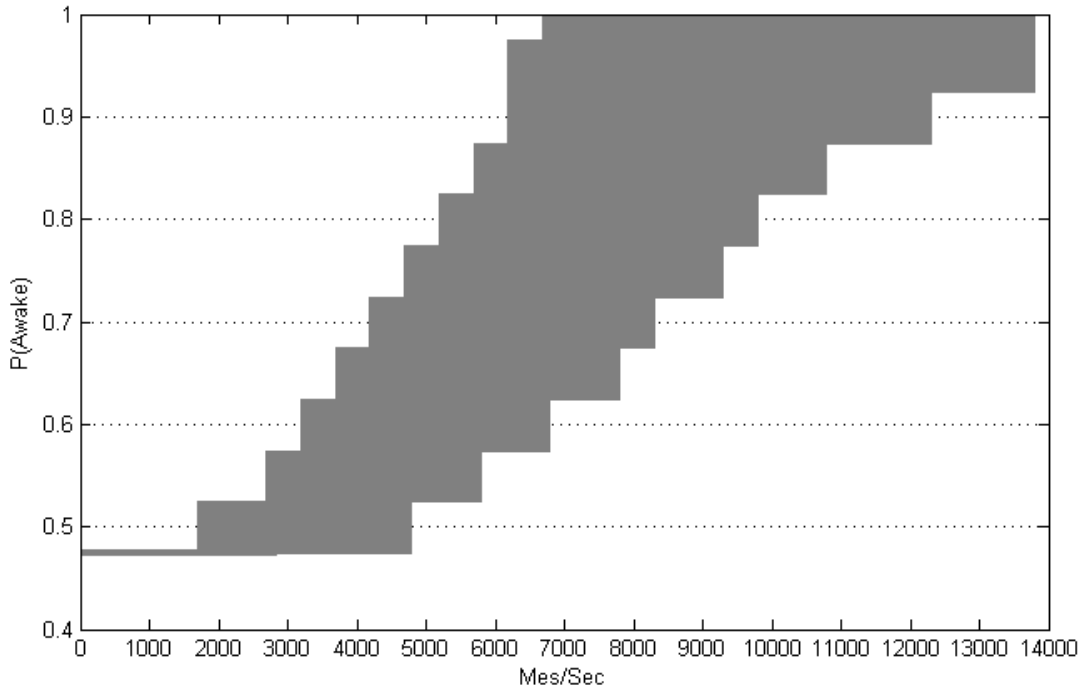


Figure 44. P^{awake} region within 5% of minimum power.

The region looks like a step function because only certain points were taken into consideration during the simulation. The purpose of the graph was not to define the region, but rather to show the flexibility of the optimum P^{awake}. Therefore, a more detailed simulation that could take a lot of time and resources to perform was not necessary.

5.3.3 Theoretical optimum active probability:

The optimum P^{awake} was shown to have a relationship only with the sampling rate μ. Other parameters of the network like the power consumption of the nodes do not affect optimum P^{awake} because they are treated as constants. As a result, a mathematical formula for the optimum P^{awake} becomes valid for any network with the same density and required μ.

Optimum P^{awake} curve can be found mathematically by finding the probability that gives the minimum power consumption in Equation 37. In order to do so, the derivative of the equation must be found and set to zero. The resultant equation is shown below:

$$\mu = \frac{\left(W_{sleep} - W_{idle}\right) * N * \gamma}{W_{TR}\left[H\acute{o}p(P^{awake}) * \left(\delta_1 + \frac{\delta_2 P^{awake} N r^2}{R^2}\right) + Hop(P^{awake}) * \frac{\delta_2 N r^2}{R^2}\right]}$$

Equation 40

In the equation above, there are only two variables which are μ and P^{awake}. For each value of μ, there exists a value of P^{awake} that gives minimum WSN power consumption. Figure 45 shows the accuracy of the equation. It is clear that the results line up with simulation results.

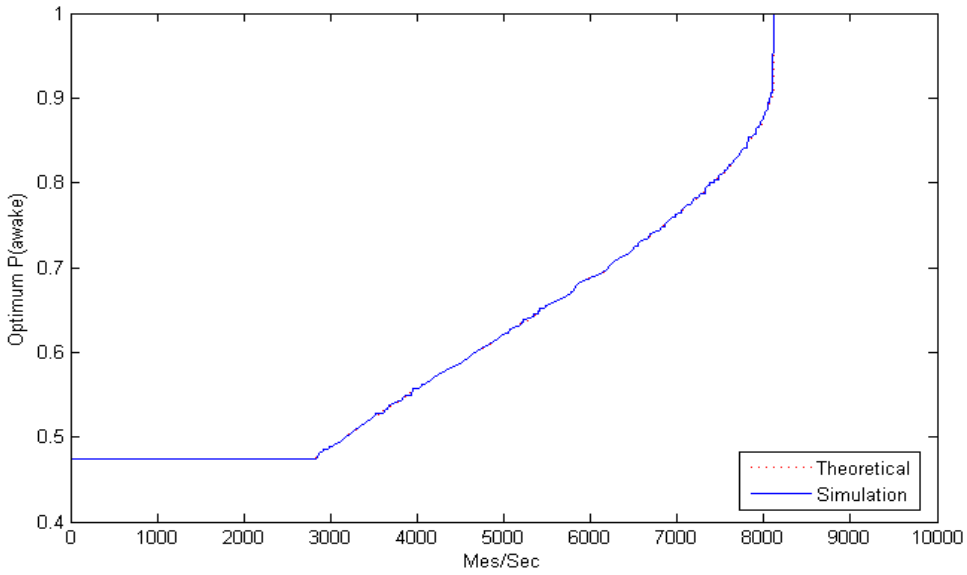


Figure 45. Accuracy of theoretical analysis of optimum P^{awake}

5.4 REGIONS OF OPERATION IN CS-PAGG

When looking at the optimum P^{awake} curve, it is noticeable that the curve has three distinct regions. The first region always has P^{th} as the optimum probability, the second one always has 1 as the optimum probability, and last region has different values of optimum probability as shown in Figure 46 below.

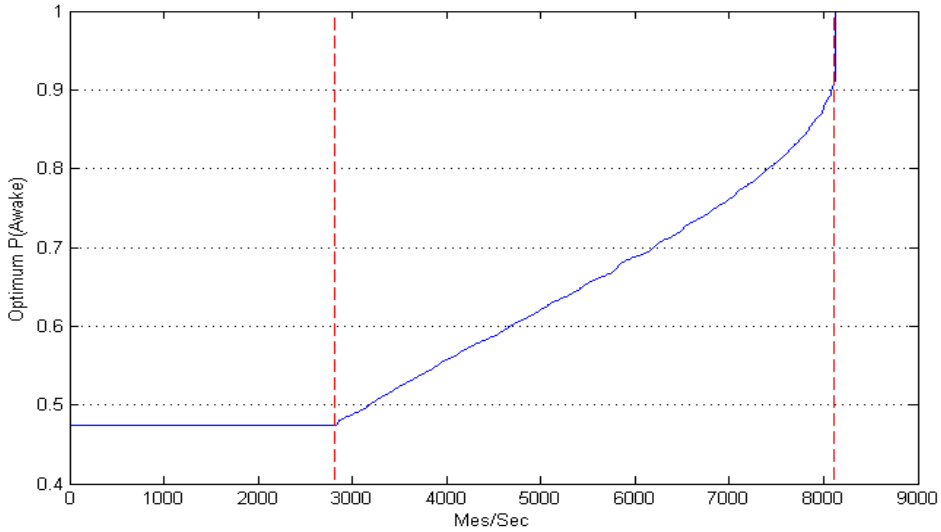


Figure 46. Regions of optimum probability in a WSN.

The sampling rates at which the regions change are referred to as μ_1 and μ_2. The values of μ_1 and μ_2 can be found by substituting P^{awake} in Figure 42 with P^{th} and 1 respectively as shown below.

$$\mu_1 = \frac{(W_{sleep} - W_{idle}) * N * \gamma}{W_{TR}\left[\acute{Hop}(P^{th}) * \left(\delta_1 + \frac{\delta_2 P^{th} N r^2}{R^2}\right) + Hop(P^{th}) * \frac{\delta_2 N r^2}{R^2}\right]}$$

Equation 41

$$\mu_2 = \frac{(W_{sleep} - W_{idle}) * N * \gamma}{W_{TR}\left[\acute{Hop}(1) * \left(\delta_1 + \frac{\delta_2 N r^2}{R^2}\right) + Hop(1) * \frac{\delta_2 N r^2}{R^2}\right]}$$

Equation 42

The optimum probability for all values of μ can be represented by:

$$P^{optimum} = \begin{cases} P^{th}, & \mu < \mu_1 \\ Using\ Equation\ 40, & \mu_1 < \mu < \mu_2 \\ 1, & \mu > \mu_2 \end{cases}$$

Equation 43

From the previous equations, it can be concluded that the sampling rate of a WSN determines the optimum probability. Sensor networks can work under low P^{awake} levels when sampling rate is low, decreasing by that the power consumption of the network. It was explained earlier that sampling rate μ is a function of the maximum change rate of the monitored signal. Therefore, CS-PAGG gives its maximum energy saving when the monitored signal has a low

maximum frequency. Just like having two thresholds for the sampling rate, the frequency of the monitored signal also has its corresponding thresholds.

$$f_1 = \frac{\mu_1}{M} \ , \quad f_2 = \frac{\mu_2}{M}$$

Equation 44

CS-PAGG gives the best performance when the maximum frequency of the measured signal is lower than f_1. When the signal's frequency increases beyond f_2, the sleep scheduling part of CS-PAGG becomes ineffective because the optimum probability becomes 1, meaning that no sleeping schedule should be implemented. Natural, slow-changing signals that have low frequency are therefore the signals targeted by the method in order to reduce power consumption.

CHAPTER VI

PERFORMANCE OF CS-PAGG

6.1 SIGNAL RECONSTRUCTION

Signal reconstruction in CS-GAPP is the same as normal compressed sensing algorithm described in Equation 7. For the sake of simplicity, we made the assumption that the signal does not change while collecting *M* measurements. If the signal changes while aggregating, the estimated signal at the base station might slightly lose accuracy. The exact effect of signal change during aggregating *M* measurements on the signal recovery is left to be discussed in our future work.

After the base station receives *M* measurements, it can start signal reconstruction process in order to recover the monitored signal. The accuracy of signal reconstruction depends on the number of collected measurements and the information each measurement carries when the monitored signal is fixed. Since each measurement is a row in the measurement matrix, acquiring many measurements helps the base station to recover the signal accurately. The length of each measurement is controlled by the average number of hops per aggregation for the network. Since CS-PAGG works on natural signals, DCT matrix was chosen to be the sparsifying matrix of the method due to its dense nature. Other sparsifying matrices can be chosen instead if the monitored signal is more compressible under them.

6.2 SIMULATION RESULTS

When a network has a high average number of hops per measurement, the information that the measurement carries becomes very valuable for signal recovery. The effect happens with specific sparsifying matrices that are sparse.

For DCT sparsifying matrix, increasing the number of entries in one row of the measurement matrix does not dramatically change the reconstruction accuracy. Therefore, for the same energy that is required to aggregate the measurement over extra nodes, extra measurements with reduced number of hops can be used for improved reconstruction accuracy. The curve in Figure 47 shows the reconstruction error of a 200-sparse signal using 700 measurements under CS-PAGG.

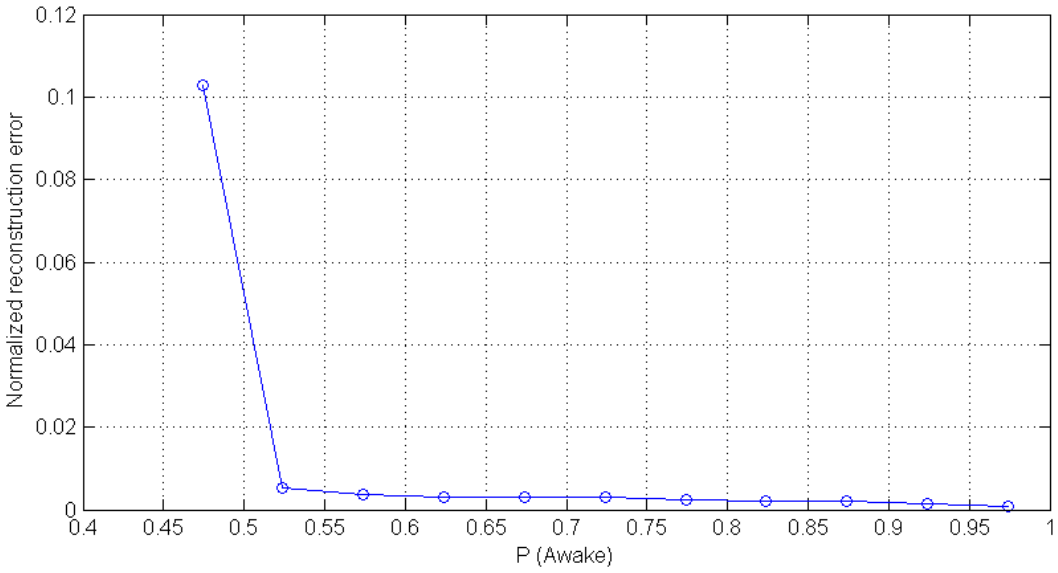


Figure 47. Normalized reconstruction error for 700 received measurements and 200-sparse signal.

The signal can be fully recovered if the number of received measurements satisfies Equation 6. However, for the same *M*, the normalized error changes when the density of the network changes as shown in the figure. The change happens because of dead measurements in

the network. When the percentage of dead measurements is high, the number of received measurements becomes less than M, and the reconstruction accuracy decreases. Comparing the previous figure to the percentage of dead measurements shown in Figure 28, it can be seen that the reconstruction accuracy is correlated to the percentage of dead measurements in the network.

6.3 MINIMUM ACTIVE PROBABILITY

Dead measurements hurt the performance of the network for both energy efficiency and signal reconstruction accuracy. Therefore, they should be avoided before CS-PAGG can achieve its full efficiency. Methods that deal with dead measurements were tested and proven to be inefficient in the previous chapters. The most efficient way to deal with dead measurements is therefore to increase the density of the network to a level at which dead measurements percentage becomes so small that it can be ignored. A good property that dead measurement has is that the percentage of dead measurements drops dramatically when node density increases as shown in Figure 28. As a result, the reconstruction error behaves in the same way as Figure 47 illustrates. Therefore, P^{th} is no longer the lowest bound of density of the network. In order to achieve maximum power efficiency, another probability that is based on dead measurements and reconstruction error had to be introduced. The new probability, which is called the minimum probability or P^{min}, is defined as the lowest active node probability that achieves a reconstruction accuracy of Δ or more when the number of aggregated measurements is M.

P^{min} also modifies the lowest sampling rate threshold μ_1 because it replaces P^{th} as the lowest bound. Therefore, the first region of operation of CS-PAGG becomes bigger and the new μ_1 becomes higher than the old one as shown in the next figure.

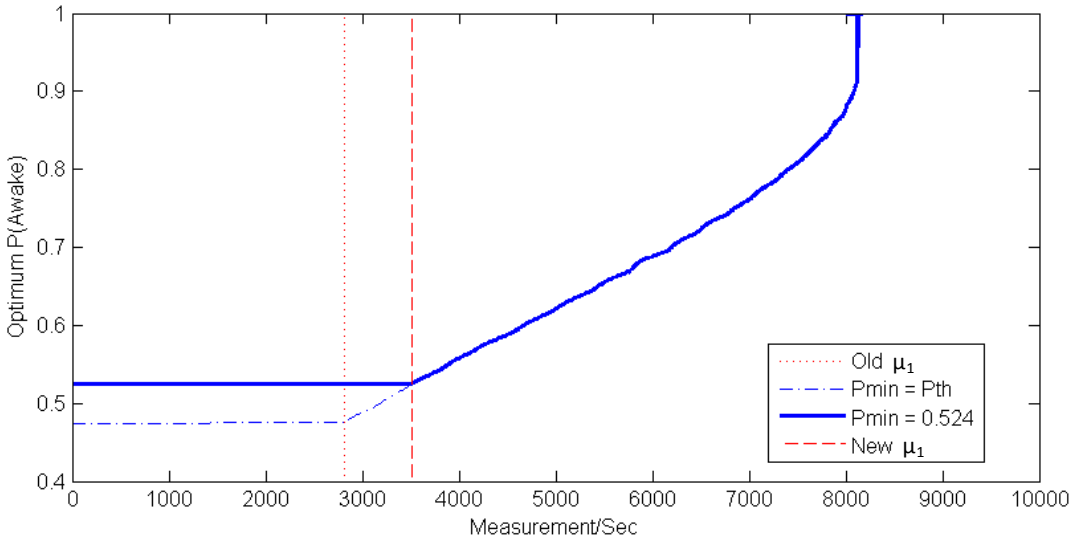


Figure 48. New CS-PAGG regions of operation after P^{min} consideration.

The upper sampling rate threshold μ_2 does not change after changing P^{min} to a value that is higher than P^{th}. The selected value of P^{awake} should be equal to at least P^{min} in order to achieve maximum power efficiency. In the results in Figure 47, P^{min} was found to be 0.525 when Δ was 99%. Therefore, P^{awake} was chosen to be 0.525 in order to save the maximum amount of power. After choosing a good value of P^{min}, dead measurements disappear and CS-PAGG works with its full power efficiency.

6.4 EFFICIENCY EVALUATION IN CS-PAGG

CS-PAGG reduces the operation power consumption of WSN considerably by turning off sensors according to a probabilistic duty cycle. When aggregating under normal conditions using traditional compressed sensing methods, all nodes in the network need to be awake, and the power consumption becomes inefficient due to having unnecessary active nodes. Fewer nodes are active when CS-PAGG is activated, leading to better power efficiency of the network. Since the sampling rate is a function of the frequency, the power consumption for traditional aggregation and CS-PAGG can be shown in relation to signal frequency like Figure 49 demonstrates.

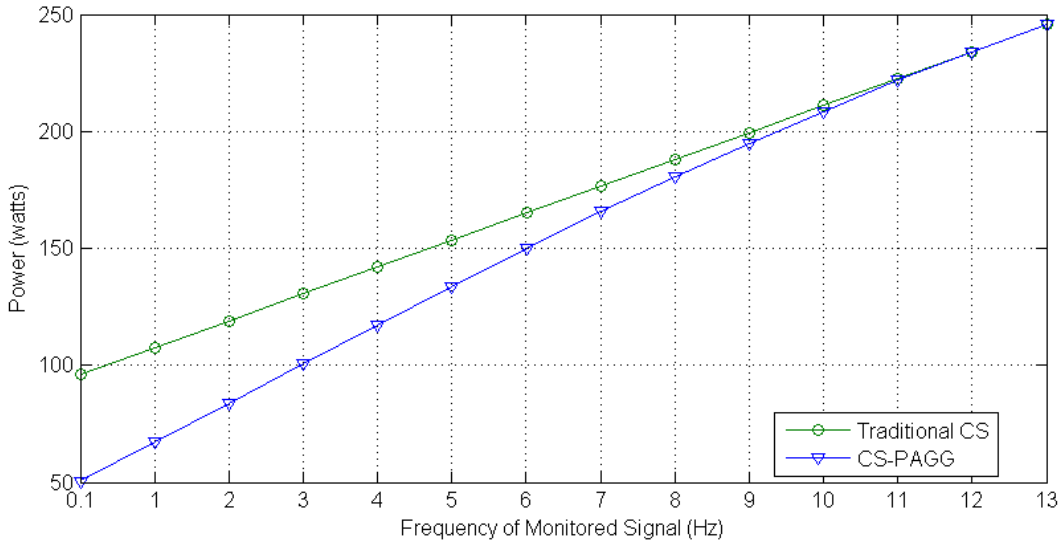


Figure 49. Total WSN power consumption with and without CS-PAGG for a network with a radius of 280m and 3000 randomly deployed Mica2 sensors.

It is clear that CS-PAGG significantly decreases the power consumption when monitoring low frequency signals. Figure 50 shows the reconstruction error of CS-PAGG. The error is slightly higher in CS-PAGG because of the effect of dead measurements. In the particular simulation, P^{min} was set to 0.525, resulting in a dead measurement rate of 10%. When the rate of dead measurement is not zero, the number of measurements that the BS receives is lower than M,

causing some reconstruction error. As a conclusion, CS-PAGG saves a lot of power for a very small loss of reconstruction accuracy that can be neglected. In this particular network, the percentage of saved power can be as high as 48% with a penalty of 0.006 of normalized error (same order as without CS-PAGG) at max.

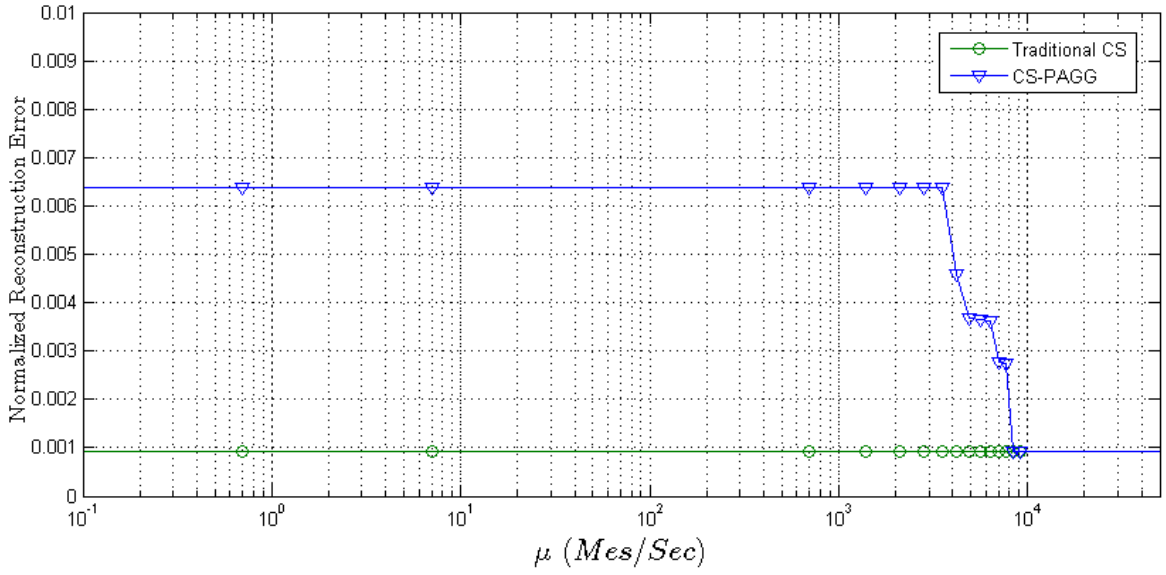


Figure 50. Signal reconstruction error with and without CS-PAGG for a 200-sparse signal and 700 measurements at P^{min} of 0.525.

The amount of saved power can be calculated from the active probability P^{awake} simply as follows:

$$W_{saved} = W_{operation} * (1 - P^{Awake})$$

Equation 45

and

$$Percentage\ of\ power\ saving\ (\zeta) = \frac{W_{saved}}{W_{WSN}|_{p^{Awake}=1}} * 100$$

Equation 46

The amount of saved power is very large when P^{awake} is the least possible value. However, P^{awake} cannot drop below P^{min} in order to fulfill signal reconstruction accuracy requirements. Therefore, using P^{min} in the power saving equation returns the maximum power saving that can be gained by using CS-PAGG. Power can be saved for any signal that needs a sampling rate of less than μ_2. Figure 51 shows the percentage of saved power in the WSN discussed earlier when using CS-PAGG against the same network when using traditional CS with no sleep scheduling.

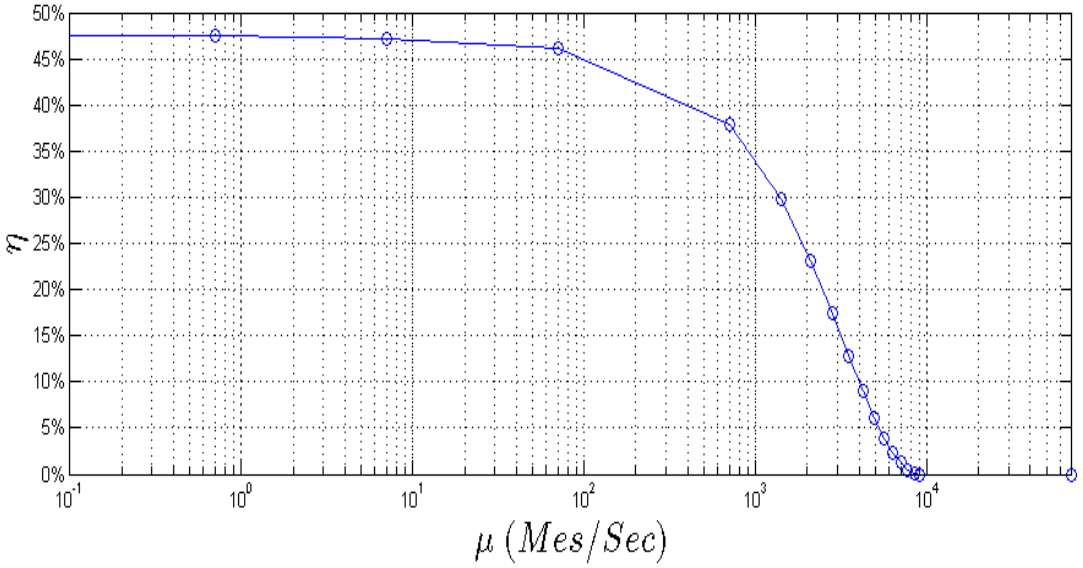


Figure 51. Power saving under CS-PAGG.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this thesis, a new aggregation method that employs both compressed sensing and probabilistic sleep scheduling was described. Compressed sensing probabilistic aggregation (CS-PAGG) improves the power efficiency of wireless sensor networks that monitor natural signals such as temperature, moisture, light, wind speed, and others. The method works based on a probabilistic sleeping probability that controls the duty cycles of each node in the network, reducing the power consumption of the network.

A duty cycle that gives optimum energy consumption, along with other important network parameters and assessments are calculated mathematically before deploying the network, resulting in a robust, simple, and optimum data collection algorithm. The ability to measure performance analytically allows fast and easy network design. Moreover, not having to use any synchronization among the nodes gives more flexibility and simplicity to CS-PAGG.

The method works best with natural signals with low frequencies such as temperature. Although the energy gained by CS-PAGG results in a very small loss in signal reconstruction accuracy, the accuracy can be greatly boosted by adding few more measurements to M. Working with increased M increases the accuracy to be more than traditional CS methods with M measurements, and also results in a big amount of power saving. Therefore, using CS-PAGG to monitor natural signals is more power efficient than traditional CS methods regardless of reconstruction accuracy.

CS-PAGG is very robust to network changes. Although the sleeping probability is determined before the network is deployed, changing the number of nodes slightly does not change the optimum probability due to having wide region of operation as was explained in Figure 43. From the same figure, by knowing the exact boundaries to each region of operation theoretically, the exact limit of network change percentage (Δ) can be determined. A network using CS-PAGG will keep its efficiency as long as the total change in network's node density caused by dying and new sensors does not exceed Δ. If the total percentage of change exceeds Δ, another optimum P^{awake} can be calculated in order to keep the network's power consumption at its lowest.

Changes in positions due to adding and removing nodes, or nodes' slight movement can also be handled by CS-PAGG. New nodes can broadcast a request for a local diffusion between them and their neighbors in order to form their routing arrays. New nodes then start working normally on sensing and aggregating. Since the sleeping probability is the same for all nodes, new nodes can easily get the probability from their neighbors. Each node can detect if one of its neighbors has stopped working by keeping a log of the neighbors' transmissions. With time, if the nodes do not hear from their neighbors, the probability of the neighbors being alive decreases. When the probability reaches a certain limit, nodes can decide if one of their neighbors has stopped working, and perform designated actions like reporting to the base station. Such detection happens with no extra communication cost, allowing the network to be maintained easily and removes the need for extra means to monitor the functionality of the network.

Another advantage of CS-PAGG is that it is implemented in the physical layer of the nodes. As a result, other methods that are application-based and are specified to compress certain signals can be implemented on top of CS-PAGG to get improved performance.

CS-PAGG is customized to work for networks with defined base stations. Therefore, messages that are not meant to be received by the base station cannot be routed. The only known path for each node is the path to its base station. For networks with more than one base station, each base station has to have its own directed diffusion when the network starts before starting aggregations. Although most of the monitoring systems have centralized base stations, there are some applications that require certain nodes to communicate with each other. CS-PAGG cannot be implemented on such networks unless different directed diffusions from each destination nodes are started.

The biggest current limitation of CS-PAGG is in its fixed sleeping probability. Fixed probability requires the network to be engineered before deployment. A future work can consider the possibility of having a dynamic sleeping probability that the nodes create after deployment. Such scheme can be implemented based on CS-PAGG by starting the network with all nodes active, then decreasing the duty cycles of nodes based on their local response received from neighbors. The probability can be controlled based on the frequency and the source of messages that each node hears. A probabilistic controller will detect the optimum probability from the information it acquires over time with no need for extra control messages among the nodes.

An extension of the research could consider giving nodes that are far from the base station higher probability of starting measurements than nodes that are close to the base station. Since each aggregation that comes from far nodes has to go through close nodes, close nodes do not have to start measurements as often as far nodes in order to contribute to the reconstruction of the signal. Such scheme will improve the power distribution among nodes and makes sure that all nodes have roughly the same life span. The drawback of such method is that each node has to know its distance from the edge of the network, which could be programmed either prior to deploying the network, or obtained dynamically while the network is operational.

Another extension of CS-PAGG that can be done is to consider event-based monitoring systems. By turning of the radio and keeping the sensor board active, events can be detected and forwarded to the base station. Nodes will turn on their radio only to listen to incoming messages based on a CS-PAGG duty cycle.

Future work might also consider the idea of dynamic sampling rate for signals that change frequency. The current CS-PAGG cannot work with signals that change frequency and need to be designed for the maximum possible frequency in order to avoid under-sampling of the signal. Adaptive frequency can be achieved by monitoring the frequency of the received signal then changing the scheme accordingly.

REFERENCES

[1] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE,* vol. 25, no. 2, pp. 21-30, 2008.

[2] D. L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory,* vol. 52, no. 4, pp. 1289-1306, 2006.

[3] Z. Xiaoyan, W. Houjun and D. Zhijian, "Wireless sensor networks based on compressed sensing," in *2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, 2010.

[4] Q. Ling and Z. Tian, "Decentralized Sparse Signal Recovery for Compressive Sleeping Wireless Sensor Networks," *IEEE Transactions on Signal Processing,* vol. 58, no. 7, pp. 3816-3827, 2010.

[5] G. Anastasi, M. Conti and M. Di Francesco, "IEEE Transactions on Industrial Informatics," *Extending the Lifetime of Wireless Sensor Networks Through Adaptive Sleep,* vol. 5, no. 3, pp. 351-365, 2009.

[6] E. Bulut and I. Korpeoglu, "Sleep scheduling with expected common coverage in wireless sensor networks," *Wirel. Netw.*, vol. 17, no. 1, pp. 19-40, 2011.

[7] N. Rahnavard, B. N. Vellambi and F. Faramarz, "CRBcast: A Reliable and Energy-Efficient Broadcast Scheme for Wireless Sensor Networks Using Rateless Codes," *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS,* vol. 7, no. 12, pp. 5390-5400, 2008.

[8] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *ScienceDirect*, vol. I, no. 346, pp. 589-592, 2008.

[9] R. G. Baraniuk, "Compressive Sensing [Lecture Notes]," *IEEE Signal Processing Magazine,* vol. 24, no. 4, pp. 118-121 , 2007.

[10] Y. Rivenson and A. Stern, "Compressed Imaging With a Separable Sensing Operator," *IEEE SIGNAL PROCESSING LETTERS,* vol. 16, no. 6, pp. 449-452, 2009.

[11] R. Baraniuk, M. Davenport, R. DeVore and M. Wakin, "A Simple Proof of the Restricted Isometry Property for Random Matrices," *Constructive Approximation*, no. 28, pp. 253-263, 2008.

[12] D. L. Donohoa, A. Malekib and A. Montanaria, "Message-passing algorithms for compressed sensing," *PNAS*, vol. 106, no. 45, pp. 18914-18919, 2009.

[13] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal.*, no. 27, pp. 265-274, 2009.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM TRANSACTIONS ON NETWORKING*, vol. 11, no. 1, 2003.

[15] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *MobiCom '00 Proceedings of the 6th annual international conference on Mobile computing and networking* , New York, USA, 2000 .

[16] S. Servetto and G. Barrenecha, "Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, New York, USA, 2002.

[17] S. A. G. Chandler, "Calculations of Number of Relay Hops Required in Randomly Located Radio Network," *Electronics Letters*, vol. 25, no. 24, pp. 1669-1671, 1989.

[18] O. Landsiedel, K. Wehrle and S. Götz, "Accurate Prediction of Power Consumption in Sensor Networks," in *Proc. of the Second Workshop on Embedded Networked Sensors*, 2005.

[19] "Modeling of Node Energy Consumption for Wireless Sensor Networks," *Scientific Research*, no. 3, pp. 18-23, 2011.

[20] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen and M. Welsh, "Simulating the Power Consumption of Large-Scale Sensor Network Applications," *SenSys*, vol. 3, no. 5, 2004.

[21] C. Rose, "Mean Internodal Distance in Regular and Random Multihop Networks," *IEEE Transaction on Communications*, vol. 40, no. 8, pp. 1310-1318, 1992.

[22] G. Lu, B. Krishnamachari and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for tree-based data gathering in sensor networks," *Wirel. Commun. Mob. Comput.*, no. 7, pp. 863-875, 2007.

VITA

Aram Almuhana

Candidate for the Degree of

Master of Science

Thesis: ENERGY-EFFICIENT DATA AGGREGATION IN WIRELESS SENSOR NETWORKS USING PROBABILISTIC SLEEP SCHEDULING AND COMPRESSED SENSING

Major Field: Electrical and Computer Engineering

Biographical:

Education:

Completed the requirements for the Bachelor of Science in electrical engineering at University of Baghdad, Baghdad, Iraq in 2008.

Experience:

- Experienced in monitoring systems, sensors, digital and analog electronics, image and signal processing, microcontrollers, FPGAs, power electronics, and RF modulation/detection.
- Proficient in MATLAB, C/C++, LabVIEW, Assembly, and TinyOS.

Professional Memberships:

IEEE.