Backbone Formation Algorithm for Wireless Sensor Networks

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Modita Behl (121276)

Under the supervision of

Dr. Ravindara Bhatt

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradessh

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **"Backbone Formation Algorithm for Wireless Sensor Networks"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of **Dr. Ravindara Bhatt** Assistant Professor, Department of Computer Science and Engineering.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Modita Behl, 121276

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Ravindara Bhatt

Assistant Professor

Dept. of Computer Science and Engineering

Dated:

ACKNOWLEDGEMENT

By the grace of God, I take this opportunity to express special thanks to all those who have been associated directly or indirectly with the accomplishment of my project work.

First and foremost, I would like to express my profound gratitude to my respected guide Dr. Ravindara Bhatt, Assistant Professor, Department of Computer Science and Engineering, JUIT, Solan, for his valued guidance, supervision, and encouragement during the course of this study.

I will fail in my duty if I do not acknowledge the efforts of my project partner for the 7th semester Sanchit Kumar [121272] for his assistance, co-operation and encouragement for completion of this work.

I also express my sincere regards and thanks to Prof. Dr. S.P Ghrera, Professor and Head of the Department, Computer Science and Engineering, JUIT, Solan, for his continuous inspiration regarding completion of project work in due time.

The co-operation from the staff members of Department of Computer Science and Engineering and the staff in the computer lab, library, and office is also gratefully acknowledged.

I cannot forget to recall with my heartiest feeling, the never ending heart felt stream of blessings and cooperation of my parents to support me with every thing for B. Tech. in the pioneer Department of Computer Science and Engineering in JUIT, Solan.

I express my deep sense of gratitude towards, Brig. (Retd.) K.K Marwah, Registrar, Jaypee University of Information Technology, Waknaghat, Himachal Pradesh, Prof. Dr. RMK Sinha Dean CSE&IT for continuous inspiration and blessings for time completion of this project.

Place: - Waknaghat [MODITA BEHL]

Date:-30th May, 2015

CONTENTS

LIST OF ABBREBRIATIONS

1. WSN	Wireless Sensor Networks
2. CDS	Connected Dominating Set
3. MST	Minimum Spanning Tree
4. DT	Delaunay Triangulations
5. MCDS	Minimum Connected Dominating Set
6. VR	Voronoi Routing
7. PUDT	3D partial unit Delaunay triangulation
8. MDT	Multi Hop Delaunay Triangulations
9. DTR	Delaunay Triangulations Routing
10. CR	Compass Routing
11. RCR	Randomized Compass Routing
12. IS	Independent Set
13. MIS	Maximal Independent Set
14. GR	Greedy Routing
15. UDG	Unit Disk Graphs
16. DBG	Disk Graphs with Bi directional
17. LEACH	Low-energy adaptive clustering hierarchy LEACH
18. UAVs	Unmanned Air Vehicles
19. UGSN	Underground Sensor Networks
20. UWSN	Underwater acoustic sensor network
21. DS	Dominating Set
22. GC	Geographical Center
23. ST	Steiner Tree

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A wireless sensor network (WSNs) are the spatially distributed sensors which are used to monitor the physical or environmental conditions. Such nodes after collecting the data co-operatively pass the data from one node to another and finally to the sink node. The wireless sensor network has a varied no. of applications from military applications such as battlefield surveillance, health care applications, the prevention of natural disasters, water pollution monitoring etc. Hence, WSN has now emerged to be a premier topic of research. In spite of the numerous applications of WSN they also face a number of challenges. A few challenges include energy constrained node deployment, lack of fixed topology, deployment in harsh environmental conditions etc. Hence, for an efficient and reliable design of such networks we require an architectural framework which counters the prevailing problems of WSN. The hierarchical framework formed for deployment of WSNs effectively overcomes the above discussed problem. In our work we have implemented the well known backbone formation algorithm which is a type of hierarchical framework for deployment of nodes. The backbone formation reduces the existing challenges of WSN deployment and has a number of advantages such as it reduces the communication overhead, increases the bandwidth efficiency, decreases the overall energy consumption, and, at last, increases network effective lifetime in a Wireless Sensor Network. In addition to this we further implemented three types of backbone based networks over two-dimensional and three-dimensional. The major reason for implementation in three-dimension is because the varied applications of WSN's are mostly in three dimensional scenarios.

In our work we implemented algorithms for backbone formation namely Connected Dominating Set, Minimum Spanning Tree, Steiner Tree in two-dimension as well as in three dimension. We also analyse the performances of the three algorithms in both two-dimensional and 3D systems. Simulations results indicate that CDS is better for a two-dimensional network, whereas for three-dimensional steiner tree based approach is beneficial.

As far as sink placement is concerned we evaluate sink placement based on three metrics: geographical centre, minimum eccentricity node and sink placement based on maximum node weight. The two routing algorithms namely greedy and voronoi routing were also implemented in both 2-D and 3-D. The greedy routing proved to be better if we need to have delivery in minimum no, of hops and if packet delivery is to be ensured the voronoi routing proved to be better.

CHAPTER-1

INTRODUCTION

1.1. What is WSN?

Wireless Sensor Networks (WSNs), are spatially distributed autonomous sensors which are used to *monitor* the physical or environmental conditions, such as temperature, sound, pressure, etc. and to mutually pass their data through the network to the base station (BS) or the sink node. Figure 1.1 illustrates the basic architecture of a WSN.

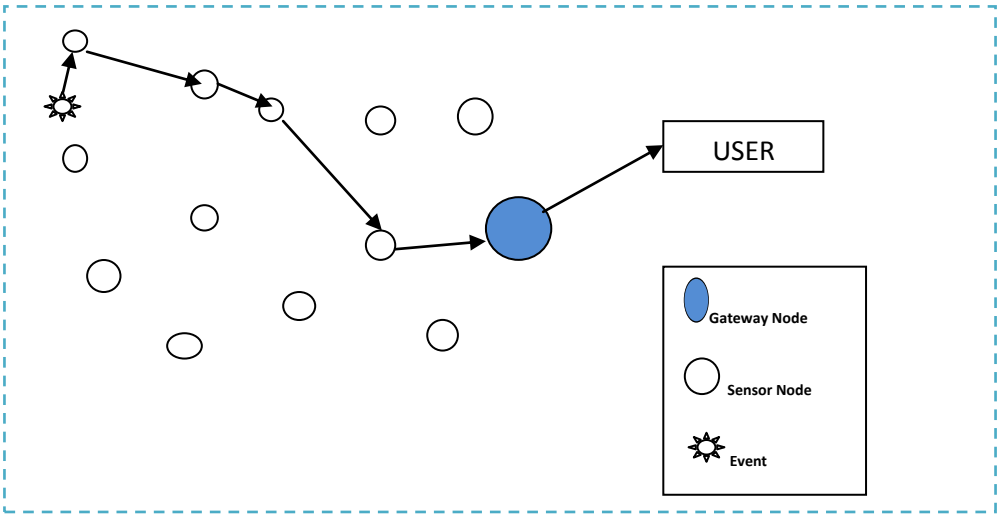

Figure No. 1.1

1.2. Applications of WSN

Nowadays, the development of wireless sensor networks for various applications, such as environmental monitoring, health monitoring, industrial process automation, battlefields surveillance has become possible due to the rapid advances in both of wireless communications and sensor technology. Figures 1.2 illustrate some of the major applications of the wireless sensor networks. The following are a few applications of WSN.

Area monitoring

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors detects enemy presence; a civilian example is the geo-fencing of gas or oil pipelines.

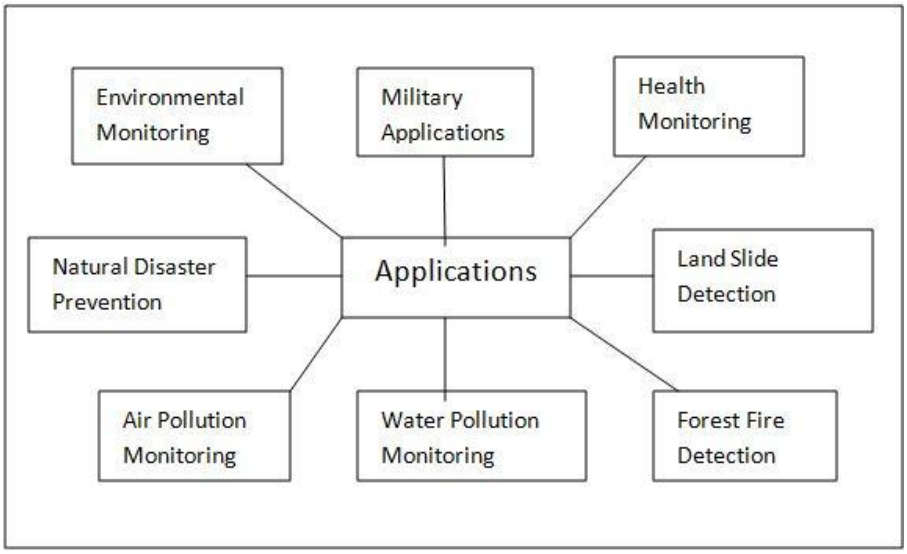

Figure No. 1.2 Application of WSN

Health care monitoring

The medical applications can be of two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user. The implantable medical devices are those that are inserted inside human body. There are many other applications too e.g. body position measurement and location of the person, overall monitoring of ill patients in hospitals and at homes. Body-area networks can collect information about an individual's health, fitness, and energy expenditure.

Environmental/Earth sensing

There are many applications in monitoring environmental parameters, examples of which are given below. They share the extra challenges of harsh environments and reduced power supply.

Air pollution monitoring

Wireless sensor networks have been deployed in several cities (Stockholm, London, and Brisbane) to monitor the concentration of dangerous gases for citizens. These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas.

Forest fire detection

A network of Sensor Nodes can be installed in a forest to detect when a fire has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the fire fighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

Landslide detection

A landslide detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.

Water quality monitoring

Water quality monitoring involves analyzing water properties in dams, rivers, lakes & oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the

permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.

Natural disaster prevention

Wireless sensor networks can effectively act to prevent the consequences of natural disasters, like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

Hence, WSN has now emerged as a topic of great importance because of the wide application area this

1.3. 2D and 3D WSN

The 2D deployment of WSN includes only distances in xy plane and all the sensors are in the same height while 3D deployment xy distance plus sensor height is considered. For some sensor applications the 2D deployment is valid but if the transmission range of a sensor is more than the height of the network then this added third dimension is too large to be neglected. This paves a path for research of 3D WSN separately. The 3D WSN have attracted major attention in past few years since most of the sensor applications e.g underwater sensor Network, under ground Sensor Network and Air borne sensor network are all in 3D.

1.4. Challenges with WSN

The major challenge in the design and development of wireless sensor networks is due to the constraints that are imposed on the sensing, storage, processing, and communication features of the sensors. The figure 1.3 illustrates a few of the major challenges imposed in implementation of WSNs are bandwidth and energy limitations, no physical backbone infrastructure in Wireless Sensor Networks, lack of the fixed topology, energy consumption constraints.

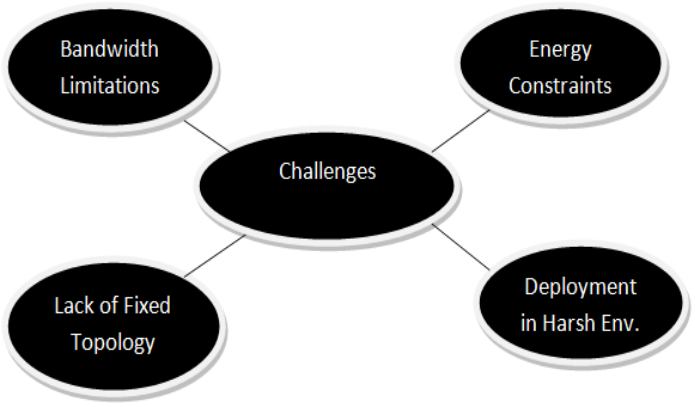

Figure No. 1.3 Major Challenges in WSN

The above mentioned challenges have paved a way for research in formation of a hierarchical structure for WSNs. More specifically formation of backbone in wireless sensor networks which has been explained in the coming sub sections.

1.5. Deployment of Nodes in WSN

The WSN can be deployed in a certain number of ways namely:

a. **Homogeneous:** Such deployment consists of sensor nodes with same abilities in terms of computing power and sensing range, transmission range, node lifetime

b. **Heterogeneous:** Such deployment consists of sensor nodes with different abilities in terms of different computing power and sensing range, transmission range, node lifetime.

c. **Hierarchical:** In this type of deployment nodes in the network may have different types, abilities or assigned roles e.g., Backbone nodes

d. **Non-hierarchical:** All nodes in the network have same types, abilities and assigned roles

1.6. What is a Backbone?

Backbone is a hierarchical network consists of a subset of nodes. The backbone nodes are more powerful as compared to normal nodes in terms of computation capabilities, range, energy etc.

The subset of nodes formed as a part of backbone is responsible for routing-related tasks. Further, the backbone also removes unnecessary transmission links through shutting down some of redundant nodes. The backbone also guarantees the network connectivity in order to deliver data efficiently in a WSN. Backbones are able to perform especial tasks and serve nodes which are not in backbone. With the help of an efficient and reliable backbone topology the routing of the network improves significantly. Further, backbone helps to improve the lifetime of the network, increases the throughput, ensures network connectivity and provides fixed topology to the WSN.

1.7. Problem Statement

WSN can be based on flat architecture which in turns leads to high energy consumption, inefficient design, lack of fixed topology, low throughput etc. In order to provide an efficient architecture WSN based networks can be arranged in a hierarchical manner. We have implemented WSN backbone based on well-known approaches as MST, CDS, and Steiner tree over 2-D and 3-D network design. Such arrangements help to improve the effective lifetime of the network, provide fixed topology, increases bandwidth efficiency etc. There is a need to have a backbone for a network which helps in designing the WSNs in 2-D and 3-D more effectively reducing the challenges otherwise faced in their deployment.

1.8. Objective

The main objectives of our work are summarised as follows:

I. We implement a backbone based WSN for 2D terrains and 3D space with the help of well-known approaches such as MST, CDS, Steiner. We then placed the sink at optimal position with respect to the backbone formed.

II. We also implement greedy and voronoi routing algorithms over the 2D and 3D networks.

1.9. Motivation

WSN is widely used in applications such as health monitoring, military surveillance, water pollution monitoring, natural disaster prevention etc. However, these nodes are energy constrained nodes, lack fixed topology, face problems related to efficient use of bandwidth etc. Hence, for an efficient and reliable design of such networks we require a architectural framework which counters the prevailing problems of WSN. The hierarchical framework formed for deployment of WSNs effectively overcomes the above discussed problem. Hence, we have implemented the well known Backbone formation algorithms which have reduced the existing challenges of WSN deployment.

Most of the most prevalent applications of WSN require them to be deployed in 3D environment. Thus these require framework architecture in 3D scenarios hence considering the major applications of WSN which are in 3-D than in 2-D we have implemented the backbone formation algorithms not only in 2-D but also in 3-D.

1.10. Methodology

The various backbone formation algorithms were designed and implemented. Further, they were executed for varying no. of nodes and various types of node distribution such as Random, Normal, Grid and exponential distribution. Then for performance analysis the various results were compared to find out which algorithm is best suited for which scenario. This whole procedure was done for both 2D distribution of nodes and 3D distribution of nodes.

1.11. Organization

The organization of our work is as follows:

In chapter titled "*Introduction*" we provide a brief in sight of WSN networks, their applications and challenges related to them and how challenges have lead to importance of construction of Backbone. Chapter 2 give the brief references of the work done in related field by various researchers. Chapter 3 discusses the methodology and briefly describes the algorithms implemented. We analysed the performance of algorithms discussed in chapter 3 in chapter 4. We further investigate the results obtained. Finally, in our last chapter we discuss the conclusions and future Scope of our work.

CHAPTER-2

LITERATURE SURVEY

In this chapter we discuss the previous work that has been done on the backbone formation algorithms, routing algorithms and algorithms for optimal sink placement.

2.1. Backbone Formation Algorithms

In "A New Classification of Backbone Formation Algorithms for Wireless Sensor Networks" by Razieh Asgarnezhad and Javad Akbari Torkestani [8] classified the backbone formation algorithms. From varied aspects, backbone formation algorithms can be classified into different types. Keeping some classifications in view, they presented a few instances of these classifications and proposed new hybrid methods.

A. Grid Partitioning-Based Backbone:

In this type of backbone formation algorithm the area of the network is divided into grids and one node in each grid is selected as a backbone node. Geographical adaptive fidelity (GAF) is a grid partitioning algorithm for backbone construction. In this method, each GAF node uses location information itself. The algorithm also divides the network into virtual grids so that nodes are distributed into small virtual grids. Any node in one grid can directly communicate with any node in the other grid. This is why that all nodes in the same grid are equivalent. Thus, one node from each grid is enough to construct a connected backbone. According to virtual grid, any node in adjacent grid can communicate with each other. The communication range is supposed deterministic. Assume r is the size of the virtual grid, and also R is the transmission range. Because any two nodes in adjacent grids can be communicate with each other, this equation can be used for grids.

B. Clustering-Based Backbone:

Clustering is method for partitioning nodes of the network into groups. CHs are used to dominate the other nodes within the clusters. Clustering can provide a hierarchical

architecture for efficient routing. At most existing solutions for clustering usually consists of two phases: construction and maintenance. In the first phase, nodes are chosen to act such as coordinators of the clusters. Then, clustering maintenance is required to reorganize the clusters due to mobility and failure of nodes. Low-energy adaptive clustering hierarchy (LEACH) is a protocol. According to this protocol randomly decide whether or not to become CHs. The parameter used in decision making is the percentage of desired CHs in the network. In this protocol, sensors that decide to become CHs broadcast their decision. Each node reports to the CH with the highest signal strength. Selection of CHs is periodically repeated to balance energy consumption of nodes. The structure of the clusters constructed through LEACH is inefficient because the sink may be very far from many CHs. A clustering algorithm proved that only clustering schemes that position their resultant clusters within the isoclusters of the monitored phenomenon are guaranteed to reduce the nodes' energy consumption and extend the network lifetime. This was the first clustering algorithm; it employs the similarity of the nodes' readings as the main criterion in cluster formation. Another algorithm proposed a mechanism as no two CHs could be direct neighbours and any other node should be adjacent to at least one CH. Each node has a unique node key and also knows the keys of its one hop neighbours. The basic idea behind the CH algorithm is to use the node key as a priority indicator when selecting CH in each cluster. Each node compares its key with the keys of its neighbours. At first, all nodes are undecided. If a undecided node has the lowest key among its undecided neighbours, the node decides to create its own cluster and broadcasts the decision and its key as the cluster key. Upon receiving a message from a neighbour so that announces itself to be a CH, each undecided node will declare itself as a non-CH node and also will inform its neighbours through transmitting a message.

C. Connected Dominating Set (CDS)-Based Backbone:

From various aspects, CDS construction algorithms can be classified into different types. Keeping some classifications in view, we exhibited a few instance of these classifications.

1) UDG and DGB: The CDS construction algorithms can classified into two types: Unit Disk Graph (UDG) based algorithms and Disk Graphs with Bidirectional (DGB) links. In UDG and DGB, the link between any pair of nodes is bidirectional. The nodes

transmission ranges in UDG are the same but in DGB are different. The MCDS in UDG and DGB has been shown to be NP-hard.

2) MIS based and Non-MIS based Independent set (IS) of a graph G is a subset of vertices so that no two vertices are adjacent in the subset. Maximal Independent set (MIS) is an IS, so that it is not a subset of any other IS. Note that in an undirected graph, a MIS is also a Dominating Set (DS). The MIS based algorithms have two kinds of realization. The optimal node selection is based on some criteria such as node degree, rest energy of node, and node id.

3) Centralized algorithm and Decentralized algorithm Algorithms that construct a CDS can be divided into two types: centralized and decentralized. The centralized algorithms in general result in a smaller CDS with a better performance ratio than that of decentralized algorithm. The decentralized algorithms also can be divided into two types: distributed and localized. In distributed algorithms, the decision process is decentralized. But in the localized algorithm, the decision process is not only distributed also requires only a constant number of communication rounds. Most of the distributed algorithms find a MIS and connect this set.

Two CDS construction approaches were proposed. The first algorithm begins through marking all vertices white. It selects the node with the maximal number of white neighbours. The selected vertex is marked black and also its neighbours are marked grey. The algorithm iteratively seeks the gray nodes and their white neighbours and selects the gray node or the pair of nodes, whichever has the maximal number of white neighbours. The selected node or the selected pair of nodes is marked black, and also their white neighbours marked grey. Finally, the algorithm terminates, when all of the vertices are marked grey or black. All the black nodes form a CDS. This algorithm results in a CDS of size at most $2(1+H(\Delta)).\left|OPT\right|$, where H is the harmonic function and OPT refers to an MCDS.

The second algorithm also begins through colouring all nodes white. A piece is defined to be either a connected black component or a white node. The algorithm includes two phases. The first phase iteratively selects a node that yield the maximum reduction of the number of pieces. A node is marked black and its white neighbours are marked grey when it is selected. The first phase terminates when no white node left. There exists at most

$|OPT|$ number of connected black components. The second phase constructs a Steiner Tree until connects all the black nodes through colouring chains of two grey and black nodes. The size of the resulting CDS formed via all black nodes is at most $(3+\ln(\Delta)).|OPT|$.

A greedy algorithm was proposed for MCDS in UDGs. At first, all nodes are coloured white. The construction of a CDS includes four phases. The first phase is computing an MIS and colouring all its members red. In the second phase, a node selects that it can decrease the maximum number of pieces. This node is coloured black and all its non-black neighbours are coloured grey. After the second phase, we still have some white nodes left. The third phase will compute a spanning tree for each connected component in the sub graph reduced through all white nodes. All non-leaf tree nodes are coloured black but leaf nodes are coloured grey. The last phase will scan chains of two gray nodes to connect disjoint black components.

The conclusion of the research paper was that the backbone has proven to be an effective construct within which to solve a variety of problems that arise in WSNs. In this paper, we classified backbone formation algorithms and a few instances of these classifications and proposed hybrid approaches of these classifications. Also, we have surveyed some famous backbone formation algorithms in term of time and message complexity. Significant attention has been paid to backbone formation algorithms yielding a large number of publications. Backbone construction depends on the task to be carried. A backbone reduces the communication overhead, increases the bandwidth efficiency, decreases the overall energy consumption and at last increases network effective lifetime of a WSN. The important issue that we can be reached is selection algorithm according to our use.

Connected Dominating set

In “Approximation Algorithms for Connected Dominating Sets” by Sudipto Guhay [9] and Samir Khuller presented two approximation algorithms for CDS problem. The first algorithm develops a greedy algorithm for solving the problem.

Algorithm I

An algorithm was introduced that calculates a connected dominating set, by growing a tree.

The idea behind the algorithm is the following:

We grow a tree T which is started from the vertex of maximum degree. At each step a v in T is picked and scanned. By Scanning a vertex, it adds edges to T from v to all its neighbours not in T. In the end we get a spanning tree T, and all the non leaf nodes are picked as the Connected Dominating set. Initially all vertices are unmarked. When we scan a vertex (colour node black), we mark all its neighbours that are not in T and add them to T (colour them grey). Thus marked nodes that have not been scanned are leaves in T (gray nodes). The algorithm continues scanning marked nodes, until all the vertices *v* are marked (gray or black). The set of scanned nodes (black nodes) will form the CDS in the end.

The main question is what rule should we use for picking a vertex to be scanned? A natural choice is to pick the vertex that has the maximum number of unmarked (white) neighbours.

Algorithm II

An alternate approach to growing one connected tree is to grow separate components that form a dominating set and to then connect them together. One straightforward approach is to find a dominating set using a greedy heuristic, and to use a Steiner tree approximation to connect it. The algorithm runs in two phases. At the start of the first phase all nodes are coloured white. Each time we include a vertex in the dominating set, we colour it black. Nodes that are dominated are coloured grey (once they are adjacent to a black node). In the first phase the algorithm picks a node at each step and colours it black, colouring all adjacent white nodes gray. A piece is defined as a white node or a black connected component. At each step we pick a node to colour black that gives the maximum (non-zero) reduction in the number of pieces. We show that at the end of this phase if no vertex gives a non-zero reduction to the number of pieces, then there are no white nodes left.

In the second phase, we have a collection of black connected components that we need to connect. Recursively connect pairs of black components by choosing a chain of two vertices, until there is one black connected component. Our final solution is the set of black vertices that form the connected component.

Steiner Algorithm -1

In "Spanning Trees and Optimization Problems," by Bang Ye Wu and Kun-Mao Chao [1], they have explained Steiner Trees as while spanning tree spans all vertices of a given graph, a Steiner tree spans a given subset of vertices. In the Steiner minimal tree problem, the vertices are divided into two parts: terminals and non terminal vertices. The terminals are the given vertices which must be included in the solution. The cost of a Steiner tree is defined as the total edge weight. A Steiner tree may contain some non terminal vertices to reduce the cost. Let V be a set of vertices. In general, we are given a set $L \subset V$ of terminals and a metric defining the distance between any two vertices in V . The objective is to find a connected sub-graph spanning all the terminals of minimal total cost. Since the distances are all nonnegative in a metric, the solution is a tree structure. Depending on the given metric, two versions of the Steiner tree problem have been studied.

• **(Graph) Steiner minimal trees (SMT):** In this version, the vertex set and metric is given by a finite graph.

• **Euclidean Steiner minimal trees (Euclidean SMT):** In this version, V is the entire Euclidean space and thus infinite. Usually the metric is given by the Euclidean distance (L2-norm). That is, for two points with coordinates (x1,y2) and (x2,y2), the distance is p $(x1 - x2)^2 + (y1 - y2)^2$.

A well-known method to approximate an SMT as has been discussed in [9] is to use a minimal spanning tree (MST). First we construct the metric closure on L, i.e., a complete graph with vertices L and edge weights equal to the shortest path lengths. Then we find an MST on the closure, in which each edge corresponds to one shortest path on the original graph. Finally the MST is transformed back to a Steiner tree by replacing each edge with the shortest path and some straightforward post processing to remove any possible cycle.

Steiner Algorithm-2

In this algorithm described by Mohamed Younis and Rahul Waknis, in "Connectivity Restoration in WSN using Steiner Tree Approximations"[7], we draw the MST over the network in Phase I. Further, we introduce new Steiner Nodes by finding centre of the circle of radius r which spans some nodes in MST (at most 3). When such centre is found, remove the edges that link these nodes in spanning tree and connecting them to the centre of the disc yields a lower cost tree

2.2. Routing in WSN

Advances in wireless sensor networks (WSNs) technology have been undergoing a revolution that promises a significant impact on society. Most existing wire-less systems and protocols are based on two-dimensional design, where all wireless nodes are distributed in a two-dimensional (2D) plane. However, 2D assumption may no longer be valid if a wireless network is deployed in space, atmosphere, or ocean, where nodes of a network are distributed over a three-dimensional (3D) space and the differences in the third dimension are too large to be ignored. In fact, recent interest in wireless sensor networks hints at the strong need to design 3D wireless networks. The characteristics of 3D wireless sensor networks require more effective methods to ensure routing and data dissemination protocols in these networks. [5]

The following are the few Routing algorithms of WSN which have be taken from Prosenjit Bose and Pat Morin in "Online Routing in Triangulations" [2]

Greedy Routing. The greedy routing (GR) algorithm always moves the packet to the neighbour gdy(vcur) of vcur that minimizes dist(gdy(vcur),vdst), where dist(p,q) denotes the Euclidean distance between p and q. In the case of ties, one of the vertices is chosen arbitrarily. The greedy routing algorithm can be defeated by a triangulation T in two ways: (1) the packet can get trapped moving back and forth on an edge of the triangulation (2) the packet can get trapped on a cycle of three or more vertices.

Compass Routing: The compass routing (CR) algorithm always moves the packet to the vertex cmp(vcur) that minimizes the angle vdst, vcur, cmp(vcur) over all vertices adjacent to vcur. Here the angle is taken to be the smaller of the two angles as measured in the

clockwise and counter clockwise directions. In the case of ties, one of the (at most 2) vertices is chosen using some arbitrary deterministic rule.

Randomized Compass Routing: In this section, we consider a randomized routing algorithm that is not defeated by any triangulation. Let cw(v) be the vertex in N(v) that minimizes the clockwise angle vdst, v, cw(v) and let ccw(v) be the vertex in N(v) that minimizes the counter clockwise angle vdst, v, ccw(v). Then the randomized compass routing (RCR) algorithm moves the packet to one of {cw(vcur),ccw(vcur)} with equal probability.

Right-Hand Routing: The folklore "right-hand rule" for exploring a maze states that if a player in a maze walks around never lifting her right-hand from the wall, then she will eventually visit every wall in the maze. More specifically, if the maze is the face of a connected planar straight line graph, the player will visit every edge and vertex of the face.

Delaunay Triangulation Routing: The Delaunay triangulation approximates the complete Euclidean graph to within a constant factor in terms of shortest path length. In the following we will use the notation x(p) (resp. y(p)) to denote the x-coordinate (resp. y-coordinate) of the point p, and the notation |X| to denote the Euclidean length of the path X. Consider the directed line segment from vsrc to vdst. This segment intersects regions of the Voronoi diagram in some order, say R_0 ... R_{m-1}, where R0 is the Voronoi region of vsrc and Rm−1 is the Voronoi region of vdst. The Voronoi routing (VR) algorithm for Delaunay triangulations moves the packet from vsrc to vdst along the path $v_0...v_{m-1}$ where v_i is the site defining Ri. An example of a path obtained by the Voronoi routing algorithm is shown in Fig. 3.2. Since the Voronoi region of a vertex v can be computed given only the neighbours of v in the Delaunay triangulation, it follows that the Voronoi routing algorithm is an O(1) memory routing algorithm. We prove an even stronger result by giving an algorithm that finds a path whose cost is at most a constant times dist(vsrc,vdst).

The results shown in this research paper included:

(1) Two deterministic memoryless routing algorithms, one that works for all Delaunay triangulations and the other that works for all regular triangulations

(2) A randomized memoryless algorithm that works for all triangulations

(3) An O(1) memory algorithm that works for all convex subdivisions

(4) An O(1) memory algorithm that approximates the shortest path in Delaunay triangulations

(5) Theoretical and experimental results on the competitiveness of these algorithms.

3D Greedy Routing in Delaunay Triangulations

In the "The Art of WSN" [2] a routing technique 3D Greedy Routing in Delaunay Triangulations is explained. Delaunay triangulation has been used as routing topology for wireless ad hoc networks since building the Delaunay triangulation needs global information and the length of a Delaunay edge could be longer than the maximum transmission range, several methods use local structures to approximate the Delaunay triangulation. This also break the delivery guarantee of 3D greedy routing over them. Recently, Lam and Qian proposed to use a virtual Delaunay triangulation to aid geographic routing. They called their routing method *multi-hop Delaunay triangulation* (MDT). The key idea is to relax the requirement that every node be able to communicate directly with its neighbour in Delaunay triangulation. In a MDT, the neighbour of a node may not be a physical neighbour. A virtual link represents a multi-hop path between them. When the current node *u* has a packet with destination **t**, it forwards to a physical neighbour closest to **t** if *u* is not a local minimum; otherwise the packet is forwarded via a virtual link to a multi-hop Delaunay neighbour closest to. Due to Theorem, MDT can guarantee the packet delivery using a finite number of hops. Simulations also show MDT has low routing stretch from efficient forwarding of packets out of local minimum. In S.S. Lam, C. Qian, Geographic routing in d-dimensional spaces with guaranteed delivery.

And low stretch, in Proceedings of the ACM SIGMETRICS joint international conference on Measurement and Modelling of Computer Systems, SIGMETRICS '11, the authors provided detailed methods to construct and maintain the multi-hop Delaunay triangulation at each node. However, such construction and maintaining are not purely localized. MDT also works for 2D networks or networks with higher dimension. Liu and Wu also used a Delaunay structure to divide the 3D network into closed subspace and then proposed a greedy-hull-greedy (GHG) routing, which uses hull routing over the subspace to escape the local minimum and guarantee the delivery. It is a 3D analogue to face routing in 2D. First, a 3D partial unit Delaunay triangulation (PUDT) is constructed to define network hulls (structures corresponding to subspaces) in 3D networks. Here, PUDT construction basically removes intersecting triangles and edges. It can be proven that if there is no intersecting edge and triangle, then there is no overlapping tetrahedra. This is because when two tetrahedra overlap, one of the four triangles on the first tetrahedron must intersect a triangle on the second tetrahedron; moreover, if two triangles intersect, an edge of one of the triangles must intersect the other triangle. Notice that unlike in Delaunay triangulation 3D greedy can encounter a local minimum in PUDT. Once a packet travels to a local-minimum during 3Dgreedy forwarding in GHG, one of the adjacent hulls of the local-minimum is selected such that the message can recover from the local-minimum by searching the nodes on this hull. GHG selects the hull whose subspace contains the segment connecting the local-minimum and destination, and uses a depth-first-search to travel this hull. Eventually, it can send the message to the node where greedy can be recovered. This local search of possible recover node over the surface of the subspace is very similar to the one used in GRG.

2.3. Optimal Sink Placement

In "Optimal sink placement in backbone assisted wireless sensor networks" [9], by Snigdh said that by placing the sink at a specific position, energy scavenging and delay constraints can effectively be controlled in WSN. Typical WSN scenarios assume the message routed towards the sink that usually is the root of the tree. This strategy suffers from the problem of hot spots ending into communication disruption due to single node failure closest to the sink. Adopting a graph based topology offers us the choice of

variable routes and sink placement which usually turns out to be the geographical centre or the end of the region under observation. The argument biasing this structure is based on achieving maximal coverage under one sink. However, their research confirmed that graph centroid placement of the sink node is better in terms of network delay and energy consumption rather than having a sink rooted tree for a communication backbone. Also, our results show that the message forwarding to the sink (in terms of the hop count) is the least for either the graph theoretic centre or centroid. This ensures minimum delay and lesser energy consumption per node and hence a longer lifetime.

CHAPTER – 3

SYSTEM DEVEOPMENT

We implemented three well known algorithms for backbone formation for Wireless Sensor Network in 2-D and 3-D. We developed an experimental model for system development. The following three objectives were to be achieved:

1. Backbone Formation in 2D and 3D using MST, CDS and Steiner Tree
2. Optimal Sink placement
3. Implementing routing algorithms in 2D and 3D i.e. Voronoi and Greedy routing

3.1. Two Dimensional System Development

The following steps were taken for the system development:

1. Topology construction using random distribution of nodes
2. Topology construction for different distribution of 100 nodes i.e uniform random, normal, grid and exponential distribution.
3. Obtaining the adjacency matrix of the graph obtained in step 1. According to the constant range of each node, checking if any two neighbouring nodes are in each others range or not.
4. Computing the Delaunay Triangulation of the obtained graph.
5. Computing the Voronoi diagram of obtained graph.
6. Computing the backbone of the network obtained.
7. Implementing the routing algorithms i.e Greedy and Voronoi Routing.
8. Optimal sink placement.

3.1.1. Delaunay Triangulations

A Delaunay triangulation for a set **P** of points in a plane is a triangulation DT(**P**) such that no point in **P** is inside the circumcircle of any triangle in DT(**P**). there are numerous algorithms for computing triangulations, it is the favorable geometric properties of the Delaunay triangulation that make it so useful. The fundamental property is the Delaunay criterion. In the case of 2-D triangulations this is often called the empty circumcircle

criterion. For a set of points in 2-D, a Delaunay triangulation of these points ensures the circumcircle associated with each triangle contains no other point in its interior.

1. We used *dt = delaunay(X,Y)* to create a 2-D Delaunay triangulation of the points (X,Y), where X and Y are column-vectors.
2. The function triplot(tri, X(:,1), X(:,2)) was used to plot the hence obtained triangulations.

3.1.2. Voronoi Diagram

A Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells. The Voronoi diagram of a set of points is dual to its Delaunay triangulation. Put simply, it's a diagram created by taking pairs of points that are close together and drawing a line that is equidistant between them and perpendicular to the line connecting them. The Voronoi diagram of a discrete set of points X decomposes the space around each point X(i) into a region of influence R{i}. Locations within the region are closer to point i than any other point. The region of influence is called the Voronoi region. The collection of all the Voronoi regions is the Voronoi diagram.

For constructing the Voronoi Diagram:

1. We first constructed the Delaunay Triangulations using *dt = delaunay(X,Y).*
2. [V, C] = *voronoiDiagram(dt)* was used to construct the voronoi diagram.

3.1.3. Minimum Spanning Tree

A minimum spanning tree is a spanning tree of a connected, undirected graph. It connects all the vertices together with the minimal total weighting for its edges. A single graph can have many different spanning trees. There are different algorithms for construction of

Minimal Spanning Tree Construction such as Prims and Kruskals. The MST was constructed using the following algorithm:

MST-PRIM ***(G,V,E,w)***

1. **for each** *u* € V[G]
2. **do** *key[u]*←∞
3. Π[u]←NIL
4. *Key[r]←0*
5. *Q*←V[G]
6. **While** ***Q≠Φ***
7. do *u* ← EXTRACT-MIN(Q)
8. for each *v* € *Adj[u]*
9. Do if *v Q and w(u,v) < key[v]*
10. Then π[v] ← *u*
11. *Key[u] ← w(u,v)*

Analysis of Algorithm: The total time for Prims algorithm is *O(E log(V)).*

3.1.4. Connected Dominating Set

In graph theory, a connected dominated set and a maximum leaf spanning tree are two closely related structures defined on an undirected graph. A connected dominating set of a graph *G* is a set *D* of vertices with two properties: Any node in *D* can reach any other node in *D* by a path that stays entirely within *D*. That is, *D* induces a connected subgraph of *G*. Every vertex in *G* either belongs to *D* or is adjacent to a vertex in *D*. We used the following algorithm for construction of CDS.

CDS-*G(V,E,w)*

1. *W*← 1 to no. of nodes, *G* ←Φ, *B* ← Φ
2. *B*← Max_Deg_Node
3. *G*← Neighbour(Max_Deg_Node)
4. **While** (union(*G*,*B*)≠ *W*)
5. **for** each node € *G*
6. *B*←Find_Next_Black_Node
7. *G*←Neighbour(B)
8. Remove B from G

The main question was how do we find the next node to be added in *B* array, for that we chose the node that we chose the node that had the maximum no. of unscanned or unmarked nodes i.e the nodes which have not earlier been entered in *G or B* array.

Analysis of algorithm: The above algorithm can be implemented in *O(m)* time.

3.1.5. Steiner Tree Algorithm-1

While a spanning tree spans all vertices of a given graph, a Steiner tree spans a given subset of vertices. In the Steiner minimal tree problem, the vertices are divided into two parts: terminals and non terminal vertices. The objective is to find a connected sub graph spanning all the terminals of minimal total cost. The cost of a Steiner tree is defined as the total edge weight.

$$\text{Cost} = \sum w_{tree} \quad (1)$$

Where w_{tree} weight (Euclidean distance) of each respective edge in the tree.

$$\text{Steiner Ratio} = \text{Cost(MST)}/\text{Cost(Steiner Tree)} \quad (2)$$

Steiner ratio in general metric is 2. The following algorithm was used to construct the Steiner Tree Algorithm:

Steiner Tree $G(V,E,w)$ and Terminal set of vertices $L \in V$

1. Construct the metric closure G_L on the terminal set L
2. Find an MST T_L on G_L.
3. $T \leftarrow \emptyset$.
4. **for** each edge $e = (u,v) \in E(T_L)$ in a depth-first-search order of T_L do
5. Find a shortest path P from u to v on G.
6. **if** Length(P) > 2 **then**
7. Add P to T;
8. **else** Let *pi* and *pj* be the first and the last vertices already in T
9. Add subpaths from u to *pi* and from *pj* to v to T.
10. Output T.

Analysis of the algorithm: The time complexity of the algorithm is O(|V ||L|2), dominated by the construction of the metric closure.

3.1.6. Steiner Tree Algorithm-2

Steiner Tree optimization to minimize overall cost by introducing a set of vertices called Steiner points. We aim at minimizing the total distance of transmission and hence the energy consumed in transmission. The following algorithm was used in constructing the Steiner Algorithm:

1. Form MST of Network
2. Introduce new Steiner Nodes by finding center of the circle of radius r which spans some nodes in MST (at most 3)
3. When such center is found, remove the edges that link these nodes in spanning tree and connecting them to the center of the disc yields a lower cost tree.

3.1.7. Greedy Routing

The greedy routing (GR) algorithm always moves the packet to the neighbour of current vertex that minimizes the distance between the neighbour and the final destination. This whole cycle is repeated until we reach the final destination. The pseudo algorithm is as follows:

Greedy Routing- *G(V,E,w)* and *scr, dest*

1. *Curnode = scr, Path* ← *NIL*
2. **While** *curnode* ≠ *dest*
3. *Nxtnode* ← Minimze_Dist(Neighbour(*curnode), dest*)
4. *Curnode = nxtnode*
5. Add *curnode to Path*
6. Output *Path*

Analysis of Algorithm: The complexity of this algorithm was $O(n^2)$ i.e for finding the next node from the set of neighbours of current node.

3.1.8. Voronoi Routing

In this algorithm we used the voronoi diagram previously constructed. The packet was transmitted based the constructed voronoi cells in following manner:

Voronoi Routing- *G(V,E,w) and src, dest*

1. *Line* ← Join(src,dest)
2. *Curnode=src, Path* ← *NIL*
3. **While** *curnode* ≠ *dest*
4. Find_Intersection(*Line*, $R_0 \ldots R_{m-1}$)
5. *Curnode* ← Center($R_{i)}$
6. Add *curnode to Path*
7. Output *Path*

3.1.9. Optimal Sink Placement

Once the backbone has been formed we need to decide on where the sink has to be placed. At fist the question what is a sink node? In wireless sensor networks (WSNs), all the data collected by the sensor nodes are forwarded to a sink node. Therefore, the placement of the sink node has a great impact on the energy consumption and lifetime of WSNs. Hence, we used three strategies for placing the sink node

1. At the geographical center
   Assuming 'N' sensors placed in a field and represented as (xi,yi) coordinates in two dimensional plane, the 'GC(X,Y)' of any topology is calculated as:
   GC(X,Y) =∑xi/n,∑yi/n.
2. At the node on backbone node with minimum eccentricity i.e called the center
   Eccentricity of vertex *x* is defined as distance from *x* to the vertex farthest from *x* in *V(T)*. Thus, *E(x)* =max *dT(x,V)*, for *V €V(T)*. Therefore, the vertex with minimum eccentricity is the centre.
3. At the node with maximum weight called the centroid

In a tree *T (V,E)*, the number of subtrees of any vertex *v* is equal to its degree and sum of branches (in a subtree) corresponds to the weight of the subtree at *v*. The weight of the heaviest subtree of vertex *v* is designated as weight of vertex *v* and the minimum weighted vertex is designated as Centroid of tree *T*. Once the sink is placed we need to compare which placement of sink is best suited in terms of Energy Consumption.

Random Event Detection

For comparing the performance of the optimal sink placement we generated a random event at any co-ordinate (x,y) and find which all nodes are in the vicinity of the event. The nodes in vicinity act as source nodes and start sending the sensed data to the sink. Meanwhile we measure the average energy used in transmitting the packet to the sink node. Hence, we measure the performance of the optimal sink placement.

3.2. Three Dimensional System Development

The following steps were taken for the system development:

1. Topology construction using random distribution of nodes.
2. Obtaining the adjacency matrix of the graph obtained in step 1. According to the constant range of each node, checking if any two neighbouring nodes are in each others range or not.
3. Computing the Delaunay Triangulation of the obtained graph.
4. Computing the Voronoi diagram of obtained graph.
5. Computing the backbone of the network obtained.
6. Implementing the routing algorithms i.e Greedy and Voronoi Routing.
7. Optimal sink placement.

3.2.1. Delaunay Triangulations

In the case of 3-D triangulations this is often called the empty circumsphere criterion. For a set of points in 3-D, a Delaunay triangulation of these points ensures the circumsphere associated with each tetrahedron contains no other point in its interior. This property is important. The circumsphere associated with a certain point should be empty. It does not contain a point in its interior.

For constructing the Delaunay triangulations:

1. *dt=delaunay(x,y,z);* function was used.
2. *tetramesh* fuction of matlab was used to plot the triangulations in 3D.

3.2.2. Voronoi Diagram

A Voronoi diagram is a partitioning of 3D plane into regions based on distance to points in a specific subset of the plane. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells in 3D. The Voronoi diagram of a set of points is dual to its 3D Delaunay triangulation. It's a diagram created by taking pairs of points that are close together and drawing a line that is

equidistant between them and perpendicular to the line connecting them. The Voronoi diagram of a discrete set of points X decomposes the space around each point X(i) into a region of influence R{i}. Locations within the region are closer to point i than any other point. The region of influence is called the Voronoi region. The collection of all the Voronoi regions is the Voronoi diagram. For constructing the Voronoi Diagram:

1. We first constructed the Delaunay Triangulations using *dt = DelaunayTri(X)*
2. [V, C] = *voronoiDiagram(dt)* was used to construct the voronoi diagram.

3.2.3. Minimal Spanning Tree

A minimum spanning tree is a spanning tree of a connected, undirected graph. It connects all the vertices together with the minimal total weighting for its edges. A single graph can have many different spanning trees. There are different algorithms for construction of Minimal Spanning Tree Construction such as Prims and Kruskals. The weights *w* in the following algorithm refer to the Euclidean distance between two points in 3D. The MST was constructed using the following algorithm:

MST-PRIM ***(G,V,E,w)***

1. Construct the metric closure G_L on the terminal set L
2. Find an MST T_L on G_L.
3. $T \leftarrow \emptyset$.
4. **for** each edge $e = (u,v) \in E(T_L)$ in a depth-first-search order of T_L do
5. Find a shortest path P from u to v on G.
6. **if** Length(P) > 2 **then**
7. Add P to T;
8. **else** Let *pi* and *pj* be the first and the last vertices already in T
9. Add subpaths from u to *pi* and from *pj* to v to T.
10. Output T.

Analysis of Algorithm: The total time for Prims algorithm is *O(E log(V))*.

3.2.4. Connected Dominating Set

In graph theory, a connected dominated set and a maximum leaf spanning tree are two closely related structures defined on an undirected graph. A connected dominating set of a graph *G* is a set *D* of vertices with two properties: Any node in *D* can reach any other

node in *D* by a path that stays entirely within *D*. That is, *D* induces a connected subgraph of *G*. Every vertex in *G* either belongs to *D* or is adjacent to a vertex in *D*. We used the following algorithm for construction of CDS and the weight *w* refers to the Euclidean distance between two points in 3D

CDS-*G(V,E,w)*

1. *W*← 1 to no. of nodes, *G* ←Φ, *B* ← Φ
2. *B*← Max_Deg_Node
3. *G*← Neighbour(Max_Deg_Node)
4. **While** (union(*G*,*B*)≠ *W*)
5. **for** each node € *G*
6. *B*←Find_Next_Black_Node
7. *G*←Neighbour(B)
8. Remove B from G

The main question was how do we find the next node to be added in *B* array, for that we chose the node that we chose the node that had the maximum no. of unscanned or unmarked nodes i.e the nodes which have not earlier been entered in *G or B* array.

Analysis of algorithm: The above algorithm can be implemented in *O(m)* time where *m* is the no. of edges in the graph.

3.2.5. Steiner Tree Algorithm-1

While a spanning tree spans all vertices of a given graph, a Steiner tree spans a given subset of vertices. In the Steiner minimal tree problem, the vertices are divided into two parts: terminals and non terminal vertices. The objective is to find a connected sub graph spanning all the terminals of minimal total cost. The cost of a Steiner tree is defined as the total edge weight.

$$\text{Cost} = \sum w_{tree} \quad (1)$$

where w_{tree} weight (Euclidean distance) of each respective edge in the tree.

$$\text{Steiner Ratio} = \text{Cost(MST)}/\text{Cost(Steiner Tree)} \quad (2)$$

Steiner ratio in general metric is 2. The following algorithm was used to construct the Steiner Tree Algorithm wherein weight *w* refers to the Euclidean distance between two points in 3D.

Steiner Tree *G(V,E,w)* and Terminal set of vertices $L \in V$

1. Construct the metric closure G_L on the terminal set L
2. Find an MST T_L on G_L.
3. $T \leftarrow \emptyset$.
4. **for** each edge $e = (u,v) \in E(T_L)$ in a depth-first-search order of T_L do
5. Find a shortest path P from u to v on G.
6. **if** Length(P) > 2 **then**
7. Add P to T;
8. **else** Let *pi* and *pj* be the first and the last vertices already in T
9. Add subpaths from u to *pi* and from *pj* to v to T.
10. Output T.

Analysis of the algorithm: The time complexity of the algorithm is O(|V ||L|2), dominated by the construction of the metric closure.

3.2.6. Greedy Routing

The greedy routing (GR) algorithm always moves the packet to the neighbour of current vertex that minimizes the distance between the neighbour and the final destination. This whole cycle is repeated until we reach the final destination. The pseudo algorithm is as follows where *w* refers to the weight i.e. Euclidean distance between two points in 3D.

Greedy Routing-*G(V,E,w)* and *scr, dest*

1. *Curnode = scr, Path* ← *NIL*
2. **While** *curnode* ≠ *dest*
3. *Nxtnode* ← Minimze_Dist(Neighbour(*curnode), dest*)
4. *Curnode = nxtnode*
5. Add *curnode to Path*
6. Output *Path*

Analysis of Algorithm: The complexity of this algorithm was $O(n^2)$ i.e for finding the next node from the set of neighbours of current node.

3.2.7. Voronoi Routing

In this algorithm we used the voronoi diagram previously constructed. The packet was transmitted based the constructed voronoi cells in following manner:

Voronoi Routing- *G(V,E,w) and src, dest*

1. *Line* ← Join(src,dest)
2. *Curnode=src, Path* ← *NIL*
3. **While** *curnode* ≠ *dest*
4. Find_Intersection(*Line*, R_0 ... R_{m-1})
5. *Curnode* ← Center(R_i)
6. Add *curnode to Path*
7. Output *Path*

3.2.7. Optimal Sink Placement

Once the backbone has been formed we need to decide on where the sink has to be placed. At fist the question what is a sink node? In wireless sensor networks (WSNs), all the data collected by the sensor nodes are forwarded to a sink node. Therefore, the placement of the sink node has a great impact on the energy consumption and lifetime of WSNs. Hence, we used three strategies for placing the sink node

1. At the geographical center

   Assuming 'N' sensors placed in a field and represented as (x_i,y_i) coordinates in two dimensional plane, the 'GC(X,Y)' of any topology is calculated as:

   GC(X,Y) =∑xi/n,∑yi/n, ∑zi/n.
2. At the node on backbone node with minimum eccentricity i.e called the center

   Eccentricity of vertex *x* is defined as distance from *x* to the vertex farthest from *x* in *V(T)*. Thus, *E(x)* =max *dT(x,V)*, for *V* €*V(T)*. Therefore, the vertex with minimum eccentricity is the centre.
3. At the node with maximum weight called the centroid

   In a tree *T (V,E)*, the number of subtrees of any vertex *v* is equal to its degree and sum of branches (in a subtree) corresponds to the weight of the subtree at *v*. The weight of the heaviest subtree of vertex *v* is designated as weight of vertex *v* and the minimum weighted vertex is designated as Centroid of tree *T*.

Once the sink is placed we need to compare which placement of sink is best suited in terms of Energy Consumption.

Random Event Detection

For comparing the performance of the optimal sink placement we generated a random event at any co-ordinate (x,y,z) and find which all nodes are in the vicinity of the event. The nodes in vicinity act as source nodes and start sending the sensed data to the sink. Meanwhile we measure the average energy used in transmitting the packet to the sink node. Hence, we measure the performance of the optimal sink placement.

CHAPTER – 4

PERFORMANCE ANALYSIS

We now provide the simulation parameters and setting for the evaluation of the performance of various algorithms implemented for backbone formation. In section 4.1 we provide topological parameters for evaluation of our network.

4.1. Simulation Environment

As shown in Table 4.1, four types of node distribution are considered in our work are uniform random distribution, normal distribution, exponential distribution and grid distribution.

Table No. 4.1 Simulation Parameters

Parameter	*Value*
Topology dimension	2-D, 3-D
Node distribution	random, normal, exponential, grid
Network size (number of nodes)	{25, 50, 75, 100}
Network area	100X100 in 2D
	100X100X100 in 3D
Range	50 m
Tool Used	MATLAB R2012A

We also taken into account the different dimensions for simulation as 2-D and 3-D. Further, the transmission range of the sensor node is taken as 50 meters of range. The number of nodes varies from 25 to 100 in steps of 25 nodes. The network model is taken as 100 *100 for 2-D , and 100*100*100 for 3-D.

Deployment of nodes: The deployment of nodes in 2D and 3D can be either known i.e deterministic and unknown i.e. un-deterministic. For deterministic deployment we have used grid distribution in 2D and 3D. For un-determinist deployment of nodes we have used uniform random, normal and exponential distribution of nodes. Figure 4.1 (a) – (d) illustrates the four types of distribution considered in our work in 2D.

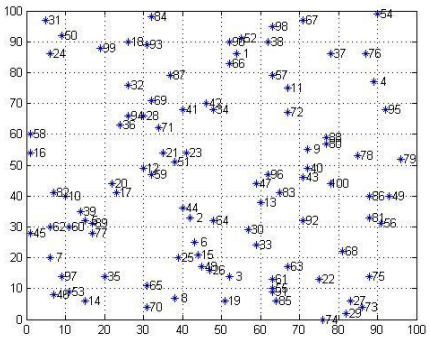

Figure 4.1 (a): Uniform random in 2D

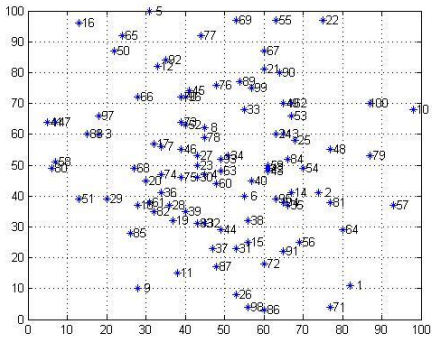

Figure 4.1 (b): Normal in 2D

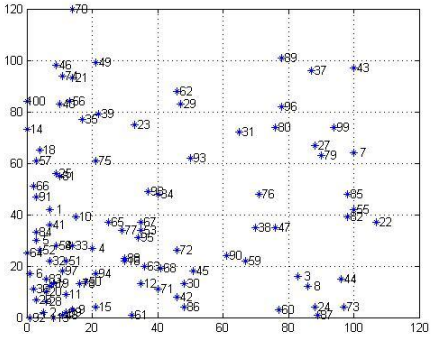

Figure 4.1 (c): Exponential in 2D

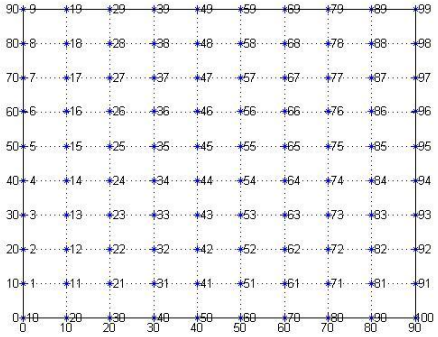

Figure 4.1 (a): Grid in 2D

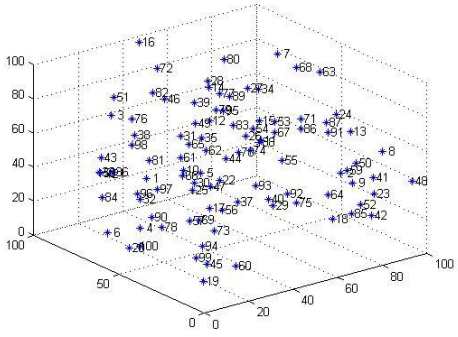

Figure 4.2 (a): Uniform Random in 3-D

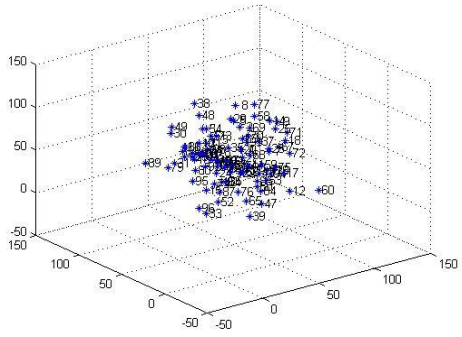

Figure 4.2 (b): Normal in 3-D

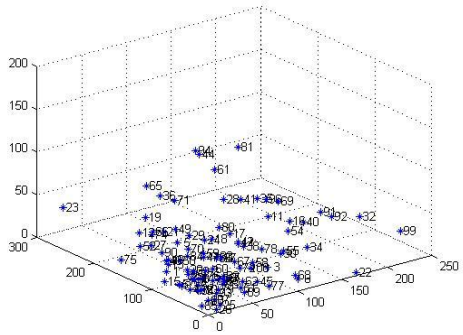

Figure 4.2 (c): Exponential in 3-D

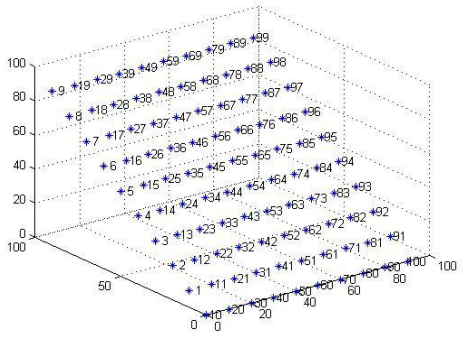

Figure 4.2 (d): Grid in 3-D

The four distributions considered are as follows:

a. Uniform Random: The 100 nodes are uniformly distributed over the 2-D or the 3-D space. The figure no. 4.1(a) and 4.2(a) shows the uniform random distribution in 2D and 3D respectively.

b. Normal Distribution: 100 nodes with range 50 m was deployed in 100X100 and 100X100X100 space. Mean and standard deviation was taken out to be 50 and 25 respectively. The hence obtained distribution is shown in figure 4.1(b) and 4.2(b) respectively for 2D as well as for 3D.

c. Exponential Distribution: 100 nodes with range 50 m was deployed in 100X100 and 100X100X100 space. Mean was taken out to be 50. The figure 4.1(c) and 4.2(c) shows the distribution of nodes hence obtained.

d. Grid Distribution**:** In this distribution the 100 nodes were distributed in 100X100 space with transmission range of 50 m in form of a grid. This type of distribution is deterministic. The figure 4.1(d) and 4.2(d) shows the distribution of nodes hence obtained.

We now construct backbone over these nodes with the help of well-known approaches available in the literature in our next section.

4.2. Simulation Results for Backbone construction of nodes over 2D

The backbone formation algorithms specified in Chapter 3 were implemented over the distribution of nodes obtained in the previous sections. Metrics selected for measuring the performance of backbone nodes is percentage of the nodes in the backbone.

Table No. 4.2 Percentage Nodes in Backbone

Distribution/Percentage of nodes in the backbone	Uniform Random Distribution	Grid distribution	Normal distribution	Exponential distribution
%age of Nodes in CDS Backbone	11	9	12	19
%age of nodes in Steiner Backbone	28	24	11	33
%age of nodes in MST Backbone	100	100	100	100

Table 4.2 illustrates the percentage nodes in the backbone for various distributions of nodes, the number of nodes were taken to be 100, dimension was taken as 100* 100 for 2D.

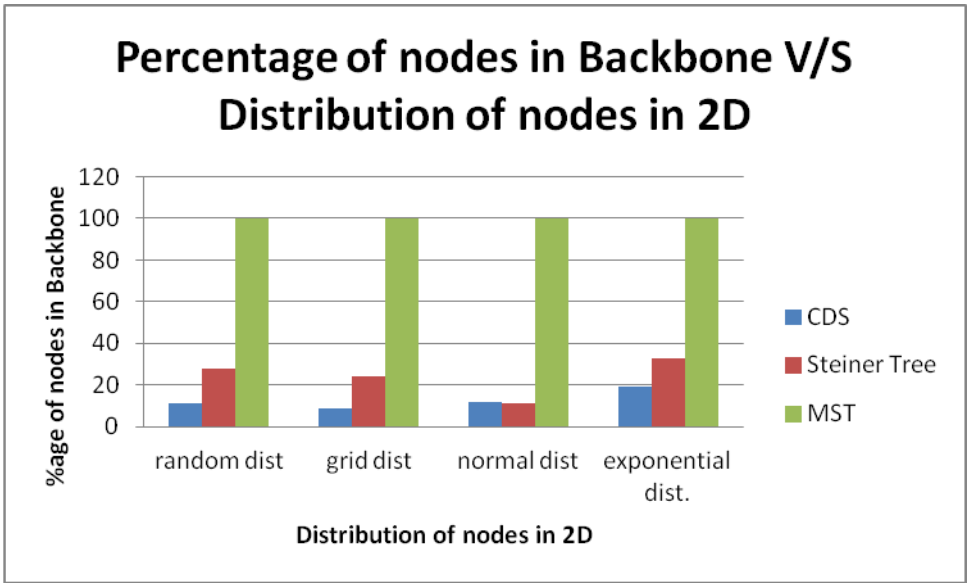

Figure No. 4.3: Percentage of Nodes in Backbone V/S Distribution of Nodes in 2D

As it is clear from the figure no. 4.3 that CDS backbone consisted for lesser no. of nodes for all the distribution of nodes than that in steiner tree. Lesser no. of backbone directly imply lesser active nodes which further imply lesser energy consumption. Hence, CDS proved to be better for backbone construction than Steiner Tree for different node distribution. Further, the algorithms for backbone formation as have been described in Chapter – 3 were implemented for variable no. of nodes i.e 25, 50, 75, 100 with transmission range of 50 m in the area of 100X100. The following figure no. 4.3 shows the plot between the percentages of nodes in backbone with the total no. of nodes.

Table No. 4.3 Percentage Nodes in Backbone

N/Percentage of Nodes in Backbone	25	50	75	100
%age of Nodes in CDS Backbone	40	24	12	11
%age of nodes in Steiner Backbone	32	32	29.3	28
%age of nodes in MST Backbone	100	100	100	100

When 25, 50, 75 and 100 nodes were deployed in 100 X 100 area as the no. of nodes increased with constant transmission range the percentage of nodes in CDS backbone showed a decreasing trend. The increased node density can be one of the reasons. As the node density increases the no. of neighbours of each node increase and a single node has a greater reach to rest of the nodes.

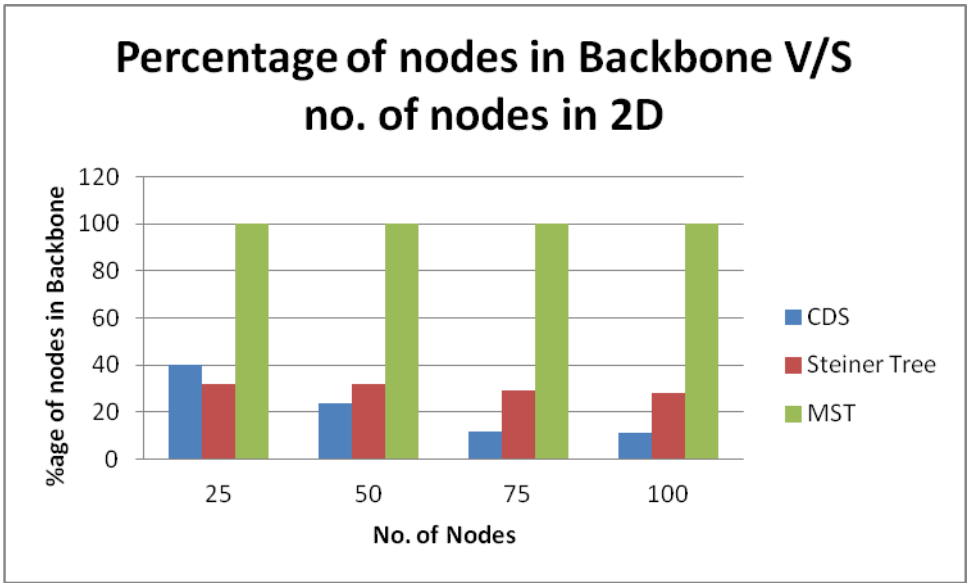

Figure No. 4.4: Percentage of Nodes in Backbone V/S No. of Nodes in 2D

While for 25 nodes 18 nodes comprised of the backbone it reduced to 12 nodes for 50 nodes and further for 75 and 100 nodes too it showed a decreasing trend. For no. of nodes in Steiner tree also the percentage of nodes in backbone slightly reduced as the no. of nodes increased. The reason behind this trend is that as the density of nodes increase the no. of terminal nodes decrease this leads to slight lesser percentage of nodes in the backbone. The following table and graph shows the variation of Steiner Cost and MST

Cost as the no. of nodes increase. The Steiner Cost has been described in Chapter 3 as formula no. 1. And Steiner Cost has also been described in Chapter 3 as formula no. 2.

Table No. 4.4 Cost

N/Cost	25	50	75	100
Steiner Tree Cost	295	370	380	505
MST Cost	650	705	986	1344
Steiner Ratio	2.2	1.9	2.95	2.6

The graph show as figure no. 4.5 the plot of comparison of Steiner Cost and MST Cost with varying no. of nodes. As it us evident the Cost increases as the no. of nodes increase.

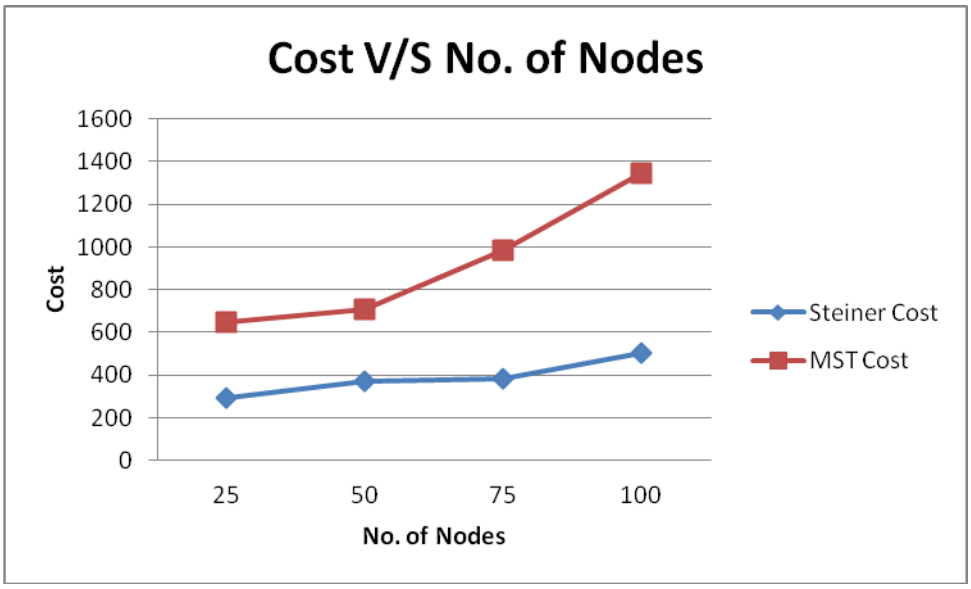

Figure No. 4.5: Cost of Tree V/S No. of Nodes in 2D

Moreover, when we calculated the steiner ratio i.e MST Cost / Steiner Cost it came out to be approximately 2 which verifies it with the original ratio value of the Steiner Algorithm.

4.3. Simulation Results for Backbone construction of nodes over 3D

The algorithms for backbone formation as have been described in Chapter – 3 were implemented for variable distributions as has been shown in figure no. 4.2(a)-(d) with transmission range of 50 m in the area of 100X100X100. The following graph shows the plot between the percentages of nodes in backbone with the total no. of nodes.

Table No. 4.5 Percentage Nodes in Backbone

Distribution/Percentage of Nodes in Backbone	Random distribution	Grid distribution	Normal distribution	Exponential distribution
%age of nodes in CDS Backbone	20	61	29	83
%age of nodes in Steiner Tree Backbone	47	25	39	63
%age of nodes in MST Backbone	100	100	100	100

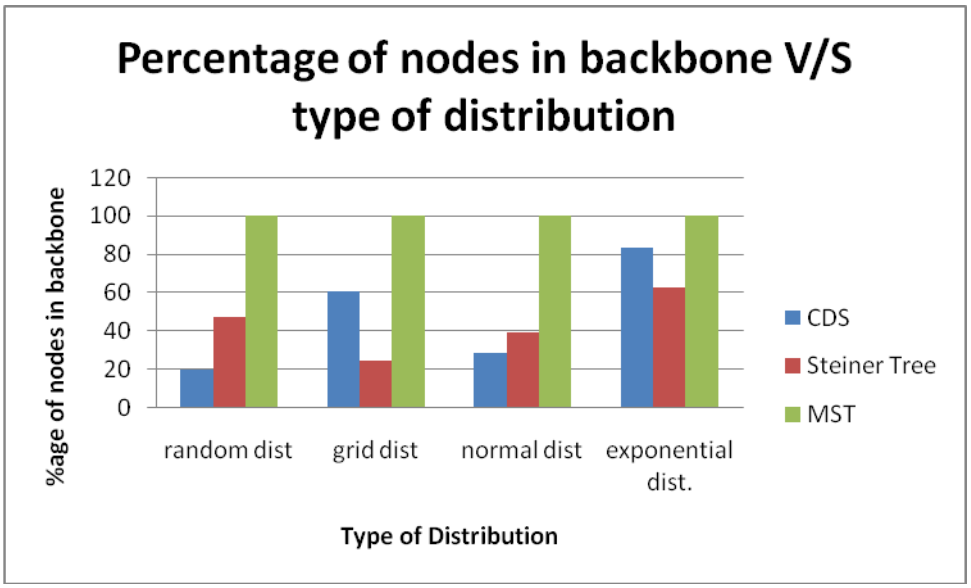

Figure 4.6: Percentage Nodes in Backbone V/S Distribution of nodes in 3D

As it is evident from the bar plot shown in figure no. 4.6 for random distribution and normal distribution CD S Backbone consisted of lesser no. of nodes than in Steiner Tree Backbone. While in Grid distribution and Exponential distribution the scenario was opposite.

Table No. 4.6 Percentage Nodes in Backbone

N/Percentage of Nodes in Backbone	25	50	75	100
%age of Nodes in CDS Backbone	80	76	54.6	20
%age of nodes in Steiner Backbone	64	56	42.6	47
%age of nodes in MST Backbone	100	100	100	100

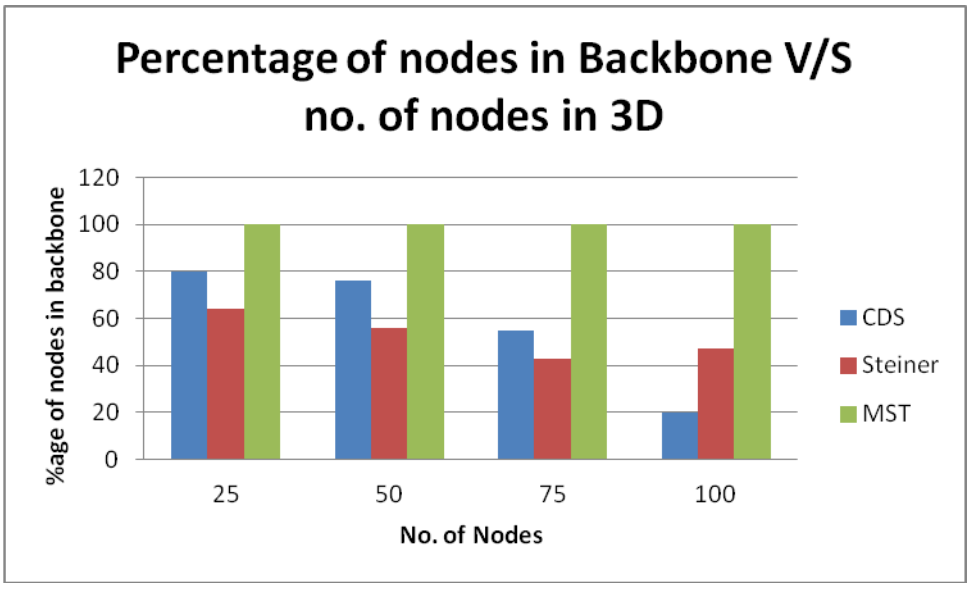

Figure No. 4.7: Percentage of nodes in backbone V/S No. of Nodes

The above graph shown in figure no. 4.7 depicts the comparison of no. of nodes in backbone for different no. of nodes. As it is quite evident from the graph that the percentage of nodes in the CDS backbone decreases as the no. of nodes increases but the transmission range and volume for deployment stays constant. This decreasing trend can be because of the fact that the density of node distribution increases hence, each node now has increases no. of neighbours and lesser no. of nodes are able to suffice the task of backbone formation. In case of Steiner Tree backbone also the trend is decreasing percentage of nodes in backbone as the no. of nodes increased the lesser no. of nodes were present in the backbone.

Further, The Steiner Cost and MST Cost (formula 1 Chapter-3) were plotted against the no. of nodes and we got the following results:

Table No. 4.7 Percentage Nodes in Backbone

N/Cost	25	50	75	100
MST cost	1250	1993	2250	2474
Steiner cost	5.26	890	1005	1459
Steiner Ratio	2.3	2.3	1.69	2.23

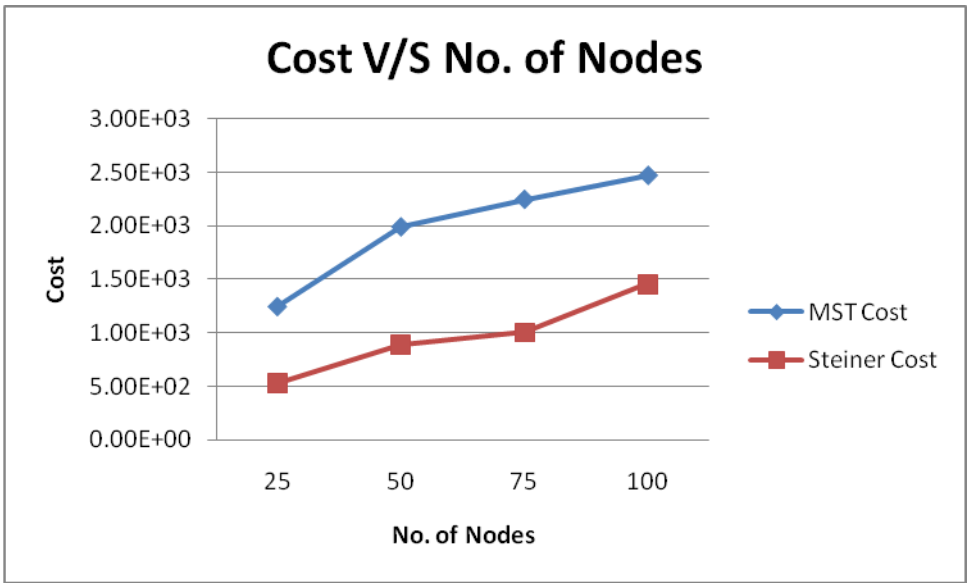

Figure No. 4.8: Cost V/S No. of Nodes

As, it is evident from the graph shown as figure No. 4.8 that as the no. of nodes increase the cost also increases. Also the steiner ratio also was calculated to be approximately 2 as it was supposed to be. We now present the routing algorithm used on our work in the next subsection.

4.4. Simulation Results for Greedy and Voronoi routing over 2D

For the same simulation parameters that have been discussed in the beginning of this chapter as table no. 4.1 we have implemented the routing algorithm for 2D WSN scenario. We took a varying no. of nodes, chose the source and destination of packet and hence plotted the no. of hops V/S the no. of nodes.

The no. of hops taken to reach the destination were plotted against the no. of nodes for Greedy routing and Voronoi routing and we got the following results:

Table No. 4.8 No. of Hops

N/No. of Hops	10	15	20	25	50	75	100
Voronoi 2D	2	2	2	5	5	7	8
Greedy Routing	2	2	0	1	2	4	4

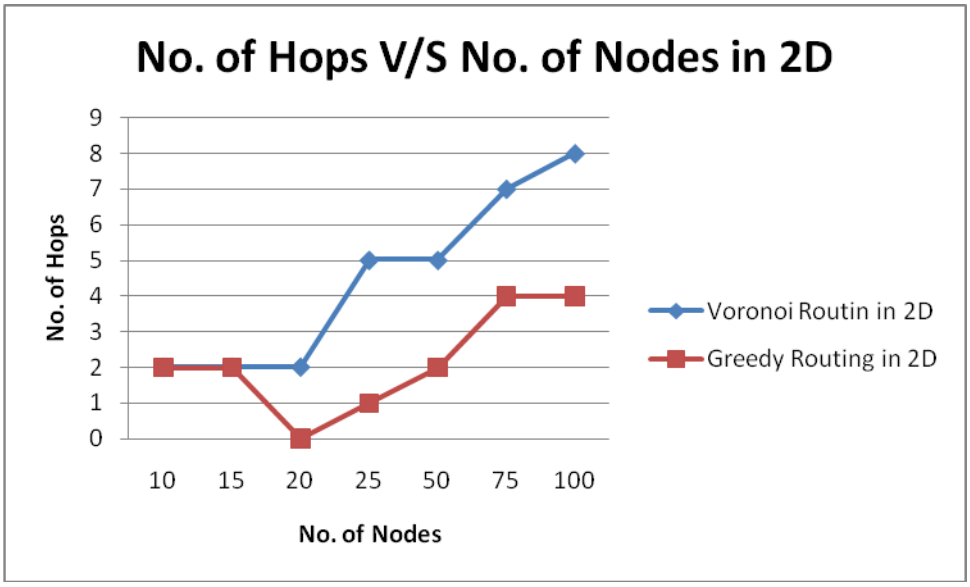

Figure No. 4.9: No. of Hops V/S No. of Nodes in 2D

The above graph in figure no. 4.9 shows the comparison of the two routing algorithms i.e Voronoi and Greedy. As it can be seen that the greedy algorithm reduces the no. of hops it transmits the packet in lesser hops while voronoi routing takes slight more no. of hops. But Greedy Routing doesn't ensure packet delivery and sometimes it can loop infinitely. The above table and graph the zero no. of hops in case of Greedy routing symbolize this looping condition. So, voronoi routing should be preferred if packet delivery is must e.g. in critical system application.

4.5. Simulation Results for Greedy and Vornoi routing over 3D

For the same simulation parameters that have been discussed in the beginning of this chapter as table no. 4.1 we have implemented the routing algorithm for 3D WSN scenario. We took a varying no. of nodes, chose the source and destination of packet and hence plotted the no. of hops V/S the no. of nodes.

The no. of hops taken to reach the destination were plotted against the no. of nodes for Greedy routing and Voronoi routing and we got the following results:

Table No. 4.9 Number of hops

N/No. of Hops	10	15	20	25	50	75	100
Greedy Routing	2	2	4	2	4	4	0
Voronoi Routing	2	2	3	2	6	4	4

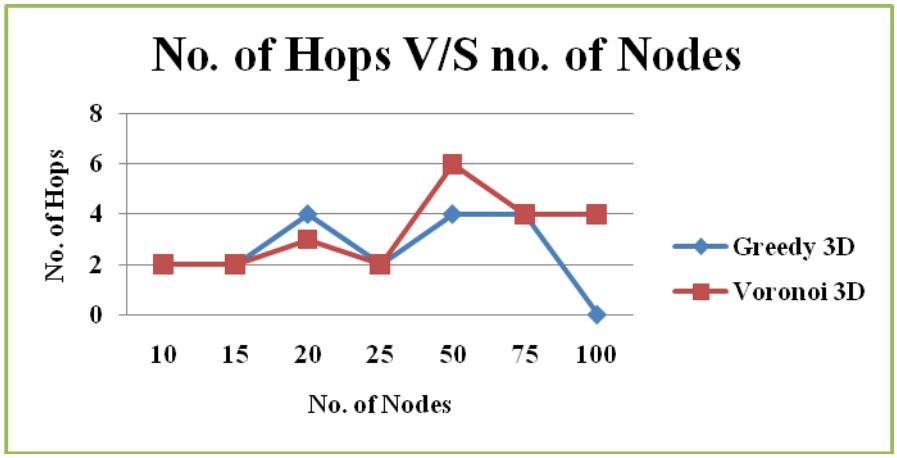

Figure No. 4.10: No. Hops V/S No. of Nodes in 3D

When the routing algorithms were implemented for varying no. of nodes the no. of hops taken by each algorithm came out to be same in most of the cases. Hence, both algorithms are efficient in case of 3D WSN. But, greedy still has the looping problem wherein the packet gets stuck in loop infinitely and is not delivered to the destination.

4.6. Simulation Results for Optimal Sink Placement over 2D

After constructing the backbone one needs to decide on where to locate the sink node. Hence for this purpose we located the sink in three different locations as have been explained in Chapter 3. The three locations were namely geographical centre, node with minimum eccentricity and node with maximum weight i,e the centroid. Ater implementing it on varying no. of nodes i.e 25, 50, 75, 100 over 100X100 area we got the following results.

Table No. 4.10 Energy Consumed

N/Energy Consumed	25	50	75	100
sink gc	0.0238	0.0119	0.0119	0.0119
sink center	0.0118	0.008	0.008	0.0118
sink centroid	0.0118	0.008	0.008	0.0118

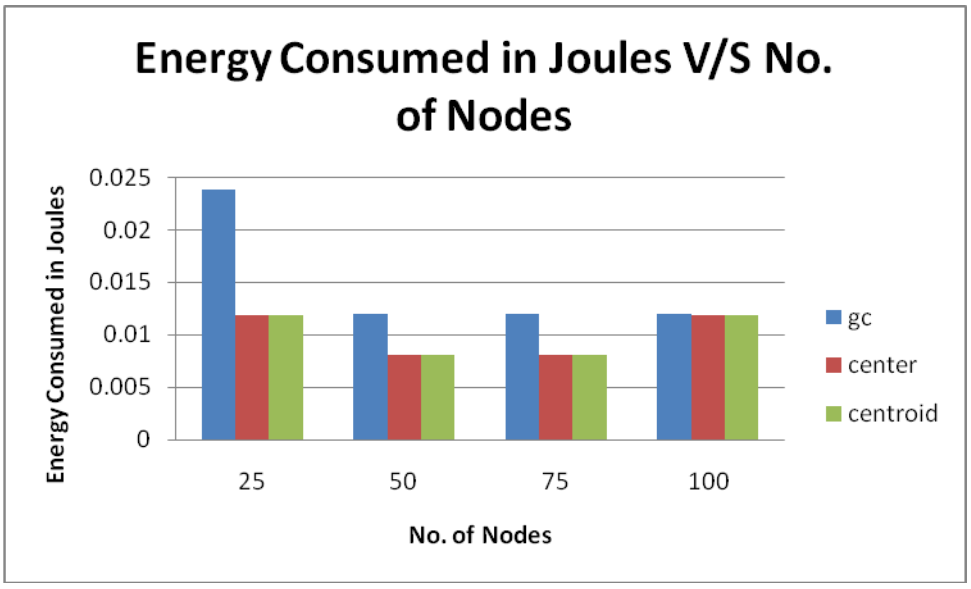

Figure No. 4.11: Energy Consumed V/S No. of Nodes

This shows that when sink is placed on node with minimum eccentricity i.e the center or the centroid lesser amount of energy is consumed rather than when it is placed at the geographical centre. But as the no. of nodes keep on increasing the nodes get uniformly distributed over the space and for 100 nodes the energy consumed is almost same for geographical centre as well.

4.7. Simulation Results for Optimal Sink Placement over 3D

After constructing the backbone one needs to decide on where to locate the sink node. Hence for this purpose we located the sink in three different locations as have been explained in Chapter 3. The three locations were namely geographical centre, node with minimum eccentricity and node with maximum weight i,e the centroid. Ater implementing it on varying no. of nodes i.e 25, 50, 75, 100 over 100X100X100 space we got the following results.

Table No. 4.11 Energy Consumed

N/Energy Consumed	25	50	75	100
sink gc	0.0079	0.0118	0.0118	0.008
sink ecc	0.0077	0.0077	0.0077	0.0079
sink centroid	0.0078	0.0118	0.008	0.0078

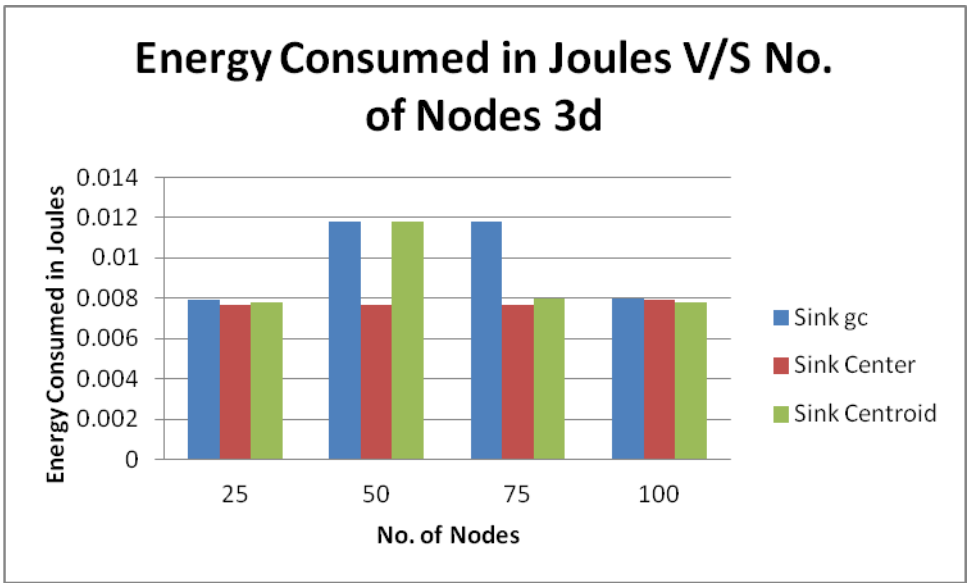

Figure No. 4.12: Energy Consumed V/S No. of Nodes

This shows that when sink is placed on node with minimum eccentricity i.e the center or the centroid lesser amount of energy is consumed rather than when it is placed at the geographical centre. But as the no. of nodes keep on increasing the nodes get uniformly distributed over the space and for 100 nodes the energy consumed is almost same for geographical centre as well.

CHAPTER – 5

CONCLUSION

1.1. Conclusions

As in past few years the wireless sensor networks have attracted great attention because of their varied applications in health care, military, environmental applications. The Wireless Sensor Networks face loads of challenges such as no fixed topology, limited energy source, no physical connection between the nodes. Hence, the formation of backbone in WSN is of great importance as it switches off the redundant nodes, acts as the master for routing activities, increases the effective lifetime of network, increases effective use of bandwidth by reducing the overhead of transmission using flooding (in case of WSN without backbone). Most of the work in WSN have been done in 2D but in real time applications the sensors are deployed in 3D rather than in 2D. Hence, we have proposed our work for 2D as well as for 3D scenario.

In this project we implemented a three algorithms for backbone formation Connected Dominating Set, Steiner Tree and Minimal Spanning Tree for 2D as well as for 3D WSN. We have also compared the efficiency of the proposed algorithms in terms for percentage of nodes in backbone. Hence, found CDS and Steiner Tree backbone came out to better than MST.

After proposing the backbone algorithm we also need to set up a sink node which acts as a collector of information sensed. Hence, Optimal Sink placement was also one of the objectives. We located the sink node at different locations such as geographical centre, center (node with minimum eccentricity), centroid and compared which sink is best located in terms of the energy consumed in delivering the packets to the sink. The sink placed at the center and centroid proved to be best for placing the sink.

The routing algorithms namely Greedy Routing and Voronoi Routing were also implemented for both 2D and 3D WSN. While Greedy Routing reduced the no. of hops the voronoi algorithm ensured packet delivery. Hence, both can be used depending on the application.

1.2. Future Work

The work done in this project can be extended to other algorithms for backbone formation and hence their performance can be analysed.

- Rather than using CDS we can also implement MCDS i.e. Minimum Connected Dominating set. To improve the quality of CDS, we can introduce routing cost constraints and obtain a sequence of results about MCDS under some routing cost constraint.

- The other algorithms which can be implemented Compass Routing or Random Compass Routing where in the algorithm always moves the packet to the vertex that minimizes the angle from neighbour of current node to destination over all vertices adjacent to current node.

- 3D Greedy Routing can be implemented on Delaunay Triangulations which guarantees packet delivery to overcome the biggest drawback of Greedy Routing.

- The recently proposed virtual Delaunay triangulation to aid geographic routing can also be implemented. This is also called multi-hop Delaunay triangulation (MDT). The key idea is to relax the requirement that every node be able to communicate directly with its neighbour in Delaunay triangulation.

Further to extend the research a new Hybrid approach using the above mentioned algorithms can be proposed which uses the best of each algorithm to provide guaranteed delivery, energy efficient routing and backbone formation.

REFERENCES

[1] Bang Ye Wu and Kun-Mao Chao (2004), "Spanning Trees and Optimization Problems,", Chapman & Hall/CRC Press, USA.

[2] Bose P, Morin P. Online routing in triangulations. *Algorithms and Computation*. Springer, 1999; 113–122.

[3] Delaunay Triangulations and Voronoi in Matlab Retrived from http://in.mathworks.com/help/matlab/ref/delaunay.html

[4] E. Kranakis, H. Singh, and J. Urrutia, Compass routing on geometric networks, in Proc. of the 11th Canadian Conference on Computational Geometry (CCCG'99), 1999.

[5] H. M. Ammari (ed.), "The Art of Wireless Sensor Networks, Signals and Communication Technology", DOI: 10.1007/978-3-642-40066-7_10, *Springer-Verlag Berlin Heidelberg*, 2014.

[6] Joseph O' Rourke, "Computational Geometry", Cambridge University Press, 1998. [BERG97].

[7] Mohamed Younis and Rahul Waknis, "Connectivity Restoration in WSN using Steiner Tree Approximations", IEEE Globecom 2010 proceedings.

[8] Razieh Asgarnezhad and Javad Akbari Torkestani, "A New Classification of Backbone Formation Algorithms for Wireless Sensor Networks" proceedings of The Sixth International Conference on Systems and Networks Communication, 2011

[9] Snigdh I et al., "Optimal sink placement in backbone assisted wireless sensor networks", Egyptian Informatics J (2016), http://dx.doi. org/10.1016/j.eij.2015.09.004

[10] Sudipto Guha and Samir Khuller "Approximation Algorithms for Connected Dominating Sets", published in Algorithmica, Pages 374-387, 4, April 1998

[11] S. Durocher, D. Kirkpatrick, L. Narayanan, On routing with guaranteed delivery in three-dimensional ad hoc wireless networks, in Proceedings of the 9th International Conference on Distributed Computing and Networking (ICDCN) (2008)

[12] T. Acharya, S. Chattopadhyay, and R. Roy, "Energy-Aware Virtual Backbone Tree for Efficient Routing in Wireless Sensor Networks," in Proc. of Int. Conf. on Networking and Services, (ICNS '07), IEEE, pp. 96-102, Athens, Greece, June 19, 2007.

[13] Wireless Sensor Networks Deployments (29th Oct, 15) Retrieved from http://theory.utdallas.edu/WebPageLink3.html#NFAR