# GRAPH BASED MODELING APPROACH FOR WIRELESS SENSOR NETWORKS

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

**Computer Science and Engineering**

under the Supervision of

***Mr. Ravindra Bhatt***

By

***Surabhi Agarwal (091294)***

***Pritika Kochar (091300)***

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

# CERTIFICATE

This is to certify that the work titled “ **Graph Based Modeling Approach For Wireless Sensor Networks** ” submitted by “ **Surabhi Agarwal** and **Pritika Kochar** ” in partial fulfillment for the award of the degree of “ **Bachelor of Technology** ” of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

**Date:** **Supervisor’s Name : Mr. Ravindra Bhatt**

**Designation : Sr. Lecturer**

# ACKNOWLEDGMENT

We express our sincere gratitude to our respected project supervisor Sr. Lect. Mr. Ravindra Bhatt, Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat under whose supervision and guidance this work has been carried out. His whole hearted involvement, advice, support and constant encouragement throughout, have been responsible for carrying out this project work with confidence. We are also grateful to him for providing us with required infrastructural facilities that have been highly beneficial to us in undertaking the above mentioned project.

We are sincerely grateful to Brig. S.P. Ghrera, Professor and Head of Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat for providing all necessary facilities for the successful completion of our project.

We would also like to thank the laboratory staff of Department of Computer Science and Engineering for their timely help and assistance.

Date: Name of the students : Surabhi Agarwal (091294)
Pritika Kochar (091300)

# TABLE OF CONTENTS

# List Of Figures and Tables

# ABSTRACT

Wireless Sensor Networks (WSN) combines the wireless communication and the limited computing facility in order to measure the physical phenomenon such as, temperature, pressure, rainfall, smoke, movement etc.

In WSN, there are no fixed or predefined infrastructures. Hence nodes in WSN communicate via sharing the medium, either through 1-hop or multi-hops. Such sharing reduces the network performance due to interference and at the same time it raises energy consumption since packet retransmission is needed. Energy consumption can also be increased if do not limit the number of active nodes used for the multi relays. So backbone can be utilized to address the above problem.

Backbone will remove unnecessary transmission links by shutting down some of the redundant nodes and still guarantee network connectivity in order to deliver the packet efficiently. A backbone reduces the communication overhead, increases the bandwidth efficiency, decreases the overall energy consumption and at last increases network effective lifetime in a WSN.

In this report we have summarized a algorithm based on Connected Dominating Set (CDS) to construct a Backbone. We have also classified and compared two Backbone Construction Techniques. Finally in our project we have implemented a Backbone Construction Technique using C language and then we have shown how any node in a network/backbone gets abandoned because of loss of energy and finally we worked on patching technique which solves a connectivity issue of a backbone.

Dominating sets have proven to be an e®ective construct within which to solve a variety of problems that arise in wireless networks. Applications that use dominating sets include media access coordination, unicast and multicast routing, and energy effciency.
A CDS could perform similar actions through forming a virtual network infrastructure. However, signifcant issues must first be addressed including the real-time assurance of the trustworthiness

of mobile hosts. Similarly, performance of the DS algorithms in the face of Denial of Service and other security attacks must be evaluated in order to ensure robust, survivable network functionality.

# CHAPTER 1

# Introduction

## 1.1 Introduction of Wireless Sensor Network

Advances in wireless networking, micro-fabrication and integration (for example, sensors and actuators manufactured using micro electromechanical system technology, or MEMS), and embedded microprocessors have enabled a new generation of massive-scale sensor networks suitable for a range of commercial and military applications. The technology promises to revolutionize the way we live, work, and interact with the physical environment. In the not-too-distant future, tiny, dirt-cheap sensors may be literally sprayed onto roads, walls, or machines, creating a digital skin that senses a variety of physical phenomena of interest: monitor pedestrian or vehicular traffic in human-aware environments and intelligent transportation grids, report wildlife habitat conditions for environmental conservation, detect forest fires to aid rapid emergency response, and track job flows and supply chains in smart factories. Unlike current information services such as those on the Internet where information can easily get stale or be useless because it is too generic, sensor networks promise to couple end users directly to sensor measurements and provide information that is precisely localized in time and/or space, according to the user’s needs or demands.

With such technological advances come new challenges for information processing in sensor networks. What are needed are novel computational representations, algorithms and protocols, and design methodologies and tools to support distributed signal processing, information storage and management, networking, and application development. While this book will primarily focus on wireless sensor networks, some of the principles, such as those of collaborative information processing and management, apply equally well to wire line sensor networks. The issues of scalability and efficient use of bandwidth are common to both wireless and wire line sensor networks.

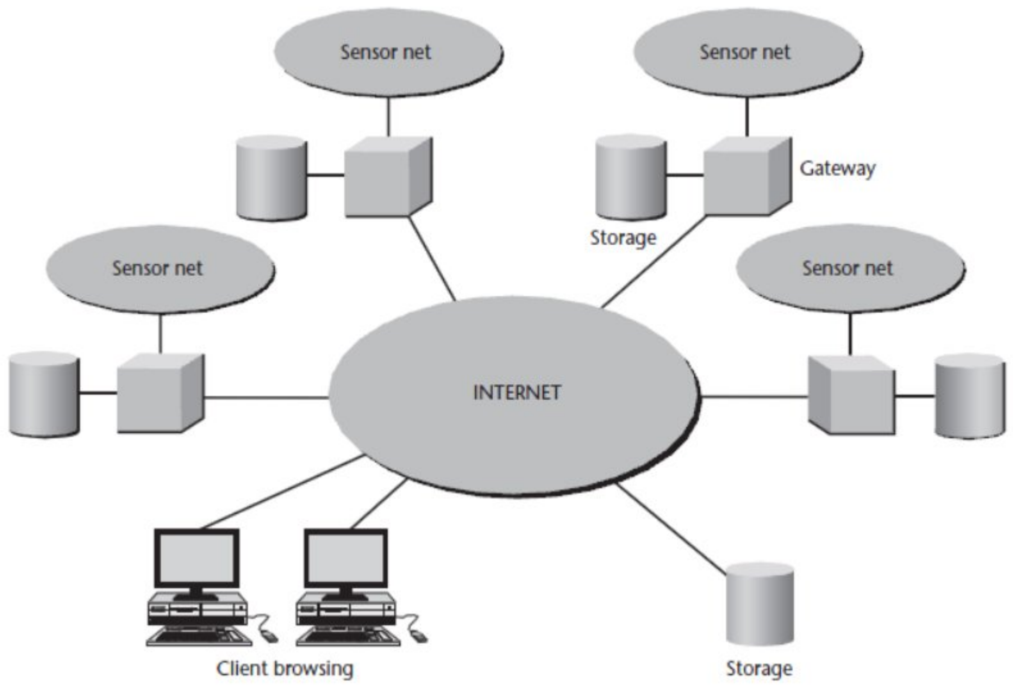


Figure 1.1 Sensor networks significantly expand the existing Internet into physical spaces. The data processing, storage, transport, querying, as well as the internetworking between the TCP/IP and sensor networks present a number of interesting research challenges that must be addressed from a multidisciplinary, cross-layer perspective.

The repository may serve as an intermediary between users and sensors providing a persistent data storage. Additionally, one or more data storage devices may be attached to the IP network, to archive sensor data from a number of edge sensor networks and to support a variety of user-initiated browsing and search functions. Despite the advances in silicon fabrication technologies, wireless communication will continue to dominate the energy consumption of networked embedded systems for the foreseeable future.

Thus, minimizing the amount and range of communication as much as possible for example, through local collaboration among sensors, duplicate data suppression, or invoking only the nodes that are relevant to a given task can significantly prolong the life of a sensor network and leave nodes free to support multiuser operations. In addition, the shorter RF transmission range improves spectrum usage and increases throughput for a sensor network. The information management and networking for this new network will require more than just building faster routers, switchers, and browsers. A sensor network is designed to collect information from a physical environment. Networking will be intimately coupled with the needs of sensing and control, and hence the application semantics.

To optimize for performance and resources such as energy, one has to rethink the existing TCP/IP stack and design an appropriate sensor network abstraction to support application development. For example, in many applications, it is more appropriate to address nodes in a sensor network by physical properties, such as node locations or proximity, than by IP addresses. How and where data is generated by sensors and consumed by users will affect the way data is compressed, routed, and aggregated. Because of the peer-to-peer connectivity and the lack of a global infrastructure support, the sensors have to rely on discovery protocols to construct local models about the network and environment. Mobility and instability in wireless links preclude the use of many existing edge-network gateway protocols for internetworking IP and sensor networks.

To summarize, the challenges we face in designing sensor network systems and applications include:

• Limited hardware:

Each node has limited processing, storage, and communication capabilities, and limited energy supply and bandwidth.

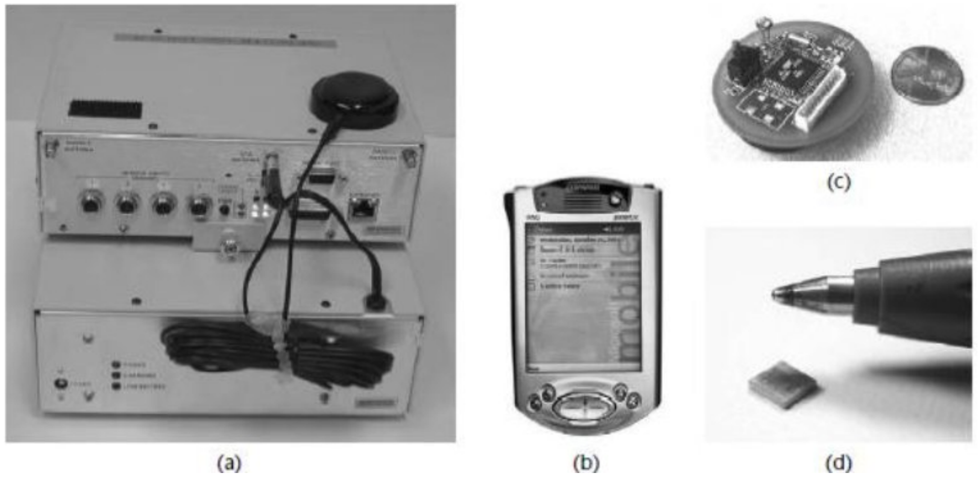
• Limited support for networking:

The network is peer-to-peer, with a mesh topology and dynamic, mobile, and unreliable connectivity. There are no universal routing protocols or central registry services. Each node acts both as a router and as an application host.

• <u>Limited support for software development</u>:

The tasks are typically real-time and massively distributed, involve dynamic collaboration among nodes, and must handle multiple competing events. Global properties can be specified only via local instructions. Because of the coupling between applications and system layers, the software architecture must be code designed with the information processing architecture.

<u>Key Hardware of Sensor Networks</u>:-



<u>Figure 1.2</u> Samples of wireless sensor hardware: (Picture courtesy of Kris Pister and Jason Hill).

1. Sensoria WINS NG 2.0 sensor node.
2. HP iPAQ with 802.11b and microphone.
3. Berkeley/Crossbow sensor mote, alongside a U.S. penny.
4. An early prototype of Smart Dust MEMS integrated sensor, being developed at UC Berkeley.

The current generation of wireless sensor hardware ranges from shoe-box-sized Sensoria WINS NG sensors with an SH-4 microprocessor to matchbox-sized Berkeley motes with an 8-bit microcontroller. A few samples of sensor hardware are shown in Figure 1.2; their corresponding capabilities are summarized and compared in Table 1.1. It is well known that communicating 1 bit over the wireless medium at short ranges consumes far more energy than processing that bit. For the Sensoria sensors and Berkeley motes, the ratio of energy consumption for communication and computation is in the range of 1000 to 10,000.

## 1.2 Key Definitions of Sensor Networks

Sensor networks is an interdisciplinary research area that draws on contributions from signal processing, networking and protocols, databases and information management, distributed algorithms, and embedded systems and architecture. In the following, we define a number of key terms and concepts that will be used throughout the book as we develop techniques and examples for sensor networks.

• *Sensor:*
A transducer that converts a physical phenomenon such as heat, light, sound, or motion into electrical or other signals that may be further manipulated by other apparatus.

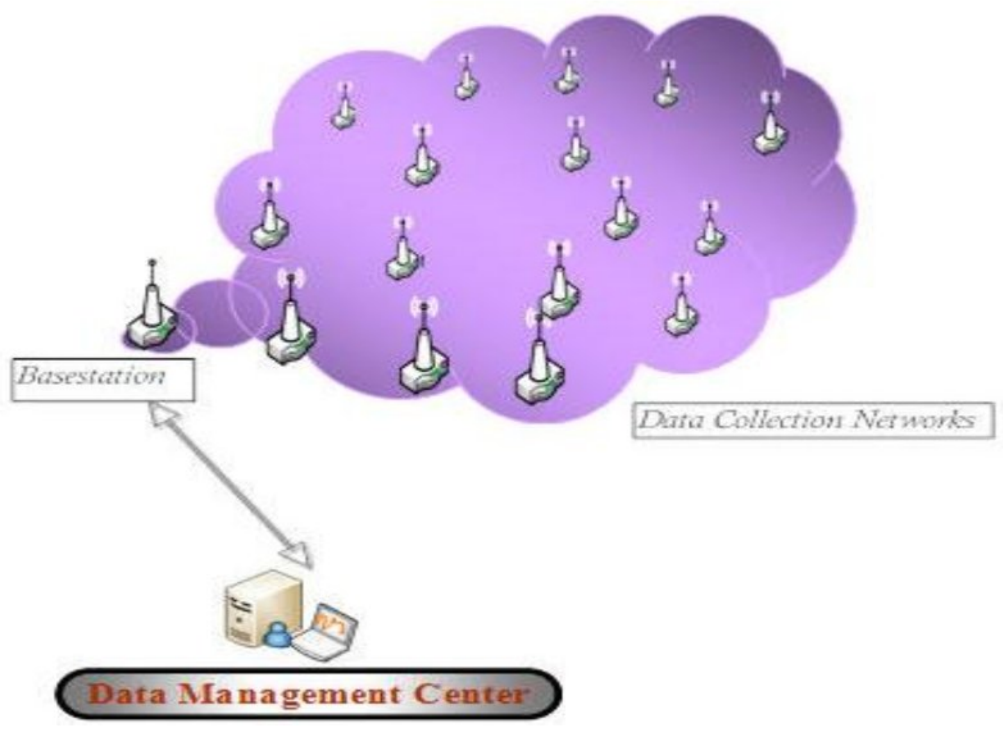
• *Sensor node:*

A basic unit in a sensor network, with on-board sensors, processor, memory, wireless modem, and power supply. It is often abbreviated as node. When a node has only a single sensor on board, the node is sometimes also referred to as a sensor, creating some confusion.

• *Network topology:*

A connectivity graph where nodes are sensor nodes and edges are communication links. In a wireless network, the link represents a one-hop connection, and the neighbors of anode are those within the radio range of the node.

## Architecture of WSN:



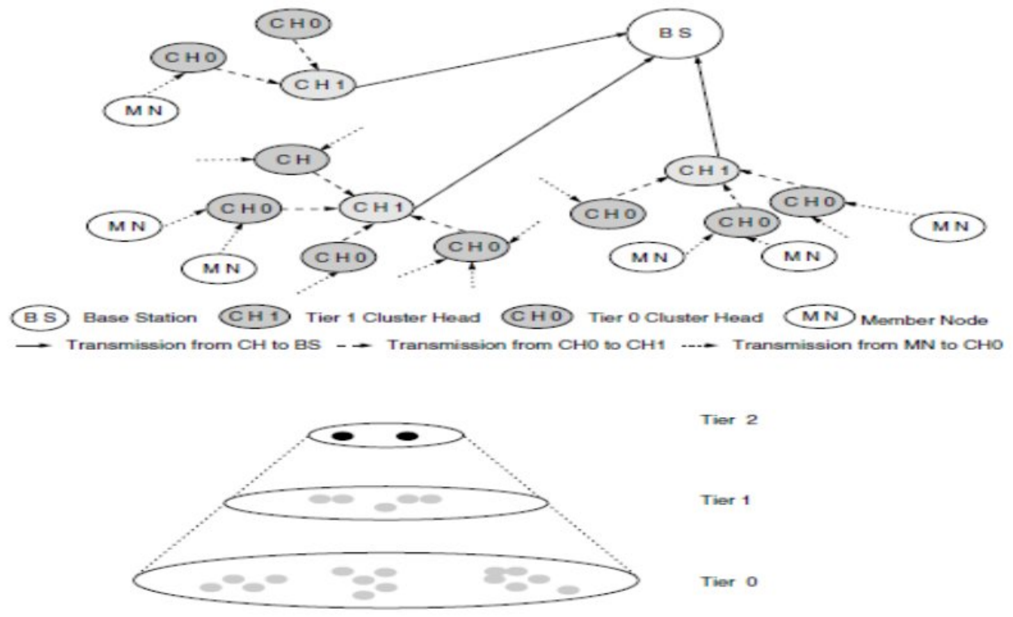


Figure 1.3 Hierarchical multi-hop clustering architecture

## Classification Of WSN : -

TABLE 1.1 Classification of WSNs according to Different Factors:-

| Factors | Distinct Groups |
|---|---|
| Distance to base station/processing centre | Single hop(nonpropogating) vs. Multiple hop(propogating) |
| Data dependency | Non aggregating vs. aggregating |
| Distribution of sensors | Deterministic vs. Dynamic |
| Control scheme | Non-self configurable vs. self-configurable |
| Application domain | Many |

## 1.3 Applications of Wireless Sensor Networks

WSNs are able to monitor a wide range of physical conditions, such as :

- Temperature
- Humidity
- Light
- Pressure
- Object motion

• Soil composition

• Noise level

• Presence of a certain object

•Characteristics of an object such as weight, size, moving speed, direction, and its latest position.

Due to WSNs' reliability, self-organization, flexibility, and ease of deployment, their existing and potential applications vary widely. As well, they can be applied to almost any environment, especially those in which conventional wired sensor systems are impossible or unavailable, such as in inhospitable terrains, battlefields, outer space, or deep oceans.

## Military Applications:

WSNs are becoming an integral part of military command, control, communications, computing, intelligence, surveillance, reconnaissance, and targeting systems. In the battlefield, a predictable tendency is that the targets will become smaller and less recognizable/detectable, have higher mobility, and usually move in extremely hostile terrain. To explore the position and strength of the opposing forces, a promising solution lies in dense arrays of sensors to be placed close to the intended targets. Because of their ability to be unattended by humans, ease of deployment, self-organization and fault tolerance.

## Environment Detection and Monitoring :

Figure 1.4 Adapted from: Xu, N.. A Survey of Sensor Network Applications. *Unpublished* 2001. Computer Science Department, University of Southern California

Spreading hundreds to thousands of tiny, cheap, self-configurable wireless sensors in a given geographical region can produce a wide range of applications in collaborative monitoring or control of the environment.

This encompasses complex ecosystem monitoring; flood detection; air and sewage monitoring; local climate control in large office buildings; soil composition detection and precise agriculture; wild land fire detection; and exploration of mineral reserves, geophysical studies, etc.

Some representative examples include:

- <u>Ecosystem monitoring</u> :

  WSNs used in ecosystem monitoring represent a class of applications with numerous potential benefits for life science study because WSNs can provide information on several environmental conditions, including soil and air chemistry as well as plant and animal species population and behaviours. It ensures the long-term automatic identification, recording, and analysis of interesting events. These long-term gathered data can help ecosystem scientists to identify, localize, track, and predict species or phenomena in areas of interest.
  Compared with traditional methods of environment monitoring, WSNs have a number of unique advantages:

1. *Non-invasive deployment*:
   Unattended wireless sensors can be dropped on remote islands or dangerous places where it would be unsafe, unwise, or even impossible to perform field study repeatedly.

2. *Anytime deployment*:

   Wireless sensor nodes can be deployed in any selected period, for example, before the producing season of some species of animal or after frozen ground melts.

3. *Minimal interference*:

   Deploying WSNs for bio systems can eliminate the disturbance impact on the measured objects. For example, some species are very sensitive to the unexpected visits necessary for large-size macro sensor equipment; this can lead to a dramatic increase of mortality in a breeding year.

4. *Less cost*:

   Deployment of WSNs also leads to a more economical solution to producing long-term observations than human-attended methods do.

5. *Higher level of robustness and accuracy*:

   By integrating data aggregation and signal processing within the neighbourhood sensors, WSNs become more robust to node failure. Self-configurable WSNs used for bio complexity mapping are adaptive to the dynamic physical world.

6. *Ease of networking*:

   Sensor nodes are capable of connecting to the Internet, thus enabling one remote user to control, monitor, and collect data for several different sensed spots or several remote users to gather data for the same spot.

- Local climate control in large buildings:

Most people who have worked in large office buildings have experienced that the temperature is seldom proper, i.e., too high or too low; the humidity level is often overly dry or overly wet; too much or too less light is present; or fresh air is lacking. Therefore, local climate monitoring and control systems are highly desirable to ensure healthy and pleasant working places.

At present, traditional systems with wired sensors are dominant in such areas. Distributed WSNs are considered a better solution than their wired counterparts in at least two respects. For one thing, the deployment of a WSN is much more flexible than a wired system. Without the restriction of wire, wireless sensors can sit wherever they are needed; they can also be moved from their original positions to more suitable places. Moreover, WSNs can produce tremendous economical gains compared to wired sensors.

According to da Silva et al and Rabaey et al, for sensing mission, 90% of the total installation cost of a low-cost temperature sensor is due to wiring. Obviously, installation cost can be greatly reduced if wireless sensors are used.

- Wild land fire detection :

Figure 1.5 Adapted from: Xu, N.. A Survey of Sensor Network Applications. *Unpublished* 2001. Computer Science Department, University of Southern California

Although significant measures have been exerted, wild land fires still cause extensive loss of lives, property, and resources each year. According to the statistics of the National Interagency Fire Center, the 10-year (1992 to 2001) average of wild land fires reached 103,112 and a total of 42,150,890 acres were burned. It costs approximately $1.6 billion (U.S.) on average for fire suppression by federal agencies only. However, because fire weather conditions are predictable, wild land fire prediction is often a possible source of help to support any geographic area before and during periods of high fire danger or fire activity.

Because of their ability to be deployed randomly and densely, WSNs are a good choice in wild land fire detection and reporting. By scattering massive numbers of wireless sensors in intended areas, early warning and origin of fires can be caught effectively.

## Disaster Prevention and Relief

WSNs may also be effectively deployed in emergency situations and disaster areas. The accurate and prompt location detection provided by the distributed WSNs could be critical in rescue operations, including detection of victims, potential hazards, or sources of the emergency and identification and localization of trapped personnel.
For example, micro sensors may be embedded/ enabled in large scale buildings during construction, through strategically dropping on the spot at the rescue site, or by automatically triggering standby sensors immediately following the disaster event. The collapse of the walls or ceiling could be predicated and estimated by the stress and motion of buildings. It is also useful to deploy WSNs for long-lasting monitoring tasks, such as detecting and tracking material fatigue, so that the evidence of harmful reaction of the building can be collected continuously and effective measures can be taken before an accident happens.

Another example, waterproof sensor arrays, can be automatically triggered to constantly report the location of sunken vessels in the ocean and to provide critically important information for the rescue and salvage operation. Furthermore, wireless sensors can also be used to track fuel, gas, and toxic substances leaked into the neighborhood ocean when a sunken vessel is raised.

## Medical Care

WSNs are very helpful in providing prompt and effective health care and will lead to a healthier environment for human beings. Some uses of WSNs in this field include:

- *Remote virus monitoring* :
  Many widespread disease-ridden regions are impoverished and lack reliable communication. Spreading large number of wireless sensors in such regions could help to collect and transmit crucial ground-based information, such as incident of disease and characteristics of the infected population; to identify features of the area; and to monitor environmental conditions, such as the amount of rainfall and humidity, that support the proliferation of virus carrying insects. WSNs can also be used to monitor and predict the breakout of some infectious diseases, such as malaria.

  A project called Health Improvements through Space Technologies and Resources (HI-STAR) proposes development of a global malaria information system. Based on the gathered air and ground-based data via wireless sensors and by integrating and analyzing epidemiological information, this system can generate malaria “risk maps” and provide early warnings about malaria outbreaks. Health officials could also allocate limited disease prevention and treatment resources on a global scale.

## Home Intelligence

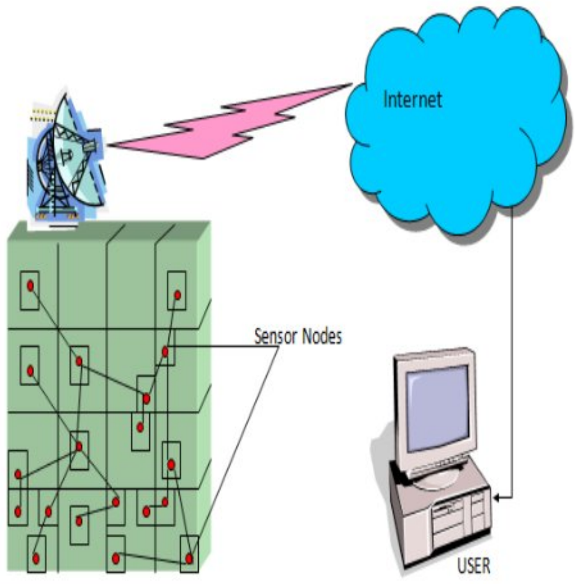


Figure 1.6 Adapted from: Xu, N.. A Survey of Sensor Network Applications. Computer Science Department, University of Southern California

WSNs can take key roles in providing more convenient and intelligent living environments for human beings.
Some predictable examples include:

- *Remote metering*.
  WSNs can be used in remote reading of utility meters, such as water, gas, or electricity, and then can transmit the readings through wireless connections. Simple attachments of wireless sensors in parking meters can send out warning signals to remind users to recharge the meter remotely before the parking time expires.

- *Smart space.*
  With recent technological development, it becomes possible to embed various wireless sensors into individual furniture and appliances, which can be connected together to form an autonomous network. For example, a smart refrigerator can understand the family's dietary requirements or doctor's orders and take inventory of refrigerators to relay information to a shopping list on a personal digital assistant. It can also create a menu according to the inventory and transmit the relevant cooking parameters to the smart stove or microwave oven, which will set the desired temperature and cooking time accordingly. Moreover, contents and schedules of TV, VCR, DVD, or CD players can be monitored and operated remotely to satisfy the different requirements of family members.

## Interactive Surroundings

WSNs produce promising mechanisms for mining information from and reacting to the physical world. By deploying cheap and tiny wireless sensors, monitors and actuators in toys and other children's familiar objects could create "smart kindergartens" to enhance early childhood education. Such a system provides a childhood learning environment with "person–physical world" interaction rather than the conventional "person–computer" or "person–person" communication. Because it allows personalized configuration to each individual child; coordinated activities of children groups; adaptation to the dynamics in children's activities; and constant and unobtrusive data collection in children's actions and learning processes, it provides effective and comprehensive problem-solving strategies in young children's education.

## Surveillance

Instant and remote surveillance inspires significant applications of WSNs. For example, a large number of networked acoustic sensors can be used to detect and track desired targets in a deterministic security area. WSNs can be deployed in buildings, residential areas, airports, railway stations, etc. to identify intruders and report to a command center immediately so that tracking actions can be initiated promptly. Similarly, installing smoke sensor nodes in strategically selected positions at homes, office buildings, or factories is critical to preventing disasters of fires and tracing the spread of fire.

## 1.4 **Thesis Challenges**

WSN design is influenced by one or more of the following technical challenges:-

*1. Massive and random deployment.*

Most WSNs contain a large number of sensor nodes (hundreds to thousands or even more), which might be spread randomly over the intended areas or are dropped densely in inaccessible terrains or hazardous regions. The system must execute self configuration before the normal sensing routine can take off. The dense deployment of sensor nodes leads to high correlation of the data sensed by the nodes in the neighborhood.

*2. Limited resources.*

WSN design and implementation are constrained by four types of resources: energy, computation, memory, and bandwidth. Constrained by the limited physical size, micro sensors could only be attached with bounded battery energy supply.

Moreover, WSNs usually operate in an untethered manner, so their batteries are non rechargeble and/or irreplaceable. At the same time, their memories are limited and can perform only restricted computational functionality.

*3. Ad hoc architecture and unattended operation.*

The attributes of no fixed infrastructure and human unattended operation of such networks require the system to establish connections and maintain connectivity autonomously.

*4. Dynamic topologies and environment.*

On the one hand, the topology and connectivity of WSNs may frequently vary due to the unreliability of the individual wireless micro sensors. For example, a node may fail to function because of exhaustion of power at any time without notification to other nodes in advance. As well, new nodes may be added randomly in an area without prior notification of existing nodes. On the other hand, the environment that the WSNs are monitoring can also change dramatically, which may cause a portion of sensor nodes to malfunction or render the information they gather obsolete. Sensor nodes are linked by the wireless medium, which incurs more errors than their wired counterpart. In some applications, the communication environment is actually noisy and can cause severe signal attenuation.

5. *QoS Concerns.*

The quality provided by WSNs refers to the accuracy with which the data reported match what is actually occurring in their environment. Different from others, accuracy in WSNs emphasizes the characteristic of the aggregated data of all sources instead of individual flows. One way to measure accuracy is the amount of data. Another aspect of QoS is latency. Data collected by WSNs are typically time sensitive, e.g., early warning of fires. It is therefore important to receive the data at the destination/control center in a timely manner. Data with long latency due to processing or communication may be outdated and lead to wrong reactions.

## 1.5 **Thesis Motivation**

As discussed in the section above, it is quite evident what lead us to do research on this thesis. Hence we have also kept some challenges in mind while doing our research on the thesis. Such as:-

1. Quality of Service
2. Energy efficiency
3. Fault Tolerance
4. Limited resources
5. Dynamic topology

Hence keeping all these WSN design challenges in mind we have studied various research papers and have concluded that these challenges can be addressed using a Backbone Construction Techniques.

A backbone will remove unnecessary transmission links by shutting down some of the redundant nodes. In backbone based WSN some nodes will be chosen as dominator nodes (backbone nodes) and the rest as non-dominator nodes. All nodes then can directly or indirectly communicate with other nodes via these nodes and the non-dominator nodes can perform sleeping schedule or turn-off the ratio to save the energy consumption. Hence a backbone reduces the communication overhead, increases the bandwidth efficiency, decreases the overall energy consumption and at last increases network effective lifetime in a WSN. Although there is no physical infrastructure, a virtual backbone can be formed by constructing a Connected Dominating Set (CDS).

## **Problem Statement:**

Through the above explained Motivation for our project, we thus compute the Problem statement that we have taken up for our project, which is:

- Study ,Evaluation and Implementation of the existing backbone construction techniques.
- Study of the graph metric characteristics :

  - Reliability
  - Energy Efficiency
  - Fault Tolerance

## The need of a backbone:

Backbones provide many efficiencies not achievable from a meshed-access network, including:

- Traffic consolidation
- Re-routing and redundancy
- Self-healing architecture
- Sharing of equipment and facilities by multiple locations
- Intelligent routing
- Flexible topologies and styles of design
- Flexibility

## Some Of The Backbone Construction Techniques:

1. **Minimum Spanning Tree (MST)**
2. **Breadth First Search (BFS)**
3. **Depth First Search (DFS)**
4. **Connected Dominating Sets(CDS)**

Different protocols of CDS are combined with one of the Techniques(MST, BFS,DFS) to construct a backbone.

- The algorithm for constructing the CDS should be efficient, distributed, and based on local information only. Since finding a minimum CDS for most graphs is NP-complete, efficient approximation algorithms are used to find a CDS of small size.
- There are many existing algorithms in the literature for broadcasting/routing in ad hoc networks using dominating-set-based approach.
- These algorithms can be evaluated by the efficiency in terms of the number of forwarding nodes, reliability in terms of delivery ratio, and running time for selecting the set of forwarding nodes

# CHAPTER 2
# Connecting Dominating Sets and Backbone Construction Techniques

## 2.1 Graph Theory Definitions on Connected Dominating Set

A number of definitions from graph theory are used in this chapter can help to illustrate the following concepts:

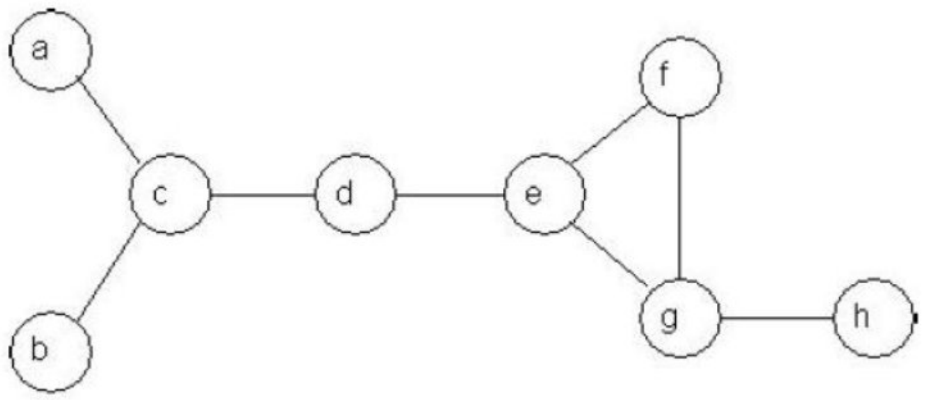


Figure 1: Representation of a wireless network with eight nodes as a graph.

**Open Neighbor Set**, N(u) = {v | (u ,v) ∈ E} is the set of nodes that are neighbors of u.

- In Figure 1, the open neighbor set of e is {d, f, g}.

**Closed Neighbor Set,** N[u] = N(u) U{u}, is the set of neighbors of u and u itself.

- In Figure 1, the closed neighbor set of e is {d, e, f, g}.

**Maximum Degree,** is the maximum count of edges emanating from a single node.

- The maximum degree of the graph in Figure 1 is three, and occurs at nodes c, e, and g.

**Independent Set,** is a subset of V such that no two vertices within the set are adjacent in V.

- For example, {a, b, f, h} is an independent set in Figure 1.

**Maximal Independent Set (MIS),** is an independent set such that adding any vertex not in the set breaks the independence property of the set. Thus, any vertex outside of the maximal independent set must be adjacent to some node in the set. The previous independent set
{a, b, f, h} must have node d added to become an MIS.

- Thus, {a, b, d, f , h} is an MIS.

**Dominating Set,** S, is defined as a subset of V such that each node in V - S is adjacent to at least one node in S. Thus, every MIS is a dominating set. However, since nodes in a dominating set may be adjacent to each other, not every dominating set is an MIS. Finding a minimum-sized dominating set or MDS is NP-Hard.

**Connected Dominating Set (CDS),** is a dominating set of G which induces a connected subgraph of G. One approach to constructing a CDS is to find an MIS, and then add additional vertices as needed to connect the nodes in the MIS.

- A CDS in Figure 1 is {c, d, e, and g}.

**Minimum Connected Dominating Set (MCDS)** is the CDS with minimum cardinality. Given that finding minimum sized dominating set is NP-Hard, it should not be surprising that finding the MCDS is also NP-Hard.

- In Figure 1, {c, g} is a minimum connected dominating set.

**Weakly Connected Dominated Set (WCDS)**, S, is a dominating set such that N[S] induces a connected subgraph of G. In other words, the subgraph weakly induced by S is the graph induced by the vertex set containing S and its neighbors. Given a connected graph G, all of the dominating sets of G are weakly connected. Computing a minimum WCDS is NP-Hard.

**Steiner Tree** is a minimum weight tree connecting a given set of vertices in a weighted graph. After finding an MIS, connecting the nodes together could be formulated as an instance of the Steiner Tree problem. Like many of the other problems that arise in CDS construction, this problem is NP-Hard.

## Basic Terminology Used

- **Maximal Independent Set (MIS),** is an independent set such that adding any vertex not in the set breaks the independence property of the set. Thus, any vertex outside of the maximal independent set must be adjacent to some node in the set.

- **Connected Dominating Set (CDS),** is a dominating set of G which induces a connected sub-graph of G. One approach to constructing a CDS is to find an MIS, and then add additional vertices as needed to connect the nodes in the MIS.

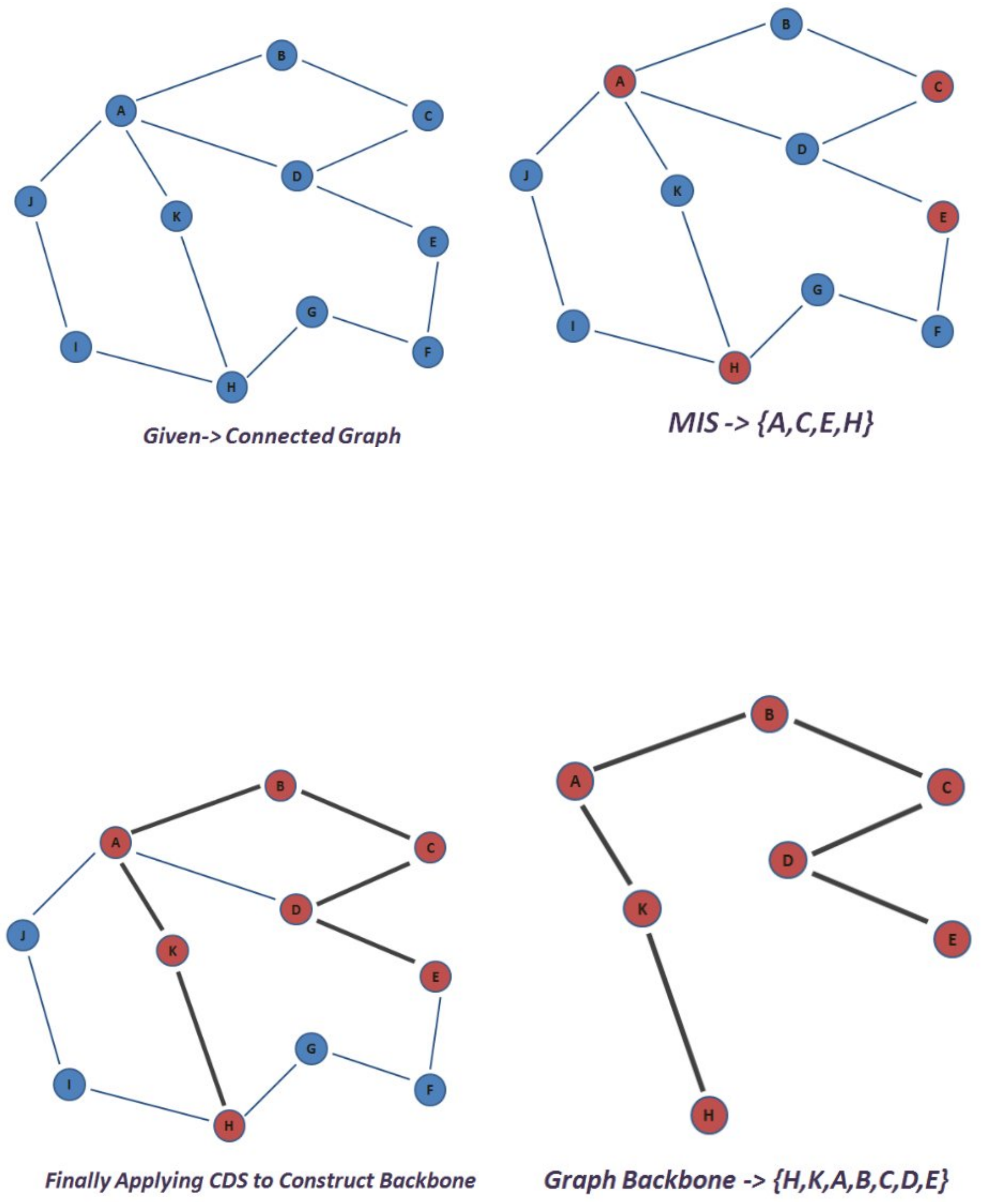

Given-> Connected Graph
MIS -> {A,C,E,H}
Finally Applying CDS to Construct Backbone
Graph Backbone -> {H,K,A,B,C,D,E}
A
B
C
D
E
F
G
H
I
J
K

Figure 2.1 Backbone Construction using CDS

## 2.2 Backbone Construction Techniques in Ad Hoc Networks

### 2.2.1 **Introduction**

Several backbone construction techniques have been proposed in recent years to address the routing problem of the ad hoc wireless sensor networks. Hence we can classify all these backbone construction techniques which are based on connected dominating set (cds) into two main routing classes. The first one is centralized routing protocols and the second one is distributed routing protocol.

### 2.2.2 Routing Approaches Based on CDS:-

#### 2.2.2.1 Guha and Khuller's Algorithm

In 1998, Guha and Khuller proposed two CDS construction strategies in their seminal work, which contains two greedy heuristic algorithms with bounded performance guarantees.

In the first algorithm, the CDS is grown from one node outward.
In the second algorithm, a WCDS is constructed, and then intermediate nodes are selected to create a CDS.

The first algorithm begins by marking all vertices white. Initially, the algorithm selects the node with the maximal number of white neighbors. The selected vertex is marked black and its neighbors are marked gray.
The algorithm then iteratively scans the gray nodes and their white neighbors, and selects the gray node or the pair of nodes (a gray node and one of its white neighbors), whichever has the maximal number of white neighbors. The selected node or the selected pair of nodes are marked black, with their white neighbors marked gray.

Once all of the vertices are marked gray or black, the algorithm terminates. All the black nodes form a connected dominating set.

This algorithm yields a CDS of size at most 2(1 + H(¢)) ¢ jOP Tj, where H is the harmonic function, and OP T refers to an optimal solution { that is, a minimum connected dominating set.

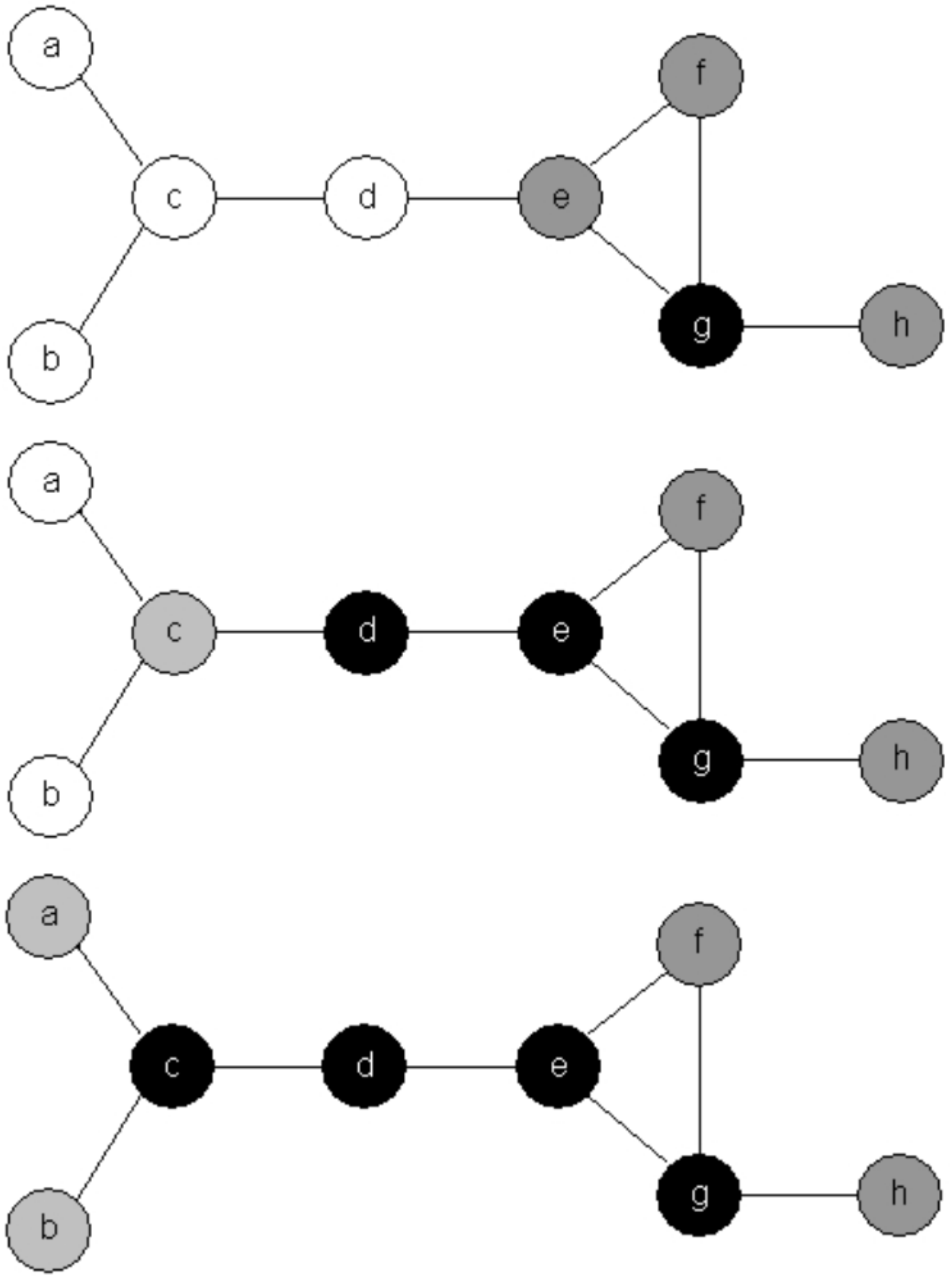


Figure 2.2: An example of Guha's first algorithm.

The second algorithm also begins by coloring all nodes white. A piece is denied to be either a connected black component, or a white node.

The algorithm contains two phases.

The first phase iteratively selects a node that causes the maximum reduction of the number of pieces. In other words, the greedy choice for each step in the first phase is the node that can decrease the maximum number of pieces. Once a node is selected, it is marked black and its white neighbors are marked gray. The first phase terminates when no white node left.

After the first phase, there exists at most jOP Tj number of connected black components.
The second phase constructs a Steiner Tree that connects all the black nodes by coloring chains of two gray nodes black. The size of the resulting CDS formed by all black nodes is at most
(3 + ln(¢)) ¢ jOP Tj.

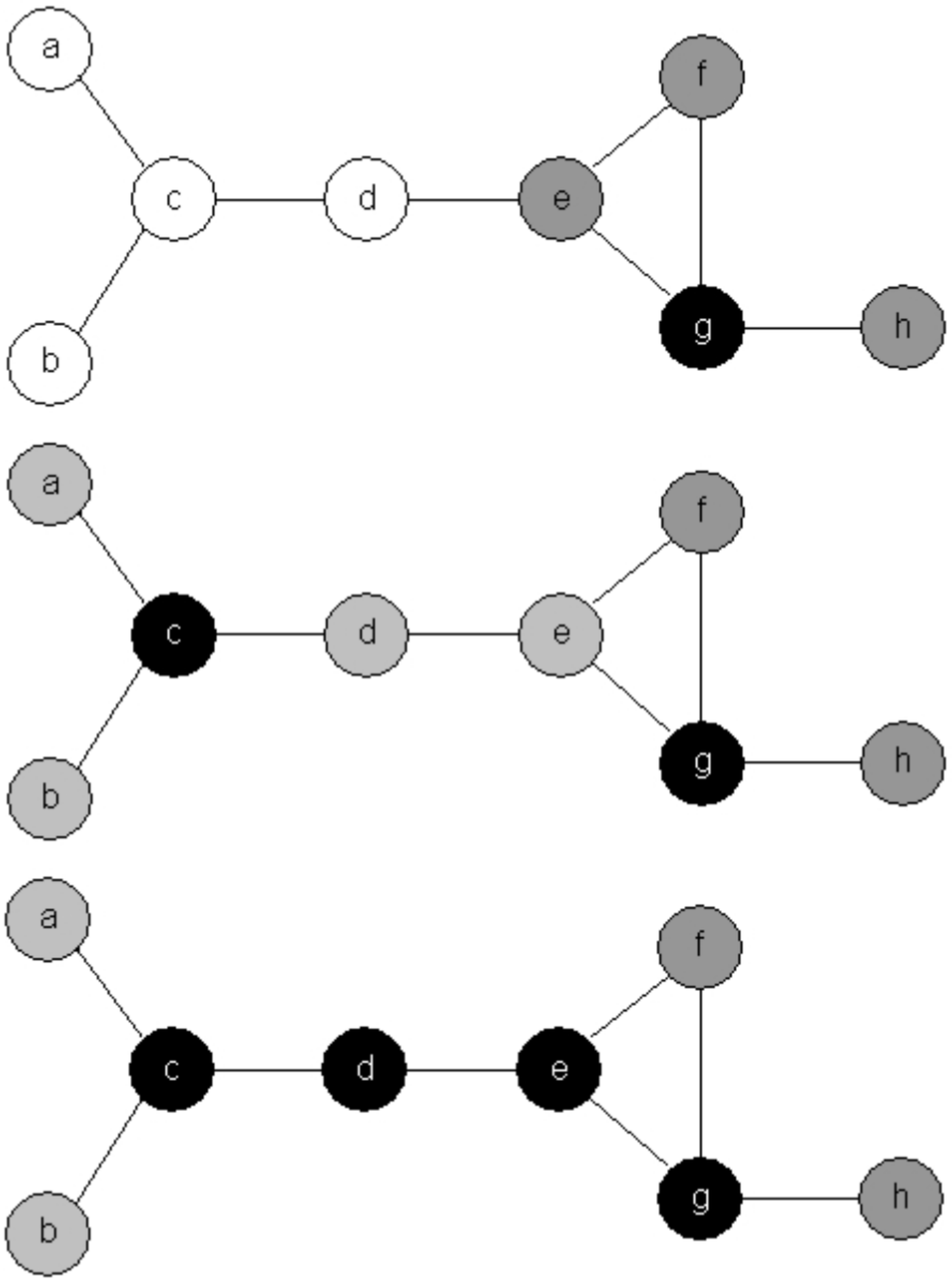
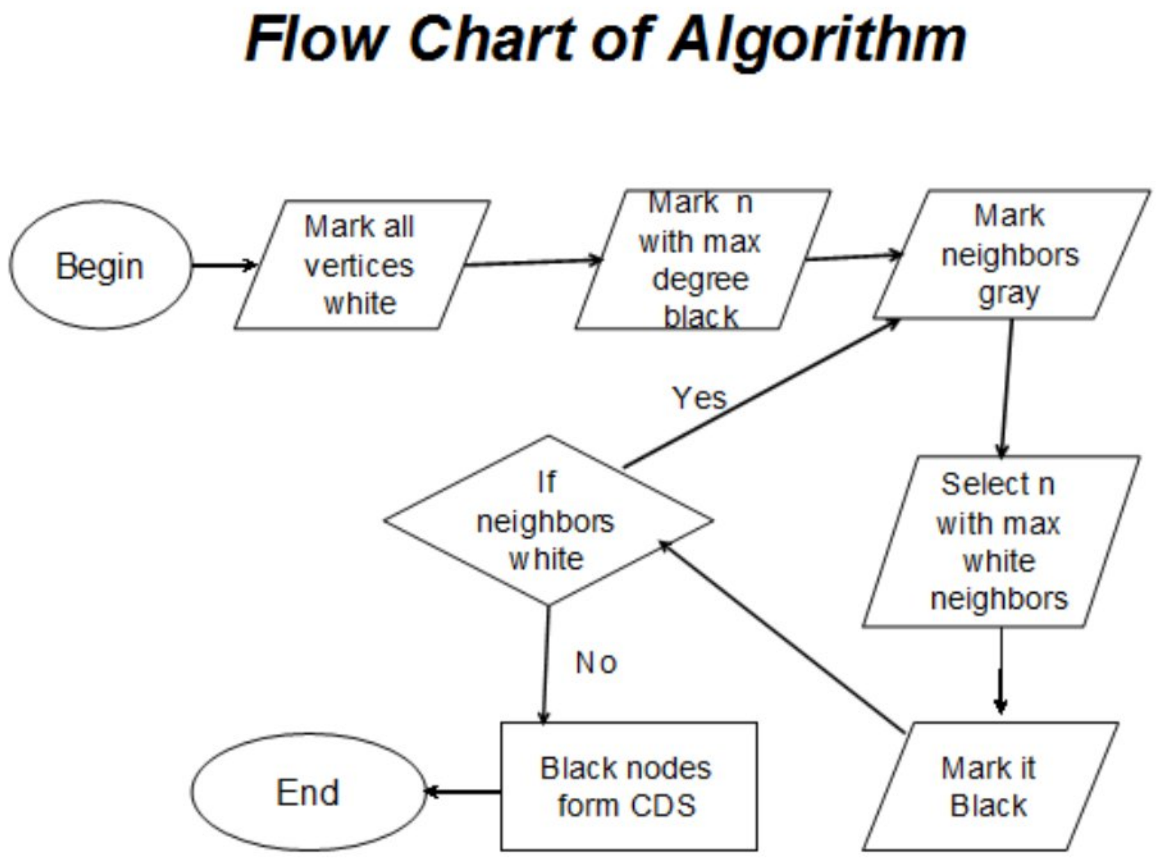


Figure2.3: An example of Guha's second algorithm



### 2.2.2.2 Ruan's Algorithm

The potential function used in the second algorithm of Guha and Khuller is the number of pieces. Each step seeks maximum reduction in the number of pieces in the first phase. By modifying the potential function, Ruan et al. proposes a one-step greedy approximation algorithm with performance ratio at most 3+ln(Δ).

This algorithm also requires each node to be colored white at the beginning. If there exists a white or gray node such that coloring it black and its white neighbours gray would reduce the potential function, then choose the one that causes maximum reduction in the potential function.

### 2.2.2.3 Cheng's Greedy Algorithm

Cheng *et al.* propose a greedy algorithm for MCDS in unit-disk graphs. This algorithm relies on an MIS but the resultant CDS may not contain all the elements in the MIS.
Assume initially all nodes are colored white.

The construction of a CDS contains four phases :

-In the first phase, an MIS is computed and all its members are colored red.

-In the second phase, a node that can decrease the maximum number of pieces is selected, where a piece is either a red node, or a connected black component. This node is colored black and all its non-black neighbors are colored gray. When the second phase is over, we still have some white nodes left.

-The third phase will compute a spanning tree for each connected component in the subgraph reduced by all white nodes. Connect each tree to the nearest black component with black nodes accordingly. All non-leaf tree nodes are colored black while leaf nodes are colored gray.

-The last phase will seek chains of two gray nodes to connect disjoint black components.

The motivation of Cheng's algorithm is two-fold :
First, the greedy choice in Guha and Khuller's second algorithm is the one that can decrease the maximum number of pieces, where a piece is either a connected black component, or a white node.
Second, a unit-disk graph has at most 5 independent neighbors. Thus intuitively one can choose the greedy choice that can connect to as many independent nodes as possible. In other words, the node to be colored black at each step will try to cover more uncovered area, if we model vertices in a unit-disk graph as nodes in a that area. Unfortunately Cheng's algorithm does not have a solid performance analysis.

### 2.2.2.4 Min's Algorithm

Min et al propose to use a Steiner tree with minimum number of Steiner nodes (ST-MSN) to connect a maximal independent set.
This algorithm contains two phase :

-The first phase constructs an MIS with the following property: every subset of the MIS is two hops away from its complement. Color all nodes in the MIS black; color all other nodes gray.

-In the second phase, a gray node that is adjacent to at least three connected black components is colored black in each step. If no node satisfying this condition can be found, a gray node that is adjacent to at least two connected black components will be colored black. This algorithm has performance ratio 6.8 for unit-disk graphs.

### 2.2.2.5 Butenko's Algorithm

The heuristic proposed in is pruning-based. In other words, the connected dominating set S is initialized to the vertex set of graph G(V,E), and each node will be examined to determine whether it should be removed or retained.
Assume all nodes in S are colored white at the beginning. Define the effective degree of a node to be its white neighbours in S. Consider a white node $x \in S$ with minimum effective degree. If removing x from S makes the induced graph of S disconnected, then retain x and color it black. Otherwise, remove x from S. At the same time, if x does not have a black neighbor in S, color its neighbor with maximum effective degree in S black. Repeat this procedure until no white node left in S.
This algorithm has time complexity $O(|V| . |E|)$. It does not have a performance analysis.

## 2.3 <u>Advantages of Connected Dominating Set(CDS)</u>

- The main advantage of connected-dominating-set-based routing is that it centralizes the whole network into small connected dominating set subnetwork, which means only gateway hosts keep routing information, so that as long as network topological changes do not affect this subnetwork there is no need to recalculate routing tables.

- In case of occurrence of any event, data from any source node in the entire network can be transmitted to one of the backbone nodes in a single hop transmission.

- A virtual backbone can be used to support short path routing , fault tolerant routing, multi-casting ,radio broadcasting etc.

- Reduces communication overhead.

- Increases bandwidth efficiency.

- Decreases energy consumption.

# CHAPTER-3
# Energy Efficient Network

## 3.1 Introduction

Wireless Sensor Network (WSN) consists of irreplaceable nodes which are equipped with limited energy resources. Necessity of power consumption becomes a prior importance for various pervasive and ubiquitous applications.

Sensor nodes can use up their limited energy supply carrying out computations and transmitting information in a wireless environment. As such, energy conserving forms of communication and computation are crucial as the node lifetime shows a strong dependence on the battery lifetime. In a multi-hop WSN, nodes play a dual role as data sender and data router. Therefore, malfunctioning of some sensor nodes due to power failure can cause significant topological changes and might require rerouting of packets and reorganization of network. Minimizing energy consumption of WSs is critical yet a challenge for the design of WSNs.

The energy consumption involves three different components: Sensing Unit, Transmission Unit and Reception Unit.

In WSNs the scarcest resource is energy. Sensor network applications requires long lifetime. So an important research issue is to provide a energy efficient network.

Many techniques and algorithms has been developed for the same like:

-Changing the transmission power of the sensor nodes.
-Turning nodes off while preserving connectivity.
-Patching

Wireless sensor networks(WSNs) are used for various pervasive and ubiquitous applications. As WSNs consist of a

large numberofnodesthat are deployed in remote and inaccessible places, there is a need forrealistic power consumptions model that can be used to estimate the network life time, connectivity, and coverage. In this work, a discrete
radio model is used to calculate more realistic power consumption and to determine the connectivity between sensor nodes.

## 3.2 Network Radio Model

A network radio energy model is presented to estimate radio energy consumptions and operation life times in multi-hop wireless ad-hoc sensor radio networks. The model is shown to be a function of radio operating duty cycles and network traffic loads. Numerical results are then given showing examples of radio energy consumptions and operation life times under various traffic conditions and radio operating cycles. It is seen that when employing a power-saving mode such as sleeping during the idling times of a sensor radio, significant energy can be saved so that the sensor operating life can be prolonged.

Using the model, system trades can be carried out by users in system planning and power management for sensor network operations, so as to achieve optimized energy-efficient solution and to provide range of system parameters that are operable under various operating conditions.

A typical sensor node consists of four major components : a data processor unit, a micro-sensor unit, a radio communication subsystem that consisting of transmitter/ receiver electronics, antenna, and amplifier; and a power supply unit. Although energy is dissipated by the first three components of a sensor node, we mainly consider the energy dissipation associated with the radio component.

### 3.2.1 First Order Radio Model

We consider the first order radio model as discussed in with identical parameter values. The energy per bit spent in transmission is given by

**etx(d) = et + ed*dn (1)**

where **et** is the energy dissipated per bit in the transmitter circuitry and **ed*dn** is the energy dissipated for transmission of a single bit over a distance d, n being the path loss exponent (usually 2.0?n?4.0). For a first order model we assume n=2 for simulation purposes.

Thus the total energy dissipated for transmitting a K-bit packet is

**Etx(K,d) = (et + ed*d2 ) * K (2)**

If er be the energy required per bit for successful reception then the energy dissipated for receiving a K-bit packet is

**Erx(K) = er * K (3)**

In our simulations we take et = 50 nJ/bit, ed = 100 pJ/bit/m2 and er = et as mentioned in [5] with K = 2000 bits. It is assumed that the channel is symmetric so that the energy spent in transmitting from node i to j is the same as that of transmitting from node j to i.

## 3.3 Energy Algorithm

1. Begin
2. Calculate Euclidean Distance between the nodes.
3. Assign Initial Total Energy to all the nodes.
4. Select a random path for frame transmission.
5. Calculate Transmission and Reception energy according to the euclidean distance for each node in the path.
6. Reduce the energies from the total energy of nodes.
7. Repeat 5, 6 unless total energy of node becomes less than zero.

Graph representing change in average energy of any randomly generated wireless sensor network with every round of transmission of packet between nodes.

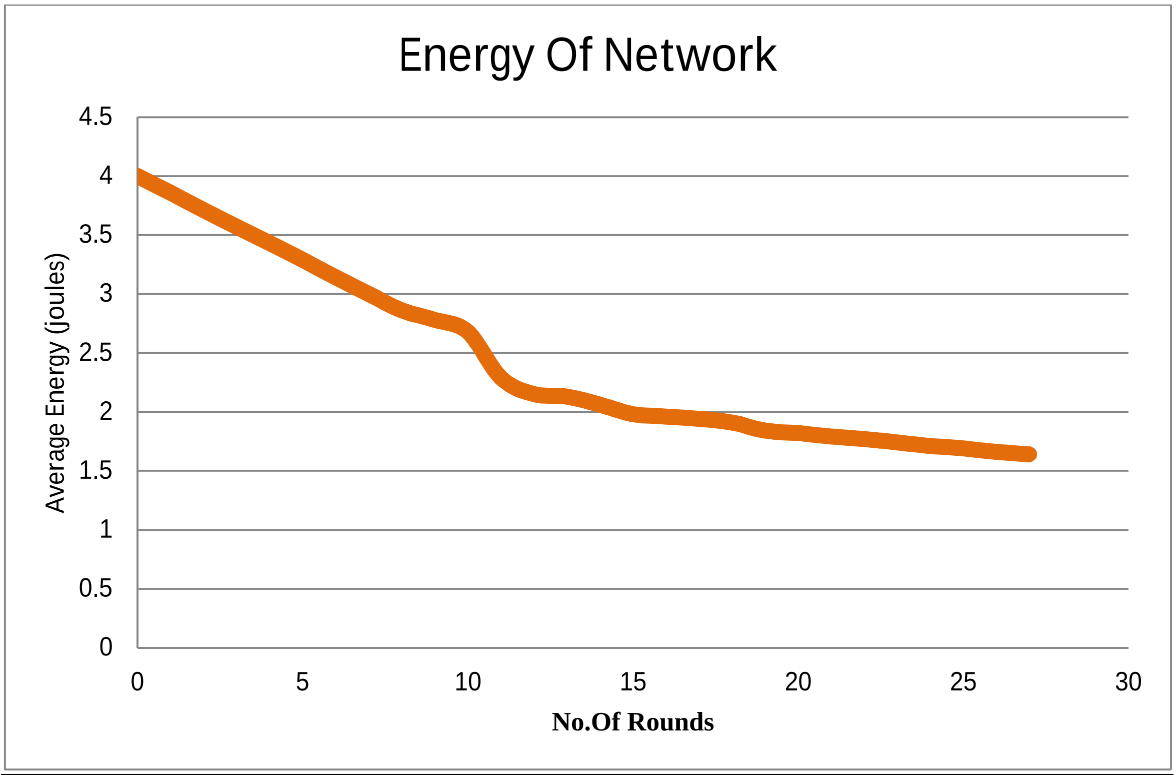


Figure 3.1 : Graph representing average energy of a network

# CHAPTER-4

# Issues Related With Backbone

- Mobile Wireless Sensor Networks (MSWNs), the continuous movement of sensor nodes , may cause complete disconnection of the network or at best a part of it.

- The design of such networks should guarantee that all sensor nodes at all times have a path to the sink node(s).

**Patching:**

When one of the sensor nodes fail, its neighboring nodes take up the failed node's position in the original backbone according to some of the established protocols.

**Back-up:**

Re-establishing the backbone every time one or more nodes abandon the backbone . This is done by removing the failed nodes and establishing a new backbone.

## 4.1 Distributed Patching for Mobile Wireless Sensor Networks(MWSNs)

- It is crucial to keep the whole network and especially the backbones of MWSNs connected at the lowest possible cost.

- Two protocols exist to efficiently patch up the backbones of WSNs and MWSNs and reconnect mobile nodes after reaching their destinations instead of rebuilding the backbones from the scratch every time the backbone is disconnected to optimize the amount of expected energy and time.

### 4.1.1 Efficient Distributed Approximation Algorithm(ECDS)

- Efficient Distributed Approximation Algorithm (ECDS) computes a sub-optimal MCDS in polynomial time for connectivity maintenance of WSNs.

i. It is fully distributed.

ii. The constructed CDS has a small size.

iii. The constructed CDS achieves load balancing.

The following 2 protocols attempt to enhance performance of the ECDS algorithm through patching technique.

### Network Model

- We assume that all nodes of the MWSNs have an equal maximum transmission range of one unit.
- Graph G=(V,E) ,where V is the set of sensor nodes and E is the set of links connecting the sensor nodes.
- Each node n has a weight of being a backbone node.
- M is the set of sensor nodes abandoning their roles due to mobility.
- Ratio of sensor nodes in M to nodes In V should not exceed some percentage R.
- If R<=20%

EEBBPv1 protocol is used.

- If 20%<=R<=35%

  EEBBPv2 protocol is used.

## <u>**Notations**</u>

- Each node of MWSN under consideration at any time is colored with one of the following:

- **<u>BLUE node</u>** :

  MCDS(backbone nodes) node of the MWSN.

- **<u>RED node</u>** :

  Non-MCDS node.

- **<u>GRAY node</u>** :

  Neighbor node to an abandoning node that caused disconnection of the backbone.

Following are some definitions of the terms that are used in the two protocols :

- <u>Node’s weight</u> :

  A combinational metric assigned to each node in the MWSN and/or WSN under consideration. It is a function of both the node’s residual energy and its degree of connectivity to the backbone.

- <u>Blue neighbour list</u> :

  Each node in the MWSN and/or WSN keeps a list of blue neighbor nodes. The list is used by the node as a gateway to send messages to the sink nodes. Moreover, the blue nodes use the nodes as the next hop in the routing.

- <u>Timer TBlue</u> :

A timer initiated by a blue node to search for a list of substitutes for an abandoning nodes. TBlue time period should be long enough to ensure that the initiated blue node has received Update messages from all its neighbours to include them in a weight comparison to choose the best substitute for the abandoning node.

- Timer TGray :

  A timer started by a red node after turning itself gray during the execution of the proposed protocols . The TGray time period is the amount of time given to a gray node to become blue, otherwise ,the node turns itself back to red.

- Region Identifier(RID) :

  A random number generated during the execution of the proposed EEBBPv2 protocol by an abandon node to uniquely identify the disconnected region due to its abandonment.

- RIDs list :

  Each node in MWSN and/or WSN keeps a list of all the region Identifiers for all the abandoning neighbouring backbone nodes that no replacement for them has been found yet.

Following are the messages concerning the two proposed protocols EEBBPv1 and EEBBPv2 :

- Abandon message :

  Broadcasted by a blue node to notify its neighbours that it is abandoning its position in the backbone . When those neighbouring nodes receive the broadcasted message trigger either one of the two proposed protocols.

- Replacement messages :

  Sent by blue nodes in the region to be patched up to enquire the weights of their neighboring gray nodes in order to be able to choose the best replacement for the abandoning backbone nodes. Receiving Replacement message causes the neighbouring gray nodes to send an Update message to the sending blue node.

- Update messages :

Sent by the gray nodes that received Replacement messages in order to report their current weights to the blue nodes sending those Replacement messages . Upon receiving the requested weights each of those blues nodes holds a comparison between all its neighbouring gray nodes weights to choose the best replacement node.

- Acknowledge Request message :

  Broadcasted by a mobile node after reaching its destination to request acknowledgement from its new blue neighbors in order to be able to create its blue neighbour list. Consequently, each blue node in the destination regions sends a Blue message upon receiving the Acknowledge message from the mobile node.

- Blue message :

  Broadcasted either by a blue node as a reply for an Acknowledge Request message ,or by a gray node chosen to be blue in order to maintain an updated blue neighbour list.

Following are the definitions that are only there in first proposed protocol EEBBPv1 :

- Invite message :

  Sent by a blue node to invite a neighboring gray node to be a blue node and a candidate to patch up a disconnected backbone. The neighbouring gray node receiving this invite message turns its colour to blue and be a part of the backbone.

- End message :

  Broadcasted by both the inviting and invited blue nodes to notify other nodes to sop searching for replacement candidates for the abandoning nodes in their vicinities . Each node receiving this End message terminates its searching and turns itself red.

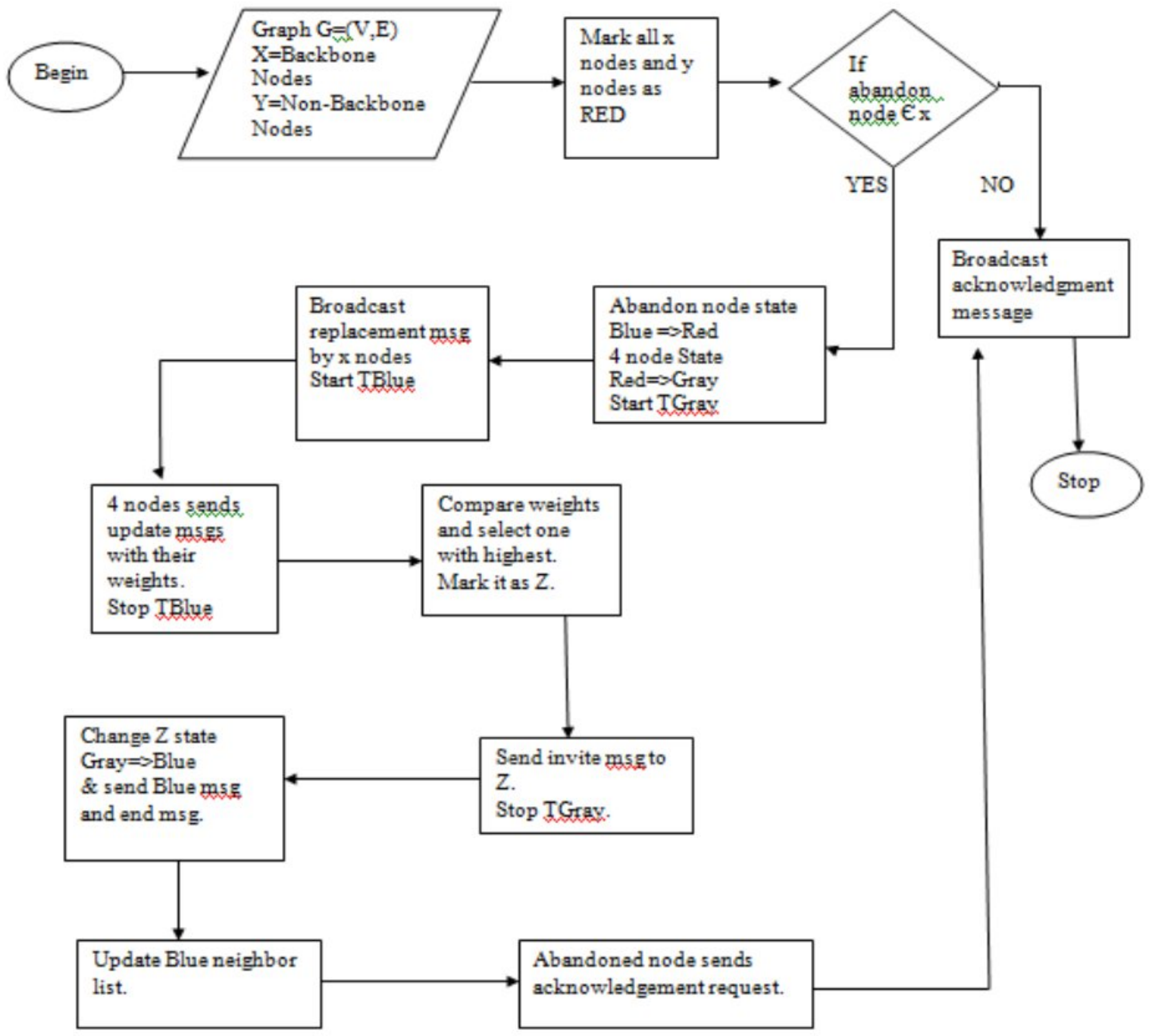
#### 4.1.1.1 Energy Efficient Backbone Patching(EEBBPv1) protocol

AlGORITHM :

```
Node_Abandon()
  { If abandon node state is BLUE Then Send ABANDON MESSAGE }
 Receive Messages()
   { If node state is BLUE Then
      Switch on message content
      Abandon: // sender is a blue abandon node
         Broadcast REPLACEMENT MESSAGE
         Remove sender from the BLUE NEIGHBOR LIST
         Start time TBLUE
      Acknowledge Request: // sender is an abandon node reached its destination
         Reply with a BLUE MESSAGE to the sender.
      Blue: //sender is a blue node
         Add sender to the BLUE NEIGHBOR LIST
    End: //sender is a blue node informing other nodes a replacement is found
      Broadcast END MESSAGE
      Add sender to the BLUE NEIGHBOR LIST
    Update: //sender is gray node sending its new weight
     Compare the reported weights to pick the  node with the highest weight
  End Switch
Else If node state is RED Then
    Switch on message content
      Abandon: //sender is a blue abandon node
          STATE = GRAY
          Remove sender from the BLUE NEIGHBOR LIST
           Start timer TGRAY
 Blue: //sender is a blue node
         Add sender to the BLUE NEIGHBOR LIST
    End Switch
  Else If node state is GRAY Then
       Switch on message content
 Abandon: //sender is a blue node abandon
```

```
        Remove sender from the BLUE NEIGHBOR LIST
 Replacement: //sender is a blue node searching a   replacement for an abandon node
        Send the node weight in an UPDATE MESSAGE
 Blue: //sender is a blue node
       Add sender to the BLUE NEIGHBOR LIST
Invite: //sender is blue node offering an invitation to gray node to be blue
      STATE = BLUE
      Broadcast END MESSAGE
      Halt.
End: //sender is blue node informing other nodes a replacement is found
      STATE = RED
      Add sender to the BLUE NEIGHBOR LIST
      Halt.
End Switch
}

TBlue_timeout()
{  If node state is BLUE Then Send an Invite message to the node with highest weight}

TGray_timeout()
{ If node state is GRAY Then state=RED }

MAIN()
{  If BLUE NODE Abandoned network Then Call Node_Abandon()
   If node received a message Then Call Receive_Messages()

 If Abandon node reached its destination Then Send Acknowledege Request Message
} End Main
```

*Flow Chart - EEBBPv1 Protocol*


Begin
Graph G=(V,E) X=Backbone Nodes Y=Non-Backbone Nodes
Mark all x nodes and y nodes as RED
If abandon node ∈ x
YES
NO
Broadcast acknowledgment message
Stop
Abandon node state Blue =>Red 4 node State Red=>Gray Start TGray
Broadcast replacement msg by x nodes Start TBlue
4 nodes sends update msgs with their weights. Stop TBlue
Compare weights and select one with highest. Mark it as Z.
Send invite msg to Z. Stop TGray.
Change Z state Gray=>Blue & send Blue msg and end msg.
Update Blue neighbor list.
Abandoned node sends acknowledgement request.

## 4.1.1.2 Energy Efficient Backbone Patching(EEBBPv2) protocol

Following are the definitions that are only there in second proposed protocol EEBBPv2 :

- Invite message RID :

Sent by a blue node to invite neighbouring gray nodes to become blue and announce that the disconnected region with the RID has been patched up. Each of the neighbouring nodes receiving an invite message RID turns itself blue if its ID matches the ID included in the invite message otherwise it turns itself red and the EEBBPv2 protocol is terminated for this RID region.

- <u>End message RID</u> :

  Broadcasted by an invited blue node to notify other nodes in its vicinity that a replacement for a certain RID has been found and to stop searching for candidate replacement for the abandoning node with that RID. All the gray nodes receiving the End message turn their colours to red and caused the EEBBPv2 protocol to terminate the search for candidates the abandoning node with that RID.

- <u>Increase weight message RID</u> :

  Broadcasted by a blue node affected by the abandonment of a backbone node with certain RID to its neighbouring gray nodes to increase their weights for this RID. All the gray nodes receiving the increase weight message increase their degrees of connectivity to the backbone and hence their weights increased.

## <u>AlGORITHM</u> :

```
Node_Abandon()
  { If abandon node state is BLUE Then Send ABANDON MESSAGE }
Receive Messages()
  { If node state is BLUE Then
    Switch on message content
    Abandon: // sender is a blue abandon node
      Broadcast an INCREASE-WEIGHT message
      Remove sender from the BLUE NEIGHBOR LIST
      Start time TBLUE
    Acknowledge Request: // sender is an abandon node reached its destination
      Reply with a BLUE MESSAGE
```

```
        **Blue:** //*sender is a blue node*
            Add sender to the BLUE NEIGHBOR LIST
            Remove RID entry from the RIDs LIST if it is already there
         **Update RID:** //*sender is gray node sent its new weight*
            Increase sender node weight in the RID entry in the RIDs LIST
      **End Switch**
**Else If** node state is GRAY **Then**
      **Switch** on message content
        **Abandon RID:** //*sender is a blue abandon node*
              Remove sender from the BLUE NEIGHBOR LIST
               Add abandon RID to the RIDs LIST
        **Replacement RID:** //*sender is a blue node searching a replacement for abandon
                                   node  with the specified RID*
               **For** each RID in the RIDs LIST **Do:**
               Send node weight for this RID in an UPDATE MESSAGE
        **Increase-Weight RID :** //*sender node is blue to increase the degree of connectivity
                                          of some gray nodes*
            Increase weight for the specified RID in the RIDs LIST
  **Blue:** //*sender is a blue node*
            Add sender to the BLUE NEIGHBOR LIST
  **Invite RID :** //*sender is a blue node offering an invitation to gray node to be blue for
                      the specified RID*
            **If** Invite.Id = node RID **Then**
                 State = BLUE
                 Send BLUE MESSAGE
            **Else**
                 Remove this RID form the RIDs LIST
                 **If** RIDs LIST is empty **Then**
                        State = RED
                        Halt
                 **End If**
```

```
                End If
            End RID : //sender is a blue node informing other nodes a replacement is found for the
                        specified RID
              Remove this RID from the RIDs LIST
              If the RIDs LIST is empty Then
                  State = RED
                  Halt
             End If
Else If Node state is RED Then
Switch on message content :
   Abandon RID : //sender is a blue abandon node
         State = GRAY
         Add this RID to the RIDs LIST
         Remove sender from the BLUE NEIGHBOR LIST
         Start timer TGRAY
   Blue : //sender is a blue node
         Add sender to BLUE NEIGHBOR LIST
End Switch
}

TBlue_timeout()
   {  If Node state is BLUE Then
      For each RID in the RIDs LIST Do:
      Send an INVITE RID MESSAGE to the node with the highest weight
      Halt
     End If
   }
TGray_timeout()
   { If Node state is GRAY Then
        State = RED
        Halt.
```

```
    **End If**
    }

**MAIN()**
    {
        **If**  a BLUE NODE Abandoned the MWSN **Then Call** *Node_Abandon()*
        **If** a node received message RID **Then Call** *Receive_Messages RID()*
        **If** Abandon node reached its destination **Then** *Send Acknowledege Request Message*
    }
  **End Main**
```

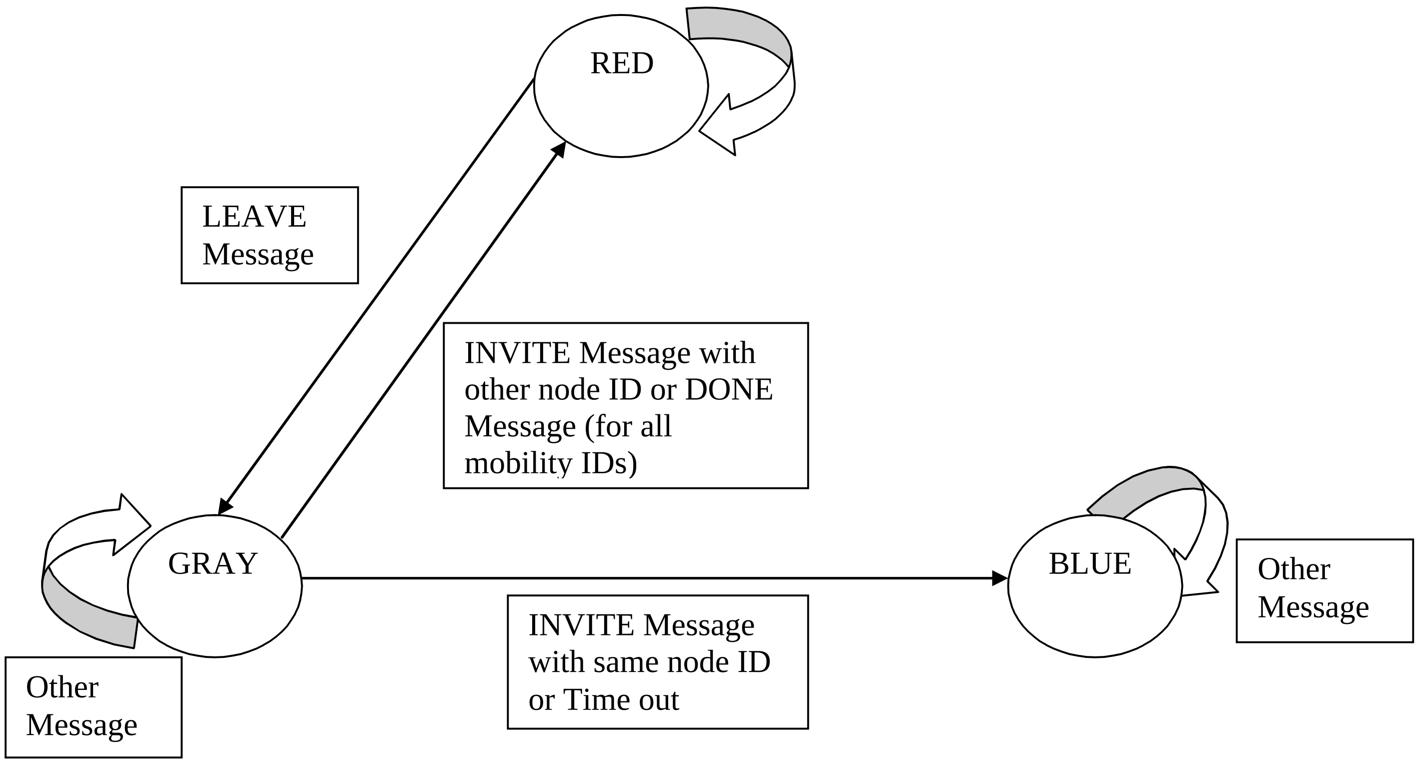
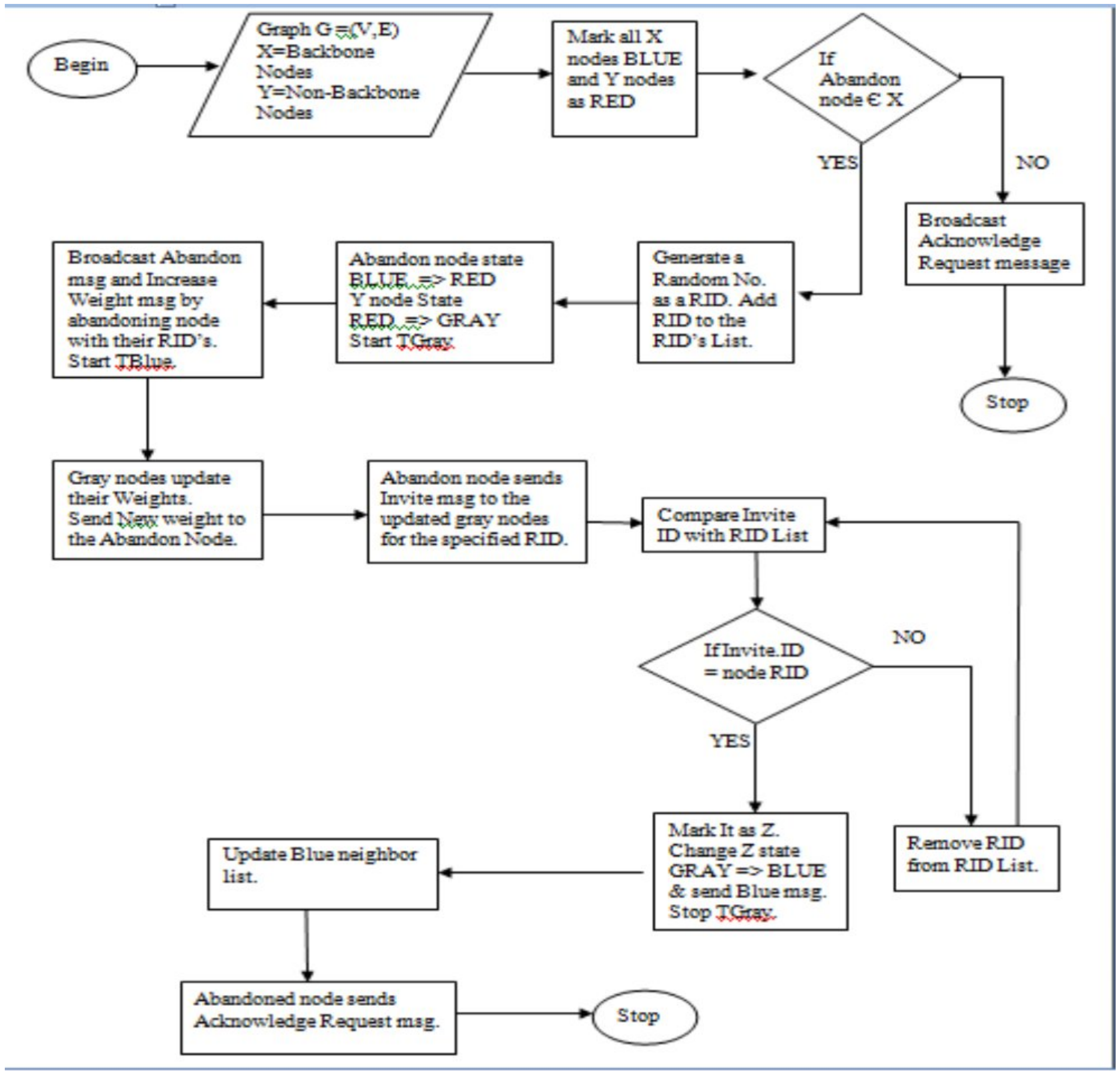

RED
LEAVE Message
INVITE Message with other node ID or DONE Message (for all mobility IDs)
GRAY
BLUE
Other Message
INVITE Message with same node ID or Time out
Other Message

Figure 4.1 State transitions during the execution of the EEBBPv2 Protocol

*Flow Chart- EEBBPv2 Protocol*

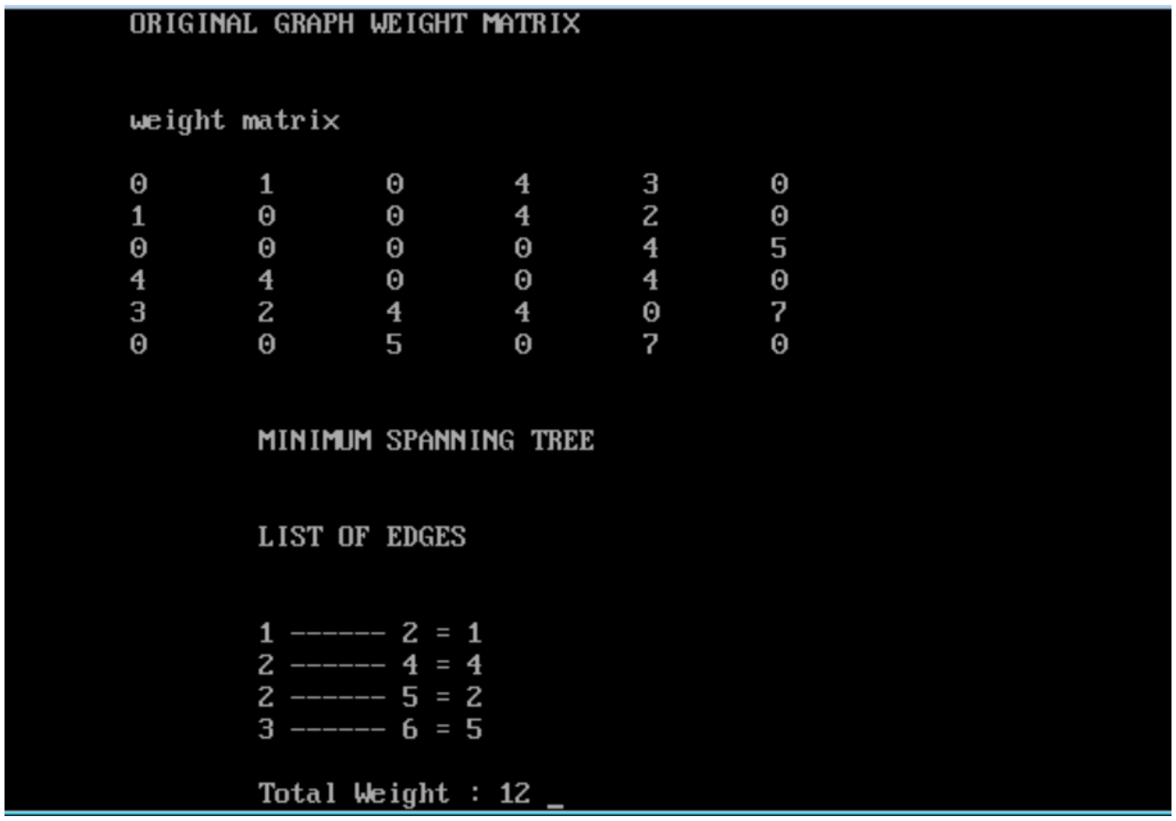

Begin
Graph G =(V,E) X=Backbone Nodes Y=Non-Backbone Nodes
Mark all X nodes BLUE and Y nodes as RED
If Abandon node € X
YES
NO
Broadcast Acknowledge Request message
Stop
Generate a Random No. as a RID. Add RID to the RID's List.
Abandon node state BLUE => RED Y node State RED => GRAY Start TGray.
Broadcast Abandon msg and Increase Weight msg by abandoning node with their RID's. Start TBlue.
Gray nodes update their Weights. Send New weight to the Abandon Node.
Abandon node sends Invite msg to the updated gray nodes for the specified RID.
Compare Invite ID with RID List
If Invite.ID = node RID
NO
YES
Mark It as Z. Change Z state GRAY => BLUE & send Blue msg. Stop TGray.
Remove RID from RID List.
Update Blue neighbor list.
Abandoned node sends Acknowledge Request msg.
Stop

# Chapter-5

# Simulations And Results

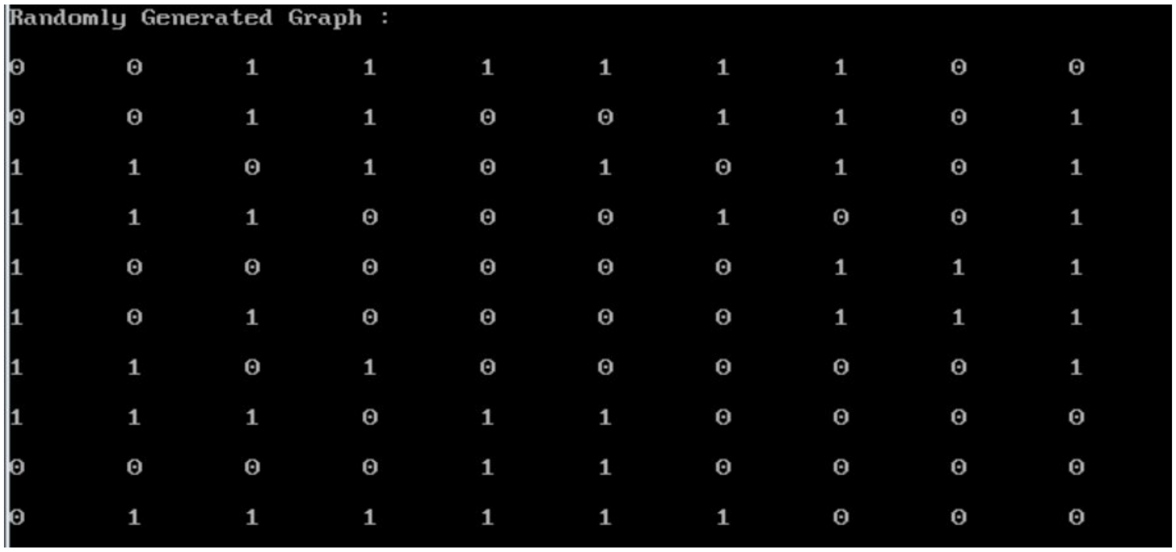
## 5.1 Performance and Comparison

Minimum Spanning Tree  c code output

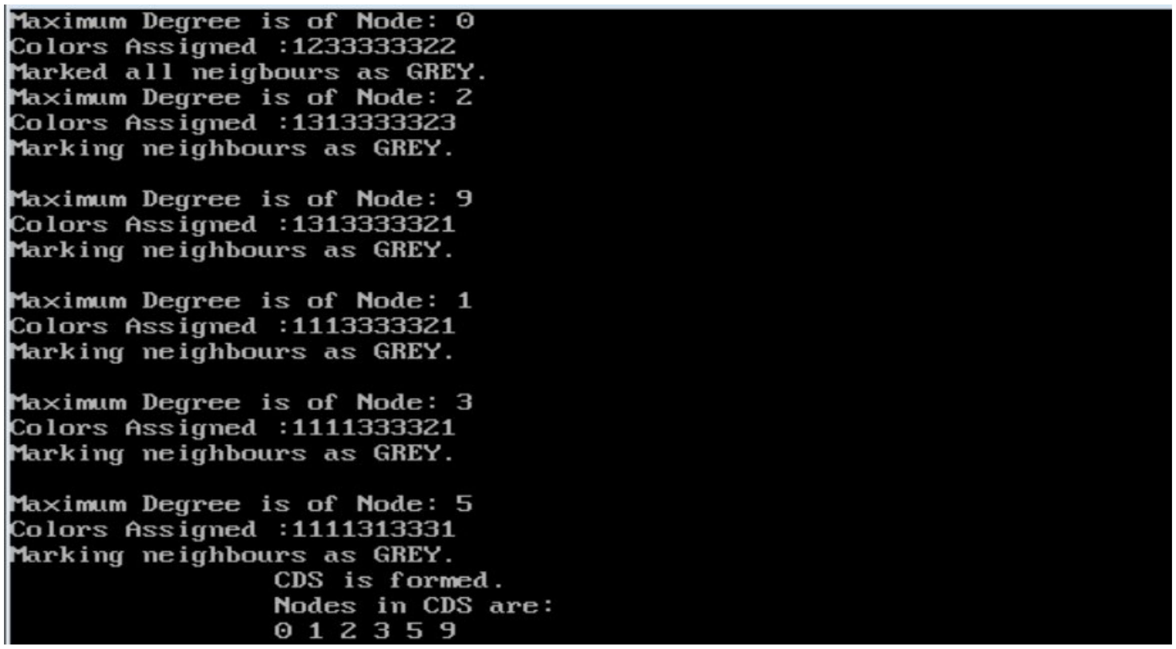


```
ORIGINAL GRAPH WEIGHT MATRIX


weight matrix

0       1       0       4       3       0
1       0       0       4       2       0
0       0       0       0       4       5
4       4       0       0       4       0
3       2       4       4       0       7
0       0       5       0       7       0


        MINIMUM SPANNING TREE


        LIST OF EDGES


        1 ------ 2 = 1
        2 ------ 4 = 4
        2 ------ 5 = 2
        3 ------ 6 = 5

        Total Weight : 12 _
```

## C code for Backbone of randomly generated graph

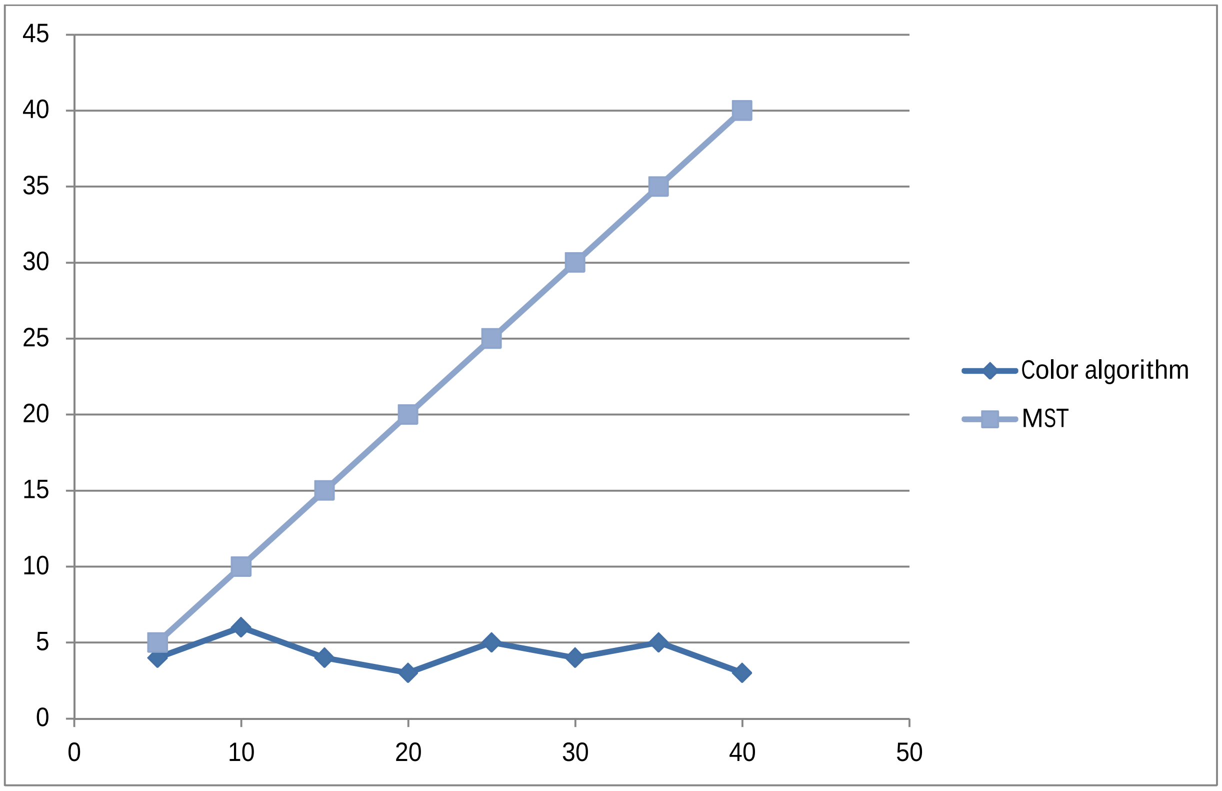


```
Randomly Generated Graph :

0       0       1       1       1       1       1       1       0       0

0       0       1       1       0       0       1       1       0       1

1       1       0       1       0       1       0       1       0       1

1       1       1       0       0       0       1       0       0       1

1       0       0       0       0       0       0       1       1       1

1       0       1       0       0       0       0       1       1       1

1       1       0       1       0       0       0       0       0       1

1       1       1       0       1       1       0       0       0       0

0       0       0       0       1       1       0       0       0       0

0       1       1       1       1       1       1       0       0       0
```



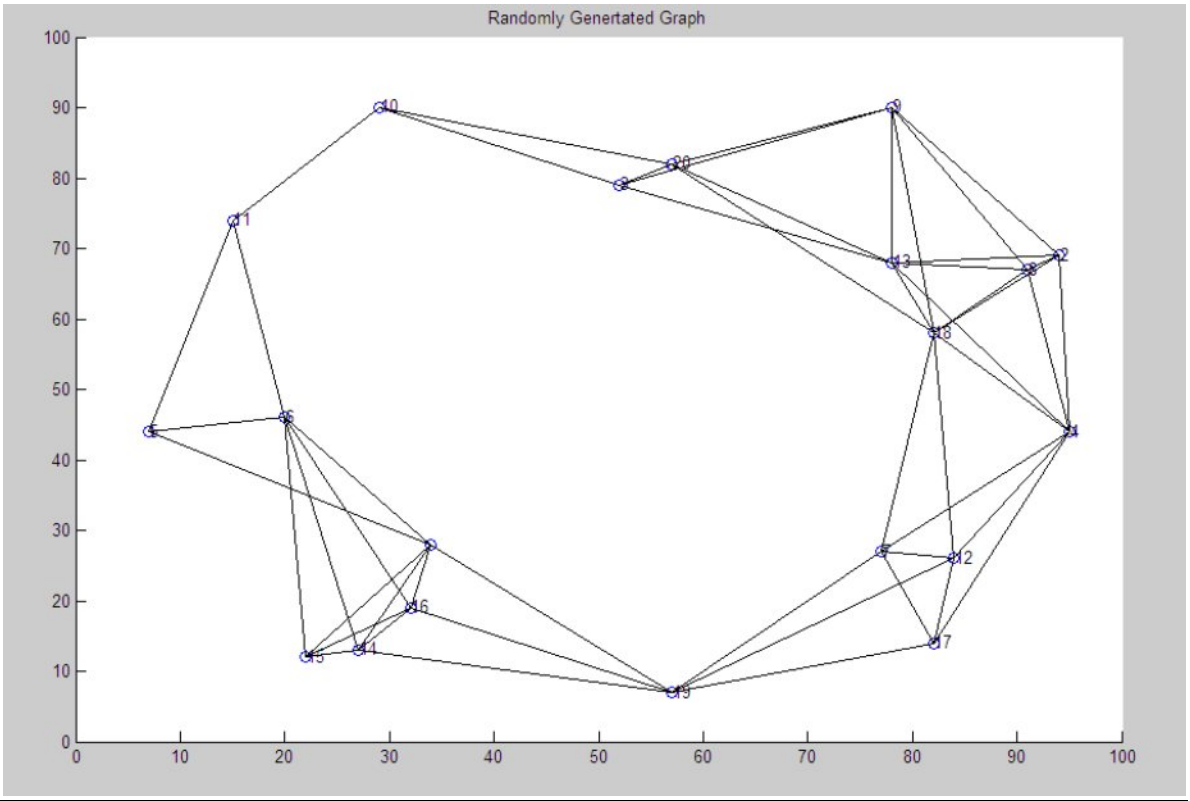
```
Maximum Degree is of Node: 0
Colors Assigned :1233333322
Marked all neigbours as GREY.
Maximum Degree is of Node: 2
Colors Assigned :1313333323
Marking neighbours as GREY.

Maximum Degree is of Node: 9
Colors Assigned :1313333321
Marking neighbours as GREY.

Maximum Degree is of Node: 1
Colors Assigned :1113333321
Marking neighbours as GREY.

Maximum Degree is of Node: 3
Colors Assigned :1111333321
Marking neighbours as GREY.

Maximum Degree is of Node: 5
Colors Assigned :1111313331
Marking neighbours as GREY.
                CDS is formed.
                Nodes in CDS are:
                0 1 2 3 5 9
```

## Comparison on the basis of nodes used for backbone formation.



In this graph, we have stimulated two techniques of Backbone Construction. We have varied the number of nodes in the network and have got the above graph.

## 5.2 MATLAB representation of Randomly Generated Graph

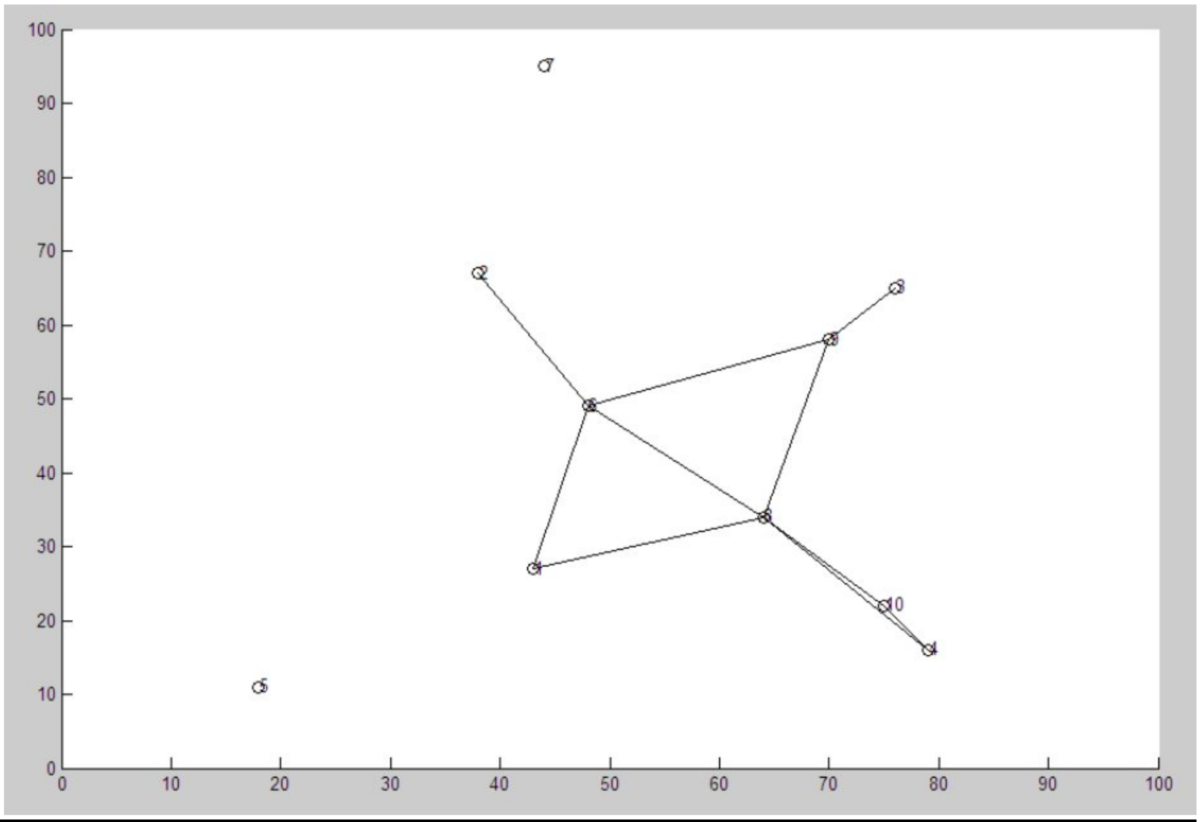


## 5.3 <u>Connectivity of nodes with different communication range</u>

We deploy the different number of nodes in a 100*100 region that are connected on the basis of their communication range. We also used different communication range for different number of nodes. We deploy 10 nodes in 100*100 region with communication range=25, then we gave communication range = 35 for the same nodes in the same region and then we gave communication range= 45 for the same. We plot oll these graphs and have shown them in the figures.

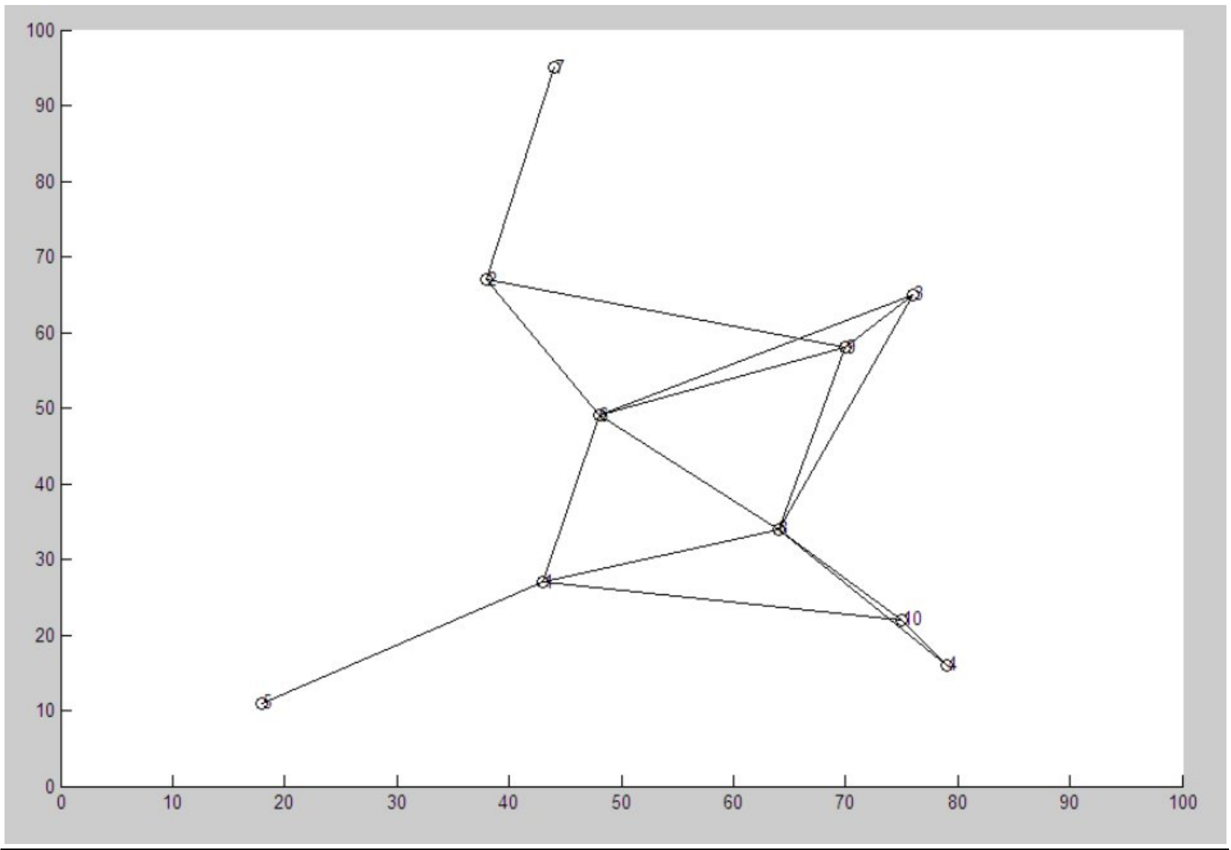
**Deploy 10 nodes in 100*100 region with different communication range.**

**Communication range = 25**



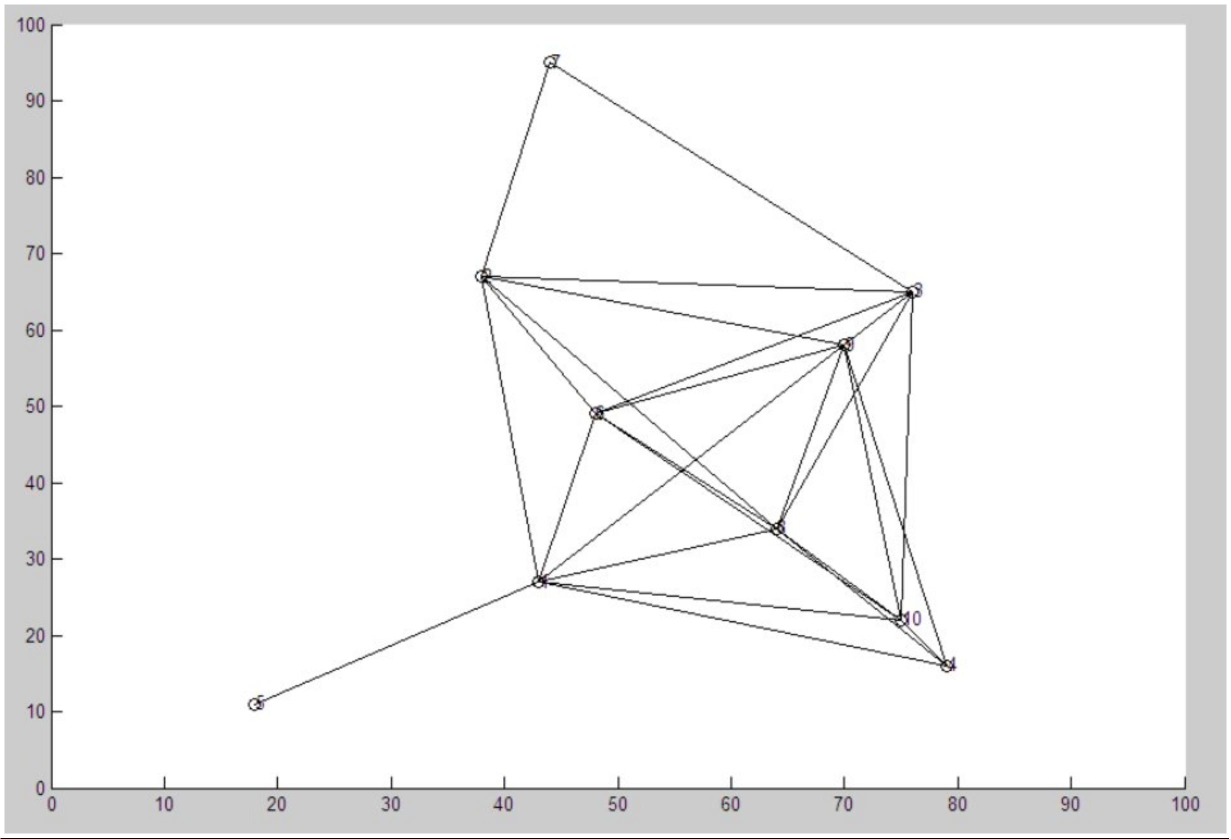
As the communication range is just 25 and the number of nodes are 10, the connectivity of the nodes in the graph is less. So, we further increase the communication range.

**Communication range = 35**



In the figure we can see that connectivity increases between the nodes as the communication range increases.

**Communication range = 45**

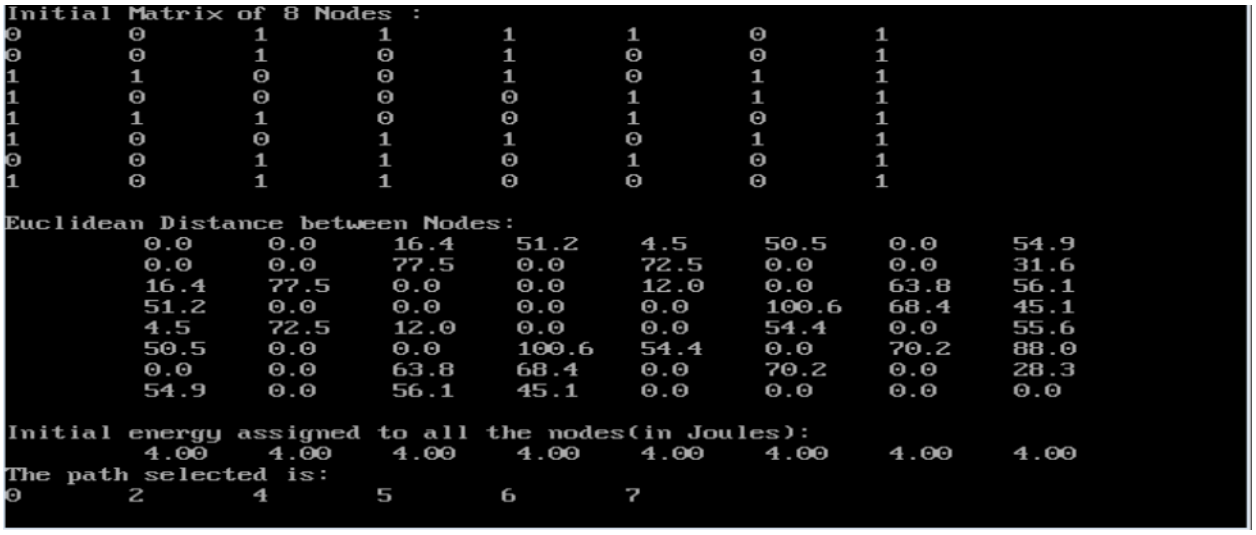


By watching all the three figures, we prove that the communication range of the nodes is directly proportional to the connectivity of the nodes.

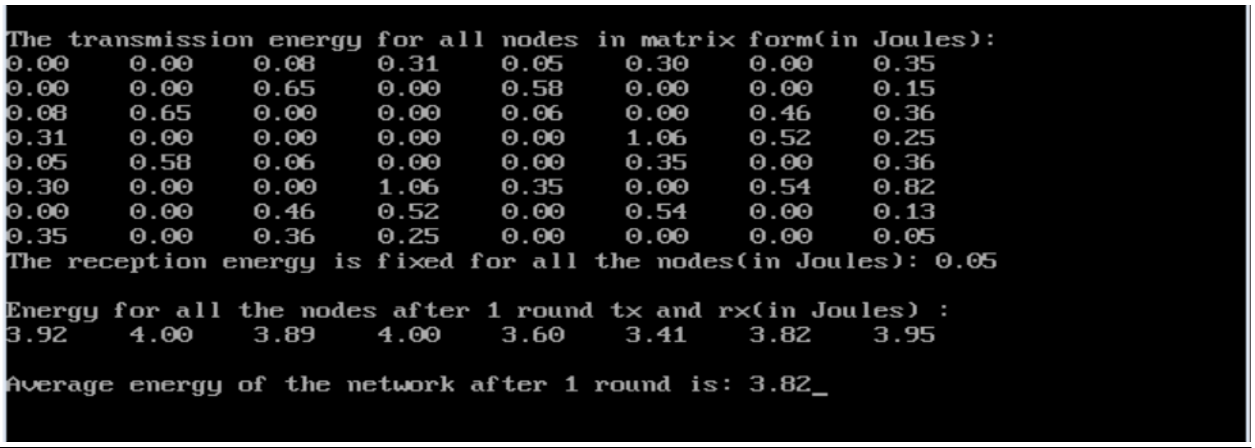
## 5.4 <u>C code output representing variation in average energy of the network</u>

**Input –** Number of nodes = 8

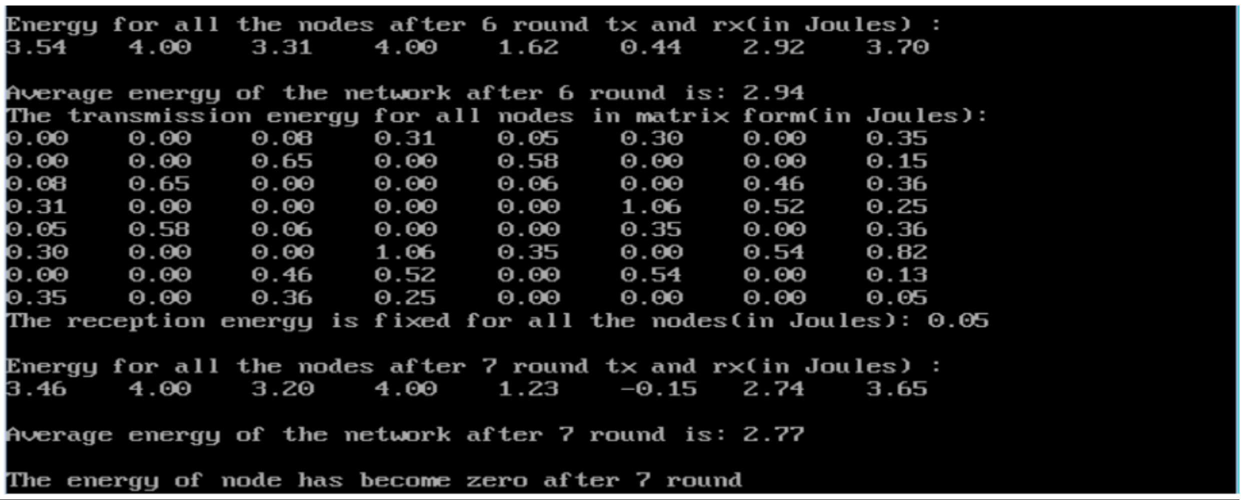
**Output**



```
Initial Matrix of 8 Nodes :
0       0       1       1       1       1       0       1
0       0       1       0       1       0       0       1
1       1       0       0       1       0       1       1
1       0       0       0       0       1       1       1
1       1       1       0       0       1       0       1
1       0       0       1       1       0       1       1
0       0       1       1       0       1       0       1
1       0       1       1       0       0       0       1

Euclidean Distance between Nodes:
         0.0     0.0     16.4    51.2    4.5     50.5    0.0     54.9
         0.0     0.0     77.5    0.0     72.5    0.0     0.0     31.6
         16.4    77.5    0.0     0.0     12.0    0.0     63.8    56.1
         51.2    0.0     0.0     0.0     0.0     100.6   68.4    45.1
         4.5     72.5    12.0    0.0     0.0     54.4    0.0     55.6
         50.5    0.0     0.0     100.6   54.4    0.0     70.2    88.0
         0.0     0.0     63.8    68.4    0.0     70.2    0.0     28.3
         54.9    0.0     56.1    45.1    0.0     0.0     0.0     0.0

Initial energy assigned to all the nodes(in Joules):
         4.00    4.00    4.00    4.00    4.00    4.00    4.00    4.00
The path selected is:
0       2       4       5       6       7
```



```
The transmission energy for all nodes in matrix form(in Joules):
0.00    0.00    0.08    0.31    0.05    0.30    0.00    0.35
0.00    0.00    0.65    0.00    0.58    0.00    0.00    0.15
0.08    0.65    0.00    0.00    0.06    0.00    0.46    0.36
0.31    0.00    0.00    0.00    0.00    1.06    0.52    0.25
0.05    0.58    0.06    0.00    0.00    0.35    0.00    0.36
0.30    0.00    0.00    1.06    0.35    0.00    0.54    0.82
0.00    0.00    0.46    0.52    0.00    0.54    0.00    0.13
0.35    0.00    0.36    0.25    0.00    0.00    0.00    0.05
The reception energy is fixed for all the nodes(in Joules): 0.05

Energy for all the nodes after 1 round tx and rx(in Joules) :
3.92    4.00    3.89    4.00    3.60    3.41    3.82    3.95

Average energy of the network after 1 round is: 3.82_
```

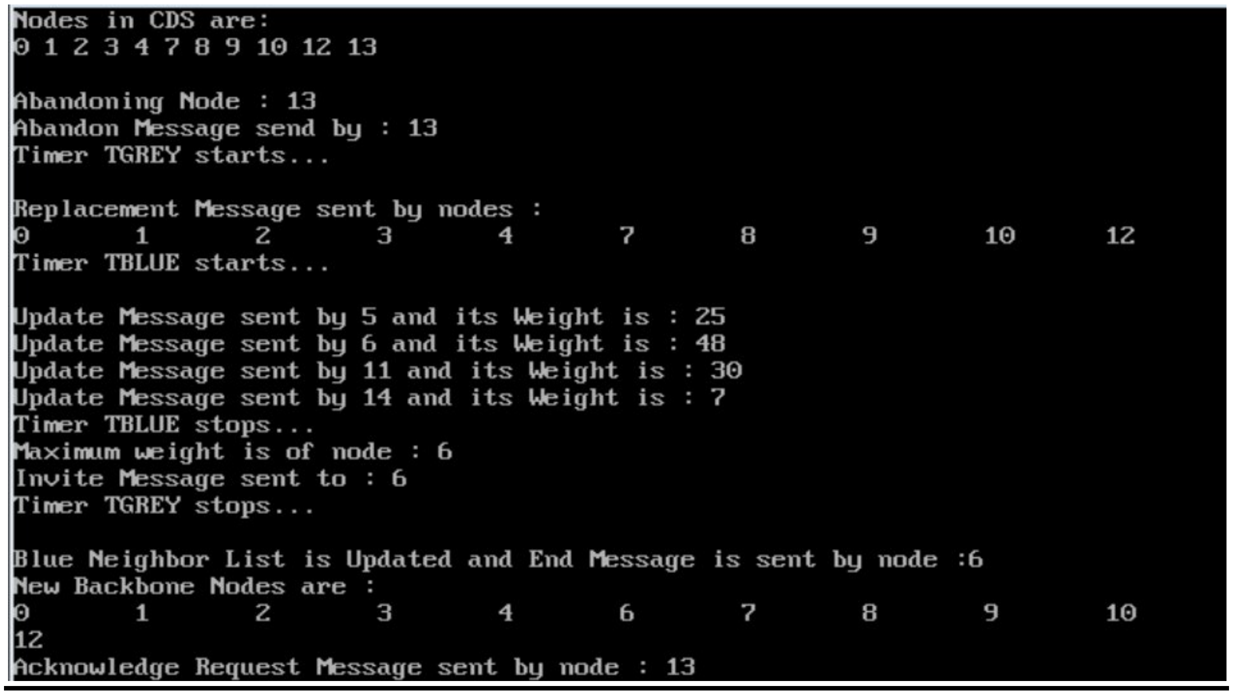


```
Energy for all the nodes after 6 round tx and rx(in Joules) :
3.54    4.00    3.31    4.00    1.62    0.44    2.92    3.70

Average energy of the network after 6 round is: 2.94
The transmission energy for all nodes in matrix form(in Joules):
0.00    0.00    0.08    0.31    0.05    0.30    0.00    0.35
0.00    0.00    0.65    0.00    0.58    0.00    0.00    0.15
0.08    0.65    0.00    0.00    0.06    0.00    0.46    0.36
0.31    0.00    0.00    0.00    0.00    1.06    0.52    0.25
0.05    0.58    0.06    0.00    0.00    0.35    0.00    0.36
0.30    0.00    0.00    1.06    0.35    0.00    0.54    0.82
0.00    0.00    0.46    0.52    0.00    0.54    0.00    0.13
0.35    0.00    0.36    0.25    0.00    0.00    0.00    0.05
The reception energy is fixed for all the nodes(in Joules): 0.05

Energy for all the nodes after 7 round tx and rx(in Joules) :
3.46    4.00    3.20    4.00    1.23    -0.15   2.74    3.65

Average energy of the network after 7 round is: 2.77

The energy of node has become zero after 7 round
```

## 5.5 Patching Results

C code representing EEBBPv1 patching protocol:

**Output**



```
Nodes in CDS are:
0 1 2 3 4 7 8 9 10 12 13

Abandoning Node : 13
Abandon Message send by : 13
Timer TGREY starts...

Replacement Message sent by nodes :
0       1       2       3       4       7       8       9       10      12
Timer TBLUE starts...

Update Message sent by 5 and its Weight is : 25
Update Message sent by 6 and its Weight is : 48
Update Message sent by 11 and its Weight is : 30
Update Message sent by 14 and its Weight is : 7
Timer TBLUE stops...
Maximum weight is of node : 6
Invite Message sent to : 6
Timer TGREY stops...

Blue Neighbor List is Updated and End Message is sent by node :6
New Backbone Nodes are :
0       1       2       3       4       6       7       8       9       10
12
Acknowledge Request Message sent by node : 13
```

From the above implementation, we can see that node number 13 is abandoning the network and sending abandoned message to other nodes in the network and finally getting replaced by node number 6.

# References

- R. Saravana Kumar , S.G.Susila, J.Raja -*"An Energy Efficient Cluster Based node Scheduling Protocol for Wireless Sensor Networks". 2010 IEEE*

- Samia A.Ali , Khaled M. Shaaban, Islam M. Alkabbanny – "*Distributed Patching For Mobile Wireless Sensor Networks*". Department of Electrical Engineering, Assuit University, Assuit, Egypt. Journal of Network and Computer Applications 35(2012). August 2012
- Journal Homepage: www.elsevier.com/locate/jnca

- Zhou Liu, Bingwen Wang and lejiang Guo,"A Survey on Connected Dominating Set Construction Algorithm for Wireless Sensor Networks", © 2010 Asian Network for Scientific Information, Informational Technology Journal 9(6): 1081-1092, 2010.

- S.Guha and S.Khuller, "Approximation algorithms for connected dominating set's", 20(4), Algorithmica, pp. 374-387, Apr. 1998.

- Feng Zhao, Leonidas J.Guibas, "Wireless Sensor Networks: An Information Processing Approach" MORGAN KAUFMAN PUBLISHERS.

# Appendices

**1. C- code for CDS Of a Network :**

```
#include<stdio.h>

#include<stdlib.h>

#define BLACK 1

#define WHITE 2

#define GREY 3

#define MAX_NODES 15

//void cds(struct node node1[]);



struct node

{

   int color;

   int ngh[10];

   int cnt;

   struct node *next;

};
```

}

```
struct node n1[MAX_NODES];
```

```
void cds(int k,struct node n1[]);

void check(struct node n1[]);

int maximum(struct node n1[]);



void degree(struct node n1[])

{

  int r,i,j,count,max,k,cd[MAX_NODES];

  for(i=0;i<MAX_NODES;i++)

  {



   count=0;

   for(j=0;j<MAX_NODES;j++)

   {



     if(n1[i].ngh[j]==1)

     {

            count++;

     }
   }

   n1[i].cnt=count;

  }

  r = maximum(n1);

  printf("\n\nMaximum Degree is of Node: %d", r);

  cds(r,n1);

  printf("\nMarked all neigbours as GREY.");

  //getchar();

  check(n1);

}




int maximum(struct node n1[])

{

  int i,max,k;

  for(i=0;i<MAX_NODES;i++)

  {

    printf("%d ",n1[i].cnt);

  }
  max=n1[0].cnt;

  k=0;

  for(i=0;i<MAX_NODES;i++)

  {

    if(n1[i].cnt > max)

    {

      max=n1[i].cnt;

      k=i;

    }

  }

  return k;

}



int maximum1(struct node n1[])

{

    int k,i,j,b1[MAX_NODES][2],p=0,q=0,max1,a,b;

    for(i=0;i<MAX_NODES;i++)

            for(j=0;j<2;j++)

                b1[i][j] = 0;
```

```
 for(i=0;i<MAX_NODES;i++)

 {

        q=0;

        if(n1[i].color==GREY)

        {

          b1[p][q]=i;

          q++;

          b1[p][q]=n1[i].cnt;

          p++;

        }

 }



/* for(a=0;a<p;a++)

 {

        printf("\n");

        for(b=0;b<2;b++)

        {

         printf("\t%d",b1[a][b]);

        }
```

```
   }*/
   max1=b1[0][1];
   k=b1[0][0];
   for(p=0;p<MAX_NODES;p++)
   {
     if(b1[p][1] > max1)
          {
            max1=b1[p][1];
            k=b1[p][0];
          }
   }
   //printf("\n%d",k);
   return k;
}



void cds(int k,struct node n1[])
{
   int i,j,cd[MAX_NODES];
```

```
if(n1[k].color!=BLACK)

{

        n1[k].color = BLACK;

        for(i=0;i<MAX_NODES;i++){

            cd[i] = k;

        }

        // Mark neighbours as GREY.

        for(j=0;j<MAX_NODES;j++)

        {

            if((n1[k].ngh[j]) == 1)

            {

              if(n1[j].color != BLACK)

                    n1[j].color = GREY;

            }

        }

        printf("\n");

        printf("Colors Assigned :");

        for(i=0;i<MAX_NODES;i++)

        {
              printf("%d",n1[i].color);

          }

   }

}




void check(struct node n1[])

{

   int r,i,cnt1=0;

   for(i=0;i<MAX_NODES;i++)

   {

          if((n1[i].color == GREY) || (n1[i].color == BLACK))

          {

           cnt1++;

          }

   }

   do

   {

          if(cnt1 == MAX_NODES)
```

```
    {
      printf("\t\tCDS is formed.\n\t\tNodes in CDS are: \n\t\t");
      for(i=0; i<MAX_NODES; i++){
            if(n1[i].color == BLACK)
                printf("%d ", i);
      }
      exit(0);
    }
    else
    {
      r = maximum1(n1);
      printf("\nMaximum Degree is of Node: %d", r);
      cds(r,n1);
      printf("\nMarking neighbours as GREY.\n");
      //getchar();
      check(n1);
    }
}
```

```
while(cnt1 != MAX_NODES);

}




int  main()

{

   int i,j;

   //

   struct node n1[MAX_NODES];

   for(i=0;i<MAX_NODES;i++)

   {

         n1[i].color=WHITE;

         n1[i].next=NULL;

         for(j=0;j<MAX_NODES;j++)

         {

          n1[i].ngh[j]=rand()%2;

          n1[j].ngh[i]=n1[i].ngh[j];

         }
```

```
        n1[i].ngh[i]=0;

   }



   printf("\nRandomly Generated Graph : \n");

   for(i=0;i<MAX_NODES;i++)

   {

        printf("\n");

        for(j=0;j<MAX_NODES;j++)

        {

          printf("%d",n1[i].ngh[j]);

          printf("\t");

        }

   }

   //getchar();

   degree(n1);

   //getch();

   return 0;

}
```

**2. C- code representing Average Energy Change of the Network :**

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

#include<conio.h>

#define MAX_NODES 8

#define k 1000



struct node

{

     float e_tot;

    int ngh[MAX_NODES];

    int cnt;

    float ed[MAX_NODES];

    struct node *next;

};
```

```
          struct node n1[MAX_NODES];

          void path(struct node n1[],int p[],int l,int m,int count);

          void energy(struct node n1[],int p[],int m,int count);

void main()

{

          int m,i,j,x[MAX_NODES],y[MAX_NODES],l,count=0;

          int p[MAX_NODES];

     //   float ed[MAX_NODES][MAX_NODES];

          clrscr();



          for(i=0;i<MAX_NODES;i++)

          {

                    x[i]=rand()%100;

                    y[i]=rand()%100;

          }

    for(i=0;i<MAX_NODES;i++)

   {
          for(j=0;j<MAX_NODES;j++)
          {
           n1[i].ngh[j]=rand()%2;
           n1[j].ngh[i]=n1[i].ngh[j];
          }
          n1[i].ngh[i]=0;
   }
          i=MAX_NODES-1;
          for(j=0;j<MAX_NODES;j++)
          n1[j].ngh[i]=1;
          printf("Initial Matrix of %d Nodes",MAX_NODES);

          for(i=0;i<MAX_NODES;i++)
   {
          printf("\n");
          for(j=0;j<MAX_NODES;j++)
          {
          printf("%d",n1[i].ngh[j]);



printf("\t");

          }

          }



        for(i=0;i<MAX_NODES;i++)

        for(j=0;j<MAX_NODES;j++)

        {

                if(n1[i].ngh[j]==1)

                n1[i].ed[j]=sqrt(pow((y[j]-y[i]),2)+pow((x[j]-x[i]),2));

                else

                n1[i].ed[j]=0;

        }

        for(i=0;i<MAX_NODES;i++)

                n1[i].e_tot=4;

         printf("\nEuclidean Distance between Nodes:");

        for(i=0;i<MAX_NODES;i++)

        {printf("\n");
```

```
for(j=0;j<MAX_NODES;j++)

printf("%0.2f \t",n1[i].ed[j]); }

 printf("\n");

 printf("Initial energy assigned to all the nodes(in Joules): \n");

for(i=0;i<MAX_NODES;i++)

printf("%0.2f \t",n1[i].e_tot);

p[0]=0;

l=0;

m=0;

path(n1,p,l,m,count);

}



void path(struct node n1[],int p[],int l,int m,int count)

{

        int i,j;

        for(j=l;j<MAX_NODES;j++)

        {

        if(n1[l].ngh[j]==1)

        {       p[++m]=j;
```

```
                        l=j;



                break;

                }

                  }

                if(l==MAX_NODES-1)

                {printf("\nThe path selected is:\n");

                for(i=0;i<=m;i++)

                printf("%d \t",p[i]);

                energy(n1,p,m,count);

        }

                path(n1,p,l,m,count);

        }



        void energy(struct node n1[],int p[],int m,int count)

          {

                  float
e_el,e_fs,i,j,e_tx[MAX_NODES][MAX_NODES],e_rx,d[MAX_NODES],avg,sum=0;

                  int l=0,z;
                  int r=0,q;



                  e_el=50*pow(10,-6);

                  e_fs=0.1*pow(10,-6);

                  e_rx=k*e_el;

                  for(i=0;i<MAX_NODES;i++)

                  for(j=0;j<MAX_NODES;j++)

                  {if(n1[i].ngh[j]==1)

                  {

                  e_tx[i][j]=e_rx+(k*e_fs*(pow(n1[i].ed[j],2))); }

                  else

                  e_tx[i][j]=0;}

                  printf("\nThe transmission energy for all nodes in matrix form(in
Joules):");

 for(i=0;i<MAX_NODES;i++)

 {

                        printf("\n");

                        for(j=0;j<MAX_NODES;j++)

                        printf("%0.2f \t",e_tx[i][j]) ;
```

```
                }
                        printf("\nThe reception energy is fixed for all the nodes(in Joules):
%0.2f",e_rx);
                        m=m+1;
                        i=p[0];
                        q=p[1];
                        j=p[m-1];
                        n1[0].e_tot=n1[0].e_tot-e_tx[i][q];
                        n1[j].e_tot=n1[j].e_tot-e_rx;
for(z=1;z<m-1;z++)
                        {i=p[z];
                        j=p[z+1];
                        n1[i].e_tot=n1[i].e_tot-e_tx[i][j];
                        n1[i].e_tot=n1[i].e_tot-e_rx;
                 }
                 for(i=0;i<MAX_NODES;i++)
                 sum=sum+n1[i].e_tot;
                 avg=sum/MAX_NODES;
```

```
  printf(" \n\nEnergy for all the nodes after %d round tx and rxin Joules) : \n",count+1);
                for(i=0;i<MAX_NODES;i++)
                printf("%0.2f \t",n1[i].e_tot);
                printf("\nAverage energy of the network after %d round is: %0.2f",count+1,avg);
                 for(i=0;i<MAX_NODES;i++)
                {
                 if(n1[i].e_tot<0)
                 { printf("\n\nThe energy of node has become zero after %d round\n",count+1);
                  //printf("\t\tNetwork Failed!!! ");
                 exit(0);
                 }
                 else
                 r=r+1;
                 }
                 if(r==MAX_NODES)
                 {count=count+1;
```

```
            m=0;

            l=0;

            for(j=0;j<MAX_NODES;j++)

            p[j]=0;

            path(n1,p,l,m,count);

            }

    }
```