Construction and Maintenance of Connected Dominating Set as Virtual Backbone in Wireless Network

Jasaswi Prasad Mohanty

CONSTRUCTION AND MAINTENANCE OF CONNECTED DOMINATING SET AS VIRTUAL BACKBONE IN WIRELESS NETWORK

Thesis submitted to
Indian Institute of Technology Kharagpur
for the award of the degree

of

Doctor of Philosophy

by

Jasaswi Prasad Mohanty

under the guidance of

Dr. Chittaranjan Mandal
Professor
Department of Computer Science and Engineering

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
West Bengal, India
May 2019

Dedicated To,

My parents,

Shri Jagnya Prasad Mohanty

Smt. Annapurna Dei

and

My beloved family,

Sushri, Subham and Shibansu

CERTIFICATE OF APPROVAL

Certified that the thesis entitled "**Construction and Maintenance of Connected Dominating Set as Virtual Backbone in Wireless Network**" submitted by **Jasaswi Prasad Mohanty** to Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

Sign:	Sign:	Sign:
Dr. Arobinda Gupta	Dr. Krothapalli Sreenivasa Rao	Dr. Raja Datta
(Member of the DSC)	(Member of the DSC)	(Member of the DSC)

Sign:	Sign:	Sign:
Dr. Dipanwita Roy Chowdhury	Dr. Chittaranjan Mandal	Dr.
(Chairman)	(Supervisor)	(External Examiner)

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India-721302

Certificate

This is to certify that the thesis entitled "**Construction and Maintenance of Connected Dominating Set as Virtual Backbone in Wireless Network**" submitted by **Jasaswi Prasad Mohanty** to Indian Institute of Technology Kharagpur, is a record of bona fide research work carried out under my supervision, and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

28th May, 2019
Kharagpur

Dr. Chittaranjan Mandal
Professor
Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur -721 302, INDIA

Declaration

I certify that

a. the work contained in the thesis is original and has been done by me under the guidance of my supervisor;

b. the work has not been submitted to any other institute for any other degree or diploma;

c. I have followed the guidelines provided by the Institute in preparing the thesis;

d. I have conformed to ethical norms and guidelines while writing the thesis;

e. whenever I have used materials (data, models, figures and text) from other sources, I have given due credit to them by citing them in the text of the thesis, and giving their details in the references, and taken permission from the copyright owners of the sources, whenever necessary.

Jasaswi Prasad Mohanty

Acknowledgment

It gives me immense pleasure to convey my deep sense of gratitude to my supervisor Prof. Chittaranjan Mandal for his expert guidance and support throughout my research work. He has taught me to work as an independent researcher and has been a steady source of wise encouragement and support. His suggestions and ideas provided the platform of my research. This work would not have been completed without his active involvement.

It gives me immense pleasure to thank my doctoral scrutiny committee (DSC) members Prof. Dipanwita Roy Chowdhury, Prof. Arobinda Gupta, Prof. Krothapalli Sreenivasa Rao, and Prof. Raja Datta for their valuable suggestions during my research tenure. My sincere thanks to the current research coordinator Prof. Debdeep Mukhopadhyay, and all former research coordinators namely Prof. Debasis Samanta, Prof. Shamik Sural who supported me in all phases of this course. I sincerely remember the support of office staff members.

I am greatly thankful to many of my friends for their constant inspiration. I thank for my parent institute Silicon Institute of Technology, Bhubaneswar for granting me three years of research leave. I am thankful to my friends and colleagues namely Sudhakar Sahu, Bikram Mishra, Bhagwat Choudhury, Sushri Rout, Pamela Choudhury, Manjula Raja, Mahesh Shirole, Prasanjeet, Barsha Mitra, Anant Nimkar, Shankar Ganesh and Shankar for always providing any help whenever I required.

It would have been impossible to achieve anything without the support of my parents. My beloved family Sushri, Subham and Shibansu support made it possible to embark on such an extraordinary journey. Continuous support and encouragement from my brother Jiban Jyoti keep me going on this destination.

Finally, I would like to acknowledge the financial support of the All India Council of Technical Education of Government of India for my study.

Jasaswi Prasad Mohanty

Abstract

A virtual backbone plays an important role in routing packets in a Wireless Sensor Network where a predefined infrastructure is absent. Some nodes of the network takes on additional responsibilities in an algorithmic framework to form the virtual backbone. A connected dominating set (CDS) can work as a virtual backbone in a wireless network. A dominating set of a graph is a subset of its vertices such that each node is either within that set or adjacent to one of the nodes present in that set. If the nodes within the dominating set are connected, then the dominating set is known as a connected dominating set. As the routing responsibilities lie only on the CDS nodes, we are interested in minimizing the CDS size. However, the construction of minimum CDS is an NP-Complete problem. In this dissertation, we first designed a new centralized degree-based greedy approximation algorithm, which constructs CDSs of smaller sizes in comparison with other existing algorithms. The proposed algorithm retains the current best approximation ratio and is also the most time efficient CDS construction algorithm. In our second work, we developed a novel distributed greedy approximation algorithm for CDS construction which reduces the CDS size effectively. Our simulation shows that this is the most size optimal distributed CDS construction algorithm with linear message complexity. The algorithm constructs the CDSs in lesser number of rounds in comparison to other degree-based algorithms. In a CDS, a node may fail or downgrade due to lack of its battery power or some other reason. In this situation, it is advantageous to repair the current CDS rather than reconstructing a fresh CDS. In our third work, we developed a distributed CDS maintenance algorithm which repairs the CDS by changing the role of only a few nodes. The proposed algorithm has linear time and message complexity. This CDS maintenance scheme handles the failure of both CDS and non-CDS nodes. To use our distributed CDS construction and maintenance algorithms each node needs its two-hop neighbours' information.

Keywords: Connected Dominating Set, Steiner Tree, Unit Disk Graph, Virtual Backbone, CDS maintenance.

Contents

List of Abbreviations

MANET	Mobile Ad Hoc Networks
VANET	Vehicular Ad Hoc Networks
WMN	Wireless Mesh Networks
WSN	Wireless Sensor Networks
DS	Dominating Set
CDS	Connected Dominating Set
MCDS	Minimum Connected Dominating Set
GG	General Graph
DG	Disk Graph
UDG	Unit Disk Graph
UBG	Unit Ball Graph
IS	Independent Set
MIS	Maximal Independent Set
ST	Steiner Tree
QoS	Quality of Service
VB	Virtual Backbone
ABPL	Average Backbone Path Length
PDS	Pseudo Dominating Set
CPDS2HI	Connected Pseudo Dominating Set using 2-Hop Information
DCMCDS	Distributed Construction of Minimum Connected Dominating Set
DMCDS	Distributed Maintenance of Connected Dominating Set
MI	Multi-Initiator
SI	Single-Initiator
PTAS	Polynomial Time Approximation Scheme
EHCDS	Energy Harvest Connected Dominating Set

List of Figures

List of Tables

CHAPTER 1

Introduction

A wireless ad hoc network consists of computing nodes connected with wireless links. There is no centralized control over the entire network. Due to the decentralized nature, it is useful in many areas where either central nodes cannot be relied upon or there is a concern of scalability. Setting up a wireless network is easy and quick due to the presence of dynamic and adaptive routing protocols. In this type of network, there is a minimal configuration requirement and the maintenance & installation cost is very less. Due to the above-mentioned reasons, a wireless network is suitable for various applications like military environment monitoring [1], natural disasters control [2], traffic information passing, search and rescue, etc. Mainly wireless ad hoc networks are classified by their application and structure. Some of the classifications include Mobile Ad Hoc Networks (MANET), Vehicular Ad Hoc Networks (VANET), Wireless Mesh Networks (WMN), Wireless Sensor Networks (WSN), etc.

Wireless Sensor Network is the most popular wireless network, which consisting of MEMS [3, 4] and some sensor nodes which are capable of sensing data from the deployment area. The sensor nodes are made of small size, low cost and low processing power devices which are used for data sensing, data processing, and communication. The major limitation of a sensor node is its battery power. So the major concern in a wireless sensor network is how to conserve the battery power of each sensor node to extend the lifetime of the entire network. One of the most vital tasks of the wireless network is to measure various event features like temperature, humidity, fire, etc. from the physical environment [5]. In case of occurrence of an event, the sensor nodes collaboratively sense the event data [6], [7] and transmit it to the sink node or the end user. The sensor nodes create organizational structures among them. The organizational structure of a wireless sensor network consists of individual sensor nodes, few clusters head [8] and the sink or base station. A sensor node rather than forwarding the sensed data directly to the sink node forwards the data to the cluster head for data fusion and transmission of combined data to the sink node.

Unlike wired networks, ad hoc network does not rely on any pre-existing infrastructure like routers. A node can send and receive messages within its transmission range. When a node broadcasts a message, the message is received by all the nodes within its communication range. In case a sender node wants to send a message to a node which is not within its communication range, the message is forwarded to the destination through some intermediate nodes. One of the simple methods of forwarding a message from a node to any other node is pure flooding. However, it is not an effective mechanism in a wireless network because it consumes high bandwidth and battery. In a wireless network, we can overcome these challenges in multi-hop routing through the use of a virtual backbone. In this type of network, some of the nodes are selected to form a virtual backbone to help in routing by forwarding data for other nodes. These nodes are selected dynamically by considering the state of each individual node as well as the state of the entire network.

A wireless network can be static or dynamic depending on the mobility of the nodes. We say a wireless network is static when all the nodes are static. A wireless network is dynamic when some or all of the nodes change their position frequently. A dynamic wireless network consists of a collection of nodes that can move on their own and interact with the physical environment. Mobile nodes have the ability to compute and communicate like static nodes. The key difference is mobile nodes have the ability to re-position and organize themselves in the network. A dynamic wireless network can start off with some initial deployment and nodes can then spread out to gather information. Information gathered by a

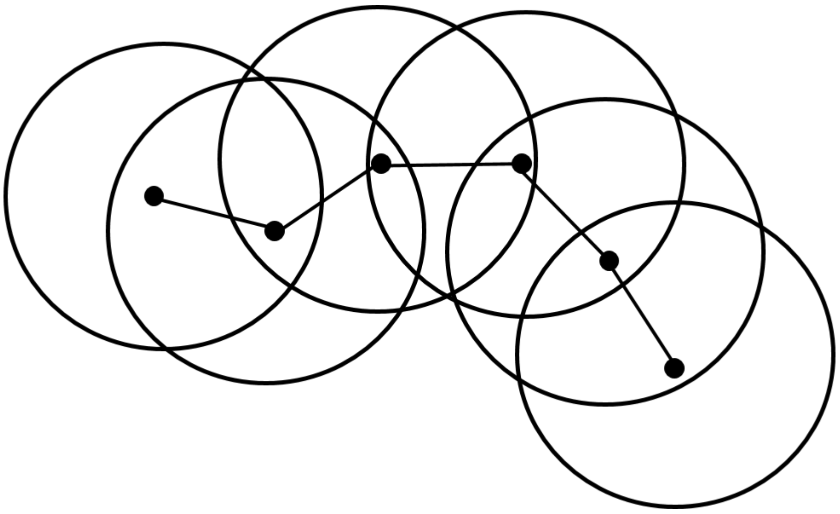


Figure 1.1: Unit Disk Graph Model representing the topology of a wireless network

mobile node can be communicated to another mobile node when they are within range of each other. Another key difference is data distribution. In a static wireless network, data can be distributed using fixed routing or flooding while dynamic routing is used in a dynamic wireless network. Challenges in the dynamic wireless network include deployment, localization, self-organization, navigation and control, coverage, energy, maintenance, and data process. In this dissertation, all three presented works are on static networks, we assume that the nodes do not change their position once they are deployed.

1.1 Network Models

The wireless network is modeled as a graph $G = (V, E)$ upon which algorithms operate. In the graph G, V represents the set of nodes in the network and E represents the set of all links between the nodes in the network. A node in the graph represents a wireless device and an undirected edge between two nodes shows that two devices are within the radio range of each other. A directed edge from node u to node v shows that node v is within the communication range of node u. Depending on the radio range of the nodes the network can be formed as the following models:

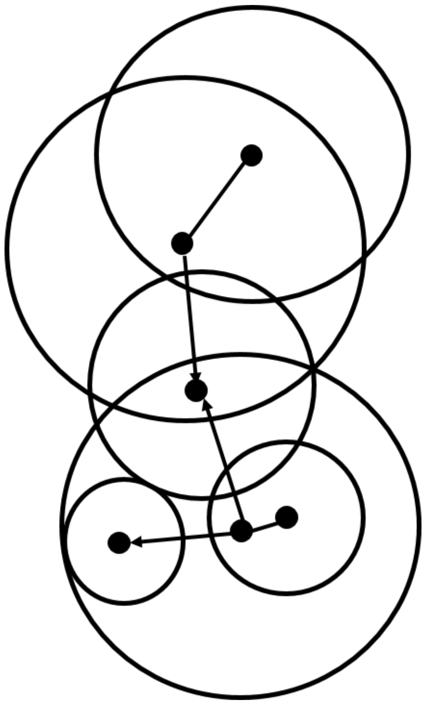


Figure 1.2: Disk Graph Model representing the topology of a wireless network

1. *Unit Disk Graph (UDG)*: Unit disk graph is used when all nodes of the wireless network can be assumed to lie in one plane. In this model, all the nodes in the network are of equal radio range r and deployed in a Euclidean plane. A graph $G = (V, E)$ is known as a unit disk graph if a pair of nodes are adjacent if and only if their Euclidean distance is less than or equal to r. Figure 1.1 shows a UDG, where each circle represents the transmission range of a node. All the nodes that fall inside this circle are said to be adjacent to the node at the center. We can observe from the diagram that the transmission range of all the nodes is the same. In the literature, one can find most of the wireless network algorithms for different problems are designed on a unit disk graph. However, in reality, the radio range of the nodes may vary and radios are not unidirectional.

2. *Disk Graph (DG)*: One can model a wireless network as a disk graph if the nodes are deployed in an Euclidean plane and their communication ranges are different. In the disk graph $G = (V, E)$ we can put an edge $(v_i, v_j) \in E$

if the Euclidean distance between the node v_i and v_j is less than or equal to the transmission range of v_i. An edge is unidirectional if $(v_i, v_j) \in E$ and $(v_i, v_j) \notin E$. An edge is bidirectional if both $(v_i, v_j) \in E$ and $(v_i, v_j) \in E$. Figure 1.2 shows a DG, where the circles represent the transmission ranges of different nodes. We can see in the figure that the undirected edges represent the bidirectional links and directed edges represent unidirectional links. Note that each undirected edge between a pair of nodes (v_i, v_j) is a composition of two directed edges from v_i to v_j and v_j to v_i.

3. *Unit Ball Graph (UBG)*: In some situations, the wireless network is modeled in a *3-dimensional* space instead of the plane [9]. For example, in a 3-dimensional under-water application, sensor nodes float at different depths in order to observe a given phenomenon. This kind of network is used for surveillance applications or monitoring of ocean phenomena (e.g. ocean bio-geo-chemical processes, water streams, pollution, etc.). Such 3-dimensional wireless networks are modeled using unit ball graphs. A graph is called unit ball graph if its vertices can be represented as points in 3-dimensional Euclidean space and two vertices are adjacent if and only if the distance between the two corresponding points is less than or equal to radio range of each node.

In this dissertation, the resultant topology of the wireless network is modeled as a unit disk graph with transmission ranges of all the nodes being considered as one unit.

1.2 Related Definitions

In this section, we discuss some of the fundamental concepts that are useful to understand our work.

Definition 1.1 (DOMINATING SET) *In graph theory, a dominating set (DS) for a graph $G(V, E)$ is a subset $V' \subseteq V$ such that for each node $v \in V - V'$, $Adj[v] \bigcap V' \neq \phi$, where $Adj[v]$ denotes set of adjacent nodes of v. The nodes in the dominating set, V' are called **dominators**. In Figure 1.3 the red coloured nodes are forming dominating sets in different networks.*

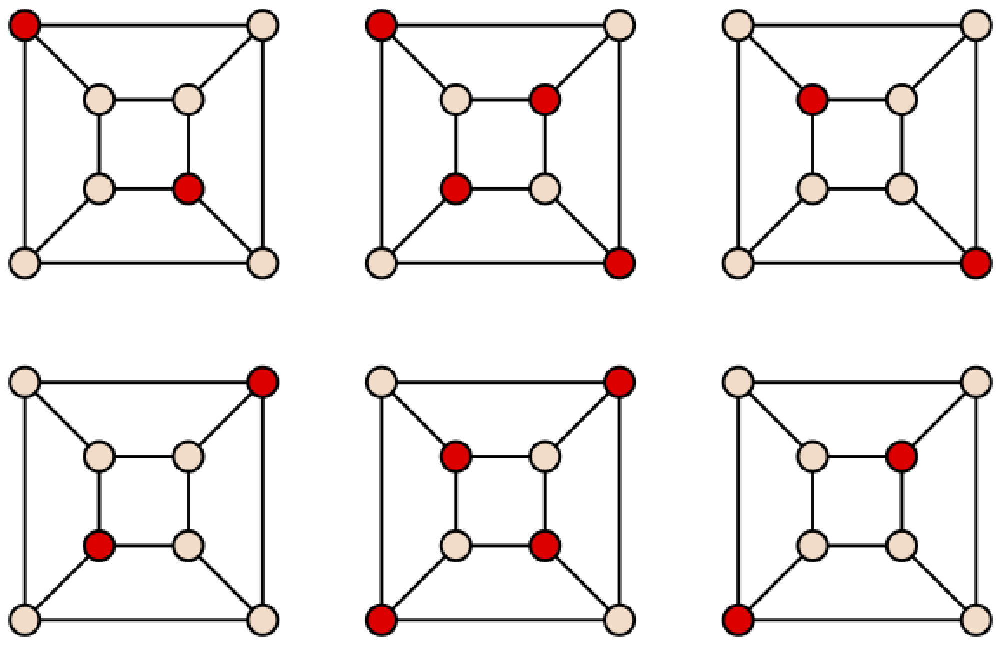


Figure 1.3: Example showing Dominating Set

Definition 1.2 (CONNECTED DOMINATING SET) *A dominating set which forms a connected sub-graph is a Connected Dominating Set (CDS). So, a CDS of a graph is a set of vertices with the following properties:*

1. *Every vertex of the graph is either belongs to the CDS or is adjacent to at least one vertex of the CDS.*
2. *We can reach from any node in the CDS to any other node in CDS by a path which stays entirely within CDS.*

The nodes which do not belong to the CDS are called as ***dominatees****. In Figure 1.4 the red coloured nodes are forming the connected dominating set in the network.*

Definition 1.3 (INDEPENDENT SET) *In a graph, a set of vertices in which no two vertices are adjacent is called an independent set or stable set.*

Definition 1.4 (MAXIMAL INDEPENDENT SET) *An independent set to which by adding any vertex outside the independent set disturbs the property of independent set is called as Maximal Independent Set (MIS) or maximal stable set.*

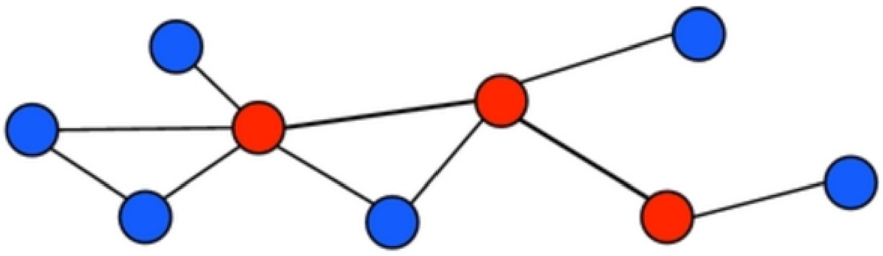


Figure 1.4: Example showing Connected Dominating Set

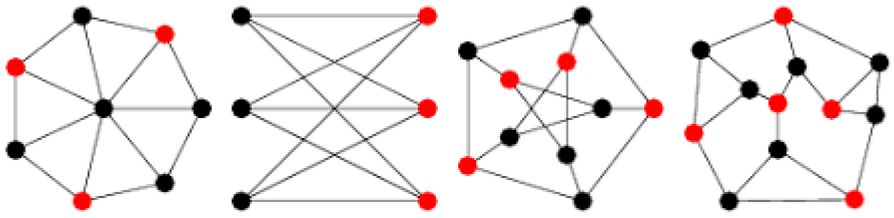


Figure 1.5: Example showing Maximal Independent Set

In other words, a maximal independent set cannot be a subset of any other independent set. In Figure 1.5 the red coloured nodes are forming maximal independent sets in different networks.

Definition 1.5 (STEINER TREE) *In a graph $G = (V, E)$, for a given subset of vertices $I \subseteq V$, a Steiner Tree is a tree which interconnects the nodes in I using a set of nodes (known as Steiner nodes) not in I.*

Definition 1.6 (m-DOMINATIING SET (m-DS)) *In a graph $G = (V, E)$, an m-DS D is a subset of V such that each node in $V - D$ is adjacent to at least m nodes in D.*

Definition 1.7 (k-CONNECTED m-DOMINATIING SET (k-m-DS)) *In a graph $G = (V, E)$, an k-m-DS D is a subset of V such that D is a m-DS of G and the sub-graph formed by the nodes of D is k-vertex connected.*

Definition 1.8 (DOMATIC PARTITION (DP)) *A domatic partition of a graph $G = (V, E)$ is a partition $D = \{D_1, D_2, ..., D_k\}$ of vertices of G such that each $D_i \in D (1 \leq i \leq k)$ is a DS of G.*

Definition 1.9 (CONNECTED DOMATIC PARTITION (CDP)) *In a graph $G = (V, E)$, a connected domatic partition $D = \{D_1, D_2, ..., D_k\}$ is a partition of vertices of G such that each $D_i \in D (1 \leq i \leq k)$ is a CDS of G.*

1.3 Backbone Network

Just like the human backbone carries signals to many smaller nerves in the body, a backbone network or network backbone is a part of computer network infrastructure that interconnects various pieces of network, providing a path for the exchange of information between different local area networks or sub-networks. A backbone can tie together diverse networks in the same building, in different buildings in a campus environment, or over wide areas. Normally, the backbone's capacity is greater than the networks connected to it [10]. A bus as a backbone network connects various local area networks and provides communication among the inter local area network hosts. Usually, bus backbones connect different buildings in an organization. Inside a single building, star backbone is used as a medium of distribution. A star backbone connects various local area networks through a switch. In fact, the switch works as the backbone in this case. At the service provider level, the routers and switches are the main components of the backbone network.

Wireless local area networks use access point as backbones. Backbone formation is a cost-effective alternative to the flooding approach that has been extensively studied in wireless sensor and ad hoc networks [11]. In this type of network, some of the selected nodes work as the backbone and perform the role of forwarding data. As there is no real backbone this backbone is known as virtual backbone. In this dissertation, we are concerned with ad hoc wireless networks that benefit by assigning forwarding roles to a few nodes belonging to a virtual backbone.

1.4 Virtual Backbone in Wireless Network

As discussed in the previous section, in a wireless network, routing related tasks are difficult to perform because neither there is any predefined physical backbone infrastructure nor there is any topology control mechanism. This is the main motivation behind the use of virtual backbone in a wireless network. The idea of virtual backbone was introduced by A. Ephremides et al. [12] in 1987. A virtual backbone is a collection of some of the selected nodes present in the entire network. The nodes in the virtual backbone besides their regular responsibilities take the additional load to help in routing. The virtual backbone nodes are connected among each other. A source node, to send a message to a destination node, forwards the message to one of the backbone node, which is adjacent to it. These backbone nodes forward any message among themselves such that the destination would receive the message. In this way of message passing, the routing path search space is reduced only to the set of backbone nodes. The backbone structure supports unicasting, multicasting with fault-tolerant routing. The major advantage of using a virtual backbone is, it can avoid the collision problem which occurs in flooding-based routing.

In a wireless ad hoc network, there is no predefined infrastructure as in wired network. Because of this reason when a source node needs to send a message to a destination node, which is outside its communication range it broadcasts the message. If the destination is within the range of the sender, it receives the message directly. However, if the destination is not within the range of sender, the sender requests its neighbouring nodes to forward the message to the destination. The neighbour of the sender forwards the message to its neighbour and this process continues until the message is received by the destination node. This method of forwarding the message is known as broadcasting which uses a lot of messages. In this approach, a lot of bandwidth, processing power and time of the node is consumed, which is unnecessary for many of the nodes in the network.

We can model a wireless network either by using a unit disk graph or disk graph or ball graph as discussed in Section 1.1. A Connected Dominating Set (CDS) can be used as a virtual backbone in a wireless network. The nodes which are not within the backbone must be adjacent to at least one node in the backbone. The non-virtual backbone nodes should keep track of their adjacent backbone nodes and the backbone nodes should keep track of their adjacent backbone nodes and also maintain a routing table. In a wireless sensor network, there is a limitation of battery power and a radio range of most of the nodes. The nodes in this network depend on their neighbours to forward messages to the base station or sink node [13]. If we can use the CDS as the virtual backbone, then they can provide the path to the base station. To conserve the battery power, the non-backbone nodes should turn off their radio when they do not have any data to send. When they are ready with the data, they should make their radio on, send the data and again should make their radio off. In this process, the routing activities are handed over to the virtual backbone nodes only, by virtue of which there is a significant reduction in message overhead due to routing.

1.5 Connected Dominating Set

Domination is one of the important research areas in graph theory and networks. There are a large number of real-world applications of this field due to which this area is thoroughly investigated for many years by researchers from different fields like Computer Science, Mathematics Communication, etc. Consider a directed graph $G(V, E)$, where V is the set of vertices and E is the set of edges. The classical *k-dominating set* D of graph $G(V, E)$ is a subset of V containing k vertices, such that for every vertex $v \in V$, either $v \in D$ or v has a neighbour in D. The minimum integer k for which G has a *k-dominating set* is called the *dominating number* of G and is denoted by $\gamma(G)$.

The DOMINATING SET problem is to check for a given graph $G(V, E)$ and an integer k, whether $\gamma(G) \leq k$ or not. There is an optimized version of the same problem in which we need to find the minimum dominating set. There is a wide range of real-world applications of the optimized version of DOMINATING SET problem. The origin of the dominating set concept traces back to the 1850's, when the following problem was noticed among chess players in Europe: *Determine the minimum number of queens that can be placed in a chessboard so that all squares are either attacked by a queen or are occupied by a queen.* In the later years, many variations of this problem were introduced. Some of the fundamental types of domination problem are given below:

1. *VERTEX COVER*: A vertex v is said to cover every edge incident to v. A vertex cover is a set S of vertices which covers every edge in E.

2. *EDGE COVER*: A set S of edges is an edge dominating set (edge cover), if for every edge $e \in E \setminus S$ there exists an edge $f \in S$, such that e and f have a common vertex.

3. *INDEPENDENT SET*: A dominating set D is an independent dominating set if no two vertices in D are adjacent.

4. *CONNECTED DOMINATION*: A dominating set D is a connected dominating set if the sub-graph induced by D is a connected subgraph of G.

The DOMINATING SET problem has been proved to be NP-Complete [14] in 1979. However, due to the practical applications of this problem, it is one of the hot subjects of many researchers during the last 40 years. Dominating sets are known to perform well in clustering [15], backbone formation [16, 11], multicast routing [17], and some other issues in wireless sensor and ad hoc networks. A dominating set is related to an independent set. An independent set is also a dominating set if and only if it is a maximal independent set. So any maximal independent set in a graph is necessarily also a minimal dominating set. Thus, the smallest maximal independent set is also the smallest independent dominating set. The minimum dominating set in a graph will not necessarily be independent, but the size of a minimum dominating set is always less than or equal to the size of a minimum maximal independent set.

If the nodes present in the dominating set (D) of a graph are connected, the set is called Connected Dominating Set. Formally, a CDS of a graph G is a sub-graph C of the given graph G (of connected nodes) such that each node in G is either in C or adjacent to a node in C.

If S is a connected dominating set, one can form a spanning tree of G in which S is the set of non-leaf nodes of the tree. Conversely, if T is any spanning tree in a graph with more than two vertices, the non-leaf nodes of T form a connected dominating set. Therefore, finding a minimum connected dominating set is the same as finding spanning trees with the maximum possible number of leaves. A *total dominating set* is a set of vertices such that all vertices in the graph (including the vertices in the dominating set itself) have a neighbour in the dominating set. Nodes in the dominating set are called *dominators*. The nodes which are adjacent to a dominator are called *dominatees*. The minimum size of a connected dominating set of G is called *Connected Dominating Number* of G and is denoted by $\gamma_c(G)$. Finding the minimum connected dominating set of a graph is known to be NP-Complete [14]. Connected-Dominating-Set is a representative technique for constructing virtual backbones of wireless networks and this facilitates the implementation of many tasks including broadcasting, routing, etc. Most of the existing works on CDS aim at constructing the minimum CDS (MCDS), so as to reduce the communication overhead over the CDS.

Application of Connected Dominating Set in Wireless Network:

Due to the unique characteristics of mobile ad hoc networks and wireless sensor networks, the necessary protocols are developed specifically to their purpose only. For better result, most of the protocols first organize the network by constructing the dominating set before doing any specific task. These protocols address routing, power management, etc.

Constructing a dominating set is nothing but forming clusters in the network. At the Data Link Layer, clustering helps in reducing collisions, providing a guarantee of Quality of Service (QoS) [18, 19, 20, 21]. The nodes in the dominating set interact with each other and use orthogonal spreading of codes in between their neighbours. This helps in improving spatial reuse with code division spread spectrum techniques [22]. Later on, these nodes can coordinate access to the wireless media by their neighbours to provide a guarantee of QoS or to avoid collisions.

In 1987, Ephremedis et al. first proposed that CDS can be used as a virtual backbone network for routing of messages [12]. Any message can be sent from a source node to one of the neighbouring CDS nodes, from this node to the closest CDS neighbour of the destination node and from that node to the destination node finally. This is known as backbone based routing [23] or dominating set based routing [24, 25], or spine based routing[26, 27]. We could reduce a significant amount of message overheads related to routing updates [28] by restricting the routing activities only to the CDS nodes. The dominating set can also be structured into a hierarchy to further minimize the control message overhead [29, 20, 21]. In location-based routing, a CDS can also be used. In this type of routing the messages are forwarded based on the geographical location of the hosts. The intermediate nodes are selected based on the vicinity of the destination node.

A CDS is also useful for location-based routing. In location-based routing, messages are forwarded based on the geographical coordinates of the hosts, rather than topological connectivity. Intermediate nodes are selected based on their proximity to the destination of messages. In this scheme, we may reach a situation where all the neighbours of the recently selected intermediate node would be farther from the destination node. In this situation, the routing must recover from this stage by using backtracking method to find another path. If messages are only forwarded to nodes in the dominating set, the inefficiency associated with this recovery phase can greatly be reduced [30].

We can also improve the multicast/broadcast routing by using CDS. In multicast/broadcast routing the intermediate nodes unnecessarily forward the messages most of the time. Many nodes receive the same messages again and again. This problem is known as broadcast storm problem [31]. If the messages would be forwarded through the CDS nodes only then the unnecessary forwarding of messages can be eliminated [32, 33, 34, 35, 36].

In a wireless sensor network, the major limitation of sensor nodes is their battery power. There is no facility of charging these sensor nodes again and again because they are deployed in remote locations most of the time. By using CDS we can allow the non-CDS nodes to be in sleep mode. By this, we can increase the number of nodes in a sleep mode. During the non-CDS nodes are in sleep mode the CDS nodes can forward the messages [37, 38]. In this way, they would be able to conserve energy among nodes [39, 40, 41, 36].

One more application of CDS is extraction of topology information in large-scale dense networks [42]. A CDS can be used to serve as database servers. We can also use CDS as the virtual backbone to spread "link quality" information for route selection in multimedia traffic [20].

1.6 Connected Dominating Set as Virtual Backbone

As the CDS can be used as the virtual backbone in a wireless network, we need to construct the CDS first before using it. After construction, we need to evaluate how efficiently we have constructed the CDS. There are many parameters used for the evaluation of the CDS, which is to act as virtual backbone [13]. Some of these parameters are:

- *Size*: The size of a CDS is the number of nodes present in it. The CDS size should be as small as possible for many reasons. Firstly, the backbone nodes are in charge of relaying packets and thus are more likely to drain their battery. In that case, fewer backbone nodes mean that fewer nodes are intensively used. So a smaller virtual backbone suffers less from the interference problem. One more reason is related to the purpose of creating such a structure. As virtual backbones are likely to be used by routing protocols, having a lesser number of backbone nodes induce less protocol-related messages, such as routing table updates, and thus increase the available bandwidth for real communications. For better performance, we should try to construct a CDS of the smallest possible size.

- *Diameter*: In a connected graph the diameter is the length of the longest path among the shortest paths between any pair of nodes in the graph. In a constructed CDS the diameter of the CDS is the diameter of the subgraph formed by the CDS nodes. So, the diameter of a CDS is the length of the longest path between any two vertices forming the CDS. For improved performance, it is desirable that the diameter is minimized. In a wireless network, when a message travels a longer distance (in terms of hop) error rate increases. Hence, for better performance, it is desirable to construct a CDS of smaller diameter also.

- *ABPL (Average Backbone Path Length)*: The average backbone path length is the average of distances traveled by different packets before reaching their destination. ABPL of a CDS is the sum of the hop distances between any pair of nodes u and v divided by the number of all possible pairs. We need to construct CDSs of smaller ABPL so that the average cost of communication can be reduced. Communication cost can be in terms of the average number of intermediate hops, processing power, bandwidth usage, battery power, etc.

- *Failure Tolerant*: The backbone should be node-failure tolerant. This characteristic is also vital as the failure of one node among the CDS nodes may result in a useless backbone because it may either disconnect the CDS or it may create a hole in the network. Many propositions have been made to increase the robustness of the structure namely *k-connectivity*, i.e. having k independent paths between any pair of nodes, empirical criteria (and their combination) such as remaining battery level, low relative speed (stable surroundings), etc.

In this dissertation, the proposed CDS construction algorithms have considered the CDS size as a parameter for CDS evaluation. Moreover, the selection of the connected dominating set must be distributed. Based on neighbourhood knowledge, a node must decide whether or not it is in the dominating set. A CDS is a good candidate for a virtual backbone of wireless networks because any non-CDS node in the network has *1-hop* distance from a CDS node. With the help of the CDS, routing is easier and can adapt quickly to network topology changes.

1.7 Construction of Minimum Connected Dominating Set and its Maintenance

For efficient routing and connectivity management in wireless networks, a CDS can be used as a virtual backbone. During routing, broadcasting responsibilities lie only with the CDS nodes, instead of all the nodes in the network. As only the CDS nodes maintain routing information, storage space can be reduced by reducing the CDS size. Due to this reason researchers are interested to construct Minimum Connected Dominating Sets (MCDS). However, MCDS construction is an NP-Complete problem. Therefore, only polynomial time approximation algorithms are practically usable. The performance of the CDS depends on its approximation ratio, which is the ratio of the size of the constructed CDS to the size of the MCDS. So, we should construct CDSs with a smaller approximation ratio. The construction cost is also measured by the overall message and time complexities.

The nodes in a CDS do some extra work of computation and communication to support routing and energy conservation of the entire network. Due to this extra load, the CDS nodes deplete their energy faster than the non-CDS nodes. If some of the CDS nodes fail, then the entire CDS would not be useful at all. In that case, we should either switch over to other CDS or repair the CDS. In case of failure of a single node, it is not wise to reconstruct a fresh CDS because very few nodes are affected and reconstruction will be performed over the entire network. So, it is better to maintain the CDS in case of failure of the CDS nodes. During the maintenance, we should ensure that we are not making too many changes to the current CDS. We should change the roles of very few nodes to maintain the current CDS.

1.8 Overview and Contribution

This section describes and lists the statements of the problems that have been addressed in this dissertation. Later on, the outline of the adopted methodologies for the solutions is given. Also, the specific contribution made in each case is mentioned in this section. The problems on wireless sensor network addressed in this thesis are:

1. **Centralized Construction of MCDS in Wireless Sensor Networks Using Pseudo Dominating Set :** In a wireless network, messages need to be sent in an optimized manner to preserve the energy of the network. A minimum connected dominating set (MCDS) offers an optimized way of sending messages. A Dominating Set of a network is a subset of nodes such that any node not in the subset is a neighbour of some element of that subset. It forms a Connected Dominating Set if the sub-graph induced by this set is connected. In a wireless network, as there is no fixed infrastructure or centralized management, a CDS can be used as a virtual backbone or spine

for efficient routing and connectivity management [43]. The CDS can receive a packet from any node in the network and can retransmit it to any other remote node. A node, which is not in the CDS can send a message to any other node through the CDS nodes. It first sends its message to one of its neighbouring CDS nodes. Now, the search space for any route is reduced to the CDS. If the destination node is within the CDS it can get the message directly, otherwise, it gets the message from one of its neighbouring CDS nodes. Thus, during routing, broadcasting responsibility lies only with the CDS nodes, instead of all the nodes in the network.

As only the CDS nodes maintain routing information, we can save the storage space by reducing the CDS size. A small sized CDS makes routing easier, reduces the communication overhead, increases the convergence speed and simplifies connectivity management. So, it is desirable to construct a minimum connected dominating set (MCDS) of the network. However, computing MCDS is an NP-complete problem [44]. So, only polynomial time approximation algorithms are practical for finding out MCDS in wireless networks. For energy-constrained wireless networks, an approximation algorithm should not only construct smaller CDSs but also construct CDSs with low computation and communication costs. Generally, the quality of the CDS is measured by its approximation ratio, which is the ratio of its size to that of the MCDS. The construction cost is measured by the overall message and time complexities. The computation time of the CDS should also be appreciably small in order to schedule speedy switches between disjoint CDSs to extend battery lifetime and optimize power consumption [45, 46].

In this work, we propose a new centralized degree-based greedy approximation algorithm which we name as **C**onnected **P**seudo **D**ominating **S**et Using **2**-**H**op **I**nformation (CPDS2HI) to construct smaller CDSs. Our scheme CPDS2HI works in three phases. In the first phase, a smaller maximal independent set (MIS), designated as a pseudo-dominating set (PDS) is constructed. The dominating set is pseudo dominating set because some of the elements may be omitted in the final dominating set. The second phase of our algorithm constructs an improved Steiner Tree which interconnects the PDS nodes in an improved way. In the final phase, some of the selected PDS nodes are excluded cleverly to reduce the CDS size further without any connectivity or coverage loss. Through simulation, we also show that our proposed algorithm CPDS2HI outperforms all the existing CDS construction algorithms in terms of CDS size and construction costs. CPDS2HI retains the current best performance ratio of $(4.8 + \ln 5)|opt| + 1.2$, $|opt|$ being the size of an optimal CDS of the network, and has the best time complexity of $O(D)$, where D is the network diameter. To the best of our knowledge, this is the most time efficient and size-optimal CDS construction algorithm.

2. **Distributed Construction of Minimum Connected Dominating Set in Wireless Sensor Network Using Two-Hop Information:** We may construct a CDS either in centralized or in a distributed manner. Although centralized algorithms provide more accurate information than distributed algorithms, they suffer from scalability problem and hence are not feasible for large size WSNs. In centralized algorithms, the reliability of the information accumulated at a centralized processor is low because of the losses involved in multi-hop transmission. Distributed algorithms are difficult to design. They require only local information exchange between neighbouring nodes. For any WSN in which the average number of hops from any node to the central processor is greater than the number of iterations required to perform a task, distributed algorithms are more energy efficient than centralized algorithms [47].

   In this work, we propose a new distributed degree-based greedy approximation algorithm which we name as **D**istributed **C**onstruction of **M**inimum **C**onnected **D**ominating **S**et (DCMCDS) to construct smaller CDSs. The proposed scheme DCMCDS works in three phases and constructs the CDS using 2-hop information only. In the first phase, it constructs the Maximal Independent Set (MIS) in a distributed manner. The MIS is designated as a Pseudo Dominating Set (PDS) because some of the elements may be omitted in the final dominating set. In the second phase, the algorithm constructs a Steiner Tree by adding some more nodes to the PDS, which are needed to interconnect the PDS nodes. In the last phase, the algorithm drops some of the selected PDS nodes to reduce the CDS size further without any loss in coverage or connectivity. Simulation results show that DCMCDS is better than existing CDS construction algorithms in terms of CDS size and construction costs. The performance ratio of the proposed algorithm, which is the best at the current moment, is $(4.8 + \ln 5)|\mathsf{opt}| + 1.2$, where $|\mathsf{opt}|$ is the size of an optimal CDS of the network. Its time complexity is $\mathsf{O}(\mathsf{D})$, where D is the diameter of the network. It has a linear message complexity of $\mathsf{O}(\mathsf{nR})$, where n is the network size and R is the maximum between number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes. The distributed greedy algorithm DCMCDS does not depend on any specific initiating node. It identifies non-trivial CDSs of smaller size for both uniform and random distribution of nodes in a distributed manner. The algorithm constructs the CDS in lesser number of rounds in comparison to other degree-based algorithms.

3. **Maintenance of CDS as Virtual Backbone in Wireless Sensor Network:** A Wireless Sensor Network (WSN) consists of a group of distributed sensor nodes deployed in the area to be monitored. As there is no fixed infrastructure, for efficient routing, some of the nodes come forward to form a virtual backbone. A Minimum Connected Dominating Set (MCDS) can be used as a virtual backbone. In the literature there are many approximated MCDS construction algorithms available as the construction of MCDS is an *NP-Hard* problem.

   The CDS nodes deplete their energy faster than non-CDS nodes because of extra computation and communication load. After a certain period of time, these CDS nodes would run out of their battery and hence would not able to handle the routing responsibilities anymore. In that situation, either we should switch over to a new CDS [45, 46] or repair the current CDS by finding some other non-CDS nodes to hand over the responsibilities of the failed CDS node. Also, it is a well-known fact that the battery performance can be greatly improved by using pulsed discharge instead of constant discharge [48, 49]. It is better to handover the routing responsibilities of the CDS node, which is about to fail in the near future rather than waiting for its failure. This will allow the recharge recovery effect in electrochemical batteries to extend their lifetime. If a node is chargeable from other sources like sunlight [50] then it can recharge itself for future use. A CDS node may also fail due to processor failure, radio failure, etc. So, whatever may be the reason for the failure of a CDS node, it may disconnect the CDS, as a result, the data transmission may stop also. In this scenario, it is not wise to re-construct the CDS fresh since very few nodes are affected and reconstruction will be performed on the entire network, so it is wastage of resource. Therefore, it is a better approach to repair the CDS by assigning new responsibilities to some of the non-CDS nodes and the affected CDS nodes are freed from their earlier responsibilities. Maintenance of a CDS is initiated by a particular CDS node when it finds that its battery power has reduced below a threshold value. The affected CDS node after being relieved as a backbone node can recharge itself from any available resources like sunlight and make itself ready for future use. However, if a CDS node fails due to any reason other than power failure, the nearby nodes should take the initiatives to repair the CDS.

In this work, we propose a new distributed CDS maintenance approach which we name as **D**istributed **M**aintenance of **C**onnected **D**ominating **S**et (DMCDS). Our proposed approach handles the following situations: (1) a CDS node finds that its battery power has reduced below a threshold value due to which it is about to fail in near future (2) a CDS node has already failed due to some reason (3) a non-CDS node finds its battery power has reduced below a threshold value due to which it is about to fail in near future (4) a non-CDS node has already failed due to some reason (5) a non-CDS node is ready to sense the required data after recharging. The proposed approach also takes care of the situation in which the CDS becomes disconnected by the failure of any CDS node. When the connectedness property of the CDS is lost due to the failure of any CDS node, multiple components are formed containing some of the CDS nodes. In this work, the non-CDS nodes come forward to connect these components by changing their role. According to the knowledge of the authors, this is the first CDS maintenance algorithm which handles multiple components. Simulation results show that the proposed algorithm is able to repair the CDS in all possible situations. The proposed distributed CDS maintenance scheme only changes the current CDS by a factor of Δ, where Δ is the maximum degree of a node in the entire network. Its time complexity is $\mathsf{O}(\mathsf{C})$, where C is the size of the largest component of the network during the application of the proposed CDS maintenance algorithm. It has a linear message complexity of $\mathsf{O}(\mathsf{n})$, where n is the number of nodes present in the network.

Contributions : The thesis has three contributions, which are summarized below:

1. We have proposed a new centralized degree-based greedy approximation algorithm to construct smaller CDSs with the current best approximation ratio of $(4.8 + \ln 5)|opt| + 1.2$, where $|opt|$ is the size of an optimal CDS of the network. The algorithm has the best time complexity of $O(D)$, where D is the network diameter. To the best of our knowledge, this is the most time efficient and size-optimal CDS construction algorithm.

2. We have developed a distributed degree based algorithm for the minimum connected dominating set problem with the current best approximation factor of (4.8 + ln 5)opt + 1.2, where $|\mathsf{opt}|$ is the size of an optimal CDS of the network. Simulation results show that DCMCDS is better than existing CDS construction algorithms in terms of CDS size and construction costs using a slightly higher expense of a number of messages exchanged as compared to previous degree-based CDS construction techniques. Its time complexity is $\mathsf{O}(\mathsf{D})$, where D is the diameter of the network. It has a linear message complexity of $\mathsf{O}(\mathsf{nR})$, where n is the network size and R is the maximum of the number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes. The distributed greedy algorithm DCMCDS does not depend on any specific initiating node. It identifies non-trivial CDSs of smaller size for both uniform and random distribution of nodes in a distributed manner. The algorithm constructs the CDS in lesser number of rounds in comparison to other degree-based algorithms.

3. We have developed a new distributed CDS maintenance approach to handle the following situations: (1) a CDS or non-CDS node is about to fail due to the depletion of its battery power below a threshold level (2) a CDS or non-CDS node has already failed due to some reason (3) a non-CDS node becomes ready to sense the required data after recharging. The proposed distributed CDS maintenance scheme only changes the current CDS by a factor of Δ, where Δ is the maximum degree of a node in the entire network. Hence, the performance ratio of the CDS holds after each maintenance. Its time complexity is $\mathsf{O}(\mathsf{C})$, where C is the size of the largest component of the network during the application of the proposed CDS maintenance algorithm. It has a linear message complexity of $\mathsf{O}(\mathsf{n})$, where n is the number of nodes present in the network.

1.9 Thesis Organization

The thesis has three working chapters, besides chapters on introduction, a review of the various works on CDS construction and maintenance in wireless sensor networks and conclusions.

The organization of the thesis is as given below.

Chapter 1: Introduction This chapter gives an overview of network models and related definitions to be used in the entire thesis. The chapter first introduces the virtual backbone and its importance in a wireless network. Later, the application of Connected Dominating Set as the virtual backbone is discussed. At the end of the chapter, the contribution of the thesis is presented.

Chapter 2: Review of CDS Construction Algorithms This chapter provides the state-of-the survey on existing CDS construction algorithms which are classified depending on various design goals. At the end of the chapter scope of our work is presented.

Chapter 3: Centralized Construction of CDS in WSN In this chapter, a centralized algorithm for the minimum connected dominating set problem based on a pseudo dominating set is proposed. An approximation factor for the computed MCDS has been derived. Simulation results show that the method constructs CDSs of smaller sizes than other similar techniques.

Chapter 4: Distributed CDS Construction in WSN Here a distributed algorithm for the minimum connected dominating set problem based on two hop information is proposed. An approximation factor, time and message complexity for the computed MCDS have been derived. Simulation results demonstrating the usefulness of this technique for effective aggregation over other competitive CDS schemes are presented.

Chapter 5: Distributed Maintenance of CDS in WSN This last working chapter contains an algorithm for CDS maintenance. The proofs describe that the method does not change the current CDS too much while maintaining. Also, the time and message complexity of the algorithm is derived.

Chapter 6: Conclusion In this chapter we summarize the contributions of this thesis and present our conclusions. Possible future extensions to this work are also identified.

CHAPTER 2

Review of CDS Construction Algorithms

2.1 Overview

Wireless ad hoc and sensor networks play a pivotal role in the next-generation network in providing flexible deployment and mobile connectivity. These types of networks are popularly used in the health-care industry, agriculture, food industry, automated battlefield, disaster control, etc. However, unlike wired networks or cellular networks, no physical backbone infrastructure is required in wireless ad hoc and sensor networks, which offers new paradigms for routing. These types of networks consist of either static or mobile or a mixture of both static and mobile nodes. Each of the nodes of these types of networks has an omnidirectional antenna and can broadcast messages within its communication range. When a node broadcast a message, the message can be reached to the nodes within the communication range of the sender. If the receiver is outside the single hop radio transmission range of the sender then the message is sent through intermediate nodes by establishing a communication session between them. This is known as multi-hop routing. In a wireless network, one of the simple and intuitive methods of forwarding messages between non-adjacent nodes is pure flooding. In flooding, each node broadcasts the packet once it has received from its adjacent node. The transmission in pure flooding is redundant due to which it consumes the available bandwidth of the channels and battery power of the nodes. Therefore, pure flooding is not an efficient communication mechanism in a wireless network. To overcome these challenges and to achieve scalability and efficiency a virtual backbone infrastructure can be used in multi-hop routing. A virtual backbone organizes ordinary nodes into a hierarchy. Generally, a Connected Dominating Set (CDS) is used as a virtual backbone in a wireless network. The protocols which use CDS can perform a wide range of communication functions. Some of the protocols which use CDSs as their underlying architecture includes media access coordination [18, 19, 20], multicast/broadcast [33, 34, 41, 36, 30], topology control [42], location-based routing [30]; energy conservation [37, 38, 39, 40, 51].

In the energy constrained ad hoc and sensor networks small size CDSs help to extend the network lifetime. The problem of finding the CDS with minimum cardinality is called Minimum Connected Dominating Set (MCDS) problem and is known to be NP-complete [44]. Therefore, most of the researchers are interested to design polynomial time approximation algorithms for small size CDS construction. In recent years many algorithms for CDS constructions have been proposed. For each of the CDS construction algorithms, the size of the CDS, its energy saving capability, time and message complexity, performance ratio, degree of localization have been studied deeply. In this chapter, we present the works on a large number of existing CDS construction techniques proposed in the context of wireless ad hoc and sensor networks.

In this chapter, we examine several centralized algorithms and their corresponding distributed implementations in detail. Distributed or even localized algorithms are useful in wireless ad hoc and sensor networks. The algorithms should rely on limited knowledge of network topology. Although there are many parameters to evaluate a CDS, most of the works are on construction of minimum connected dominating set (MCDS) in unit-disk graphs. So their main design goal is performance ratio. In fact, minimizing the size of the CDS can help us to decrease the control overhead since broadcasting and topology update is restricted to a small subset of nodes [32]. Therefore, by using size optimal CDSs, one can greatly decrease the overhead due to the broadcast storm problem [31] in a wireless network. The existing routing algorithms adopt different methodologies to construct virtual backbones. We first describe various network models and then in the following sections discuss various CDS construction techniques and examine their characteristics.

In a wireless network, the sensor nodes are randomly deployed in an inaccessible terrain due to which it is difficult to recharge the sensor nodes. Therefore, people are interested to design energy efficient protocols for wireless sensor networks. Generally, in a sensor network, the data are gathered in individual nodes combined in different cluster-head and then communicated to the base station or sink node. Clustering helps them to improve the lifetime [52] and scalability goals for data gathering applications. In a wireless network where a CDS is used as a virtual backbone, the dominators can be treated as cluster-heads. The cluster-heads or dominators do some extra amount of work for combining and forwarding data to the base station due to which they exhaust their battery power more often. The nodes may also fail due to other reason like hardware failure etc. Because of the failure of either a single node or a group of nodes, the entire network fails from data transmission which needs maintenance. In this scenario, the adaptive protocols identify a new set of cluster-heads for their next round [53] whereas, in the fixed clustering scheme, the new set of nodes are identified within the fixed clusters [54]. In the scenario where CDS nodes are used as a virtual backbone, to maximize the lifetime of the network large number of disjoint dominating sets [55] can be used and can be activated one after another in round wise. The problem of finding a maximum number of disjoint dominating sets is called domatic partitioning and the maximum number of disjoint dominating sets is called domatic number of a graph. Thus, the problem of rotating the responsibility of being cluster-head (or coordinator) is abstracted as the domatic partition problem. The last part of the chapter discusses the works on this problem.

The remaining of this chapter is organized as follows. In Section 2.2, we provide the network models which are used by various researchers during the CDS construction. Section 2.3, provides the classification of various CDS construction algorithms. In the next section (Section 2.4), we discuss the various types of CDS construction algorithms in detail. At the end of this chapter, in Section 2.5, we summarize the CDS construction algorithms.

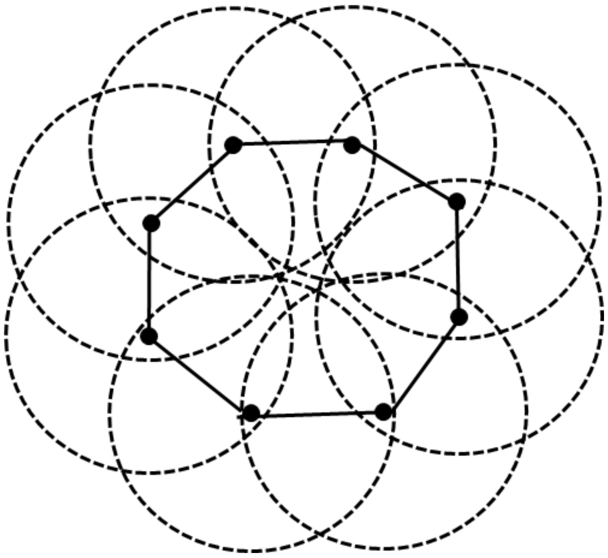


Figure 2.1: UDG Containment Model representing the topology of a wireless network

2.2 Network Models

Here, we discuss a network model which is used by most of the authors in their CDS construction algorithm. This model is the most popular network model in the community working on the virtual backbone. Consider a wireless sensor network with n nodes, each with an omnidirectional antenna of the maximum transmission range of R. We can model this network as a unit disk graph in the Euclidean plane as discussed in Section 1.1 of Chapter 1. The Unit disk graphs are nothing but the intersection of equal sized circles in the plane which represents the communication range of each node. In unit disk graph for representing the ad hoc network there are three kinds of models available:

1. *Proximity Model*: In this model, the network is represented by a graph in which each vertex represents a node. A pair of vertices are connected by an edge if the nodes corresponding the vertices are within some specified Euclidean distance bound of d.

2. *Intersection Model*: In this model also the nodes in the network form the vertices of the graph and two vertices are connected with an edge when circles formed around the nodes (which represent the transmission range

of each node) intersect. Two tangent circles are considered as intersecting circles.

3. *Containment Model*: In this model, each vertex of the graph represent a node in the wireless network. Each vertex is at the center of a disk formed around the corresponding node which represents the communication range R. Two vertices u and v are connected through an edge if their Euclidean distance is less than or equal to R. In other words, there will be an edge between u and v if the circle centered at u contains v and the circle centered at v contains u. Figure 2.1 shows the containment model of UDG. This model is used by the authors in their approaches for CDS construction to be used as a virtual backbone.

We have restricted our discussion to the Containment Model of UDG only and skipped other network models.

2.3 Classification of CDS Construction Algorithms

In the literature, many CDS construction algorithms have been proposed by many authors. In this section, we discuss three major types of CDS construction algorithms.

2.3.1 According to the use of topology information:

According to the topology information of the network the CDS construction algorithms can be divided into following two categories as shown in Figure 2.2.

1. *Centralized Algorithms*: These types of algorithms use the complete topology information present at a single node. The decision is taken at a single node where the topology information is present. These algorithms construct the CDSs of smaller size with a better approximation factor.

2. *Decentralized Algorithms*: The decision taking capability is distributed among all the nodes of the network. These algorithms are further divided into two categories.

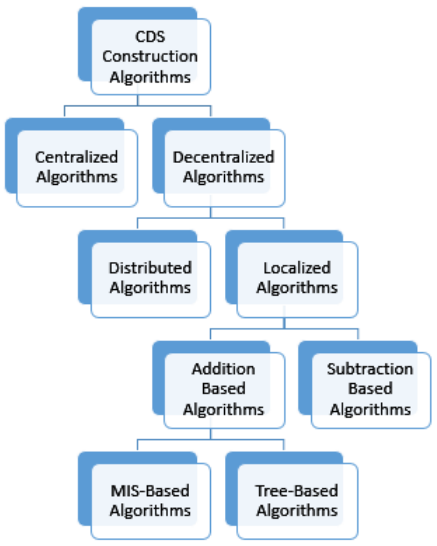


Figure 2.2: Classification of CDS construction algorithms

(a) *Distributed Algorithms*: The decision process is decentralized and the decision may be taken after a number of rounds which may depend on the size of the network.

(b) *Localized Algorithms*: The decision making process is fully distributed among the nodes in the network. The decision should be taken after a constant number of communication rounds. These algorithms are further divided into two categories.

   i. *Addition-based CDS construction*: These types of algorithms start from a subset of CDS nodes and keep on adding the nodes to make it connected to form the CDS. Addition-based algorithms further divided into the following two categories.

      A. *MIS-based CDS algorithms*: These types of algorithms works in two stages. In the first stage, they construct a maximal independent set in a distributed manner using the local information. In the later stage, these MIS nodes are connected to form the CDS.

      B. *Tree-based CDS algorithms*: These type of algorithms execution start from one or more nodes called initiators. The tree is grown from each of the initiators. These algorithms work on three phases. In the first phase the initiators are chosen, in the second phase, the trees are grown so that each node belongs to at least one tree or adjacent to a node of any tree. In the last phase, additional nodes are chosen to connect the trees. The communication overhead is less in these types of algorithms [56].

ii. *Subtraction-based CDS construction*: These types of algorithms starts with all nodes in the network and then removes some of the nodes in a systematic manner to find the CDS. Some of the algorithms of this kind are the algorithm proposed by Wu and Li [25] and Dai and Wu [57]. In general addition-based algorithms constructs CDSs of smaller size in comparison to subtraction-based algorithms.

2.3.2 According to the network models:

According to the network models CDS construction algorithms are divided into two types.

1. *Undirected Graphs*: The undirected graphs are further divided into two types.

   (a) *General Undirected Graphs*: The wireless network is modeled as a general undirected graph G where the communication range of the nodes are different. The approximation factor of the algorithms depends on the maximum degree of G.

   (b) *Unit Disk Graph*: Here, the network is modeled as a graph with nodes having the same communication range. The approximation factor is constant. The UDG model is more idealistic than practical.

2. *Directed Graphs*: Here, the transmission range of the nodes are not necessarily the same. We may have a situation where a node u is adjacent to v and v may not be adjacent to u. In this situation, the network is modeled as

a directed graph. Instead of normal CDS, a strongly connected dominating set is used as a virtual backbone.

2.3.3 According to the efficiency of the algorithms:

According to the efficiency of the algorithms in forming size optimal CDSs, message and time complexities the algorithms are classified into the following types.

1. *Global Protocols*: These protocols use the global topology information present at a central point. Although these protocols construct small sized CDSs, their maintenance cost is high.

2. *Quasi-global Protocols*: In these type of protocols, the computation for CDS construction starts from a central point and spread to the entire network. They have a small constant approximation ratio in UDG. These are not preferable in dynamic networks due to the high overhead of global infrastructure.

3. *Quasi-local Protocols*: These protocols do not consider a central point. The information is still spread over the network. Although these protocols have a large constant approximation ratio in UDG, the overhead is less as the nodes are selected in parallel to form an MIS.

4. *Local Protocols*: These protocols use only the local information. Each node changes its status by looking at the information within its k-hop neighborhood, where k is a small constant. These protocols have a constant approximation ratio.

2.4 CDS Construction Algorithms

The idea to use the CDS as a virtual backbone for routing was first proposed by Ephermides in 1987 [12]. Since then many CDS construction algorithms are reported. In the literature, one can find many CDS-related problems. Depending on the use of the CDS, the CDS construction algorithms have different design goals. Most of the CDS construction algorithms try to minimize the size of the CDS.

However, finding an MCDS in a UDG is an NP-Hard problem. Therefore, most of the authors have proposed polynomial time approximation algorithms for CDS construction. Besides the size of the CDS, the other parameter considered in CDS construction is its time and message complexities, fault tolerance, degree of localization, etc. In the first subsection, we discuss the CDS construction algorithms which focused on minimizing the CDS size. In the later subsections, we discuss other CDS construction algorithms which have different design goals. The major CDS construction algorithms found in the literature are summarized in Table 2.1.

2.4.1 CDS Construction Algorithms with the design goal of minimizing CDS size

The CDS nodes working as the backbone of the wireless network send most of the control messages. So by minimizing the size of the CDS can reduce the control messages. A minimum size CDS can also reduce the global flooding caused by the broadcast storm. Therefore, most of the CDS construction algorithms focus on reducing the CDS size. In this subsection, we discuss the CDS construction algorithms whose major design objective was to reduce the CDS size.

In 1998, Guha and Khullar [58] first proposed two centralized greedy algorithms for CDS construction in general graphs. The first algorithm initially builds a spanning tree rooted at the node with maximum degree. The tree is grown to contain all the nodes of the network. In the end, the internal nodes and the root node form the CDS. In the beginning, the colour of all the nodes in a network is white. The algorithm uses a greedy function which chooses and adds a single or a pair of nodes to the tree with maximum white neighbours. The color of the chosen node(s) are changed to black and the colour of its (their) neighbouring nodes are changed to grey. When all the white nodes are changed either to black or grey, the CDS construction is over. The approximation ratio of the algorithm is $2(H(D)+1)$, where H is a harmonic function. Their second algorithm constructs the CDS in two phases. In the first phase, a dominating set is constructed and in the second phase, these dominating sets are connected using the approximation algorithms for the Steiner Tree problem. The second algorithm has improved the approximation ratio to $H(D)+2$. In 1997, Das et al. [26] provided a centralized

CDS construction. Later on, he provided its distributed version and proposed a routing algorithm based on the constructed MCDS.

Guha's second algorithm uses a potential function which is based on the number of pieces available in the network. It selects the vertex which reduces the number of pieces more. Ruan et al. [59] improved the potential function and proposed another greedy algorithm. This algorithm consists of a single step. Here also the initial colour of each node is white. The algorithm selects either a white or a grey node such that by colouring it to black and all its neighbours to grey, the potential function is reduced to the maximum. The approximation ratio of the algorithm is $O(3 + \ln D)$.

Mostly CDS construction algorithms are distributed in nature. Wu and Li [25] proposed a distributed CDS construction algorithm in which they first construct a trivial CDS and then delete the redundant nodes based on two sets of pruning rules. The algorithm requires each node to have knowledge of its 2-hop neighbourhood. The approximation ratio of Wu and Li's algorithm as reported in [60] is $O(n)$, where n denotes the total number of nodes present in the network. The approximation ratio of distributed algorithms reported by Stojmenovic et al. in [61] is also $O(n)$, while that of Das et al. [43] is $O(\log n)$. None of these algorithms guarantee to generate a CDS of small size and also incur high message and time complexities. Most of the recent distributed CDS construction algorithms construct the CDS by first selecting an MIS and then connecting the nodes in the MIS. Note that an MIS is also a DS in an undirected graph. Wan in [62] gave an ID based two-phase single leader initiated distributed algorithm to construct a CDS tree rooted at the leader. They use a spanning tree. After the construction of the spanning tree, each node in the tree is labeled as either a dominator or a dominatee. For UDGs, the algorithm has an approximation factor of $8|opt| + 1$, time complexity of $O(n)$ and message complexity of $O(n \log n)$, where $|opt|$ being the size of an optimal CDS in the network. Adjih [63] reported a localized algorithm for CDS construction based on multipoint relays (MPR). However, no approximation analysis of that algorithm is known till now. Based on the MPR approach, several extensions have been reported, leading to localized MPR based CDS construction. However, the localized approaches provided in [63] are not highly effective without an approximation factor to ensure an upper bound on

CDS size.

The above-discussed CDS construction algorithms first construct the MIS and then add more nodes to it to make it connected. Islam et al. in [64] used a different approach to construct the CDS. In spite of constructing the MIS of the whole network, the algorithm first constructs a small connected sub-graph of the network and then finds the MIS of that sub-graph. Later on, it connects other nodes to the already constructed MIS. The approximation ratio of the algorithm is 38.

In [25], Wu and Li proposed a CDS construction algorithm which uses connectivity information of 2-hop neighbours to construct the CDS quickly. The algorithm uses a marking process in which it marks a node as true if it has two unconnected neighbours. All the true marked nodes form the CDS. They have also used some pruning rules to drop some of the nodes from the CDS to reduce the CDS size further. Based on the pruning rule given by Wu and Li, Dai and Wu proposed a general pruning rule known as Rule K [57]. This rule omits the marked nodes which are covered by other k marked nodes. Using Rule K, they reduced the CDS size further. The computational complexity and message complexity of the proposed approach are $O(\Delta^2)$ and $O(\Delta)$ respectively, where Δ is the maximum degree of the network.

Cardei et al. [65] improved the ID-based, two-phase, single leader initiated, distributed algorithm proposed by Wan et al. [62]. They improved the approximation ratio to $8|opt|$. Cardei's 2-phase distributed algorithm grows from a single leader and uses 1-hop connectivity information with degree-based heuristics and degree aware optimization for identifying Steiner nodes as connectors in the CDS construction. In the proposed method, the improvement is that the root not necessarily waits for the COMPLETE messages from the farthest nodes. The algorithm has $O(n)$ message complexity and $O(\Delta n)$ time complexity. Alzoubi's multiple leaders based localized distributed 2-phase approach [60] uses local information at each node. It first constructs an MIS by comparing node IDs within a 1-hop neighbourhood without spanning a tree or selecting a leader. If the node has the smallest ID within its 1-hop neighbourhood, it becomes a dominator. In the next phase, the MIS nodes are interconnected to form a CDS. The algorithm has an approximation factor of $192|opt| + 48$. In a similar work [66], the authors reported a distributed algorithm with a approximation ratio of 172.

Cheng et al. [67] provided a multiple leaders based CDS construction algorithm which first selects the nodes with minimum ID in their 1-hop neighbourhood as the leaders. These leader nodes become the root of different trees. Later, each adjacent pair of constructed trees are connected through one or two nodes. The approximation ratio of this algorithm is 147. In [68], Funke et al. proposed another distributed algorithm with a better approximation ratio of 6.91. This algorithm first constructs a connected set S and then finds an independent set I of the set S. The nodes are represented with different colors black, grey, blue, red, and white. The colour of the nodes belongs to I are black. The node which is in S but not in I is of colour grey. Blue coloured nodes are in S and are connected with a node of I. If a node is of colour red then it is a neighbour of either a grey node or a blue node. Otherwise, the node is white in colour.

In [69] Qayyum et al. introduced a broadcast scheme called multi-point relying (MPR). In this scheme, each host selects some set of 1-hop neighbours to cover its 2-hop neighbours. Other tree-based algorithms like Single-Initiator (SI) version [56] and Multi-Initiator (MI) version [70] start from some initiator nodes which are some subset of nodes and grow CDS trees from each of the initiators. These tree-based algorithms, work in three phases. In the first phase, the initiators are selected from the entire network. In the second phase, these initiators grow their respective trees by adding nodes which have more neighbours. In the last phase, additional nodes are added to connect the neighbouring trees.

Among all the approximation algorithms for distributed CDS construction in UDGs, Li's S-MIS algorithm [71] and Misra's collaborative cover heuristic [72] achieve the best approximation factor of $(4.8 + ln5)|opt| + 1.2$. Both these algorithms, first construct an MIS and then tap the MIS nodes through a Steiner Tree construction. The collaborative cover heuristic [72] constructs the MIS using effective coverage as a metric. However, it has a high message complexity of $O(n\Delta^2)$ and time complexity of $O(n)$. Both these algorithms use the property that any node in a UDG is adjacent to at most five nodes. In [73], Du et al. proposed a polynomial-time constant approximation algorithm that leads to a CDS with bounded CDS size and guaranteed routing cost in terms of routing path length. In the CDS constructed by their algorithm, for each pair of vertices

u and v there exists a routing path of maximum length of seven times the shortest path between u and v. The algorithm produces the CDS of size less than or equal to $148|opt| + 208$ which is very high in comparison to collaborative cover [72].

In the literature, some CDS construction algorithms with an energy model can also be found. Wu et al. [40] first proposed a CDS construction algorithm which uses an energy model. Using this idea Yuanyuan et al. [74] proposed a distributed algorithm which uses energy parameters to construct a CDS. Later on, Meghanathan [75] proposed an energy-aware CDS based data gathering algorithm for collection, fusion, and transmission of data periodically in WSN.

The design goal of all the above-discussed algorithms was to minimize the CDS size. Kim et al. [13] first investigated the problem of constructing quality CDS in terms of size, backbone path length, diameter, etc. They also proposed two centralized algorithms with constant approximation ratio. Later on Yu et al. introduced diameter is the new quality factor for CDS construction algorithms. They also proposed two algorithms based on new heuristics with constant approximation ratio for the size and diameter of the constructed CDS.

Neiberg and Hurink [76] developed a local algorithm where any vertex in the network itself decides whether or not it is part of the dominating set depending only on the vertices which are a constant number of hops away from it. Their polynomial time approximation scheme (PTAS) presents a theoretical method for computing a dominating set of a connected UDG without the geometric embedding of the graph in polynomial time with $1 + \epsilon$ approximation ($\epsilon > 0$). The processing time of a vertex is bounded by a polynomial in the number of vertices present in its locality (i.e. the radius of the area that needs to be explored). The better the approximation (smaller ϵ) required, the larger will be the locality distance. This, in turn, implies that the processing time for each vertex increases and that the network messages may propagate uncontrollably far. Furthermore, the work also includes the concept of a 2-separated collection. It emphasizes the fact that cardinality of the dominating set can be reduced if the topology is divided into local 2-separated collections, each separated from its nearest collection by two intermediate hops. Our approach develops this concept and also investigates the trade-off required between the approximation ratio and the locality that needs to be surveyed.

Table 2.1: **A comparative study of CDS Construction algorithms**

Algorithm Ref	**Classification**	**Model**	**Time and Msg Complexity**	**Approximation Ratio**
Guha et al. [58] - 1	Centralized	GG	$O(n + \|C\|\Delta)$, $O(n\|C\| + m + \lg n)$	$2(H(\Delta) + 1)$
Guha et al. [58] - 2	Centralized	GG	$O(\|C\| + (\Delta + \|C\|))$, $O(n(\|C\|))$	$H(\Delta) + 2$
Ruan et al. [59]	Centralized	GG	-	$3 + \ln \Delta$
Li et al. [71]	Centralized	GG	-	$4.8 + \ln 5$
Kim et al. [13] - 1	Centralized	UDG	-	10.395
Kim et al. [13] - 2	Centralized	UDG	-	6.906
Wan et al. [62]	Distributed	UDG	$O(n)$, $O(n \lg n)$	8
Carei et al. [65]	Distributed	UDG	$O(n)$, $O(n\Delta)$	8
Cheng et al. [67]	Distributed	UDG	$O(n)$, $O(n \lg n)$	8
Zeng et al. [**?**]	Distributed	UDG	$O(n)$, $O(n)$	7.6
Funke et al. [68]	Distributed	UDG	$O(n)$, $O(n^2)$	6.91
Alzoubi et al. [60]	Distributed	UDG	$O(n)$, $O(n)$	192
Islam et al. [64]	Distributed	UDG	$O(\Delta)$, $\Omega(\Delta)$	38
Wu & Li[25]	Localized	GG	$O(\Delta^3)$, $\Theta(m)$	$O(n)$
Dai & Wu[57] - 1	Localized	UDG	$O(\Delta^2)$, $O(\Delta^2)$	-
Dai & Wu[57] - 2	Localized	UDG	$O(\Delta)$, $O(\Delta)$	-
Zou et al. [77]	Localized	UBG	-	$13 + \ln 10$
Butenko et al. [9]	Localized	UBG	-	22
Misra et al. [72]	Distributed	UDG	$O(n)$, $O(n \triangle^2)$	$(4.8 + \ln 5)\|OPT\|$ 1.2

All the above-discussed CDS construction algorithms are 2-dimensional. That means the entire network of nodes is deployed in a 2-dimensional plane. However, we may have a situation where the nodes of the network may be present in the 3-dimensional space. Under water sensor network is one such example. Zou et al. [77] proposed an MCDS construction algorithm in a UBG. The algorithm first constructs an MIS in the UBG. Initially, all the nodes are white in colour. The root node is selected randomly and colored black. All its neighbours become grey. As long as there exists a white node in the network it is coloured as black and its neighbours become grey. Later on, the algorithm selects some other nodes to connect the nodes in the MIS. The approximation ratio of the algorithm is $13 + \ln 10$. Later, Butenko et al. [9] proposed another MCDS construction algorithm on UBG with a constant approximation ratio of 22.

2.4.2 CDS Construction Algorithms to improve fault tolerance and robustness

In wireless networks, there is a greater chance of failure of nodes. Nodes may fail due to energy depletion or external damage. In case a CDS is used as a virtual backbone and if some of the CDS nodes fail, then the virtual backbone would be of no use. If we do not provide enough fault-tolerance and robustness to the dominators then we have to reconstruct a fresh CDS. In case of failure of backbone nodes, k-connected m-dominating sets are really useful for their extended lifetime.

In 2000, Amis et al. [78] introduced the concept of k-dominating set problem. They proved that k-dominating set is NP-complete and also proposed a distributed cluster-based algorithm with constant time complexity. Later, Dai et al. [79] proposed three localized algorithms on the same problem, out of that one algorithm is deterministic and the other two algorithms are probabilistic. In 2008, Wu et al. [80] proposed a new distributed algorithm to construct k-m-CDS with a low message overhead, where k and m are arbitrary integers. The approximation ratio of the proposed approach is constant for the networks with constant maximum node degree.

In [81], Thai et al. proposed two approximation algorithms for 1-m-CDS and k-k-CDS problems. Later, they developed an algorithm for the general k-m-CDS problem by combining the ideas of the algorithms for the above two problems. In [82], Zhang et al. proposed an algorithm for minimum 3-connected m-dominating set ($m \geq 3$) problem and in [83] Wang et al. constructed a minimum 4-connected m-dominating set in UDG. In [84] Kim et al. proposed an algorithm for constructing (3, m)-CDS. The algorithm is a polynomial time algorithm and the approximation ratio is also constant.

To improve the fault tolerance of the virtual backbone, Wang et al. [85] proposed a 2-connected CDS construction algorithm. They also proved that $|MCDS| \leq |opt| - 2$, where *opt* is the minimum 2-CDS. The algorithm first constructs a CDS using any CDS construction algorithm. Then all the CDS blocks are computed. Then, the shortest path in the original graph meeting the requirements is found out. Finally, the nodes in the shortest path are added to the CDS.

2.4.3 CDS Construction Algorithms to prolong the network lifetime

The major limitation of the wireless node is its battery as it is not easy to recharge the battery of a node once deployed. In order to extend the lifetime of the network, we need to save the energy of each node as much as possible. We can use domatic partition for this purpose. In this sub-section, we discuss some of the major works in this field.

In [86], Slijepcevic et al. proposed a novel energy saving scheme which selects and activates a mutually exclusive set of nodes. Each set covers the entire network to be monitored. In [37], a randomly distributed algorithm is proposed which takes the decision locally to determine whether a node would be in a sleeping state or join the DS. Each node in the network becomes a dominator rotation wise.

Cardei et al. [87] proposed another two-phased efficient algorithm to save energy. The algorithm organizes the nodes of the network into a maximum number of disjoint sets. These disjoint sets are activated one after another. In the first phase of this algorithm, all the nodes are coloured using a sequential colouring algorithm. Based on the colouring, a heuristic is used to construct disjoint dominating sets. Misra et al. [45] first introduced the connected dominating partition problem. It divides the entire network of nodes into a number of disjoint CDSs. These CDSs are rotated to reduce the energy consumption which increases the network lifetime. The proposed algorithm first divides the entire network into a number of clusters and find the level of each cluster approximately. The level of nodes is compared to generate disjoint CDS trees. The main advantage of this algorithm is that it maximizes the size of the CDP and the rotation of CDSs are through a local switching operation.

Shi et al. [88] defined a unique problem named as Energy Harvest CDS (EH-CDS) to discover the maximum number of CDSs in a static network which takes the advantages of rechargeable nodes to become energy harvest networks.

2.5 Summary of CDS Construction Algorithms

The CDS construction algorithms found in the literature are summarized as follows:

- The workload of the CDS nodes are much more than normal nodes in a wireless network. Therefore, by minimizing the size of the CDS can greatly reduce the control messages. This would extend the battery power of CDS nodes and hence indirectly extend the virtual backbone lifetime. However, the construction of minimum CDS is an NP-Complete problem. Therefore, researchers are interested to design approximated MCDS construction algorithms of lower approximation ratio in polynomial time.
- Some CDS construction algorithms also considered the diameter of the network, average backbone path length, etc. while designing the CDS construction algorithms.
- In wireless network node failure is not rare. Therefore, the CDSs should provide fault-tolerance and robustness to the dominators. The algorithms provide fault-tolerance and robustness to the dominators by constructing k-m-CDSs or r-k-DSs.
- Wireless nodes are battery powered and it is not easy to recharge them as and when required. To prolong the battery power, we can rotate the role of dominators in a wireless network. To achieve this, algorithms for Domatic Partition problem were proposed by the researchers.

2.6 Scope of Work

To use CDS as a virtual backbone in a wireless network (especially in WSN), the CDS size should be as small as possible. This would help in reducing the generation of control messages by the CDS nodes and also extend the battery power of the CDS nodes. Although many authors have worked on this problem to construct CDS of smaller size, we felt that the CDS size can further be reduced. To take a decision about a node (whether to include it in the DS or not), we use its one-hop information. For a better approximation, the information of larger locality may be used. However, to get the information of a larger locality we need more processing time and propagation of more number of messages. Therefore, there is a definite scope of investigation of the trade-off required between the approximation ratio and the locality that needs to be surveyed. Many of the algorithms first construct the MIS (which is a DS) and later connect the MIS nodes using some connectors (Known as Seiner nodes) to form a CDS. There is always a chance of dropping some of the dominators from the CDS if the dominatees dominated by the dominators are now dominated by some of the selected Steiner nodes. Before downgrading some of the dominators to dominatees, we should ensure the connectivity between the remaining dominators.

Centralized algorithms provide more accurate information. However, they are not applicable to networks of larger sizes. In the other hand, although distributed algorithms are difficult to design, they need local information exchange only. So the use of distributed algorithms would be more feasible. Hence, we should design distributed CDS construction algorithms to reduce CDS size further.

In the literature, most of the works are on CDS construction. In case of failure of even a single CDS node, either we need to reconstruct another CDS or replace the failed CDS node with some non-CDS nodes. We find some works on CDS rotation among some pre-computed CDSs. However, there is no work reported on CDS maintenance. We should look for CDS maintenance algorithms for the failure of CDS nodes either due to battery or due to any other reason like hardware failure etc.

CHAPTER 3

Centralized Construction of CDS

3.1 Overview

In a wireless ad hoc and sensor network, unlike wired networks or cellular networks, no physical backbone infrastructure is required, thus offering new paradigms for routing. Each node in the network broadcasts messages to all the nodes within its transmission range. Therefore, through broadcasting, a node can reach all of its nearby nodes with one emission. If communicating nodes are not within the single hop radio transmission range of each other, then a communicating session is established through multi-hop links by some intermediate nodes for relaying messages (multi-hop routing). One simple and intuitive method for multi-hop routing between non-adjacent nodes in wireless networks is flooding, in which each node retransmits a packet only once after receiving it. However, owing to the low available bandwidth of the wireless channels and the redundant retransmissions generated through pure flooding, the latter is not used as a communication mechanism in wireless networks. The most popular means of multi-hop routing in wireless networks is through the use of a virtual backbone. In this chapter, we mainly focus on constructing a minimum size Connected Dominating Set (CDS) as a virtual backbone of the network. A CDS is more particularly used as a data aggregation backbone in remote data gathering applications to optimize network communication, which in turn saves communication energy and extends the network lifetime [89].

A Dominating Set of a network is a subset of nodes such that any node not in the subset is a neighbour of some element of that subset. It forms a Connected Dominating Set if the sub-graph induced by this set is connected. In a wireless network, as there is no fixed infrastructure and centralized management, a CDS can be used as a virtual backbone or spine for efficient routing and connectivity management in such networks [43]. The CDS can receive a packet from any node in the network and can retransmit it to any other remote node. A node, which is not in the CDS can send a message to any other node through the CDS nodes. It first sends its message to one of its neighbouring CDS nodes. Now, the search space for any route is reduced to the CDS. If the destination node is within the CDS it can get the message directly, otherwise, it gets the message from one of its neighbouring CDS nodes. Thus, during routing, broadcasting responsibility lies only with the CDS nodes, instead of all the nodes in the network. As only the CDS nodes maintain routing information, we can save the storage space by reducing the CDS size. A small sized CDS makes the routing easier, reduces the communication overhead, increases the convergence speed and simplifies connectivity management. So, it is desirable to construct a minimum connected dominating set (MCDS) of the network. However, computing MCDS is an NP-complete problem [44]. So, only polynomial time approximation algorithms are practical for finding out MCDS in wireless networks. For energy-constrained wireless networks, an approximation algorithm should not only construct thinner CDSs but also construct CDSs with low computation and communication costs. Generally, the quality of the CDS is measured by its approximation ratio, which is the ratio of its size to that of the MCDS. In a centralized CDS construction algorithm, the construction cost is its time complexity. The computation time of the CDS should also be appreciably small in order to schedule speedy switches between disjoint CDSs to extend battery life and optimize power consumption [45, 46].

Although theoretically any centralized algorithm can be implemented in a distributed fashion, with the trade-off of higher protocol overhead, distributed algorithms are very important for sensor networks and MANETs. CDS must be constructed efficiently to be applicable in a mobile or large scale network. Due to the dynamism of wireless links and nodal mobility, algorithms should rely on limited knowledge of the current network topology. However, for applications where node mobility is rare, centralized CDS construction algorithms are very useful. In those applications, once the topology of the entire network is known to a particular node (maybe the sink node), it can construct the CDS without waiting for the information from the other nodes in every step.

In this chapter, we propose a new centralized degree-based greedy approximation algorithm which we name as **C**onnected **P**seudo **D**ominating **S**et Using **2**-**H**op **I**nformation (CPDS2HI) to construct smaller CDSs. The proposed scheme CPDS2HI works in three phases. In the first phase, a smaller maximal independent set (MIS), designated as a pseudo-dominating set (PDS) is constructed. The dominating set is pseudo dominating set because some of the elements may be omitted in the final dominating set. The second phase of our algorithm constructs an improved Steiner Tree which interconnects the PDS nodes in an improved way. In the final phase, some of the selected PDS nodes are excluded cleverly to reduce the CDS size further without any connectivity or coverage loss. Through simulation, we also show that our proposed algorithm CPDS2HI outperforms all the existing CDS construction algorithms in terms of CDS size and construction costs. CPDS2HI retains the current best performance ratio of $(4.8 + \ln 5)|opt| + 1.2$, $|opt|$ being the size of an optimal CDS of the network, and has the best time complexity of $O(D)$, where D is the network diameter. To the best of our knowledge, this is the most time efficient and size-optimal CDS centralized construction algorithm.

The rest of the chapter is organized as follows. Section 3.2 discusses the motivation behind this work and state its objective. In Section 3.3, we state our assumptions regarding the development of ad hoc network model. Section 3.4 explains our algorithm in detail. Section 3.5 provides an analysis of our algorithm. Supporting simulation results are given in Section 3.6. Our conclusions are finally presented in Section 3.7.

3.2 Motivations and Objectives

3.2.1 Motivations

The recent competitive distributed algorithms in [62, 65, 71, 72], to achieve constant satisfactory approximation ratio, construct MISs with a specific property stated in [71]. The characteristic property of such MISs is that *any pair of complementary subsets of the MIS must have a distance of exactly two hops between them*. This underlying property assists in interconnecting the MIS nodes in the next phase. However, intuitively in an MIS, a node can be separated from its nearest node by at most three hops. These MISs with lower cardinalities can effectively reduce the CDS size further and improve the ratio of number of connectors to number of independent nodes. The ratio has a significant impact on the lifetime of the network. We cite an example to demonstrate this point. For the graph shown in Fig. 3.1, these MIS selection schemes would select an MIS of size 5 comprising nodes 1, 3, 5, 8 and 10. Each MIS node is separated from its nearest neighbour in the MIS by exactly two hops. However, nodes 1, 4 and 9 alone can also form an MIS of the same graph. In the latter case, each MIS node is separated from its nearest MIS neighbour by three hops. However, to construct such MISs, at least 2-hop neighbourhood connectivity information of each node is required. Furthermore, Steiner Tree construction [71], [90] in the post-MIS selection phase will also incur a higher number of message exchanges. Consequently, the communication overhead will be higher.

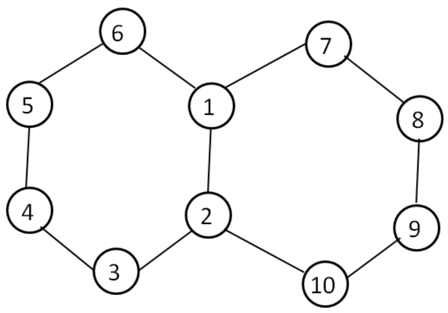


Figure 3.1: Network to illustrate an alternative MIS construction technique

We also note that in post Steiner Tree construction, some dominators from the

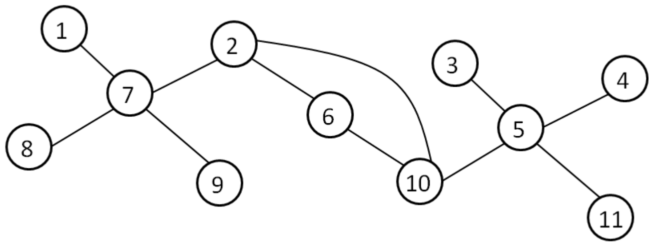


Figure 3.2: Network to illustrate how PDS can improve CDS size over MIS

MIS can be downgraded to dominatees without any loss in connectivity or coverage of the CDS. This is intuitively true when the neighbours (if any) of the MIS node to be downgraded are covered either by some Steiner nodes or by some other MIS nodes. We illustrate this through an example. In the graph given in Fig. 3.2 the MIS with minimum size consists of nodes 5, 6 and 7. We will essentially require nodes 2 and 10 to connect the MIS nodes. Thus, nodes 7, 2, 6, 10 and 5 form the CDS. But after CDS construction, node 6 in the CDS becomes redundant. Nodes 7, 2, 10 and 5 alone can also form a CDS. Thus, ideally, an MIS should be treated as a pseudo-dominating set (PDS), since post-CDS construction some of the MIS nodes can be removed from the CDS to reduce CDS size further. We are motivated towards dealing with the issues cited above to reduce the CDS size further at an optimal trade-off in the number of messages exchanged.

3.2.2 Objectives

Centralized Construction of CDS in a static Wireless Network: Consider a wireless network consisting of n number of nodes deployed in a geographical region. All the nodes are static. Each node is mounted with an omnidirectional antenna with the transceivers having the same maximum transmission range of R. The ad hoc network is a unit disk graph $G = (V, E)$ where $|V| = n$ be all the nodes, E be the edges and edge between any pair of node exists if the distance is at most R, taken as a unit radius. The problem is to find a minimum cardinality CDS of G. Since this is an NP-complete problem, our aim is to develop a heuristic based approximation algorithm to construct minimum size CDS. Assume that the topology information of the entire network is available at a single node where the centralized algorithm would be executed. Our objective is to design a centralized

CDS construction algorithm which could contribute towards improving the CDS size further than previous approximation algorithms. In the final CDS, we should not have any redundant dominators.

3.3 Network Model for Centralized CDS Construction

The assumptions regarding the development of ad hoc network model for the above-mentioned objective are as follows:

- Each node has a unique ID.
- All the hosts are deployed in a 2-dimensional plane and their maximum transmission ranges are the same.
- The nodes do not have any topological information. They do not even have knowledge of their distances to their neighbours.
- Two nodes are adjacent to each other if both are present within the communication range of the other.
- The number of nodes present within the communication range of a node is known as its *degree*.
- The topology information of the entire network is available at a single node where the centralized CDS construction algorithm would run.
- The centralized CDS construction algorithm has the information of *1-hop* and *2-hop*two neighbours of each node with their degree.

3.4 CDS Construction by CPDS2HI

Our CDS construction scheme CPDS2HI works in the following three phases:

A. Pseudo-dominating set construction

B. Improved Steiner tree construction

C. Removal of redundant dominators

3.4.1 PDS Construction

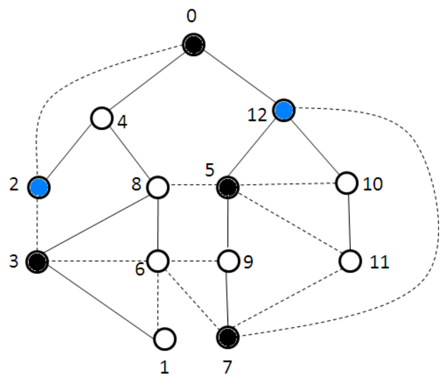
In this phase of our proposed algorithm, we greedily construct a PDS as an MIS with lower cardinality as compared to other MIS algorithms. Any pair of complementary subsets of our PDS can have a distance of either two or three hops. Some of the nodes in the PDS may not be included in the CDS after the second phase depending on the coverage of the connectors connecting them to the rest of the CDS. We construct the PDS through a simple degree-based algorithm. The algorithm uses 1-hop and 2-hop neighbours information of each node. As an MIS node can be separated from its nearest MIS nodes by at most three hops, it is sufficient for a node to examine only its 1-hop and 2-hop neighbours during this phase. In each round of the algorithm, several nodes may get elected as dominators from their respective local 2-hop neighbourhood pockets; this, in turn, yields an excellent linear time complexity. Thus, the first phase of our approach for an optimal CDS construction maintains overall low time complexity retaining the basic characteristic of degree-based algorithms and at the same time successfully reduces the CDS size by reducing the cardinality of MIS. Our greedy algorithm, for determining the PDS of a graph is given in Algorithm 1. The algorithm starts with all nodes coloured white and produces sets of dominators (D) coloured black and virtual-dominators (VD) coloured grey. The PDS is the union of black and grey nodes. Some white nodes are temporarily coloured yellow during the process to indicate that they have been removed from consideration in identifying the dominators and virtual dominators.

Next, we compare our PDS selection approach with a popular MIS construction algorithm reported in [62] for the same example cited there. Their dominating tree construction mechanism forms an MIS of size 4, as shown by the black nodes in Fig. 3.3a. The black nodes in Fig. 3.3b presents the PDS constructed from our first phase. In the figure, the blue nodes are coloured later and should be regarded as white at this stage. The cardinality of our PDS is 3. In our PDS, first node 5 is selected as a dominator over all its rivals in its 1-hop and 2-hop neighbourhood

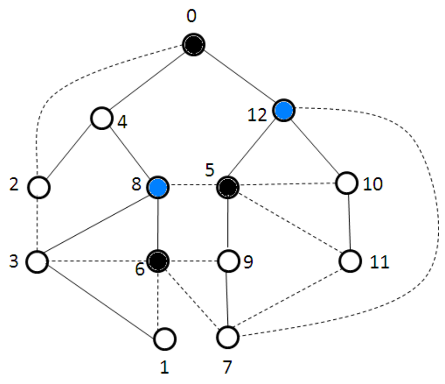
Algorithm 1 Construction of PDS of a graph

Input: A connected graph $G(V, E)$ in which the colour of each node is white.
Output: PDS of the graph $G(V, E)$ formed by black and grey nodes.

```
1:  $PDS \leftarrow \phi$, $D \leftarrow \phi$, $VD \leftarrow \phi$, $i \leftarrow 1$
2:  while there is a white node in $G$ with $degree > 0$ do
3:      $Round_i \leftarrow \phi$
4:      for all white nodes $u \in V$ do
5:          if $u$ satisfies any one of the following conditions:
              (i) $u$ has a degree strictly higher than each of its 1-hop and 2-hop white
                  neighbours.
             (ii) Each of the white nodes in the 1-hop and 2-hop neighbourhood of $u$ has a
                  degree lower than or same as that of $u$, but $u$ has a higher original degree
                  than its white 1-hop and 2-hop neighbours with the same current degree
                  as that of $u$.
            (iii) None of the white nodes in the 1-hop and 2-hop neighbourhood of $u$ has
                  a degree higher than that of $u$ and there exists 1-hop or 2-hop white
                  neighbour(s) of $u$ that has the same current and original degrees as that
                  of $u$, but $u$ has the least node ID among them.
            then
6:              Add $u$ to $Round_i$      ▷ $u$ is selected as a dominator in the $i^{th}$ round
7:          end if
8:      end for
            ▷ $Round_i$ contains dominator(s) selected at the $i^{th}$ round where no two
        dominators are within a 2-hop neighbourhood of each another
9:      for all white nodes $v \in Round_i$ do
10:         Change colour of v from white to black
11:         $E \leftarrow E - ((N_1(v) \times V) \cup (V \times N_1(v)))$
12:         $V \leftarrow V - N_1(v)$      ▷ Delete all the adjacent nodes of $v$ by changing its
        colour to yellow and edges incident on them from $G$
13:         Update degree of the remaining white nodes in $G$
14:     end for
15:     $D \leftarrow D \cup Round_i$
16:     $i \leftarrow i + 1$
17: end while
18: $VD \leftarrow V - D$      ▷ Each of the undeleted white nodes (with degree 0) are
    considered as virtual-dominators
19: All the nodes in $VD$ are marked grey
20: $PDS \leftarrow D \cup VD$      ▷ Dominators & Virtual-Dominators both form the PDS
```



(a) CDS formed by dominating tree construction



(b) CDS formed by CPDS2HI

Figure 3.3: Example showing CDS Construction

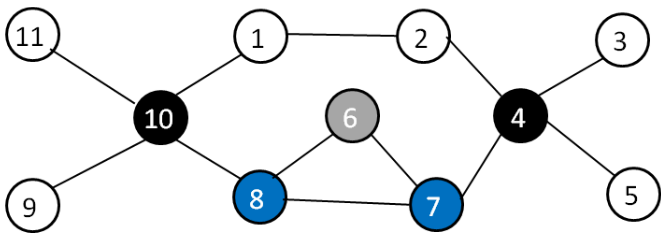


Figure 3.4: CDS formed by the black dominators and blue connectors selectively discarding grey virtual-dominators

as node 5 has a higher initial degree and a lower node ID (when compared with node 6 which also has the same initial degree). In the next turn, node 6, with an effective degree of 3, is chosen as a dominator. Although node 3 has the same degree as that of node 6 in this turn, it lost to node 6 as the latter has a higher original degree. In the next round, 0 is selected as the dominator.

We illustrate the PDS selection process with one more example. For a distribution of nodes shown in Fig. 3.4, nodes 4, 6 and 10 form the PDS. In the first round, both nodes 4 and 10 are selected as dominators. In the next round, the remaining white node is 6 with an effective degree of zero. Hence, node 6 is chosen as a virtual-dominator and its colour is changed to grey. The recent collaborative cover heuristic [72], which produces smaller MISs than previous MIS selection techniques [62], [65] and gives marginally better bound when the distribution of nodes is uniform, yields an MIS size of 3 or 5 for the same graph in Fig. 3.4 depending on whether node 6 is selected as the initiator or not. Selecting a leader to initiate the MIS construction can also add to the time and message complexities. The best distributed leader-election algorithm takes time $O(n)$ and message $O(n \log n)$ [91]. In addition, the border effect that arises when dealing with the border nodes such as 3, 5, 9 and 11 is also responsible for a 67% increment in MIS size as compared to our PDS size. Our scheme does not suffer from leader selection problems or border effects.

3.4.2 Improved Steiner Tree Construction

In this phase, we tap all the dominators and virtual-dominators in the PDS by selecting Steiner nodes from the dominatees greedily to construct a Steiner Tree spanning all the nodes in the PDS. For a graph $G(V, E)$, our objective is to find a Steiner Tree with a minimum number of Steiner nodes from $\{V - PDS\}$, thereby reducing the CDS size. At the beginning of this phase, each of the dominators and virtual-dominators forms separate components. The dominatees are selected as Steiner node based on their connection with separate components. Based on this idea, we formulate our approach for the second phase in Algorithm 2. The algorithm starts with a mixture of black (dominator), grey (virtual-dominator), and white (other) nodes. It proceeds to convert some white nodes to blue (Steiner connector) nodes. The resulting CDS consists of the sub-graph formed by the union of the grey, black, and blue nodes.

Post-PDS construction in the network shown in Fig. 3.4, only nodes 7 and 8 are adjacent to two independent components. All the remaining dominatees are adjacent to exactly one component (either dominator 4 or 10). Node 7 has a *connection-load* of 2 higher than that of dominatees 2, 3 and 5 and a lower node-id than dominatee 8 sharing the same *connection-load* and nodal degree. So, in the first round of the improved Steiner Tree construction, node 7 is chosen as a Steiner node over rival dominatees 2, 3, 5 (adjacent to dominator 4) and 8 (adjacent to virtual-dominator 6) to tap dominator 4 and virtual-dominator 6. In the subsequent round, only dominatee 8 connects two separate components (dominator 10 and component 6-7-4). Therefore, in the next round, node 8 is

Algorithm 2 Improved Steiner Tree Construction

```
Input: A connected graph $G(V, E)$ with its PDS formed by black and grey nodes.
Output: CDS of the graph $G(V, E)$ formed by black, grey and blue nodes.
 1: All (black) dominators and (grey) virtual-dominators form separate components (as isolated vertices).
 2: Dominatees which are adjacent to the same component are rival dominatees.
 3: for all white dominatees $u \in V - PDS$ do
 4:     connection-load of $u \leftarrow$ Number of independent components adjacent to $u$.
 5: end for
 6: $i \leftarrow 1$
 7: repeat
 8:     $Round_i \leftarrow \phi$
 9:     for all white dominatees $u \in V - PDS$ do
10:         if $u$ satisfies any one of the following three conditions:
              (i) Vertex $u$ has a connection-load strictly higher than each of its rival dominatees.
             (ii) The white rival dominatees of $u$ has a connection-load lower than or same
                  as that of $u$, but $u$ has higher nodal degree than the rival dominatees with
                  the same connection-load.
            (iii) None of the white rival dominatees of $u$ has a connection-load higher than
                  that of $u$ and there exists rival dominatees that have the same connection-
                  load and nodal degree as that of $u$, but $u$ has the least node ID among them.
          then
11:             Add $u$ to $Round_i$  ▷ $u$ is selected as a connector from its white rival dominatees in the $i^{th}$ round
12:         end if
13:     end for  ▷ $Round_i$ contains connector(s) selected as Steiner nodes at the $i^{th}$ round
14:     for all connectors $v \in Round_i$ do
15:         Make connector $v$ and the separate components, that $v$ connects, a single component.
16:         Change color of $v$ from white to blue
17:     end for
18:     $i \leftarrow i + 1$
19:     Update the connection-load of each of the remaining white nodes (if any).
20: until all dominators and virtual-dominators in PDS are in the same component
21: $CDS \leftarrow$ connected component formed by black, grey and blue nodes.  ▷ PDS nodes are joined to form a single component.
```

selected as a connector to connect dominator 10 to the component 6-7-4. So, we have the component 10-8-6-7-4. For the network shown in Fig. 3.3b, CPDS2HI will choose node 8 as the first Steiner node followed by the selection of node 12 as a connector to produce a CDS size of 5; whereas the dominating tree construction [62] produces a CDS of size 6 as shown in Fig. 3.3a.

3.4.3 Removal of redundant dominators

This phase of CPDS2HI helps to reduce the CDS size as much as possible. In this phase, we downgrade all the redundant (black) dominators and (grey) virtual-dominators to (white) dominatees. The remaining virtual-dominators are upgraded to (black) dominators. A dominating node is redundant if by removing it from the CDS, the resultant CDS is still connected and covers all the nodes of the network. We know already a virtual-dominator does not cover any uncovered node, so we can omit virtual-dominators from the CDS if its removal does not disconnect the CDS nodes. We can also omit a dominator from the CDS if the dominatees covered by it can be covered by some other dominators/connectors and its removal does not disconnect the CDS nodes. The CDS after discarding the redundant dominator(s) / virtual-dominator(s) still covers the entire network, as the downgraded nodes are covered by their adjacent connectors. Based on this, we formulate our approach for the third phase in Algorithm 3. At the end of this phase, the CDS is the sub-graph formed by the union of the remaining black and blue nodes. All the remaining nodes are white.

After getting the initial CDS shown in Fig. 3.4, we can reduce the size of it by downgrading the virtual-dominator 6 as it is connected to two connectors 7 & 8 which are adjacent. Therefore, node 6 is discarded from the CDS. The nodes 4, 7, 8 and 10 form a CDS of size 4 even without virtual-dominator 6. The CDS size obtained from collaborative cover heuristic [72] for the same graph shown in Fig. 3.4 will be 5 or 7 depending on the choice of the leader to initiate the construction.

Algorithm 3 Removal of redundant dominators

```
Input: A connected graph $G(V, E)$ with its CDS formed by black, grey and blue nodes.
Output: A probable smaller CDS of the graph $G(V, E)$ formed by black and blue nodes.
 1: for all (grey) virtual-dominators $w \in VD$ do
 2:     if w is connected to the CDS by one connector or by exactly two connectors and they are adjacent to each other then
 3:         $CDS \leftarrow CDS - \{w\}$                    ▷ Omit virtual-dominator
 4:         Change the colour of w from grey to white.
 5:     else
 6:         Change the color of w from grey to black.
 7:     end if
 8: end for
 9: for all (black) dominators $w \in D$ do
10:     if All the dominatees of w are connected to some other dominators/connectors then
11:         if w is connected to the CDS by one connector or by exactly two connectors that are adjacent to each other then
12:             $CDS \leftarrow CDS - \{w\}$                ▷ Omit dominator
13:             Change the colour of w from black to white.
14:         end if
15:     end if
16: end for
```

3.4.4 Working Example

In this subsection, we illustrate the complete working procedure of our centralized algorithm CPDS2HI through two examples.

Example 1:

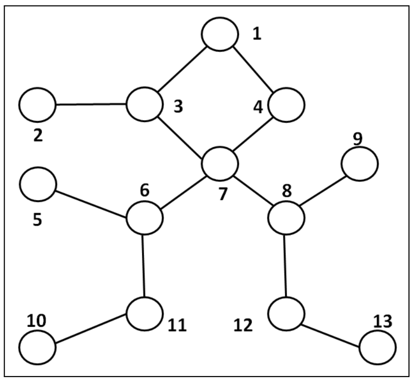
We illustrate the PDS construction process with a distribution of nodes shown in Fig. 3.5a, in which the nodes 1, 2, 5, 7, 9, 11 and 12 form the PDS. Fig. 3.5b - 3.5f shows the construction of the PDS in stepwise. In the first round, node 7 is selected as dominator as it is having highest degree among all its 1-hop neighbours (3, 4, 6, 8) and 2-hop neighbours (1, 2, 5, 9, 11, 12). Nodes 10 and 13 are not selected as dominators because they have degree lesser than their 1-hop neighbours 11 and 12 respectively. After the selection of node 7 as dominator, all its 1-hop neighbours with their incident edges are removed from the network. In the next round, among the nodes with an effective degree greater than zero (10, 11, 12, 13), node 11 and 12 are selected as dominators. Although node 11 has an effective degree the same as that of node 10, node 11 is selected because of its higher original degree. Similarly, node 12 is chosen over 13. After the selection of dominator (7, 11, 12), the remaining white nodes 1, 2, 5, 9 are with effective degree zero. So, these nodes are selected as virtual dominators and are coloured grey.

We illustrate the Steiner Tree construction phase through the PDS computed in Phase 1 of this algorithm. Post PDS construction is shown in Fig. 3.6. Fig. 3.6a shows the PDS in which we find the nodes 7, 11, 12 are dominators and nodes 1, 2, 5, 9 are virtual-dominators. The remaining nodes 3, 4, 6, 8, 10, 13 are dominatees. The dominators and virtual-dominators form individual components. The *connection-load* of the dominatees 3, 4, 6, 8, 10, 13 are 3, 2, 3, 3, 1, 1 respectively. In the first round, node 3 is chosen as the connector although its rival dominatees 6 and 8 have same connection-load (3) and degree (3), however, node 3 is with least node-ID. Node 3 forms a new component 1-2-3-7. Now, the connection-load of the remaining dominatees 4, 6, 8, 10 becomes 1, 3, 3, 1, 1 respectively. So, among nodes 6 and 8, node 6 is chosen as the connector in the second round since it has smaller node-ID (connection-load and degree are the same as node 8). Node 6 forms a new component 1-2-3-6-7-11. In a similar way, in the next round, node 8 is chosen as the connector and forms a single component consisting of all the dominators and virtual-dominators.

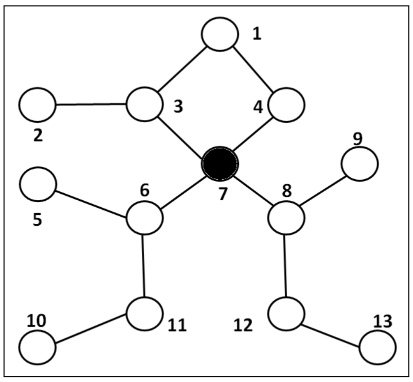
After getting the initial CDS as shown in Fig. 3.6d, we can reduce the size of it by downgrading the virtual-dominators 1, 2, 5 and 9 as they are connected to the CDS by one connector. The final CDS is shown in Fig. 3.7

Example 2:

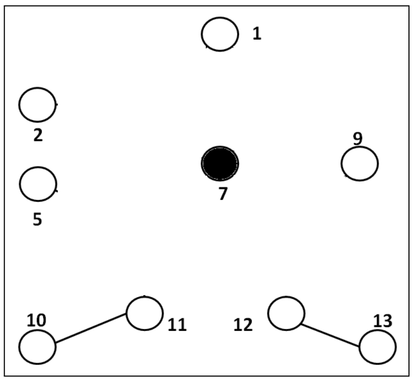
Consider a network shown in Fig. 3.8a. The nodes in the network are randomly positioned. An edge between a pair of a node indicates that the nodes are within their communication range. Assume that the colour of each node is white, each node's 1-hop and 2-hop neighbours information is known. The stepwise construction of PDS is shown in the Fig. 3.8b- 3.8e. In the first round of this phase, node 1 becomes the dominator. Nodes coloured yellow were neighbours of a dominator and have been disconnected (removed from consideration) in identifying further dominators. In the second round node 5, 19 and in the third round node 4, 15



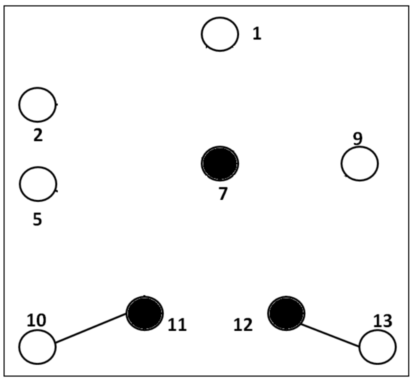
(a) Initial Network



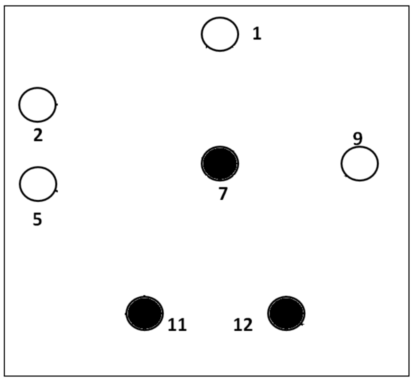
(b) Node 7 becomes dominator



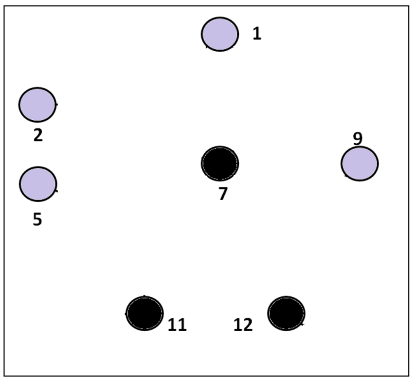
(c) Adjacent nodes of node 7 with their edges removed



(d) Node 11, 12 become dominators

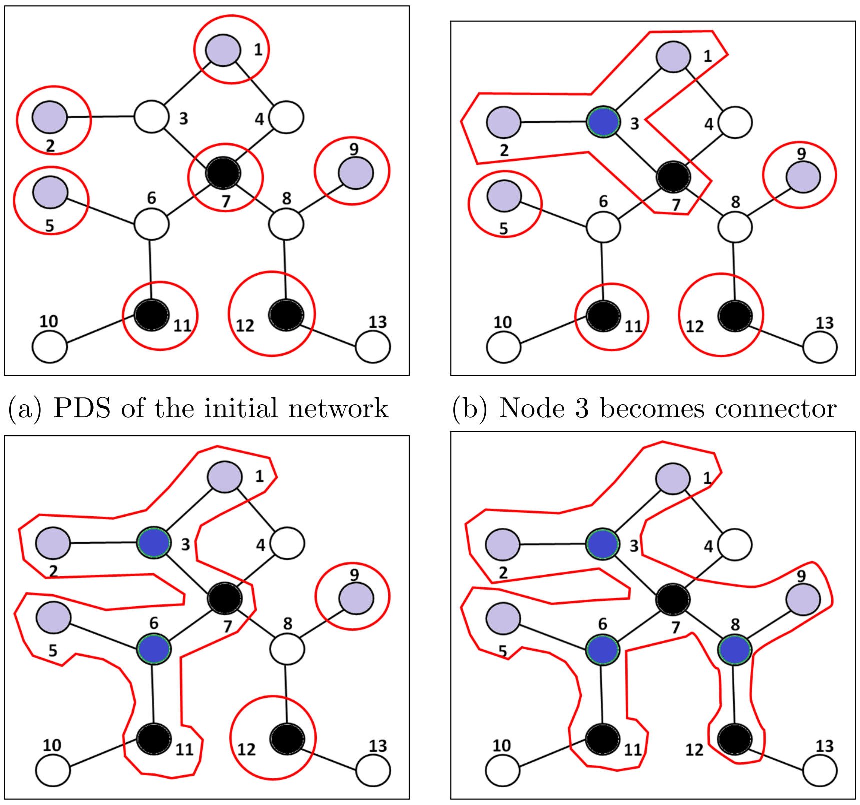


(e) Adjacent nodes of node 11, 12 with their edges removed



(f) Nodes 1, 2, 5, 9 becomes virtual dominator

Figure 3.5: Example showing PDS Construction (Phase 1 of CPDSTHI)



(a) PDS of the initial network (b) Node 3 becomes connector

(c) Node 6 becomes connector (d) Node 8 becomes connector

Figure 3.6: Example showing Steiner Tree construction (Phase 2 of CPDSTHI)

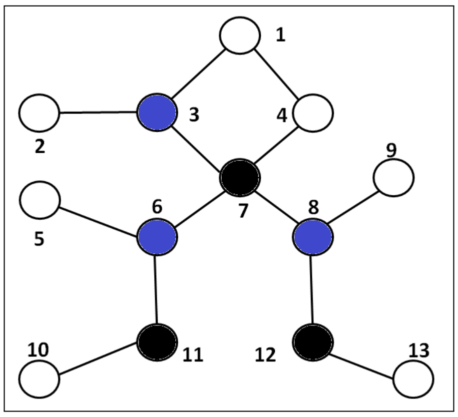


Figure 3.7: Final CDS after removing redundant nodes

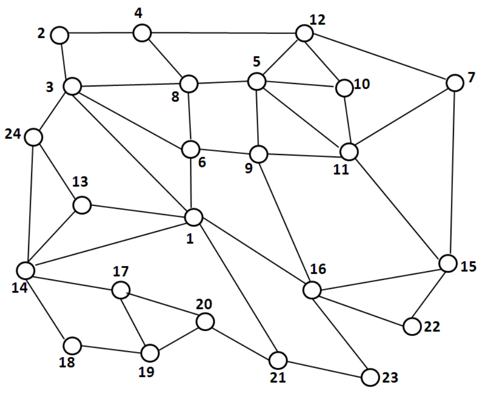
become dominators. At the end nod, 23 and 24 become virtual-dominators. The stepwise construction of Steiner Tree is shown in Fig. 3.8f- 3.8k. At the start of this phase, the dominators and virtual-dominators construct the isolated components (marked by the red line) shown in Fig. 3.8f. In the first round of this phase, the dominatees 16 and 17 becomes the connectors. They form new components by merging some earlier components. In the subsequent rounds node 14, 11 and 8 become connectors. The CDS after this phase is shown in Fig. 3.8k. In the last phase of CPDS2HI the redundant dominating nodes 19, 23 and 24 are removed from the CDS. The virtual-dominators 23 and 24 are connected to the CDS by one connection. Similarly, the dominator 19 is connected to the CDS by one connector and its dominatees are adjacent to some other connectors. So these redundant dominating nodes are downgraded to dominatees. The final CDS is shown in Fig. 3.8l.

3.5 Algorithm Analysis

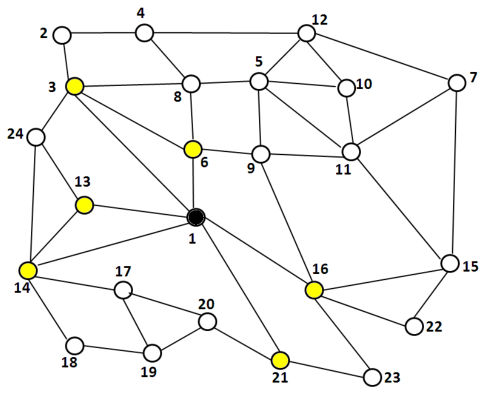
In this section, we find the approximation ratio of our proposed algorithm with its time complexity.

Lemma 3.1 *All black and grey nodes resulting from PDS construction (steps of Algorithm 1) forms an MIS.*

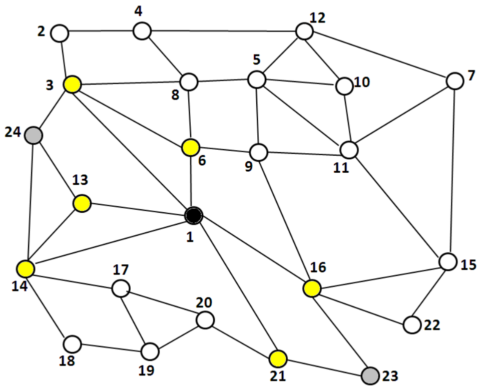
Proof. *The while loop present in step 2 of the Algorithm 1 would continue as long as there is a white node present in the graph with degree greater than zero. When a node is selected as dominator, its colour is changed to black and all its adjacent nodes are deleted. At the end of the while loop, each node is either selected as a dominator (black in colour) or deleted (as it is adjacent to some dominator) or its degree has become zero. The white nodes with degree zero are changed to colour grey and also included in the PDS. In the PDS, the colour of each node is either black or grey. The neighbours of black nodes (yellow nodes) are not there in the PDS and the grey coloured nodes are not the neighbour of any black coloured nodes (in that case we would have deleted it before). Hence, the PDS is independent. The PDS is maximal because we can't add any more nodes to the list because the other nodes (yellow nodes) are dominated by black nodes.* ■



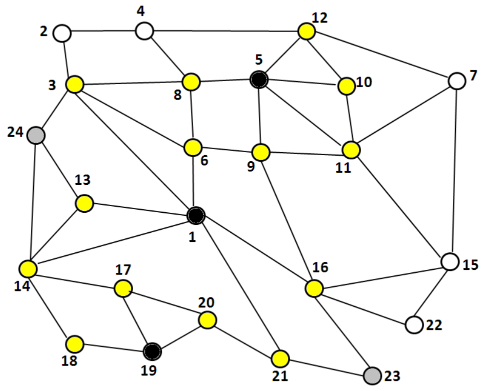
(a) Initial Network



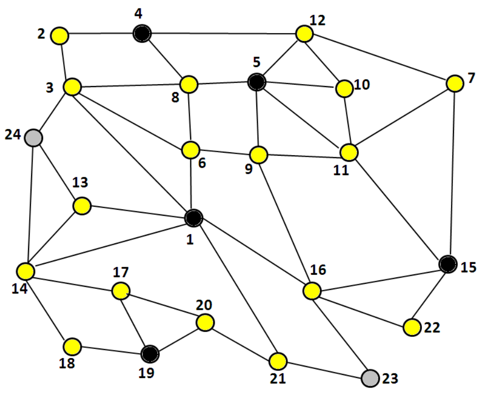
(b) Node 1 becomes dominator



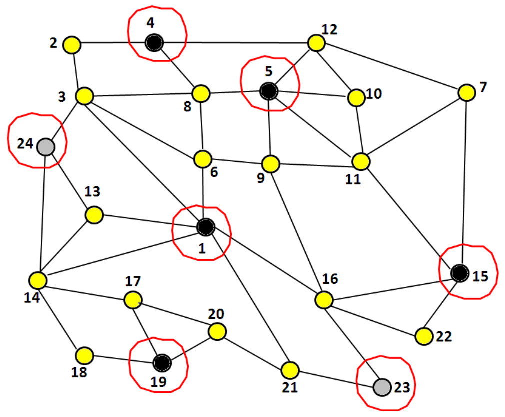
(c) Node 23, 24 become VD



(d) Node 5, 19 become dominators

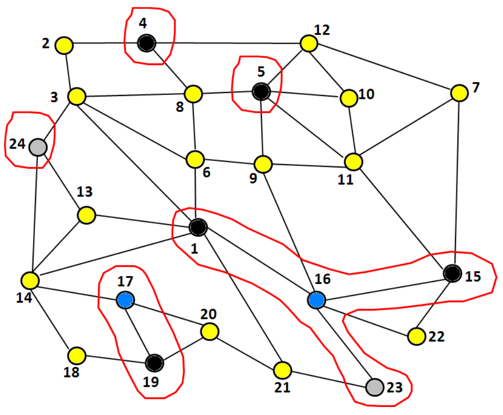


(e) PDS of initial network

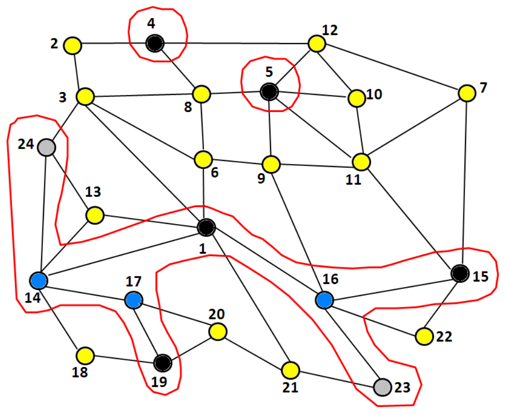


(f) Initial components

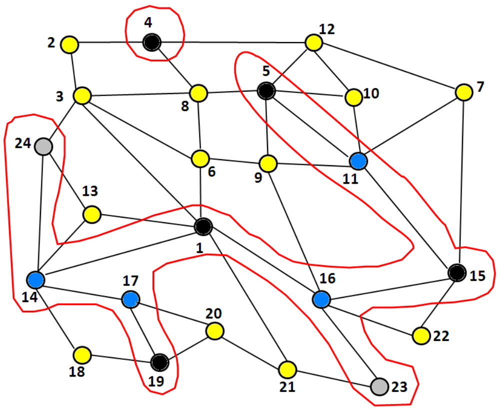
Figure 3.8: Example showing CDS Construction by CPDSTHI



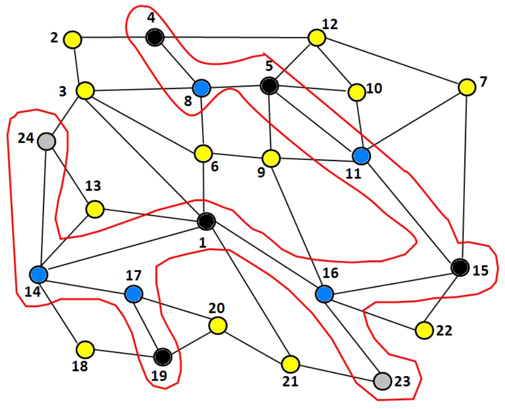
(g) Node 16, 17 become connectors



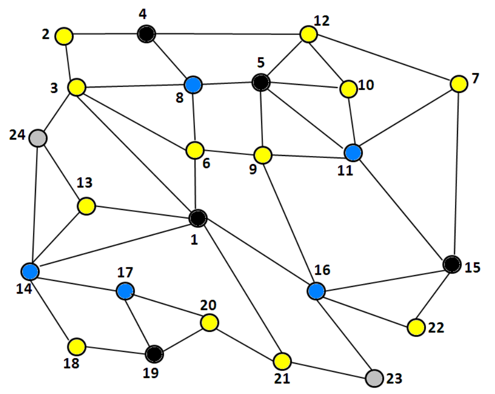
(h) Node 14 becomes connector



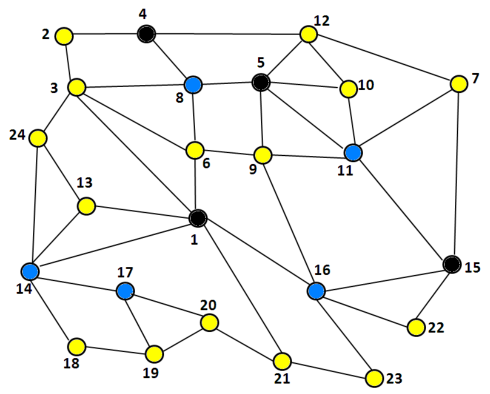
(i) Node 11 becomes connector



(j) Node 8 becomes connector



(k) Initial CDS of the network



(l) Final CDS

Figure 3.8: Example showing CDS Construction by CPDSTHI - contd...

Lemma 3.2 *Improved Steiner Tree construction on a PDS (Algorithm 2) forms a single connected component.*

Proof. *The algorithm for Steiner tree construction (Algorithm 2) first forms separate components by using the black and grey nodes. Next, in each iteration, from the remaining yellow nodes, it selects the nodes as the connector which connects a maximum number of components among its rivals. So, each connector either connects two or more components or increases the size of any existing component at least by one. At the beginning of the Steiner tree construction phase, each component is separated from its nearest component by a maximum of two hops. Therefore, in each iteration, after the addition of a connector to an existing component, the distance of some component from its nearest component either would become zero or one. That means in each iteration, the selected connector either decreases the number of components or reduces the distance between any pair of components. Therefore, after a finite number of iterations, all the components would be connected and a single component would be formed.* ■

Lemma 3.3 *Removal of redundant dominators (by Algorithm 3) does not disconnect the CDS nodes.*

Proof. *Algorithm 3 removes a virtual dominator w from the CDS in the following situations: (1) w is connected to the CDS by one connector. In this case, w can be dominated by the connector. (2) w is connected to the CDS by two connectors and they are adjacent. In this case, by removing w from the CDS does not disconnect the CDS.*

The algorithm also removes some of the dominators whose dominatees are connected to some other dominators or connectors. Let s be a dominator whose dominatees are connected to some other dominators or connectors. So, if s is connected to the CDS either by one connector or by two connectors and they are adjacent then by the similar argument (for virtual dominator w) we can conclude that s can be removed from the CDS without disconnecting the CDS. ■

Theorem 3.1 *Given a network, CPDS2HI determines the corresponding CDS in finite time.*

Proof. *In Lemma 3.1 we proved that Algorithm 1 constructs an MIS. In each iteration of Algorithm 1, a positive number of nodes are removed from considera-*

tion, so the number of iteration is bounded. The process terminates when all the remaining nodes have degree zero (and become virtual dominators).

Algorithm 2 starts with a PDS which is a dominating set of nodes (each considered as a separate component). Adding a positive number of connector nodes in each iteration preserves the dominating set property, strictly increases the size of one or more components so the number of iterations is bounded. The connectors added may also reduce the number of components, and since the original graph is connected, the process must terminate with a dominating set forming a single component (hence a CDS).

In phase 3, any virtual dominator or dominator removed from the CDS is chosen in such a way that the reduced CDS remain both connected and a dominating set. All the virtual dominators and the dominators are considered for down gradation to dominatee one at a time. As there is a finite number of virtual dominators and dominators, so it will take a finite time. ■

Lemma 3.4 *In any unit disk graph, the size of every MIS is upper-bounded by* $3.8|opt| + 1.2$*, where* $|opt|$ *is the size of the MCDS in this unit disk graph.*

Proof. *From the result reported in [92].* ■

Lemma 3.5 *The size of Steiner nodes obtained from CPDS2HI is at most* $(1 + \ln 5)|opt|$*, where* $|opt|$ *is the size of any optimal CDS.*

Proof. *The proof follows directly from theorem 2 of [71].* ■

Theorem 3.2 *CPDS2HI produces a CDS with size bounded by* $(4.8 + \ln 5)|opt| + 1.2$*, where* $|opt|$ *is the size of the MCDS.*

Proof. *CPDS2HI in its first phase constructs the PDS as an MIS. In the second phase it obtains the Steiner nodes. In the last phase it removes the redundant dominating nodes.*
Therefore, we have,

$$|CDS| \leq |PDS| + |Steiner\ nodes|$$

Since PDS is an MIS, it follows from lemma 3.4 and 3.5 that:

$$\begin{aligned}|CDS| &\leq 3.8|opt| + 1.2 + (1 + \ln 5)|opt| \\ &= (4.8 + \ln 5)|opt| + 1.2\end{aligned}$$

Thus we conclude that the performance ratio of CPDS2HI is $(4.8+\ln 5)|opt|+1.2$. ■

Theorem 3.3 *CPDS2HI has time complexity of $O(D)$ time, where D is the network diameter.*

Proof. *In our PDS construction algorithm, multiple dominators may be selected simultaneously. Each dominator is chosen from its local 2-hop neighbourhood and all its adjacent nodes become dominatees. Then, the search for the next set of dominators, from the remaining white nodes in the locality, commences in the subsequent iterations. The worst case time for the PDS construction is the maximum time required to select the largest stretch of dominators and virtual-dominators, where each member is selected one after another because the former choice of the dominator influences the choice of the next dominator. In the worst case, this longest stretch of dominators and virtual-dominators can exist along the network diameter, which is largest of all the shortest distances of the farthest nodes from the first set of chosen dominators. In the worst case, as discussed, the number of iterations is at most $O(D)$. Therefore, the time complexity for PDS construction is $O(D)$ time. However, as multiple dominators can be selected in each iteration, the average time complexity of this phase is much lower than $O(D)$.*

In the Steiner Tree construction, multiple Steiner nodes are selected simultaneously. Each connector is selected to connect the components surrounding it. By a similar argument, Steiner Tree construction will also require $O(D)$ time.

Finally, to remove the redundant dominating nodes, each PDS nodes is checked once whether to be removed or not from the PDS. The maximum distance between any pair of PDS nodes in the network is the diameter D of the network. If we consider each PDS nodes one by one whether to remove or not, then the maximum required time is $O(D)$. Hence, the proposed algorithm has an overall time complexity of $O(D)$ time. ■

3.6 Simulation Results

In this section, we present the simulation results to show the accuracy and performance of our CPDS2HI scheme and compare it with the existing approaches. In the experimentation, the deployment area (M) is of dimension 100×100 square units. We generate N number of hosts in the area M randomly, by choosing each of their abscissa and ordinate using a uniform random number generator. We used randomly generated instances because although deterministic node deployment has many advantages, in large WSNs we deploy the nodes randomly to reduce the installation cost. The transmission range of each node is considered as R. In the network, two nodes are connected through an edge, if the distance between them is at most R. As the transmission range of all the nodes is the same, the underlying network is a UDG. In experimentation, we consider only connected networks. In the following simulations, we have considered $R = 25$. We run the algorithm 100 times for each of the different network sizes. The average results are reported in the accompanying figures and table. The complete simulation is carried out in NS-2, a network simulator for wireless networks.

In the first experiment, we perform a simple experiment to substantiate that our PDS has smaller cardinality than the MIS selected from other CDS construction schemes. For varying sizes of connected networks in UDGs, we compare the cardinality of our PDSs with that of the MISs obtained from collaborative cover. The results reported in Fig. 3.9 illustrates that our PDS has smaller sizes compared to the MIS selected from collaborative cover.

Next, we compare the Steiner nodes required to connect the independent set nodes (i.e. the dominators and virtual-dominators) as a function of network size. We use a metric, which is the ratio of a number of Steiner nodes to the number of independent set nodes. We compare the results with the best existing algorithm, collaborative cover heuristic [72], for the network sizes varying from 25 to 225 as given there. The results are reported in Fig. 3.10. The ratio provides a good measure for the average effective connector degree. From the figure, one can observe that for large size networks, the ratio for collaborative cover becomes less than 0.3, which indicates that on average a Steiner node connects more than three independent set nodes. However, the same ratio for our CPDS2HI scheme

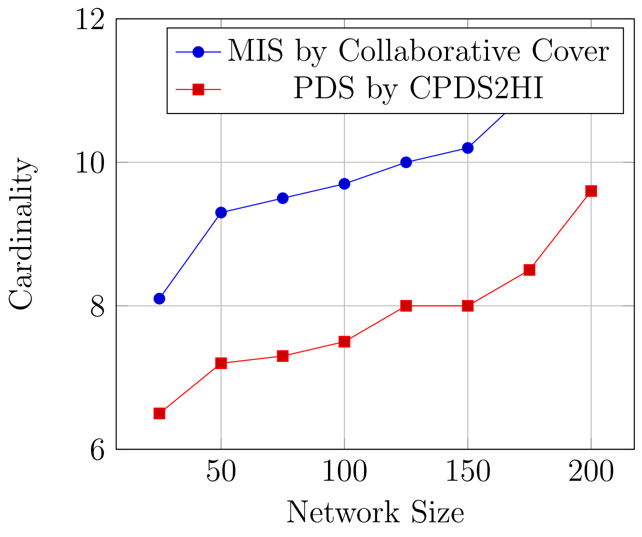


Figure 3.9: Performance comparison of PDS construction phase with MIS selection scheme

in large network sizes tends to be around 0.5, which implies that one Steiner node often connects nearly two PDS elements. So, the average effective connector degree resulting from CPDS2HI scheme is less than that of collaborative cover approach. Less effective degree of a connector indicates that the connection load of the connector is less which helps in enhancing the battery life and hence the network lifetime.

In the next experiment, we analyse the significance of ignoring virtual-dominators from PDS in the post-CDS construction. The network size is varied from 25 to 250 to determine the fraction of total virtual-dominators being discarded and its impact on the reduction of CDS size. Fig. 3.11 shows that post-Steiner Tree construction, nearly all of the virtual-dominators are discarded, occasionally at most one virtual-dominator is retained as a connector for bridging two disjoint components of the CDS. We also find that post-Steiner Tree construction some of the dominators are also discarded. Results reported in Fig. 3.12 describe that for networks of large sizes, ignoring virtual-dominators and dominators according to the specified criteria, results in a reduction of around 1/10$^{\text{th}}$ of CDS size.

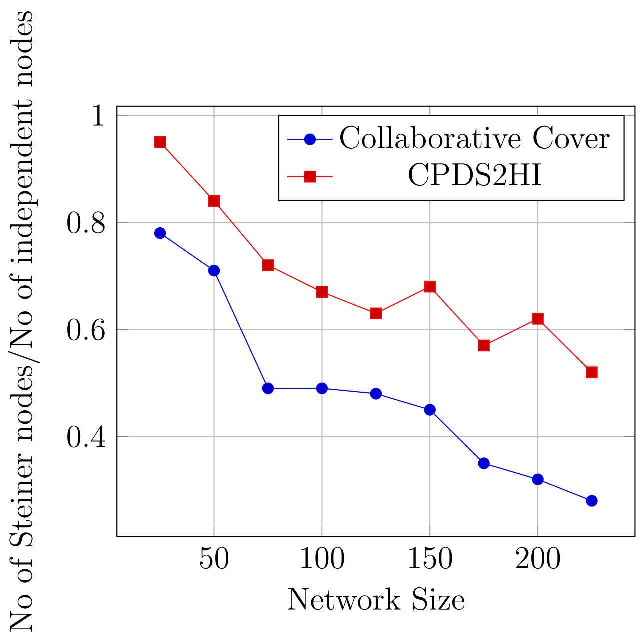


Figure 3.10: Performance comparison of number of Steiner nodes and number of independent nodes.

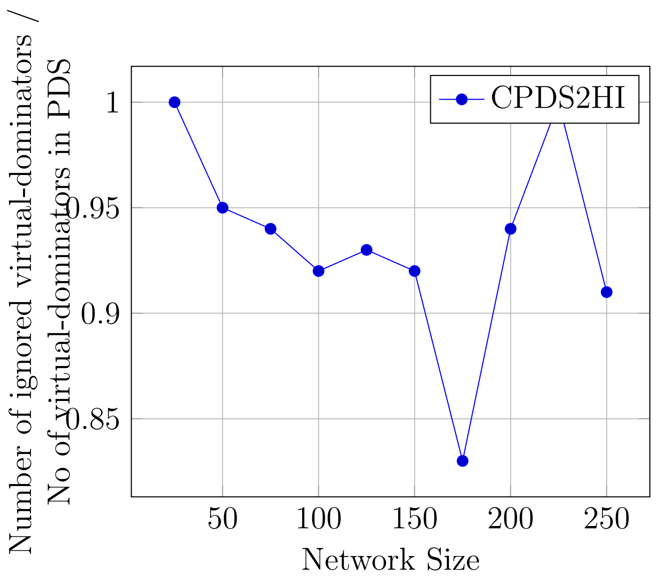


Figure 3.11: Ratio of ignored virtual-dominators to total virtual-dominators in pseudo-dominating set for different network sizes.

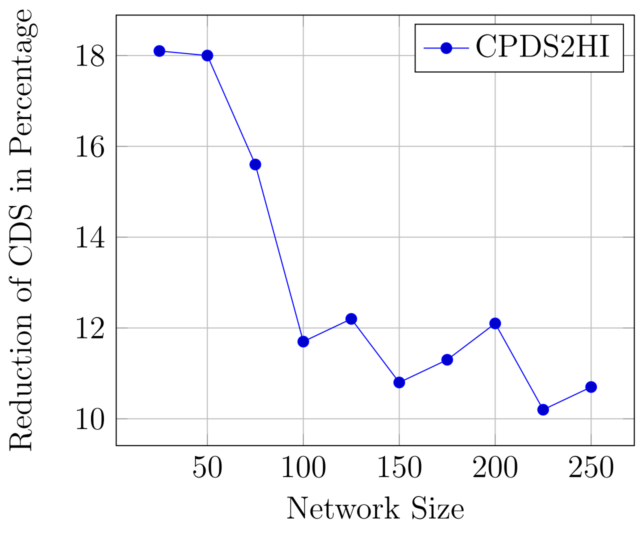


Figure 3.12: Reduction in CDS size after discarding redundant dominators and virtual-dominators in post-Steiner Tree construction for different network sizes

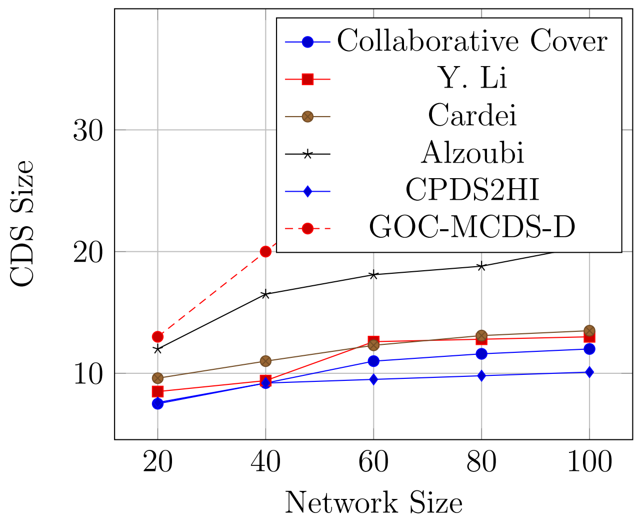


Figure 3.13: Performance comparison of CDS construction algorithms

Next, we compare the performance of our CPDS2HI scheme by comparing its CDS size with the CDS sizes of the existing techniques reported in [62], [65], [71], [72], [73]. In comparison, we considered the random distribution of nodes. Firstly, we determine the CDS size for the networks of sizes 20, 40, 60, 80 and 100. The result shown in Fig. 3.13 demonstrates that CPDS2HI outperforms the dominating tree construction technique [62], 8-approximate CDS algorithm [65], S-MIS approach [71], collaborative cover heuristic [72] and GOC-MCDS-D [73] in identifying a smaller size CDS. The result shows that our approach reduces the CDS size by 16% compared to the previous best collaborative cover heuristic.

Furthermore, the S-MIS [71] algorithm involving Steiner Tree construction and the degree-based 8-approximate CDS algorithm [65] result in 29% and 34% higher CDS sizes respectively, when compared with CPDS2HI. Our proposed algorithm is able to construct smaller CDS because of the tricks in Phase 2 and Phase 3. In Phase 2, it constructs the Steiner Tree by including the nodes which can connect a maximum number of components. In Phase 3, it omits the redundant nodes cleverly to reduce the CDS size further without any coverage or connection loss.

3.7 Summary

In this work, we designed a new centralized degree-based greedy approximation algorithm CPDS2HI. The CDS is constructed in three steps, firstly by Pseudo-Dominating Set (PDS) selection the PDS nodes are selected, then through an improved Steiner Tree construction technique the PDS nodes are connected and finally, the redundant virtual-dominators and dominators from the CDS are discarded. The constructed PDS helps in identifying the MISs of smaller sizes. An improved Steiner Tree is constructed to connect the PDS nodes and some of them are selectively removed in the later stage to build the CDS of smaller size. CPDS2HI identifies non-trivial CDSs of smaller sizes for any random distribution of sensor nodes. The simulation results show that our algorithm constructs smaller sized CDSs in comparison to all other existing schemes for a wide variety of distributions of sensor nodes. The performance ratio of our reported algorithm is $(4.8 + \ln 5)|opt| + 1.2$, where $|opt|$ is the size of an optimal CDS. The results indicate that our algorithm outperforms all the existing approaches that were surveyed, in terms of CDS size. Our algorithm constructs the CDS once the topology information is known. It does not wait for the messages from other nodes in each stage of construction. Although the network model used in the algorithm is UDG, our reported algorithm is applicable for CDS construction when hosts in a network have different transmission ranges.

CHAPTER 4

Distributed Construction of CDS

4.1 Overview

In a wireless network, each node of the network helps in routing by forwarding data of other nodes and which node will forward the data is decided dynamically based on the state of the network. WSN is ad-hoc because it does not depend on any existing infrastructure such as a router in a wired network. In WSN, nodes can communicate efficiently through the use of a virtual backbone. A virtual backbone is a connected subset of nodes deployed in the entire network which helps in routing. A pair of nodes in the network can communicate with each other through the backbone nodes. As there is no fixed infrastructure and centralized control in a WSN, a Connected Dominating Set (CDS) can work as a virtual backbone for efficient routing and connectivity [43].

A Dominating Set (DS) of a network is formed by any subset of nodes of the entire network such that, each node either belong to the subset or neighbour of some element of that subset. If the nodes of a DS are connected, then they form a CDS. The CDS is responsible for transmitting messages from any node to any other node. A source node which does not belong to the CDS sends its message to the destination node by first sending it to one of its neighbouring CDS nodes. If the destination node belongs to the CDS, it gets the message directly, otherwise, it gets the message from one of its neighbours which belongs to the CDS. During routing, a CDS node forwards the message to its CDS neighbours only. So, these CDS nodes only maintain the routing information. Therefore, reduction of CDS size can save the storage space and also makes the routing easier and faster. Also by using a smaller sized CDS as a virtual backbone the total energy consumption of the network can be reduced if the non-CDS nodes switch off their radio when they don't have any data to send. Therefore, the construction of Minimum Connected Dominating Set (MCDS) of the network is desirable. However, MCDS construction is an NP-complete problem [44]. For this reason, researchers are interested in polynomial time approximation algorithms for CDS construction. As there is no centralized management in WSN, distributed algorithms can be useful for finding the MCDS. Energy is vital in WSN because the nodes can't be recharged. Therefore, the distributed approximation algorithms should construct smaller CDSs with low computation and communication costs. The quality of CDS is measured by its approximation ratio, which is the ratio of the size of the constructed CDS (by the proposed algorithm) to the size of MCDS. The construction cost is also measured by the overall message and time complexities. To extend the lifetime of the network, in spite of relying on a single CDS, the network should switch between disjoint CDSs [45, 46]. Therefore, to switch between CDSs quickly the computation time of the CDS construction algorithm should be small enough.

We can construct a CDS either in centralized or in a distributed manner. Although centralized algorithms provide more accurate information than distributed algorithms, they suffer from scalability problem and hence not feasible for large size WSNs. In centralized algorithms, the reliability of the information accumulated at a centralized processor is low because of the losses involved in multi-hop transmission. Distributed algorithms are difficult to design. They require only local information exchange between neighbouring nodes. For any WSN in which the average number of hops from any node to the central processor is greater than the number of iterations required to perform a task, distributed algorithms are more energy efficient than centralized algorithms [47]. In this chapter, we propose a new distributed degree-based greedy approximation algorithm which we name as

Distributed **C**onstruction of **M**inimum **C**onnected **D**ominating **S**et (DCMCDS) to construct smaller CDSs.

The proposed scheme DCMCDS works in four phases and constructs the CDS using 2-hop information only. In the first phase, it initializes the data items stored in each individual node and creates the 1-hop and 2-hop neighbours table. In the second phase, it constructs a Maximal Independent Set (MIS) in a distributed manner. The MIS is designated as a Pseudo Dominating Set (PDS) because some of the elements may be omitted in the final dominating set. In the third phase, the algorithm constructs a Steiner Tree by adding some more nodes to the PDS, which are needed to interconnect the PDS nodes. In the last phase, the algorithm drops some of the selected PDS nodes to reduce the CDS size further without any loss in coverage or connectivity. Simulation results show that DCMCDS is better than existing CDS construction algorithms in terms of CDS size and construction costs. The approximation ratio of the proposed algorithm, which is the best at the current moment, is $(4.8 + \ln 5)|\mathsf{opt}| + 1.2$, where $|\mathsf{opt}|$ is the size of an optimal CDS of the network. Its time complexity is $\mathsf{O}(\mathsf{D})$, where D is the diameter of the network. It has a linear message complexity of $\mathsf{O}(\mathsf{nR})$, where n is the network size and R is the maximum between the number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes.

The remaining of the chapter is organized as follows. In Section 4.2, we discuss the motivation behind the work and state our objective. In Section 4.3, we state our assumptions regarding the development of ad hoc network model. In the next section (Section 4.4), the distributed CDS construction algorithm is presented in detail. The analysis of our proposed distributed algorithm is discussed in Section 4.5. Supporting simulation results are given in Section 4.6. Finally, we present the summary in Section 4.7.

4.2 Motivations and Objectives

4.2.1 Motivations

Centralized algorithms provide more accurate information. However, they are not applicable to networks of larger sizes. In the other hand, although distributed algorithms are difficult to design, they need local information exchange only. So the use of distributed algorithms would be more feasible. The centralized CDS constructions algorithms construct small sized CDSs with better approximation ratio, the construction is based on the assumption that the complete network topology is known to a single node where the CDS construction is carried out. However, this assumption is not practical in a wireless network. Therefore, we were motivated to design distributed CDS construction algorithms which can be used practically in a wireless network to construct smaller size CDSs.

4.2.2 Objectives

Distributed Construction of CDS in a static Wireless Network We can construct a CDS either in centralized or in a distributed manner. Although centralized algorithms provide more accurate information than distributed algorithms, they suffer from scalability problem and hence not feasible for large size WSNs. In centralized algorithms, the reliability of the information accumulated at a centralized processor is low because of the losses involved in multi-hop transmission. Distributed algorithms are difficult to design. They require only local information exchange between neighbouring nodes. For any WSN in which the average number of hops from any node to a central processor is greater than the number of iterations required to perform a task, distributed algorithms are more energy efficient than centralized algorithms [47]. Our second objective was to design a distributed approximation algorithm to construct smaller size CDSs.

4.3 Network Model for Distributed CDS Construction

The assumptions regarding the development of ad hoc network model for the above-mentioned objective are as follows:

- The nodes do not have any topological information. They do not even have knowledge of their distances to their neighbours.
- Each node has a unique ID.

- Nodes exchange HELLO messages to identify their *1-hop* neighbours and ascertain their degree. Later on, they build and maintain two tables to store their *1-hop* and *2-hop* neighbours information by sending other required messages.
- All the hosts are deployed in a 2-dimensional plane and their maximum transmission ranges are the same.
- The resultant topology of the network is modelled as a unit disk graph (UDG) with the transmission range of the nodes considered as one unit i.e. two nodes are connected by a wireless link if their distance does not exceed the transmission range. All the nodes are bidirectional.
- The communication overhead due to interference is negligible.
- The computation is partitioned into rounds, where the nodes receive the messages sent in the previous round, execute local computations and send messages to the neighbours in the next round.

4.4 Distributed DCMCDS scheme

In this section, we discuss the details of the distributed algorithm of DCMCDS scheme. During the execution of the algorithm, each node of the network u, maintains the following variables:

- **colour (u_{colour}):** This variable shows the current status of the node. The initial colour of each node is white. The nodes change their colours either to black, grey, yellow or blue when their status changes to either dominator, virtual-dominator, dominatee or connector respectively.
- **nodeID (u_{ID}):** An ID, which is unique for each node.
- **originalDegree ($u_{odegree}$):** This variable stores the initial degree of the node in the graph.

- **effectiveDegree ($u_{edegree}$):** This variable stores the effective degree of a node u in the graph. Effecive degree of a node varies from time to time. Effective degree of a node at a particular moment is the number of white nodes adjacent to that node at that particular moment.

- **componentID (u_{cID}):** An ID to demarcate nodes belonging to different components. The nodes present in a single component have the same *componentID*, which is the least *nodeID* of all the dominators / connectors forming the component.

- **1HopNebsTable ($N_1(u)$):** A table stored at node u which records the *nodeID*, *colour*, *originalDegree* and *effectiveDegree* of all its adjacent nodes.

- **2HopNebsTable ($N_2(u)$):** A table stored at node u which records the *nodeID*, *colour*, *originalDegree*, *effectiveDegree*, *mutualNeighbour*, *mnColour* for its distance-2 neighbours (excluding itself). $N_2(u)$ contains even those 2-hop neighbours of u which are also adjacent to u. The multi-valued attribute *mutualNeighbour* in $N_2(u)$, corresponding to a 2-hop neighbour v, contains the *nodeID*s of all the nodes that are adjacent to both u and v. The multi-valued attribute *mnColour* stores the colour of the corresponding *mutualNeighbour*.

- **cdsList ($u_{cdsList}$):** This list contains the *nodeID*s of the members (dominators / virtual-dominators / connectors) of the component, to which node u belongs.

- **connectionCount (u_{ccnt}):** This variable records the number of independent components adjacent to u.

- **rivalList ($u_{rivalList}$):** This list contains the *nodeID*s of the dominatees which are adjacent to the same component, to which node u is adjacent.

In the following sub-sections, first, we discuss each of the phases of our distributed DCMCDS scheme in detail. At the end of this section, we discuss the phase transition of the proposed distributed algorithm.

4.4.1 Node Initialization and neighbourhood table creation

In this phase of the distributed DCMCDS scheme, each of the nodes initializes their variables and neighbourhood tables by sending and receiving the following messages:

- HELLO: Each node broadcasts this message to inform about its presence to its neighbours.
- OWN_INFO: Through this message, a node informs its *originalDegree* to its neighbours.
- NEB_INFO: This message is sent by a node, to pass on its detailed neighbour information, to all of its neighbours.

Algorithm 4 describes the detail initialization procedure.

Algorithm 4 Node Initialization

```
1: Each node $u$, initializes its variables as $\langle u_{colour} \leftarrow white \rangle$, $\langle u_{cID} \leftarrow nil \rangle$, $\langle u_{odegree} \leftarrow 0 \rangle$, $\langle u_{edegree} \leftarrow 0 \rangle$, $\langle u_{ccnt} \leftarrow 0 \rangle$, $\langle N_1(u) \leftarrow nil \rangle$, $\langle N_2(u) \leftarrow nil \rangle$.
2: Each node broadcasts a HELLO message and receive the same message from other nodes.
3: A node $u$ ascertains its number of neighbours from the number of HELLO messages received and updates its state variable originalDegree and effectiveDegree as $u_{odegree} \leftarrow u_{edegree} \leftarrow$ number of HELLO messages received.
4: A node $u$ after updating its state variable originalDegree, broadcasts a message OWN_INFO $= \langle u_{ID}, u_{odegree} \rangle$.
5: A node $v$ adjacent to $u$, on receiving OWN_INFO message from $u$, adds a tuple $\langle u_{ID}, white, u_{odegree}, u_{odegree} \rangle$ to $N_1(v)$.
6: When all the OWN_INFO messages are delivered, each node $v$ broadcasts a message NEB_INFO $= \langle v_{ID}, N_1(v) \rangle$.
7: Every node $w$, which is a 2-hop neighbour of $u$, on receiving message NEB_INFO from $v$, adds all tuples in $N_1(v) - \{\langle w_{ID}, white, w_{odegree}, w_{odegree} \rangle\}$ to $N_2(w)$ with mutualNeighbor $\leftarrow v_{ID}$ and $mnColor \leftarrow$ white.
```

4.4.2 Distributed PDS construction

In this phase, each node uses its neighbourhood information (stored in its *1HopNebTable* and *2HopNebTable*) to decide whether it can become a dominator or not. A node on becoming a dominator, virtual-dominator or a dominatee, spread its new status information up to two hops so that its 1-hop and 2-hop neighbours can update their tables. In each round, one or more nodes become either a dominator or a virtual-dominator. At the beginning of each round, the white nodes check their updated *1HopNebsTable* and *2HopNebsTable*, to decide whether they can become a dominator in the current round or not. When all the nodes change their colour from white to some other colour, the PDS construction is over.

This phase constructs the PDS in a distributed manner using the following messages:

- DOMINATOR: A node broadcasts this message when it becomes a dominator.
- DOMINATEE: A node broadcasts this message when it becomes a dominatee.
- VIRTUAL_DOMINATOR: A node broadcasts this message when it becomes a virtual-dominator.
- UPDATE_NEB_INFO: When the *effectiveDegree* of a node is changed, it informs this to its neighbours through this message.
- UPDATE_NODE_COL: This message is sent by a dominatee node, to inform about the change in colour of any of its neighbours to its other neighbours.

The detailed procedure for distributed PDS construction is given in Algorithm 5. It is worth mentioning that unlike PTAS [76], which first selects the approximation $1 + \epsilon$ and then determines the locality around a node that needs to be explored, in our algorithm a node surveys only up to its 2-hop neighbourhood. If we expand this locality further, then a higher number of message exchanges will be required. Hence, this agreement of exploring up to 2-hop neighbourhood only in our algorithm can significantly reduce CDS size without any considerable compromise in a number of messages exchanged as is evident from our simulation results. Besides this locality distance of two hops in our algorithm is sufficient to divide the connected graph into 2-separated independent dominating sets (i.e. neighbouring local smaller MISs separated from each other by two intermediate hops). If we reduce the locality distance further, then this will result in 1-separated local MISs that will increase the approximation ratio.

4.4.3 Distributed Steiner Tree construction

At the beginning of this phase, the colour of each node is either black, grey, or yellow. Each of the black and grey nodes form separate components and store their own *nodeId* in their *cdsList* as they are the only node of their component. In each round of this phase, the CDS members (black / grey / blue nodes) of each component send a request message to their adjacent yellow nodes (dominatees) to get their *connectionCount* (number of independent components they are connected to). The dominatees after getting all the request messages, reply to their adjacent CDS members with their own *connectionCount*. The CDS members of each component after getting these reply messages, prepare a list of their adjacent dominatees (known as *rivalList*). They circulate their own *rivalList* among other CDS members of the same component to prepare the complete *rivalList* of the whole component. Once the CDS members of a component prepare the complete *rivalList*, they send this *rivalList* along with their *cdsList* together, to their yellow neighbours. The *rivalList* contains the $node - ID$s of the rival members along with their original degree. Each of the yellow dominatees, after getting the rival information (*rivalList*) of the components they are connected with, arranges the rival nodes in non-increasing order of *connectionCount*. After this, it also decides to participate in this process or not. If a yellow node finds all its rivals are connected to the same component to which it is connected, and each of its 2-hop black neighbours are present in one of the received *cdsList*s, then it decides not to participate in this process anymore. If a yellow node decides to participate and finds itself ranked first in its *rivalList*, then it becomes a connector. A node on becoming a connector changes its colour to blue, forms a new component by merging the components it connects with itself. It assigns the *componentID* of the new component as the minimum of the *componentID*s of the merged components and its own *nodeID*. The *cdsList* of the new component contains all the existing CDS member of the merged components and the connector. The connector sends the updated component information to all its neighbours which in turn spread the component information to all the members of the same component. The colour of the connector is also sent to its 2-hop neighbours. All the CDS nodes start their next round by sending the request messages for getting the *connectionCount* of their yellow neighbours. This phase continues until there is no yellow node that can still participate in this phase. This phase uses the following messages to constructs the Steiner Tree:

- CONN_INFO_REQ: The black/grey/blue nodes of a component broadcast this message to their yellow neighbours to get their *connectionCount*.
- CONN_INFO_REP: The yellow dominatees send their *connectionCount* to their blue/grey/black neighbours through this message.
- COMP_RIVAL_INFO: The black/grey/blue nodes of a component broadcast this message, to inform the yellow nodes about their *rivalList*. They also send their *cdsList* with this message, which is used by the yellow nodes to decide whether to participate in this phase or not.
- RIVAL_INFO: The black/grey/blue nodes of a component, send this message to their component members, to prepare the complete *rivalList* of the whole component to which they belong.
- CONNECTOR: A node broadcasts this message when it becomes a connector to notify its neighbours about its new role.
- UPDATE_COMP_INFO: Black/grey/blue nodes of a component, send this message to their component members, to update their *componentID* and *cdsList*.
- UPDATE_NODE_COL: This message is sent by a dominatee node, to inform about the change in colour of any of its neighbours to its other neighbours.

The detailed procedure for distributed construction of Steiner Tree is given in the Algorithm 6.

Algorithm 5 Distributed PDS Construction

```
Input: A connected graph $G(V, E)$.
Output: PDS of the graph $G(V, E)$ formed by black and grey nodes.
1: In each round, a white node $u$ checks itself to decide whether it can be a dominator or not.
2: A white node $u$, elects itself as a dominator, if any of the following conditions apply:
     (i) $u_{edegree} > v_{edegree} \forall$ white nodes $v \in N_1(u) \cup N_2(u)$.
     (ii) $u_{edegree} \geq v_{edegree} \forall$ white nodes $v \in N_1(u) \cup N_2(u)$, but $u_{odegree} > w_{odegree} \forall$ white nodes $w \in N_1(u) \cup N_2(u)$ where $u_{edegree} = w_{edegree}$.
     (iii) $u_{edegree} \geq v_{edegree}$ and $u_{odegree} \geq v_{odegree} \forall$ white nodes $v \in N_1(u) \cup N_2(u)$, but $u_{ID} < w_{ID} \forall$ white nodes $w \in N_1(u) \cup N_2(u)$ where $u_{edegree} = w_{edegree}$ and $u_{odegree} = w_{odegree}$.
3: A white node $u$ on becoming a dominator, performs the following operations:
     (i) Updates its colour as $\langle u_{colour} \leftarrow black \rangle$.
     (ii) Updates the colour of each of its 1-hop white neighbours to yellow in $N_1(u)$
     (iii) Update its componentID as $\langle u_{cID} \leftarrow u_{ID} \rangle$.
     (iv) Broadcasts message DOMINATOR($u_{ID}$).
4: A white node $v$ on receiving DOMINATOR($u_{ID}$) message from a node $u$, performs the following operations:
     (i) Updates its state variable as $\langle v_{colour} \leftarrow yellow \rangle$
     (ii) Updates the colour of node $u$ in $N_1(v)$ and $N_2(v)$ as $\langle u_{colour} \leftarrow black \rangle$.
     (iii) Changes the colour of the node $x \in N_2(v)$ to yellow if $x_{colour} = white$ and the mutualNeighbour of $x$ is $u$.
     (iv) Broadcasts message DOMINATEE($v_{ID}, u_{ID}$).
5: A white node $w$ on receiving DOMINATEE($v_{ID}, u_{ID}$) message from node $v$, performs the following operations:
     (i) Updates its effectiveDegree as $\langle w_{edegree} \leftarrow w_{edegree} - 1 \rangle$
     (ii) Updates the colour of node $v$ in $N_1(w)$ and $N_2(w)$ as $\langle v_{colour} \leftarrow yellow \rangle$.
```

Algorithm 5 Distributed PDS Construction - **contd...**

```
        (iii) Updates the colour of node $u$ in $N_2(w)$ as $\langle u_{colour} \leftarrow black \rangle$.
        (iv) Updates the colour of the mutualNeighbor $v$ in $N_2(w)$.
        (v) Broadcasts UPDATE_NEB_INFO $(w_{ID}, w_{edegree}, v_{ID})$ message .
            [Note that when $v$ becomes a dominatee it is deleted from the network
        (refer to Step 12 of Algorithm 1). So, the 2-hop neighbours of $w$, only through
        $v$, are no more the 2-hop neighbours. Henceforth, the 2-hop neighbours with
        non-white mutual neighbour only are not considered as 2-hop neighbours of $w$
        during dominator election process in the next round (Step 2).]
 6: A yellow node $w$ on receiving DOMINATEE$(v_{ID}, u_{ID})$ message from node $v$,
    performs the following operations:
        (i) Updates the colour of node $v$ in $N_1(w)$ as $\langle v_{colour} \leftarrow yellow \rangle$.
        (ii) Updates the colour of node $u$ in $N_2(w)$ as $\langle u_{colour} \leftarrow black \rangle$.
        (iii) Broadcasts UPDATE_NODE_COL $(v_{ID}, yellow)$ message.
 7: A node $p$ on receiving UPDATE_NEB_INFO $(w_{ID}, w_{edegree}, v_{ID})$ message from
    $w$, performs the following operations:
        (i) Updates the colour of node $v$ in $N_1(p)$ and $N_2(p)$ as $\langle v_{colour} \leftarrow yellow \rangle$.
        (ii) Updates the effectiveDegree of $w$ in $N_1(p)$ as $w_{edegree}$.
 8: At any instance, when the effectiveDegree of a white node $u$ gets
    decremented to zero, it become a virtual-dominator and updates its
    colour as $\langle u_{colour} \leftarrow grey \rangle$. It informs its new role by broadcasting
    VIRTUAL_DOMINATOR $(u_{ID})$ message to its neighbours.
 9: A yellow dominatee $v$ on receiving the DOMINATOR message from $u$:
        (i) Updates the colour of node $u$ in $N_1(v)$ as $\langle u_{colour} \leftarrow black \rangle$.
        (ii) Broadcasts UPDATE_NODE_COL $(u_{ID}, black)$ message.
10: A yellow node $v$ on receiving the VIRTUAL_DOMINATOR message from $u$:
        (i) Updates the colour of node $u$ in $N_1(v)$ as $\langle u_{colour} \leftarrow grey \rangle$.
        (ii) Broadcasts UPDATE_NODE_COL $(u_{ID}, grey)$ message.
```

Algorithm 5 Distributed PDS Construction - **contd...**

```
11: A node $p$ on receiving UPDATE_NODE_COL $(nid, ncolor)$ message, updates
    the colour of node $x \in N_1(p) \cup N_2(p)$ with $x_{ID} = nid$ as $\langle x_{colour} \leftarrow ncolor \rangle$.
12: Each white node $u$ broadcasts the message NEB_INFO $= \langle u_{ID}, N_1(u) \rangle$, if the
    effective degree of any of its 1-hop neighbours has changed in the last round.
13: A node $v$ on receiving NEB_INFO $= \langle v_{ID}, N_1(u) \rangle$ message from $u$ updates its
    2-hop neighbours' information present in $N_2(u)$.
14: Phase-II terminates when eventually all nodes change their colour from white
    to some other colour.
```

4.4.4 Distributed removal of redundant dominators

In this phase, each grey and black node checks whether to downgrade itself or not to reduce the overall CDS size. If a grey node finds that either it is connected to the CDS by only one CDS node or the CDS nodes (in case of multiple connections with CDS nodes) are connected without it, then it downgrades itself to a dominatee, otherwise, it upgrades itself to a dominator. After it upgrades/downgrades it sends its new role to its neighbours, which in turn inform their neighbours. However, if a black node satisfies the same condition (as discussed above for a grey node), it has to check whether all its dominatees have some alternative dominators or not. If it finds that all its dominatees have some alternative dominators, then it downgrades itself to a dominatee and informs its neighbours, which in turn inform their neighbours, otherwise, it remains as a dominator. A black node, to find out the availability of the alternative dominators of its dominatees, sends a request message to its dominatees and waits for their replies. If it gets the TRUE reply from all of them, then it downgrades itself, otherwise, it cancels its previous request by sending a cancel message to all of them. A dominatee which gets a request message to check its alternative dominators, sends a TRUE reply to the first dominator from which it has received the request. After that, it waits for either the change in the status of that dominator (to which it has sent the TRUE reply) or the cancel message from it. If it finds that the dominator has downgraded to a dominatee, it sends a FALSE reply to all of the alternative dominator requests.

Algorithm 6 Distributed Steiner Tree Construction

```
Input: A connected graph $G(V, E)$ with its PDS formed by the black and grey nodes.
Output: Connected Dominating set of the graph $G(V, E)$ formed by black, grey and blue nodes.
1: At the end of PDS construction phase, each black and grey node $x$, forms isolated separate components and initiates the Steiner Tree construction phase as follows:
     (i) Updates its *cdsList* as $\langle x_{cdsList} \leftarrow \{x_{ID}\}\rangle$.
    (ii) Initializes its *componentID* as its *ID* by $\langle x_{cID} \leftarrow x_{ID}\rangle$
   (iii) Broadcasts the CONN_INFO_REQ($x_{ID}, x_{cID}, x_{cdsList}$) message.
2: Each yellow node $u$, after getting the CONN_INFO_REQ messages from all of its black/grey/blue neighbours:
     (i) Calculates its *connectionCount* ($u_{ccnt}$), which is the count of number of independent components it is connected to.
    (ii) Broadcasts a reply message CONN_INFO_REP($u_{ccnt}, u_{ID}, u_{odegree}$) to all its black / grey / blue neighbours.
3: Each black / grey / blue node $w$, on receiving a CONN_INFO_REP message from one of its yellow neighbours $u$, updates its *rivalList* by including the node $u$ in it with its details.
4: Each black / grey / blue node $w$, after receiving CONN_INFO_REP messages from all of its yellow neighbours:
     (i) Circulates its *rivalList* among other component members (if any) to prepare the *rivalList* of the whole component.
    (ii) After preparing the complete *rivalList* of the whole component, it broadcasts the message COMP_RIVAL_INFO($w_{ID}, w_{RivalList}, w_{cdsList}$).
5: Each black / grey / blue node $w$, to prepare the complete *rivalList* of the whole component, circulates its *rivalList* among other component members in the following way:
     (i) If it is the only member of the component, then its *rivalList* is the *rivalList* of the whole component.
```

Algorithm 6 Distributed Steiner Tree Construction - **contd...**

```
        (ii) Else if it has only one component neighbour, then it sends the message
             RIVAL_INFO($w_{cID}, w_{RivalList}$) to that component neighbour.
        (iii) Else it waits to receive the RIVAL_INFO message from all its component
             neighbours except one.
 6: Each black / grey / blue node $z$, on receiving RIVAL_INFO($w_{cID}, w_{rivalList}$)
    message from the same component neighbour $w$:
        (i) Updates its cdsList as $\langle z_{rivalList} \leftarrow z_{rivalList} \cup w_{rivalList} \rangle$.
        (ii) If it has received the RIVAL_INFO message from all its component neigh-
             bours except one, then it sends the message RIVAL_INFO($z_{cID}, z_{rivalList}$) to
             the component neighbour from which it has not received the RIVAL_INFO
             message.
        (iii) Else if it has received the RIVAL_INFO message from all its component
             neighbours, then it sends the message RIVAL_INFO($w_{cID}, w_{RivalList}$) to the
             component neighbours to which it has not sent the RIVAL_INFO message.
 7: Each yellow node after receiving COMP_RIVAL_INFO messages from all its
    adjacent black / grey / blue nodes, orders its received rival nodes according
    to the following criteria:
        (i) All the dominatees are arranged in non-increasing order of their
             connectionCount.
        (ii) If two dominatees have the same connectionCount, then the one with a
             higher originalDegree is ranked higher.
        (iii) If two dominatees have the same connectionCount and originalDegree,
             then the one with a smaller nodeID is ranked higher.
 8: Each yellow node $w$, after preparing the rivalList, decides whether to further
    participate in this phase or not. It participates no longer in the process if it
    satisfies the following two conditions:
        a) The conectionCount of itself and its rivals is 1. This indicates that the
             yellow node and its rivals are adjacent to the same component.
        b) There is no black node in its 2HopNebsTable that does not occur in any
             of the received cdsLists from their black/grey/blue nodes.
```

Algorithm 6 Distributed Steiner Tree Construction - **contd...**

```
9: If a yellow node $w$ decides to participate in the process and finds itself ranked
   first in its *rivalList*, then it becomes a connector and executes the following
   actions:
    (i) Updates its state variable as $\langle w_{colour} \leftarrow blue \rangle$
    (ii) $w_{cdsList} \leftarrow$ union of *cdsList* of the black/blue nodes it is connected with
         and its own ID, $w_{ID}$.
    (iii) $w_{cID} \leftarrow$ minimum of *componentID*s of the black / blue nodes it is con-
          nected with and its own ID, $w_{ID}$.
    (iv) Broadcasts CONNECTOR($w_{ID}$, $w_{cID}$, $w_{cdsList}$) message for its neighbours
         to notify them about its new role.
10: A node $x$, on receiving CONNECTOR ($w_{ID}$, $w_{cID}$, $w_{cdsList}$) message from $w$,
    executes the following:
    (i) Update the colour of $w$ as $\langle w_{colour} \leftarrow blue \rangle$ in its $N_1(x)$.
    (ii) Broadcasts UPDATE_NODE_COL ($w_{ID}, blue$) to its neighbours.
    (iii) If $x_{ID} \in w_{cdsList}$
        a) Update its *componentID* as $\langle x_{cID} \leftarrow w_{cID} \rangle$
        b) Update its *cdsList* as $\langle x_{cdsList} \leftarrow w_{cdstList} \rangle$
        c) Broadcast UPDATE_COMP_INFO ($w_{cID}$, $w_{cdsList}$) to its neighbours if
           it has not send the same $\langle w_{cID}, w_{cdsList} \rangle$ before through any of the
           CONNECTOR or UPDATE_COMP_INFO message.
11: A node $y$, on receiving UPDATE_COMP_INFO ($w_{cID}$, $w_{cdsList}$) message from
    $w$, executes the following: **If** it has not send the same $\langle w_{cID}, w_{cdsList} \rangle$ before
    through either of the CONNECTOR or UPDATE_COMP_INFO message and
    $y_{ID} \in w_{cdsList}$ **then**
    (i) Update its *componentID* as $\langle y_{cID} \leftarrow w_{cID} \rangle$.
    (ii) Update its *cdsList* as $\langle y_{cdsList} \leftarrow w_{cdstList} \rangle$.
    (iii) Broadcasts UPDATE_COMP_INFO ($w_{cID}$, $w_{cdsList}$) message.
```

Algorithm 6 Distributed Steiner Tree Construction - **contd...**

```
12: In the next round, each black / grey / blue node sends the
    CONN_INFO_REQ($x_{ID}$, $x_{cID}$, $x_{cdsList}$) message to all its neighbours again and
    the procedure from step 2 onwards is repeated.
13: This phase of connector selection ends when no yellow dominatee participates
    further. When no yellow node participates further, no new connectors will
    be created. Due to which no more UPDATE_COMP_INFO messages will be
    sent. So black / grey / blue nodes will not send any more CONN_INFO_REQ
    messages.
```

However, if it gets a cancel message from the dominator to which it has already sent the TRUE reply, then it sends the TRUE reply to the next dominator out of the dominators waiting in the queue for its reply. This phase removes some of the redundant dominating nodes in a distributed manner by using the following messages:

- UPGRADE_DOM: A grey virtual-dominator broadcasts this message to its neighbours when it decides to change its role from a virtual-dominator to a dominator.
- DOWNGRADE_DOM: This message is sent by either a virtual-dominator or a dominator when it decides to downgrade itself to a dominatee.
- ALT_DOMINATOR_REQ: A black node sends this request message to its dominatees to know whether they have some alternative dominators or not.
- ALT_DOMINATOR_REP: A yellow dominatee sends a TRUE reply with this message if it is adjacent to some dominator/connector other than the dominator from which it received the ALT_DOMINATOR_REQ message. Otherwise, it returns FALSE reply with this message.
- ALT_DOMINATOR_REQ_CANCEL: If a black node receives a FALSE message from any one of the yellow nodes through the ALT_DOMINATOR_REP message it sends this message to all its yellow neighbours.

Algorithm 7 Distributed removal of redundant dominators

```
Input: A connected graph $G(V, E)$ with its CDS formed by the black, grey and
    blue nodes.
Output: A potentially smaller CDS of the graph $G(V, E)$ after removing some of
    the redundant dominators and virtual-dominators.
 1: Each grey node $v$ changes its state according to the following (Steps 2 - 10):
 2: if there exists only one connector $x \in N_1(v)$ with $x_{colour} = blue$ or there exists
    at least two connectors $x$, $y \in N_1(v) \cap N_2(v)$ with $x_{colour} = y_{colour} = blue$ and
    mutualNeighbour corresponding to $x$ being $y$ and vice-versa then
 3:     Updates its colour as $\langle v_{colour} \leftarrow yellow \rangle$.
 4:     Broadcasts the UPDATE_NODE_COL($v_{ID}, yellow$) message.
 5:     Broadcasts the DOWNGRADE_DOM($v_{ID}$) message.
 6: else
 7:     Updates its colour as $\langle v_{colour} \leftarrow black \rangle$
 8:     Broadcasts the UPDATE_NODE_COL($v_{ID}, black$) message.
 9:     Broadcasts the UPGRADE_DOM($v_{ID}$) message.
10: end if
11: Each black node $v$ may changes its state according to the following (Steps 12
    - 21):
12: if there exists only one connector $x \in N_1(v)$ with $x_{colour} = blue$ or there exists
    at least two connectors $x$, $y \in N_1(v) \cap N_2(v)$ with $x_{colour} = y_{colour} = blue$ and
    mutualNeighbour corresponding to $x$ being $y$ and vice-versa then
13:     Node $v$ broadcasts a request message ALT_DOMINATOR_REQ for all its
    yellow neighbours and wait for their replies.
14:     if It receives ALT_DOMINATOR_REP(TRUE) message from all its yellow
    neighbours then
15:         Updates its colour as $\langle v_{colour} \leftarrow yellow \rangle$.
16:         Broadcasts the UPDATE_NODE_COL($v_{ID}, yellow$) message.
17:         Broadcasts the DOWNGRADE_DOM($v_{ID}$) message.
18:     else
19:         Broadcasts the ALT_DOMINATOR_REQ_CANCEL message.
20:     end if
21: else
22:     The black node $v$ does not change its state.
23: end if
```

Algorithm 7 Distributed removal of redundant dominators - **contd...**

```
24: A yellow node after receiving the first ALT_DOMINATOR_REQ message, sends
    the ALT_DOMINATOR_REP(TRUE) message to that node and enters into the
    waiting state if it is connected to some other dominator or connector. Other-
    wise, it sends the ALT_DOMINATOR_REP(FALSE) message.
25: A yellow node in the waiting state remains in that state until it receives either
    ALT_DOMINATOR_REQ_CANCEL or DOWNGRADE_DOM message from the
    node to which it has already sent the ALT_DOMINATOR_REP(TRUE) message.
26: A yellow node in the waiting state, inserts the new nodes in a queue from
    which it receives the new ALT_DOMINATOR_REQ messages.
27: When a yellow node in the waiting state receives the DOWNGRADE_DOM($v_{ID}$)
    message:
      (i) It sends ALT_DOMINATOR_REP(FALSE) message to the nodes in the
          queue and comes out of the waiting state.
      (ii) After this if it receives ALT_DOMINATOR_REQ messages from any node,
          it sends the ALT_DOMINATOR_REP(FALSE) message to that node imme-
          diately.
28: When a yellow node in the waiting state receives
    the ALT_DOMINATOR_REQ_CANCEL message, it sends
    ALT_DOMINATOR_REP(TRUE) message to the first nodes in the queue
    (if the queue is non-empty) and remains in the waiting state. In case of
    empty queue it comes out of the waiting state.
29: A node $x$ on receiving DOWNGRADE_DOM($v_{ID}$) from node $v$:
      (i) Updates the colour of node $v$ in $N_1(x) \cup N_2(x)$ as $\langle v_{colour} \leftarrow yellow \rangle$
      (ii) Broadcasts the UPDATE_NODE_COL($v_{ID}, yellow$) message..
30: A node $x$ on receiving UPGRADE_DOM($v_{ID}$) from node $v$:
      (i) Updates the colour of node $v$ in $N_1(x) \cup N_2(x)$ as $\langle v_{colour} \leftarrow black \rangle$
      (ii) Broadcasts the UPDATE_NODE_COL($v_{ID}, black$) message.
```

The detail distributed procedure for removing the redundant dominating nodes is given in the Algorithm 7.

4.4.5 Phase Transition

In any distributed algorithm, the phase transition is very important. We handle the phase transition of our distributed algorithm in the following way. Each node after creating its *1HopNebTable* and *2HopNebTable* should start the Distributed PDS Construction phase. A non-white node can begin the Distributed Steiner Tree Construction phase if it finds all its neighbours are non-white. The Distributed Steiner Tree Construction phase messages are queued by any node that still has white neighbours until all its neighbours become non-white. These queued messages are handled by the node when it finds all its neighbours have become non-white. In the Distributed Steiner Tree Construction phase, each black dominator and grey virtual-dominator keeps on sending CONNECTION_INFO_REQ messages while they find some of the yellow nodes participate in this phase (See the Step number 8 of Algorithm 6 for yellow node participation condition). If none of the yellow nodes participates in this phase, then the black or grey nodes will not receive any UPDATE_COMP_INFO messages. So, if a black or grey node do not receive any UPDATE_COMP_INFO message, then it can ensure that the Distributed Steiner Tree Construction phase is over. Any black dominator or grey virtual-dominator which finds the Distributed Steiner Tree Construction phase is over can start the last phase of the distributed CDS construction algorithm to remove the redundant dominators or virtual-dominators.

4.5 Algorithm Analysis

In this section, first, we find the approximation ratio of our proposed distributed algorithm. Later, we also find the time and message complexity of our proposed scheme. To do this, we use certain lemmas and theorems. The detail proofs of all of these can be found in this section.

Lemma 4.1 *At the end of distributed PDS construction phase, an MIS is formed by the black and grey nodes resulting from the Algorithm 5.*

Proof. *The distributed PDS construction terminates when each white node changes its colour. Algorithm 5 ensures that every yellow node is adjacent to at least one black node. Hence, by definition, the set of black and grey nodes form a Dominating Set. We can also observe that when a node changes its colour to black all its neighbours become yellow. Similarly, a node changes its colour to grey when it finds that all its neighbours have changed their colour to yellow. So, no node in the DS will find its neighbour in the set. So, the DS is independent. Also, DS is maximal because every omitted (yellow) node in the graph is dominated. Hence, by definition, the set of black and grey nodes form an MIS.* ■

Theorem 4.1 *Distributed DCMCDS constructs a PDS with the property: the distance between any pair of complementary subsets of the PDS have a distance of exactly two or three hops.*

Proof. *In order to prove this property about our constructed PDS, we first need to show that for a* $|\mathsf{PDS}| > 1$, *if* $\mathsf{u} \in \mathsf{PDS}$, *then the nearest black or grey neighbour of* u *in terms of number of hops is separated from* u *by at most three hops. We prove this by contradiction for any PDS whose cardinality is greater than 1. Let us assume that* $\mathsf{u} \in \mathsf{PDS}$ *and the nearest black or grey node to* u, *in terms of a number of hops, is separated from* u *by more than three hops. Let* v *be a strictly 2-hop neighbour of* u *which is not adjacent to* u. *If such a* v *does not exist, then it implies that all 2-hop neighbours of* u *are also its 1-hop neighbours, which in turn indicates that* u *dominates the whole connected graph. This contradicts* $|\mathsf{PDS}| > 1$. *So, for* $|\mathsf{PDS}| > 1$, *let* v *be a non-adjacent 2-hop neighbour of* u.

Case I: v is either a dominator or a virtual-dominator. *This implies that* v *is in PDS. So, we have a node* $\mathsf{v} \in \mathsf{PDS}$ *which is two hops away from* u. *This contradicts our assumption. So this case is not possible.*

Case II: v is neither a dominator nor a virtual-dominator. *By lemma 4.1, the PDS, which comprises all the black and grey nodes, is a maximal independent set. This implies that* v *is adjacent to at least one node in the PDS. Let* $\mathsf{w} \in \mathsf{PDS}$ *be adjacent to* v. *This means that* u *and* w *are 3-hop neighbours. This also contradicts our assumption. So, this case is not possible as well. Thus, our assumption does not hold true for any* $\mathsf{u} \in \mathsf{PDS}$. *This implies that* u *is separated from its nearest black or grey neighbour by at most three hops. Again, from lemma 4.1, it follows* *that any two nodes in the PDS are separated by at least two hops. Therefore, any pair of complementary subsets of the PDS have a distance of exactly two or three hops.* ■

Lemma 4.2 *The Distributed Steiner Tree construction phase of the proposed scheme (Algorithm 6) constructs a single connected component from the PDS obtained from the Distributed PDS Construction phase.*

Proof. *Here, we focus on the situation at the end of the connector selection phase. From the step 8 of Algorithm 6, we know that at the end of the connector selection phase, each yellow node* w *satisfies:*

(i) The conectionCount *of itself and its rivals is 1.*

(ii) There is no black node in its 2HopNebsTable *that does not occur in* $\mathsf{w}_{\mathsf{cdsList}}$.

We show by contradiction that all black, blue and grey nodes are in one component. Suppose otherwise, let A be one component and let B be a nearest different component (minimum number of hops away). Since, we have considered the network as a connected graph, A and B must be connected by one or a chain of yellow nodes. Let us consider the shortest chain *joining A and B.*

Case I*:* A and B are joined by a single yellow node, *let u be that node. So, the* connectionCount *of* u *will be 2, which violates the above condition (i).*

Case II*:* A and B are joined by a chain of two yellow nodes, *say* u *(adjacent to* A*) and* v *(adjacent to* B*). Dominatee* u *must be adjacent to at least one black node. If a black node is adjacent to* u *belongs to* B*, then* A *and* B *can be joined only by* u*. In that case, the shortest chain length joining components* A *and* B *will be one which contradicts the assumption of this case. In the other hand, if the dominator adjacent to* u *belongs to a separate component (other than* A *and* B*), then* B *no longer remains the nearest component to* A*. Therefore, these contradictions imply that* u *is adjacent to at least one black node that belongs to component* A*. Similarly,* v *is adjacent to at least one black node that belongs to the component* B*. Hence, without any loss of generality, we can consider the end nodes in both components* A *and* B *to be black. Let* u *be adjacent to a black node* x *of component* A *and* v *be adjacent to a black node* y *of component* B*. So* y *is a 2-hop neighbour of* u*. In this* *case, as* y *belongs to a different component,* u *will not find* y *in its* cdsList. *This violates the above condition (ii).*

Case III*:* A and B are joined by a chain of yellow nodes with a chain length greater than two*. Let us consider the second yellow node from the end (nearest to component* A*). If the second yellow node is adjacent to a black node belonging to* A*, then we could have made this the first node in the chain contradicting this as our choice of the shortest chain. If the second yellow node is not adjacent to* A*, then it must have a black node in its 1-hop neighbourhood that is not in* A*. Either this node is in* B *(contradicting that the shortest chain is more than 2), or it is in some other component different from both* A *and* B *(contradicting B being the nearest neighbouring component). So, in all these cases we have a contradiction. Therefore, we conclude that there is only one component at the end of the process. This concludes the proof.* ■

Theorem 4.2 *From a given a network, DCMCDS constructs a CDS in finite time period.*

Proof. *We present the correctness proof of our proposed scheme in two parts. First, we show that DCMCDS operates in a finite time and then, we prove that a CDS is definitely obtained. In order to prove that DCMCDS works in a finite time, we individually prove that all three phases of the algorithm namely distributed PDS construction, distributed Steiner Tree construction and distributed removal of redundant dominating nodes all takes a finite time. In each round, the distributed PDS construction algorithm searches for a potential dominator locally from the remaining white nodes in the local 2-hop neighbourhood. At every round, a white node is selected as the dominator and its colour is updated to black and all its adjacent nodes are updated as dominatees by changing their colours to yellow. When any white node discovers all its adjacent nodes to be yellow, it updates itself as virtual-dominator by changing its colour to grey. The PDS construction algorithm terminates when there is no white node left. We now prove by contradiction that this terminating condition must result in termination of the algorithm after a few rounds. Let* u *be a node which is still white.*

Case I*:* If all adjacent nodes of u are yellow*, then* u *must be a virtual-dominator. Hence,* u *must change its colour to grey.*

Case II: If a black node v is adjacent to u, *then* u *is a dominatee. Hence,* u *must change its colour to yellow.*

Case III: If there are one or more white nodes around u, *then one white node among them can be selected as dominator. If* u *is selected, then* u *changes its colour to black, otherwise cases I, II and III are followed until* u *changes its colour after a few rounds. Therefore,* u *will eventually change its colour from white. Thus, each white node will eventually change its colour either to black, yellow or grey accordingly completing the PDS construction. Lemma 4.2 shows that distributed Steiner Tree construction post-PDS selection results in a single connected component after a finite number of operations. Now, the selective removal of virtual-dominators and dominators (Algorithm 7) takes constant time as each grey node performs these steps independently. Hence, DCMCDS completes execution in a finite time. We next show that the proposed algorithm determines a CDS. Lemma 4.1 proves that the set of all black and grey nodes, obtained from PDS construction, is an independent dominating set (MIS). Lemma 4.2 shows that the Steiner Tree construction forms one connected black-blue-grey component. The latter itself is a CDS as it connects all the nodes in the PDS. It can also be shown that after the removal of the selected virtual-dominators and dominators the resulting component is still a CDS as the algorithm takes care of connection and coverage while removing these nodes from the CDS. This concludes the proof.* ■

Lemma 4.3 *In any UDG, each MIS size is upper-bounded by* $3.8|\mathsf{opt}| + 1.2$*, where* $|\mathsf{opt}|$ *is the size of the MCDS.*

Proof. *Directly from the result found in [92].* ■

Lemma 4.4 *Maximum number of Steiner nodes obtained from distributed DCMCDS is* $(1 + \ln 5)|\mathsf{opt}|$*, where* $|\mathsf{opt}|$ *is the size of any optimal CDS.*

Proof. *The proof is direct from the theorem 2 of [71].* ■

Theorem 4.3 *The size of the CDS obtained by DCMCDS is upper bounded by* $(4.8 + \ln 5)|\mathsf{opt}| + 1.2$*, where* $|\mathsf{opt}|$ *is the size of the MCDS.*

Proof. *In the first phase, DCMCDS constructs the PDS as an MIS. In the second phase, it finds the Steiner nodes to construct the Steiner Tree. In the last*

phase, it removes the redundant dominating nodes (both dominators and virtual-dominator) to reduce the CDS size.
Therefore, we have,

$$|\mathsf{CDS}| \leq |\mathsf{PDS}| + |\mathsf{Steinernodes}|$$

As PDS is an MIS, from lemma 4.3 and 4.4 we have:

$$\begin{aligned}|\mathsf{CDS}| &\leq 3.8|\mathsf{opt}| + 1.2 + (1 + \ln 5)|\mathsf{opt}| \\ &= (4.8 + \ln 5)|\mathsf{opt}| + 1.2\end{aligned}$$

Therefore, the performance ratio of DCMCDS is $(4.8 + \ln 5)|\mathsf{opt}| + 1.2$. ■

Theorem 4.4 *The time complexity of DCMCDS is* $\mathsf{O(D)}$ *time or* $\mathsf{O(D)}$ *rounds, where* D *is the network diameter.*

Proof. *In the proposed distributed scheme multiple dominators are selected in a single round. After the selection of a dominator from its 2-hop neighbours, all its adjacent neighbours become dominatees. In the next round, the algorithm selects the dominators from the remaining white nodes. The worst case occurs when in each round only one node is selected as the dominator or virtual-dominator, that means the dominator are selected one after another. The longest stretch of dominators and virtual-dominators should exist along the network diameter. Note that network diameter is the largest of all the shortest distances between any pair of nodes. In the worst case, as discussed, the number of rounds is at most* $\mathsf{O(D)}$. *Therefore, the time complexity for PDS construction is* $\mathsf{O(D)}$ *time or* $\mathsf{O(D)}$ *rounds. However, in the proposed scheme there is a chance of selection of multiple dominators in each round. So on average, the time complexity is much lower that* $\mathsf{O(D)}$. *Also in the second phase of distributed Steiner Tree construction, there is a chance of selection of multiple Steiner nodes in each round. After the selection of each single connector number of component decrease by one. By a similar argument, the Steiner Tree construction will also need* $\mathsf{O(D)}$ *time or* $\mathsf{O(D)}$ *rounds. In the last phase each dominators and virtual-dominators checks itself whether to upgrade or downgrade. All the dominators and virtual-dominators can do this checking simultaneously. Therefore, only one round is needed to do this.*

Hence the overall running time of the proposed algorithm is $\mathsf{O(D)}$ *time or* $\mathsf{O(D)}$ *rounds.* ■

Theorem 4.5 *DCMCDS has message complexity of* $\mathsf{O(nR)}$*, where* n *is the network size and* R *is the maximum between number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes.*

Proof. *We present the message complexity of each phase of the distributed DCMCDS to find out the message complexity of the whole algorithm. In the initialization and neighbourhood table creation phase, each node broadcasts the messages* HELLO*,* OWN_INFO *and* NEB_INFO *once each. Therefore, the message complexity of this phase is* $\Theta(\mathsf{n})$.

In the distributed PDS construction phase, the total number of DOMINATOR *and* VIRTUAL_DOMINATOR *messages broadcast is* $\Theta(|\mathsf{PDS}|)$*. Similarly, the number of* DOMINATEE *messages sent is* $\Theta(\mathsf{n} - |\mathsf{PDS}|)$*. So, for each* DOMINATOR *or* VIRTUAL_DOMINATOR *message, a total of* Δ DOMINATEE *and* UPDATE_NODE_COL *messages are generated in the worst case, where* Δ *is the maximum degree of all the nodes. As we have a total of* $|\mathsf{PDS}|$ *dominators/virtual-dominators, the total number of* DOMINATEE *and* UPDATE_NODE_COL *messages generated will be* $\Delta|\mathsf{PDS}|$*. For each* DOMINATEE *message, a total of* Δ UPDATE_NEB_INFO *and* UPDATE_NODE_COL *messages are generated in the worst case. For* $\mathsf{n} - |\mathsf{PDS}|$ DOMINATEE *messages, a total of* $\Delta(\mathsf{n} - |\mathsf{PDS}|)$ *number of* UPDATE_NEB_INFO *and* UPDATE_NODE_COL *messages will be generated.*

Therefore, the total number of UPDATE_NEB_INFO *and* UPDATE_NODE_COL *messages* $= \Delta(\mathsf{n} - |\mathsf{PDS}|) + \Delta|\mathsf{PDS}| - (\mathsf{n} - |\mathsf{PDS}|) = |\mathsf{PDS}| + \Delta\mathsf{n} - \mathsf{n} = \mathsf{O}(\mathsf{n}\Delta)$.

At the end of each round of this phase, some of the white nodes (those who find a change in the effective degree of their 1-hop neighbours in last round) broadcast their updated neighbour information through the NEB_INFO *message. So, the message count of this message will be* $\mathsf{O(nR_{PDS})}$*, where* $\mathsf{R_{PDS}}$ *is the number of rounds needed to construct the PDS. Hence, the message complexity of this phase is* $\mathsf{O(nR_{PDS})}$*, assuming* $\mathsf{R_{PDS}}$ *is greater than* Δ.

To find out the message complexity of distributed Steiner Tree construction phase, let us assume the algorithm runs for $\mathsf{R_{ST}}$ *rounds to interconnect the PDS nodes. In each round, the total number of* CONN_INFO_REQ *and* CONN_INFO_REP *messages sent is* n*. So the total count of these two messages in all rounds is* $\mathsf{O}(\mathsf{nR_{ST}})$*. The* COMP_RIVAL_INFO *messages are sent by each node of the components once in each round. In the first round, the count of this message is* $|\mathsf{PDS}|$*. It keeps on increasing up to* $|\mathsf{CDS}|$ *in the last round. So, the total count of this message in all rounds is* $\mathsf{O}(\mathsf{R_{ST}}|\mathsf{CDS}|)$*. As the* RIVAL_INFO *message is sent by the component members twice in each round, the total count of this message in all rounds is* $\mathsf{O}(\mathsf{R_{ST}}|\mathsf{CDS}|)$*. The number of* CONNECTOR *messages sent in all rounds is* $\Theta(|\mathsf{CDS}| - |\mathsf{PDS}|)$*. The* UPDATE_COMP_INFO *message is sent by the component members once in each round. So, the total number of* UPDATE_COMP_INFO *messages sent in all rounds is* $\mathsf{O}(\mathsf{R_{ST}}|\mathsf{CDS}|)$*. For each dominator,* Δ *number of* UPDATE_NODE_COL *messages are sent. Total* UPDATE_NODE_COL *messages sent in all rounds is* $\mathsf{O}(\mathsf{n}\Delta)$*. Hence, the total message complexity of this phase is* $\mathsf{O}(\mathsf{nR_{ST}})$ *assuming* $\mathsf{R_{ST}}$ *is greater than* Δ*.*

In the last phase of removing redundant dominating nodes, some of the virtual-dominators who want to upgrade themselves to dominators, send the UPGRADE_DOM *message, only once. So, the total count of this message sent is* $\mathsf{O}(|\mathsf{VD}|)$*. Similarly, the dominators or virtual-dominators who want to downgrade themselves to dominatees send the* DOWNGRADE_DOM *message only once. So, the total count of this message is* $\mathsf{O}(|\mathsf{CDS}|)$*. A dominator to downgrade itself needs to send the* ALT_DOMINATOR_REQ *message once to its dominatees. The dominatees send the* ALT_DOMINATOR_REP *message for their dominators. So, the total count of these two messages is* $\mathsf{O}(|\mathsf{DS}|)$*. The dominators which do not receive the TRUE reply through the* ALT_DOMINATOR_REP *messages from all their dominatees, withdraw their intent to become dominatees by sending the* ALT_DOMINATOR_REQ_CANCEL *message. So, the count of this cancel message sent is* $\mathsf{O}(|\mathsf{DS}|)$*. For each upgrade/downgrade of virtual-dominators, and downgrade of dominators,* Δ *number of* UPDATE_NODE_COL *messages are sent. So, the total* UPDATE_NODE_COL *messages sent is* $\mathsf{O}(|\mathsf{DS}|\Delta)$*. Hence, the message complexity of this phase is* $\mathsf{O}(\mathsf{n}\Delta)$*.*

Thus, the overall message complexity for DCMCDS is $\mathsf{O}(\mathsf{nR})$*, where* R *is the maximum of* $\mathsf{R_{PDS}}$ *and* $\mathsf{R_{ST}}$*. This completes the proof.* ■

4.6 Simulation Results

In this section, we present the results of the simulations conducted by us to compare our proposed scheme with the existing approaches. The WSN is modelled in a fixed area of dimension 100×100 square units. We have generated the hosts randomly by choosing their abscissa and ordinate using a uniform random number generator. The transmission range of each node was taken as R for each node. Two nodes are connected if their distance is less than equal to R. In the entire experimentation, we considered connected networks only. In the simulation, we considered $\mathsf{R} = 25$. The proposed algorithm is run for 100 times for different network sizes from 50 to 250. The average results are reported in the figures and table. We conducted the entire simulation in NS-2, a network simulator for wireless networks.

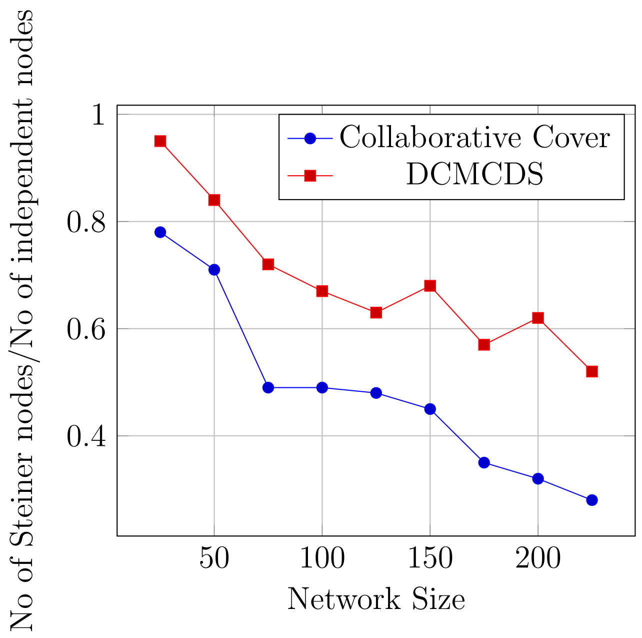


Figure 4.1: Performance comparison of number of Steiner nodes and number of independent nodes.

In our first experiment, we found the average effective degree of a connector. It is the ratio of the total number of connectors to the number of accepted PDS nodes (connector and virtual-connector). We calculated the same ratio for the

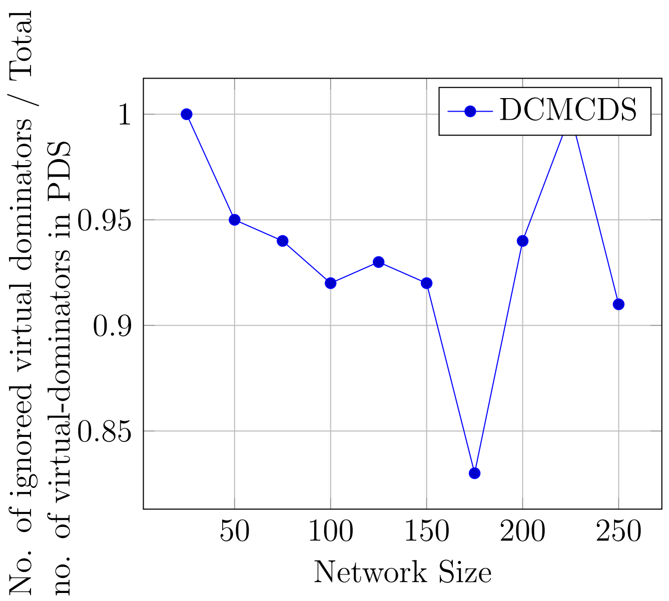


Figure 4.2: Ratio of ignored virtual-dominators to total virtual-dominators in pseudo-dominating set for different network sizes

best existing algorithm, collaborative cover heuristic [72], for the network sizes varying from 25 to 225. The results are shown in fig. 4.1. We found that the average effective degree for collaborative cover is 0.3 and for our proposed scheme it is 0.5. That means in the collaborative cover a Steiner node connects more than three PDS nodes whereas in our algorithm a Steiner node connects nearly two PDS nodes. This is a significant result and it has many positive consequences. Less effective degree of a connector indicates that the connector is less loaded. This enhances the lifetime of the network.

In our second experiment, we did an analysis of how much PDS nodes change their status in the post-Steiner Tree construction phase and what is its impact on the reduction of overall CDS size. For varied network sizes from 25 to 250, we determine the ratio of total virtual-dominators being downgraded to dominatee and its impact on the reduction of CDS size. The results in fig. 4.2 shows that almost all of the virtual-dominators are downgraded to dominatee. Very few of them retained their earlier state because they are actually bridging two disjoint components of the CDS. The results found in fig. 4.3 implies that for networks of

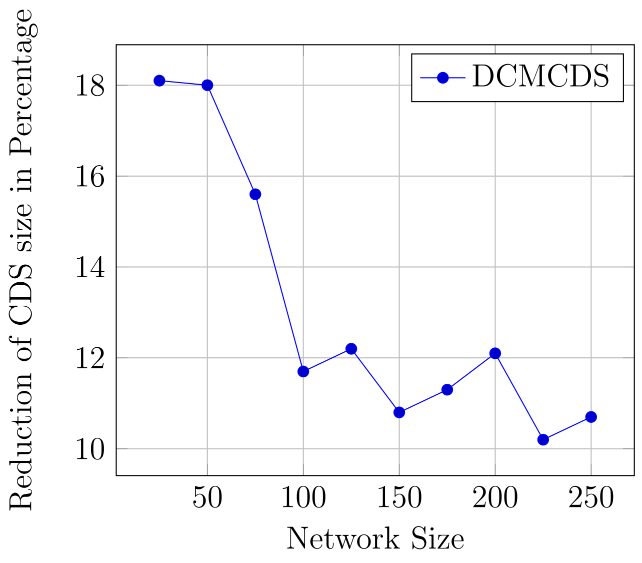


Figure 4.3: Reduction in CDS size after discarding redundant dominators and virtual-dominators post-Steiner Tree construction for different network sizes

larger sizes, by changing the status of some of the PDS nodes from dominator or dominatee according to the specified criteria reduces the CDS size by 10%.

In the next experiment, we compare the performance of our proposed scheme with the existing CDS constructions techniques found in [62], [65], [71], [72], [73] in two steps. Firstly, we found the CDS for the network where the nodes are distributed randomly. We considered the network of sizes 20, 40, 60, 80 and 100. The results found are shown in fig. 4.4. It is clear from the result that our scheme outperforms all above mentioned CDS construction techniques in identifying smaller CDSs. Our result is at least 16% better than the collaborative cover heuristic which is best among the above mentioned algorithms. We are getting better results because of our "Steiner Tree Construction" phase and "Removal of redundant dominator" phase. During Steiner Tree construction we select the connectors which connects maximum number of components. In the redundant dominator removal phase it omits the redundant dominating nodes cleverly without any loss in coverage or connectivity.

Secondly, we compare our proposed scheme with the collaborative cover based heuristic [72], for ideal uniform distribution of nodes. Illustrations provided in [72] shows that the coverage based heuristic achieves significantly better results in op-

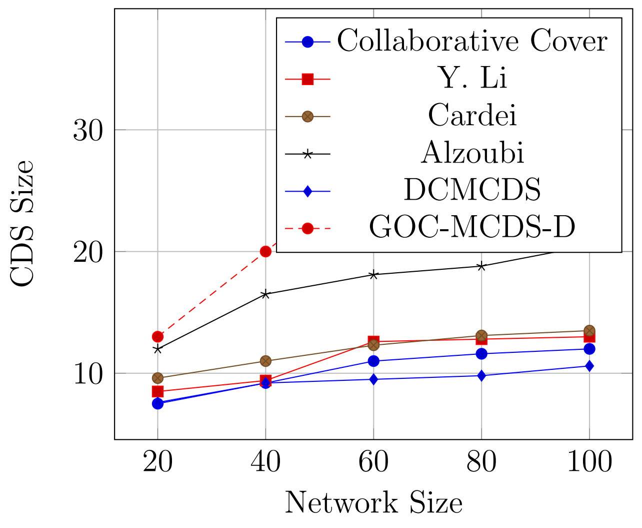


Figure 4.4: Performance comparison of CDS construction algorithms

Table 4.1: Comparison with Collaborative Cover for uniform distribution of nodes

For Uniform Distribution of Nodes	**Network Sizes**				
	50	100	150	200	250
Ratio of CDS size by DMCDS to CDS size by collaborative cover	0.74	0.75	0.67	0.78	0.84

timizing CDS size for uniform distributions by identifying optimal sub-structures than previously reported degree-based schemes. We vary the network size from 50 to 250 and determine the ratio of CDS sizes obtained by the DCMCDS scheme and collaborative cover heuristic for uniform distribution. The results summarized in Table 4.1 illustrate that DCMCDS produces smaller CDS sizes for the above-mentioned distribution.

A good CDS construction algorithm not only should construct the CDSs of smaller sizes but also should construct it in less time. In our next experiment, we compare the number of rounds needed to construct a CDS by our proposed scheme with 8-approximate CDS algorithm [65]. We use the number of rounds as the metric to compare the performance of our scheme with the existing approaches where, a round is a total time needed by the nodes to receive messages from their neighbours in the previous round, execute local computations and consequently send messages to their neighbours. The reason for comparing with only 8-approximate

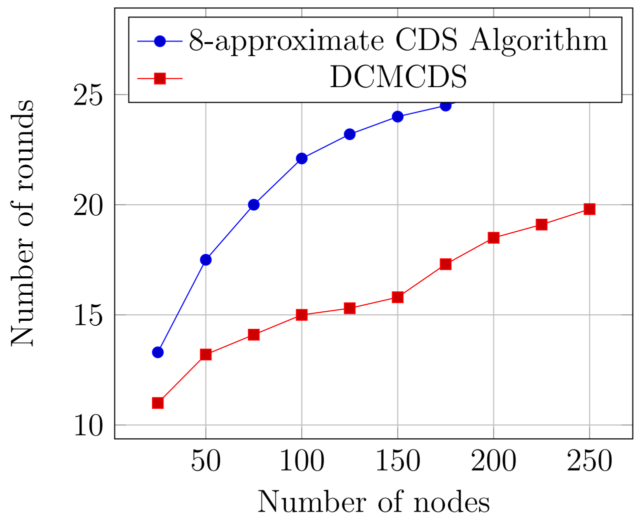


Figure 4.5: Performance comparison of number of rounds in CDS construction

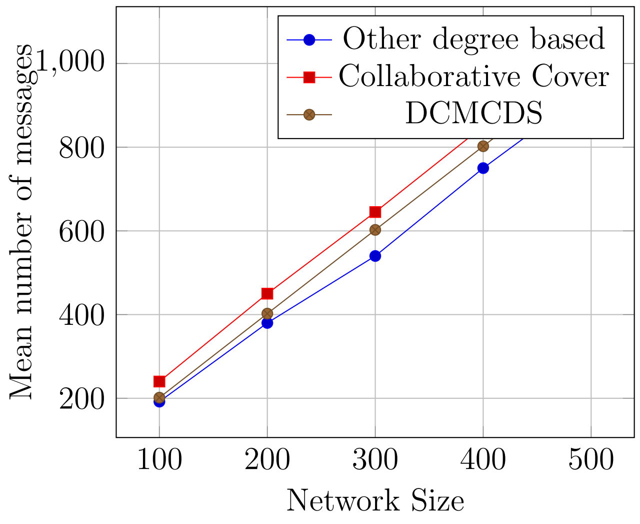


Figure 4.6: Comparison of message exchanges in CDS construction algorithms

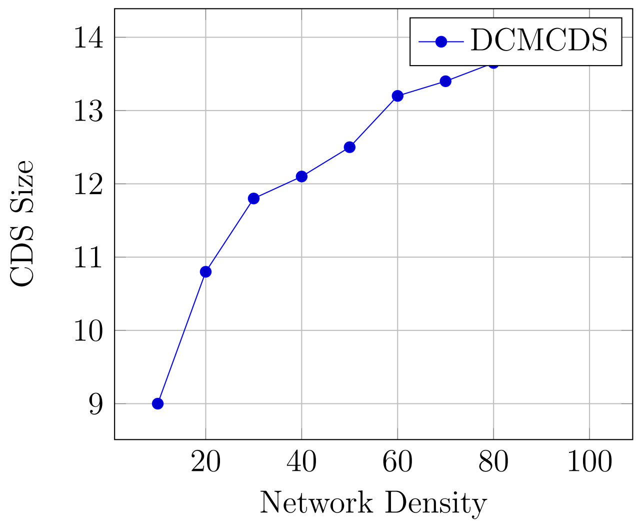


Figure 4.7: CDS Size corresponding to DCMCDS for different network densities

CDS algorithm [65] is, it represents the class of CDS construction techniques which first forms an MIS and then connect the MIS nodes greedily using different techniques. The S-MIS approach [71] and collaborative cover heuristic [72] belongs to this class of algorithms. However, the number of rounds required by all of the algorithms is hardly any different. For the comparative study, we varied the size of the networks from 25 to 250. For each of the network size, we find the number of rounds for constructing the CDS both by DCMCDS and 8-approximate CDS algorithm. The comparison is shown in fig. 4.5. Theoretically, the upper bound of both these algorithm is $O(D)$ rounds, D, being the network diameter. However, our proposed scheme needs fewer rounds than 8-approximate CDS. For large network sizes the number of rounds required by the proposed algorithm is nearly $(\frac{2}{3})\mathsf{rd}$ of that required by 8-approximate CDS. This reduces 33% execution time. We also observe that the slope of the curve representing the proposed scheme is significantly smaller than that for 8-approximate CDS. This means with an increase in network size, the corresponding increment in the number of rounds is much smaller for the proposed scheme than other CDS construction techniques.

Next, we analyse the message exchanges needed for CDS construction of DCMCDS by comparing with previous degree-based [65] and collaborative cover [72] approaches. We vary the network size N from 100 to 500 and compare the mean number of messages required. From Fig. 4.6, one can observe that the mean number of broadcasted messages by our proposed algorithm is closer to degree-based approach [65] and better than the collaborative cover heuristic [72]. The result is justified because the message complexity of 8-approximate degree based CDS scheme is $O(n\Delta)$ and that of collaborative cover is $O(n\Delta^2)$, where n is the number of nodes in the network and Δ is the maximum degree of a node. The message complexity of DCMCDS is $O(nR)$, where R is the number of rounds. Although the message complexity of both degree-based 8-approximate CDS algorithm [65] and DCMCDS are linear, the former requires slightly fewer messages than the latter. This is because 8-approximate CDS technique [65] uses only 1-hop connectivity information whereas DCMCDS uses 2-hop neighbourhood knowledge to obtain a better CDS. So from this experiment, we can conclude that DCMCDS constructs the CDSs of smaller sizes using a slightly higher expense of a number of messages exchanged as compared to previous degree-based CDS construction techniques.

The advantages of DCMCDS is that it identifies much smaller CDSs than 8-approximate CDS or collaborative cover for all network sizes. In fact, after the network density crosses a certain threshold, the CDS size will hardly change much irrespective of how many nodes more added to the fixed deployment area. Fig. 4.7 explains this scenario. For DCMCDS this threshold is reached earlier than collaborative cover or 8-approximate CDS. Therefore a significant increment in average dominator degree coupled with a marginal increment in CDS size for DCMCDS would lead to a slower rise in energy dissipation for small as well as large scale networks.

4.7 Summary

In this chapter, we presented our novel distributed degree-based greedy approximation algorithm for Minimum Connected Dominating Set problem where there is no specific initiating node. The algorithm first identifies smaller size, MISs, using PDS constructed in a distributive manner. Next, Steiner Tree is constructed to connect the PDS nodes in a distributed manner. In the later stage, the algorithm selectively removes some nodes of the Steiner Tree to minimize the CDS size. The time and message complexities of our algorithm are $O(D)$ and $O(nR)$ respectively, where n is the network size, D is the diameter of the network and R is the maximum between the number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes. The approximation ratio of our proposed scheme is found to $(4.8 + \ln 5)|\mathsf{opt}| + 1.2$, where $|\mathsf{opt}|$ is the size of an optimal CDS. The simulation results show that the proposed algorithm constructs smaller CDSs in comparison to all other existing CDS construction algorithms for both uniform and random distribution of nodes. Also, it needs a small number of rounds than all degree-based algorithms. The distributed nature of our algorithm makes it useful for situations where a centralised approach is not suitable. The proposed algorithm can construct CDS construction in the network of nodes with different transmission ranges.

CHAPTER 5

Distributed Maintenance of CDS

5.1 Overview

In a wireless sensor network, the nodes are operated through their battery power. They communicate either through single hop or through multiple hops. The nodes consume their battery power while performing computation and communication. So, after a series of communications, the nodes become powerless and hence are not be able to communicate. Therefore, researchers are interested to design protocols to optimize the power consumption of the nodes to extend their battery life. Unlike a wired network, a wireless ad hoc network does not have a predefined network infrastructure. In a wireless network, when a node is interested to send a message to some other node, which is not within its communication range, it broadcasts the packet. The intermediate nodes keep on broadcasting the message until it is not reached the destination node. This method of message passing from any source to destination could cause broadcast storm which wastes bandwidth, time and processing power of the nodes. One way of avoiding this problem is to use Connected Dominating Set (CDS) as virtual backbone [43]. Each node in the network becomes either a member of the CDS or adjacent to at least one of the CDS nodes. Each non-CDS node should keep track of its adjacent CDS nodes.

To reduce the power consumption, each non-CDS node should keep its radio off by default. When it becomes ready to send any data it should make its radio on, send the data to its adjacent CDS node and should again make its radio off after sending. The CDS nodes should work as a virtual backbone for sending messages to any destination node. So, by restricting the routing activities only to CDS nodes we are able to reduce a significant amount of messages related to routing.

The CDS nodes deplete their energy faster than non-CDS nodes due to their extra computation and communication load. After a certain period of time, these CDS nodes would run out of their battery and hence would not be able to handle the routing responsibilities anymore. In that situation, either we should switch over to a new CDS [45, 46] or repair the current CDS by finding some other non-CDS nodes to hand over the responsibilities of the failed CDS nodes. Also, it is a well-known fact that the battery performance can be greatly improved by using pulsed discharge instead of constant discharge [48, 49]. It is better to handover the routing responsibilities of the CDS node, which is about to fail in the near future rather than waiting for its failure. This allows the recharge recovery effect in electrochemical batteries in extending their lifetime. If a node is chargeable from other sources like sunlight [50] then it can recharge itself for future use. A CDS node may also fail due to processor failure, radio failure, etc. So, whatever may be the reason for the failure of a CDS node, it may disconnect the CDS, as a result, the data transmission may stop as well. In this scenario, it is not wise to re-construct the CDS fresh since very few nodes are affected and reconstruction would be performed on the entire network, so it is wastage of resource. Therefore, it is a better approach to repair the CDS by assigning new responsibilities to some of the non-CDS nodes and the affected CDS nodes are freed from their earlier responsibilities. Maintenance of a CDS is initiated by a particular CDS node when it finds that its battery power has reduced below a threshold value. The affected CDS node after relieved as a backbone node can recharge itself from any available resources like sunlight and make itself ready for future use. However, if a CDS node fails due to any reason other than power failure, the nearby nodes should take the initiatives to repair the CDS. In the literature, one can find many approaches for CDS construction. However, algorithms for CDS maintenance are very rare.

In this chapter, we propose a new distributed CDS maintenance approach which we name as **D**istributed **M**aintenance of **C**onnected **D**ominating **S**et (DM-CDS). Our proposed approach handles the following situations: (1) a CDS node finds its battery power has reduced below a threshold value due to which it is about to fail in near future (2) a CDS node has already failed due to any reason (3) a non-CDS node finds its battery power has reduced below a threshold value due to which it is about to fail in near future (4) a non-CDS node has already failed due to any reason (5) a non-CDS node has become ready to sense the required data after recharging. The proposed approach also takes care of the situation in which the CDS becomes disconnected by the failure of any CDS node. When the connectedness property of the CDS is lost due to the failure of any CDS node, multiple components are formed containing some of the CDS nodes. In this work, the non-CDS nodes come forward to connect these components by changing their role. According to the knowledge of the authors, this is the first CDS maintenance algorithm which handles multiple components. Simulation results show that the proposed algorithm is able to repair the CDS in all possible above mentioned situations. The proposed distributed CDS maintenance scheme only changes the current CDS by a factor of Δ, where Δ is the maximum degree of a node in the entire network. Its time complexity is $\mathsf{O}(\mathsf{C})$, where C is the size of the largest component of the network during the application of the proposed CDS maintenance algorithm. It has a linear message complexity of $\mathsf{O}(\mathsf{V})$, where V is the network size.

The remaining of the chapter is organized as follows. In the following section (Section 5.2), we discuss the motivation and objective of the proposed work. In Section 5.3 we state our assumptions regarding the development of the ad hoc network model. Section 5.4 discusses the proposed distributed CDS maintenance algorithm in detail with a sufficient number of working examples. The analysis of the proposed distributed algorithm is discussed in Section 5.5. Supporting simulation results are given in Section 5.6. Finally, the conclusion is presented in Section 5.7.

5.2 Motivations and Objectives

5.2.1 Motivations

In the literature, one can find many distributive CDS construction algorithms like [62, 65, 71, 72] to achieve a good approximation ratio. Many researchers are trying to improve the time and message complexities of CDS construction algorithms also. A CDS works efficiently as long as there is no issue of battery power in any of the CDS nodes. A node may fail either due to lack of its battery power or due to hardware failure. In case of failure of any CDS node, the CDS may become disconnected and may not be able to pass the sensed information to the sink or base station. In this situation we have two options, either construct a new CDS or repair the existing CDS. Construction of new CDS is not wise because the current CDS is useless due to very few nodes. It is rare that a group of CDS nodes fail together at a time. Therefore, it is wise to repair the current CDS rather than constructing a fresh CDS in this situation. To demonstrate this point, let us consider a graph shown in Fig. 5.1. In this graph, each node represents one sensor node and an edge between a pair of nodes indicate that these sensor nodes are within the communication range of each other. Suppose, the current network contains a CDS consisting of node 2, 5, 8 as shown in Fig. 5.2. Now, suppose a CDS node 8 fails. In that case, the current CDS would be of no use. We can construct a new CDS as shown in Fig. 5.3. However, without constructing a new CDS if we could replace the failed node 8 by node 7 as a CDS node (shown in Fig. 5.4), then we can avoid many changes in the network. Clearly, it is a better technique than constructing the entire CDS once again.

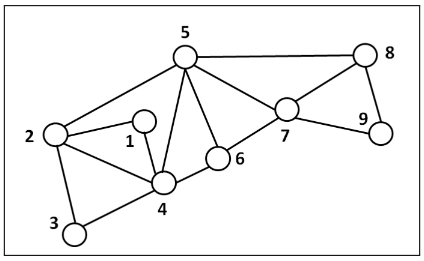


Figure 5.1: Network showing a set of nodes

Further, if a CDS node's battery power is reduced below a level, then without waiting for the complete discharge of its battery if we could allow that CDS node to downgrade itself from a dominator to a dominatee then it could again work as a dominator after recharge. This is better for the lifetime of that particular node and hence for the entire network.

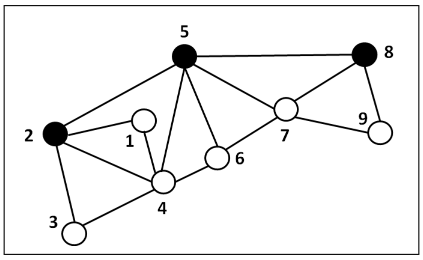


Figure 5.2: CDS of the network shown in Fig. 5.1

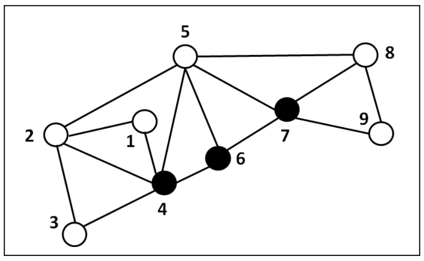


Figure 5.3: An alternative CDS of the network shown in Fig. 5.1

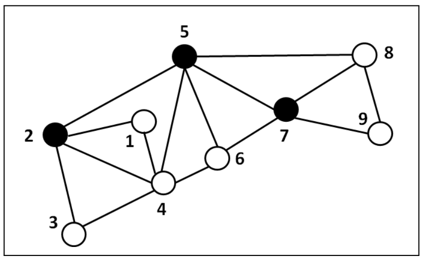


Figure 5.4: Demonstration on repair of CDS due to the failure of node 8 shown in Fig. 5.2

The role of a dominatee is to sense the data and send the information to its dominator. So, when a dominatee is discharged below a certain level we could allow the dominatee to enter into sleep mode, during which it can recharge and make itself ready to sense data again in the future. In the case of depletion of

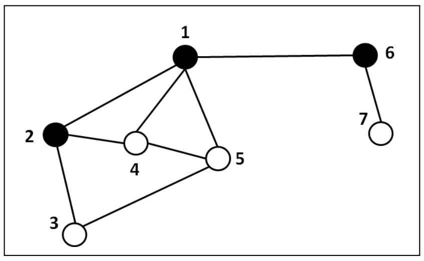


Figure 5.5: Demonstration on repair of CDS in case of failure of a dominatee

battery power or complete failure of a dominatee, maintenance of CDS is necessary. To understand this point, let us consider a graph as shown in Fig. 5.5 with a CDS formed by nodes 1, 2 and 6. Node 3, 4, 5 and 7 are dominatees. Suppose node 7 either completely failed or discharged below a level. In that case, node 7 would no longer sense the data for the network. Node 6 which was working as the dominator of node 7 only, would not have any other dominatees. So, if node 6 can be downgraded to a dominatee, then it can save its battery power. In the later stage, when node 7 after recharging would be ready to work as a dominatee, node 6 can again become a dominator. From this, it is very clear that downgradation of a dominatee to a normal node is also needed in CDS maintenance.

5.2.2 Objectives

Maintenance of CDS in a static wireless network: Consider a situation where initially all the nodes in the wireless network are static. Let us assume, initially, the virtual backbone (CDS) for the entire network was working fine and then a single node failed. If the failed node is a CDS node then either the CDS may be disconnected or the network may have some nodes which do not have any dominator at all. In that case, the CDS need to be repaired by assigning new responsibilities to some of the non-CDS nodes. In case of failure of the non-CDS node, there is a need for CDS maintenance also. The CDS maintenance algorithm needs to be executed for failure of both a CDS and non-CDS node. The maintenance algorithm needs to be executed in only those nodes that have failed or those that have been affected due to the failure of some other nodes. So our third objective is to design a distributed CDS maintenance algorithm which could repair the CDS by changing the role of very few nodes so that the new CDS would work fine for the current state of all the nodes.

5.3 Network Model for Distributed CDS Maintenance

In order to describe the proposed algorithm the following assumptions are made:

1. The nodes do not have any geometric or topological information. They do not have knowledge of their distances to their neighbours.
2. Each node has a unique ID.
3. All the nodes are deployed in a 2-D plane and their maximum communication ranges are the same.
4. The resultant topology of the network is modelled as a unit disk graph (UDG) with the transmission range of the nodes considered as one unit i.e. two nodes are connected by a wireless link if their distance does not exceed the transmission range. All the links are bidirectional.
5. The communication overhead due to interference is negligible.
6. Each node maintains its 1-hop and 2-hop neighbours' information in it.
7. Initially some CDS is working fine.
8. The nodes belong to the CDS transmits the data and the non-CDS nodes sense the information from their surroundings.
9. At a time only one CDS node fails due to shortage of its battery power or any other reason.
10. Node density remains almost the same (not change drastically) during the network lifetime.
11. The nodes whose battery power is reduced currently can always join as a CDS node later after recharging.

12. Each node knows its remaining battery power. The maximum and minimum battery power of a node is 1.0 and 0.0 respectively.

5.4 Distributed CDS Maintenance by DMCDS

This section describes the proposed distributed CDS maintenance approach DM-CDS. It is assumed that an already constructed CDS is working fine as a virtual backbone in a wireless network and a single node either failed due to any reason or about to fail due to depletion of its battery power. Failure of CDS nodes is frequent since these nodes have some extra load of computation and communication, due to which their battery depletes at a faster rate. Therefore, during processing and transmitting the data some of the CDS nodes may find their battery has been drained out below a predefined threshold value and would not able to participate in the data transmission process. A CDS node may also fail due to processor failure, radio failure etc. Failure of any CDS node may disconnect the CDS, as a result, the data transmission may stop as well. In this scenario, it is not wise to re-construct the CDS fresh since very few nodes are affected and re-construction will be performed on the entire network, so it is wastage of resource. Therefore, it is wise to repair the CDS by assigning new responsibilities to some other nodes and the affected CDS nodes are made free from their earlier responsibilities. Maintenance of CDS is initiated by each CDS node when it finds that its battery power has reached below a predefined threshold value. However, if a CDS node has already failed due to any reason, its nearby nodes should take the initiative to repair the CDS. An affected CDS node may recharge itself from any of the resources like sunlight and make itself ready to work as a CDS node in the future. A non-CDS node or dominatee can also deplete its battery power below the predefined threshold value or it can fail altogether. If a dominatee fails, and it was the only dominatee of its dominator, then that dominator can also downgrades itself to a dominatee. Therefore, maintenance of CDS is also necessary in case of failure of a non-CDS node.

In the proposed approach, two thresholds for battery power are considered: upper threshold θ_1 and lower threshold θ_2, where $0.0 < \theta_2 < \theta_1 < 1.0$. For example we may take $\theta_1 = 0.4$ and $\theta_2 = 0.25$. Initially, the battery power of all nodes is the same (1.0). So, all nodes are eligible to work as either a dominatee or a dominator. When the battery power of a dominator is reduced below θ_1, it would not be interested to work as a CDS node and may work as a dominatee. During that period, it may re-charge itself and again becomes eligible for a dominator when its battery power becomes more than 90%. Similarly, a dominatee enters into sleep mode when its battery power is reduced below θ_2. In the sleep mode, it never senses any data and only charges its battery. After re-charging, it would be again ready to work as a dominatee when its battery power reaches above θ_1.

During the discussion of CDS maintenance, it is assumed that the network already has a CDS running successfully. All the nodes store their 1-hop and 2-hop neighbourhood information in two tables *1HopNebTable* and *2HopNebTable* respectively. Each node knows about its own status (dominator or dominatee) and its 1-hop and 2-hop neighbours status. A node notifies to its neighbours about any changes in either its own status or its neighbours status. Whenever a CDS node receives any information from any of its CDS neighbours it circulates that information immediately among its other CDS neighbours.

The proposed algorithm takes care of each of the following situations:

1. A CDS node (dominator) find its battery power has reached below the upper threshold value.

2. A CDS node (dominator) has completely failed.

3. A non-CDS node (dominatee) finds its battery power has reached below the lower threshold value.

4. A non-CDS node (dominatee) has completely failed.

5. After recharging, a node is interested to work as a dominatee.

In the following discussion, both dominators and connectors are considered as CDS nodes, the non-CDS nodes are generally the dominatees if their battery power is more than the lower threshold value.

In the proposed distributed CDS maintenance algorithm, the nodes send and receive the following messages:

- DOMINATOR: A node sends the message DOMINATOR when it becomes a dominator.
- DOMINATEE: A node sends the message DOMINATEE when it becomes a dominatee.
- CRTD: A node u sends the message CRTD (Change Role To Dominator) to a node v to notify node v about its new role as a dominator.
- RYS: A CDS node sends the message RYS (Rescue YourSelf) to a non-CDS node to be attached with other CDS node (if possible).
- UCI: A node sends the message UCI (Update Component-Id) to its neighbours when its component-ID has changed.
- CCID: A CDS node sends the message CCID (Change Component-ID) to its CDS neighbours to change their component-ID.
- DPROMREQ: A non-CDS node (dominatee) sends the message DPROMREQ (Dominatee PROMotion REQuest) to its adjacent CDS nodes to inform them about its willingness to become a dominator.
- GDPROMREQ: Two adjacent non-CDS nodes (dominatees) send the message GDPROMREQ (Group Dominatee PROmotion REQuest) to their respective adjacent CDS nodes to inform them about their willingness to become dominators.
- GETPOW: A node sends the message GETPOW (Get Power) to its adjacent node to inform about its current battery power and also requests for the current battery power of its adjacent node.
- POW: A node sends the message POW (Power) to its adjacent node to inform its current battery power.
- TOSLEEP: A dominatee node sends the message TOSLEEP to its adjacent CDS node when its battery power has reached below a lower predefined threshold value.

In the following subsections, we discuss each of the cases that our proposed algorithm handles in detail.

5.4.1 A CDS node finds its battery power has reached below the upper threshold value

When a CDS node D_i finds that its battery power has reached below the upper threshold value θ_1, it would be interested to downgrade itself from dominator to dominatee to stop the quick discharge of its battery. The CDS node $\mathrm{D_i}$ itself initiates the CDS maintenance process by executing Algorithm 8. First, D_i notifies to its dominatees (if any) to be connected with other alternative CDS nodes by sending the RYS message. Next, the dominator D_i checks about its connectedness with other CDS nodes. If D_i finds that it is adjacent to only one CDS node D_j, then the dominator D_i becomes a dominatee and starts recharging. It notifies its new role by sending the DOMINATEE message. If D_i finds that it is adjacent to multiple CDS nodes, then the failure of D_i would make the CDS disconnected assuming there is no cycle in between the CDS nodes. In that case, multiple components would be created. So the dominator D_i sends the message CCID to its adjacent CDS nodes to form new components. The CDS node which receives the CCID message changes its component-ID to its own node-ID and forwards the new component-ID to other members of the same component by sending the UCI message. The other members update their component-ID and forward the UCI message. In this way, all the members of each newly created components would update their new Component-ID which would be used later to combine all components together. Later, the dominator D_i becomes a dominatee and starts recharging. It notifies its new role by sending the DOMINATEE message. One of the adjacent CDS nodes of D_i becomes the dominator of the new dominatee D_i.

5.4.2 A CDS node has completely failed

When a CDS node fails completely due to some reason, its adjacent nodes (both CDS and non-CDS) initiate the maintenance process. If a CDS node notices that its adjacent CDS node has failed, it executes the Algorithm 9. When a dominator

Algorithm 8 To be executed by any CDS node $\mathrm{D_i}$ when its battery power has reached below a predefined upper threshold value θ_1.

```
1: The dominator $\mathrm{D_i}$ sends the message RYS to all its dominatees (if any).
2: if the dominator $\mathrm{D_i}$ is adjacent to exactly one CDS node $\mathrm{D_j}$ then it executes
   the following steps:
    (a) The dominator $\mathrm{D_i}$ becomes a dominatee and sends the DOMINATEE
        message to all its neighbours.
    (b) The CDS node $\mathrm{D_j}$ becomes the dominator of the new dominatee $\mathrm{D_i}$.
3: else if the dominator $\mathrm{D_i}$ is adjacent to more than one CDS nodes then it
   executes the following steps:
    (a) The dominator $\mathrm{D_i}$ informs its adjacent CDS nodes to change their
        component-IDs to their own node-IDs by sending the CCID message.
          (i) The CDS nodes on receiving the message CCID, should change their
              component-ID and pass their updated component-ID to their neigh-
              bours by sending the message UCI.
         (ii) A CDS node on receiving the message UCI, should update its
              component-ID and send the same CCID message to its neighbours.
        (iii) A non-CDS node (dominatee) on receiving the message UCI, updates
              its dominators component-ID in its 1HopNebTable and broadcasts it
              to its neighbours to update their 2HopNebTable.
    (b) The dominator $\mathrm{D_i}$ becomes a dominatee and sends the DOMINATEE
        message to all its neighbours. One of the adjacent CDS nodes of $\mathrm{D_i}$
        becomes the dominator of the new dominatee $\mathrm{D_i}$.
4: end if
```

D_i finds that its adjacent CDS node D_j has failed, it first checks whether the failed dominator was a leaf node of the CDS or not. If it was a leaf node that means D_i is the only adjacent CDS node of D_j, then D_i updates its status in its *1HopNebTable*. It also updates the *2HopNebTable* if there was any 2-hop neighbour of D_i through D_j. However, if D_i finds that there is some other CDS neighbour of D_j present in the CDS, then it understands that the failure of D_j has created multiple numbers of components and it is in one of them. CDS node D_i creates a new component by updating the component-ID as its own node-ID and forwards the updated component-ID to its other adjacent CDS neighbours through UCI message. If a CDS node receives the UCI message it updates its component-

ID and again forwards the UCI message to its neighbours. If a non-CDS node receives the UCI message it only updates its CDS neighbour's component-ID in its *1HopNebTable* and forwards it to its neighbours to update their *2HopNebTable*. So, in this manner, all the members of the component to which D_i belongs would update their component-ID. In a similar manner, the other CDS neighbours of D_j would be able to pass the new Component-ID among other members of the same component. The adjacent non-CDS neighbours of each of the newly created components would come to know about the change in component structure by receiving UCI messages. In the latter stage, they would try to connect some of the partitioned components by using Algorithm 11.

Algorithm 9 To be executed by any CDS node D_i when it finds one of its CDS neighbour D_j has failed.

```
1: if a dominator $D_i$ finds that its neighbouring dominator $D_j$ has failed and $D_j$'s only adjacent CDS node was $D_i$ then $D_i$ removes the node $D_j$ from its 1HopNebTable and forwards it to its neighbours to update their 2HopNebTable.
2: else if a dominator $D_i$ finds that its adjacent dominator $D_j$ has failed and $D_j$ was adjacent to other dominators also then $D_i$ executes the following steps:
     (a) Removes the node $D_j$ from its 1HopNebTable and forwards it to its neighbours to update their 2HopNebTable.
     (b) The dominator $D_i$ updates its own component-ID to its node-ID and pass its new component-ID to its neighbours by sending the UCI message.
     (c) A CDS node on receiving the UCI message updates its component-ID and sends the same message UCI to its neighbours.
     (d) A non-CDS node (dominatee) on receiving the UCI message, updates its dominators component-ID in its 1HopNebTable and sends it to its neighbours to update their 2HopNebTable.
3: end if
```

When a non-CDS node (dominatee) E_i either notices about the failure of any of its adjacent CDS node D_i, or receives a RYS message from one of its adjacent CDS node D_i, then it executes the Algorithm 10. In that situation, the dominatee

E_i first checks about its adjacency with any dominator other than D_i if any. If it finds, then it becomes attached to one of it and updates its dominator information. However, if E_i does not find any 1-hop CDS neighbour other than D_i but finds a 2-hop CDS neighbour D_j through a mutual neighbour M_i, then it sends a CRTD message to M_i which becomes a dominator after receiving this message. Later, E_i become attached to the newly created dominator D_i. If the non-CDS node (dominatee) E_i neither finds a 1-hop CDS neighbour nor a 2-hop CDS neighbour then it becomes an isolated dominatee and waits for any of the above-discussed situations arise.

Algorithm 10 To be executed by any non-CDS node (dominatee) $\mathrm{E_i}$ when either it receives a RYS message from its dominator $\mathrm{D_i}$ or finds its dominator $\mathrm{D_i}$ has failed.

```
1: if the dominatee $\mathrm{E_i}$ is adjacent to some dominators other than $\mathrm{D_i}$ then it is
   attached to one of them and updates its dominator information.
2: else if the dominatee $\mathrm{E_i}$ is not adjacent to any dominator other than $\mathrm{D_i}$,
   however, it has a 2-hop neighbouring dominator other than $\mathrm{D_j}$ then
     (a) It sends a CRTD message to the mutual neighbour $\mathrm{M_i}$ which changes to
         a dominator by sending the DOMINATOR message to all its neighbours.
     (b) Dominatee $\mathrm{E_i}$ should be attached to the mutual neighbour $\mathrm{M_i}$ (now be-
         comes a dominator) and updates its dominator information.
3: else
4:     the dominatee $\mathrm{E_i}$ becomes an isolated node until one of the above condition
   (1 or 2) is satisfied.
5: end if
```

Now, we discuss how a non-CDS (dominatee) node repairs the CDS by connecting multiple components. The detailed procedure can be found in the Algorithm 11. In fact, when a dominatee finds that it is adjacent to more than one components and its current battery power is more than 75% of the full battery power, it initiates the CDS maintenance process (step 1 of Algorithm 11) by sending a promotion request DPROMREQ to its adjacent CDS nodes. In the request, it sends the component-ids of the components it is adjacent with and its current battery power. Any CDS node which receives the DPROMREQ message forwards this message to the other CDS members of the same component and also receives DPROMREQ messages (if any) from other CDS members of the same component. Each CDS node which receives DPROMREQ message from a dominatee prepares a list of dominatees interested to become dominators and selects the dominatee, which is connecting more number of components. In case of a tie, it selects the dominatee with highest battery power. Finally, the CDS node sends the reply to the selected dominatee as a CRTD message. The dominatees who receive the CRTD message from their respective CDS nodes become the dominator and send the DOMINATOR message to notify their new status. It sends the new component-ID, which is the minimum of the component-IDs of the components it has connected, through the UCI message.

When a dominatee E_{i} finds that although it is not adjacent with CDS nodes of different components, but one of its 1-hop adjacent neighbour D_{i} and one of its 2-hop adjacent neighbour D_{j} are from different components then also E_{i} initiates the maintenance process (step 2 of Algorithm 11). Suppose the mutual neighbour of E_{i} and CDS node D_{j} is E_{j}. Dominatee E_{i} sends its power and requests for the power of E_{j} through the message GETPOW. Both E_{i} and E_{j} calculates the minimum power of E_{i} & E_{j} and send group promotion requests to their respective adjacent CDS nodes D_{i} and D_{j} through the message GDPROMREQ by providing the calculated minimum power and component-ids of the components they are connected with. Both the CDS nodes D_{i} and D_{j} spread the group promotion request among other component members and take the decision of selecting the best dominatee group which can connect a maximum number of components in the similar way as discussed above. Finally, they send the CRTD message to the selected dominatees. The dominatees become the dominator and connect the components exactly in a similar way as discussed above.

5.4.3 A non-CDS node (dominatee) finds its battery power has reached below the lower threshold value

When a dominatee finds that its battery power has reached below a predefined threshold value θ_2 it enters into sleep mode by sending a message TOSLEEP

Algorithm 11 To be executed by any non-CDS node (dominatee) when it finds that the component-ID of its adjacent CDS node has changed.

```
1: if a dominatee $E_i$ finds that it is adjacent to more than one CDS nodes with
   different component-ID then it executes the following steps:
   (a) The dominatee $E_i$ should send a request to its adjacent CDS nodes by
       sending the message DPROMREQ(cmid[], power) message where cmid
       is an array of unique component-IDs of the CDS nodes adjacent to $E_i$
       and power is the current battery power of dominatee $E_i$. The dominatee
       $E_i$ should send this request only if its battery power is above 75% of the
       full battery power.
   (b) A CDS node $D_i$ on receipt of message DPROMREQ from a dominate
       $E_i$ stores the request information and forwards it to its CDS neighbours,
       and receives the same message (if any) from its CDS neighbours.
       (i) It combines the requests from the dominatees adjacent to different
           CDS nodes of the same component and prepares a list. Referring to
           the prepared list, $D_i$ selects dominatees which can connect to different
           components. If multiple dominatees connect same components then
           their battery power is considered for their selection.
       (ii) The CDS node $D_i$ sends CRTD message to the selected dominatees.
       (iii) A dominatee after receiving the CRTD message becomes a domina-
           tor to connect these components and sends a DOMINATOR message
           to all its neighbours. It also sends the UCI message with the new
           component-ID which is the minimum of component-IDs of the com-
           ponents it has connected.
2: else if a dominatee $E_i$ finds that the component-ID of its 1-hop CDS neighbour
   $D_i$ and component-ID of its 2-hop CDS neighbour $D_j$ are different then it
   executes the following steps:
   (a) It sends a message GETPOW(power) message to its neighbour $E_j$ (mutual
       neighbour of $E_i$ and $D_j$) to notify its current power. The dominatee $E_i$
       should send this message only if its battery power is above 75% of the
       full battery power.
```

Algorithm 11 To be executed by any non-CDS node (dominatee) when it finds that the component-ID of its adjacent CDS node has changed. - **contd...**

```
    (b) On receipt of GETPOW message by $E_j$:
        (i) $E_j$ sends a POW(power) to $E_i$ to notify its power to $E_i$ if its battery
            power is above 75% of full battery power.
        (ii) $E_j$ sends a GDPROMREQ(cmid[], minpower) message to $D_j$ where
            cmid is an array of component IDs of the CDS nodes $D_i$ and $D_j$ are
            adjacent with and power is the minimum battery power of dominatee
            $E_i$ and $E_j$.
    (c) On receipt of POW by $E_i$ it sends the same message GDPROMREQ
        (cmid[], minpower) to $D_i$ also.
    (d) A CDS node $D_i$ on receipt of message GDPROMREQ from a dominate
        $E_i$ stores the request information, forwards it to its CDS neighbours, and
        receives the same message from its CDS members.
        (i) It combines the requests from different dominatee groups and selects
            the dominatee group which is having more battery power.
        (ii) The CDS node $D_i$ sends CRTD message to dominate $E_i$
    (e) Dominatee $E_i$ (or $E_j$) after receiving the CRTD message becomes a dom-
        inator to connect these components and also notifies its group member
        $E_j$(or $E_i$) by sending CRTD message. $E_i$ (or $E_j$) sends a DOMINATOR
        message to all its neighbours. It also sends the UCI message with the new
        component-ID which is a minimum of component-IDs of the components
        it has connected.
3: end if
```

Algorithm 12 To be executed by a non-CDS node (dominatee) E_i when its battery power has reached below a predefined lower threshold value θ_2.

```
1: The dominatee $E_i$ should send the message TOSLEEP to its dominator $D_i$.
2: The dominatee $E_i$ enters in the sleep mode and recharge.
```

5.4.4 A non-CDS node (dominatee) has completely failed

When a dominator D_{i} either finds that one of its dominatee E_{i} has completely failed or received a TOSLEEP message from one of its dominatee E_{i}, first it removes the dominatee from its dominatee list. If the removed dominatee E_{i} was the only dominatee of the dominator D_{i}, then D_{i} becomes a dominatee by sending the DOMINATEE message to its neighbours. The detail steps can be found from Algorithm 13.

Algorithm 13 To be run by a CDS node (dominator) $\mathrm{D_i}$ when either it receives a TOSLEEP message from one of its dominatee $\mathrm{E_i}$ or finds that one of its dominatee $\mathrm{E_i}$ has failed completely.

```
1: Dominator $D_i$ removes $E_i$ from its 1HopNebTable.
2: if the dominatee list of $D_i$ (after removal of $E_i$) is empty and $D_i$ is adjacent
   to only one CDS node then
3:     Node $D_i$ becomes a dominatee and informs to its neighbours about its new
       role by sending the DOMINATEE message.
4: end if
```

to its dominator. In the sleep mode, the dominatee does not sense any information and recharges only. The steps a dominator follows after getting the TOSLEEP method is described in the next subsection. The exact steps can be found from Algorithm 12.

5.4.5 After recharging, a node is interested to work as a dominatee

In the proposed approach, when a dominator finds its battery power has reached below the upper threshold θ_1, it downgrades itself to dominatee. It remains as dominatee unless it finds a situation where it needs to be promoted to a dominator to connect the CDS nodes once again. In that period, it senses the data only. However, when a dominatee finds its battery power has reached below the lower threshold θ_2, it enters into the sleep mode and does not sense any information during that period. After recharging enough, when it finds that its battery power has become more than the predefined upper threshold θ_1, it promotes itself to a dominatee and notifies its neighbours by sending the DOMINATEE message. After its promotion, when it finds that one of its adjacent nodes is a dominator, it becomes the dominatee of one of the dominators. The detail steps can be found in Algorithm 14.

Algorithm 14 To be executed by a normal node (neither dominator nor dominatee) $\mathrm{E_i}$ when its battery power has reached above the predefined upper threshold of θ_1.

```
1: The non-CDS node E_i becomes a dominatee and informs to its neighbours
   about its new role by sending the DOMINATEE message.
2: if the dominatee E_i finds that it is adjacent to some dominators, it is attached
   to one of them and updates its dominator information then
3: else if the dominatee E_i is not adjacent to any dominator, however it has a
   2-hop neighbouring dominator D_i then it executes the following:
     (a) It sends a CRTD message to the mutual neighbour M_i which changes to
         a dominator after receiving the CRTD message and sends the DOMINA-
         TOR message to all its neighbours.
     (b) Dominatee E_i is attached to the mutual neighbour M_i (now becomes a
         dominator) and updates its dominator information.
4: else
5:     The dominatee E_i becomes an isolated node and waits.
6: end if
```

5.4.6 After recharging a dominatee is looking for its dominator

As discussed in the above sub-section when a node becomes a dominatee after recharging it looks for a dominator to be attached. If it finds one dominator as its 1-hop neighbour then its task becomes easier. However, if it does not find a 1-hop dominator, but finds a 2-hop dominator then it executes the steps given in Algorithm 14. It sends a CRTD message to the mutual neighbour between the 2-hop dominator and itself to change the role of the mutual neighbour from the dominatee to dominator. After the mutual neighbour changes its role, the dominatee is attached to the new dominator (mutual neighbour). If a dominatee neither finds a 1-hop CDS neighbour nor finds a 2-hop CDS neighbour it becomes an isolated dominatee and waits for the time until it finds one of its 1-hop or 2-hop neighbour becomes a dominator.

5.4.7 Working Example

This subsection illustrates the proposed distributed CDS maintenance approach through a network diagram shown in Fig. 5.6. The nodes in the diagram represent randomly placed sensor nodes. In the above network, any two nodes are connected through an edge if they are within the communication range of each other.

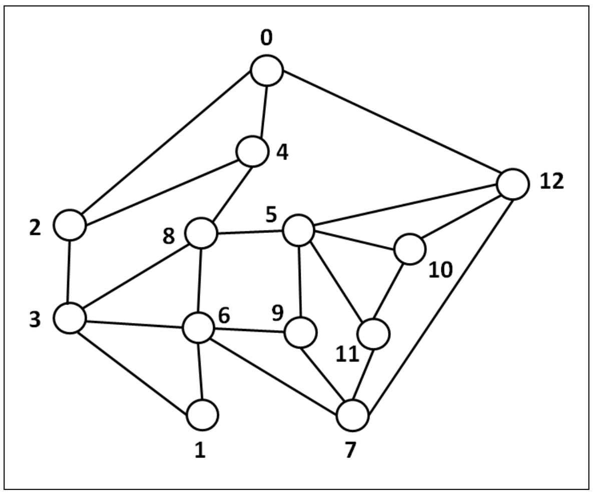


Figure 5.6: Initial Network

Assume that the communication range of all the nodes is the same. Let us also assume that the nodes {0, 5, 6, 8, 12} have formed a CDS as shown in the Fig. 5.7. Note that the black coloured nodes are dominators (CDS nodes) and the grey coloured nodes are dominatees (non-CDS nodes). The remaining of the subsection discusses the proposed approach by considering one example from each of the cases as described at the beginning of the current section. For easier understanding, without loss of generality let us assume that from the beginning node 0, 5, 6, 8, and 12 are providing the service of dominators to the group of nodes {2, 4}, {9, 11}, {1, 7}, {3}, and {10} respectively.

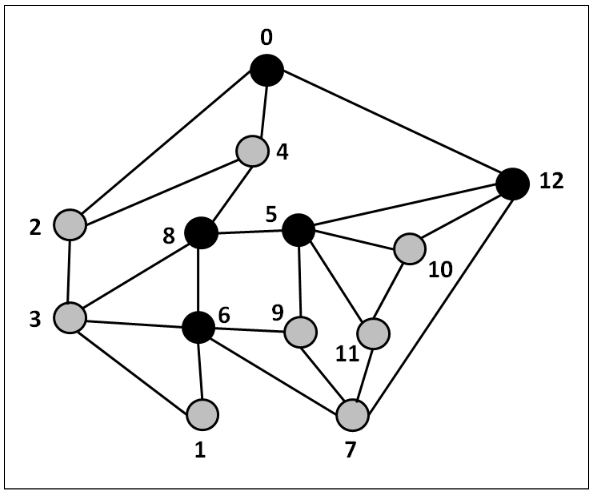


Figure 5.7: Black coloured nodes forming the CDS of the network shown in Fig. 5.6

Case 1: *Battery power of CDS node 0 (shown in Fig. 5.7) has reduced below the predefined upper threshold value θ_1.*

First, the dominator 0 would send an RYS message to its dominatees 2 and 4. In this case, node 0 is a CDS node (dominator) which is adjacent to only one CDS node 12. Node 0 would become a dominatee and node 12 would become its dominator. Node 4 after getting the RYS message would find that it is adjacent to another alternative dominator 8, so it would update its dominator from 0 to 8. Node 2 after getting the RYS message would find that it is not adjacent to any alternative dominator other than 0. However, its 2-hop neighbour 8 is a CDS node. Node 2 would send a CRTD message to the mutual neighbour 3, which becomes a dominator after getting the CRTD message. When node 3 becomes a dominator, node 2 would be attached to it. After all these changes, the updated CDS consisting of CDS nodes 3, 5, 6, 8 and 12 is shown in Fig. 5.8.

Case 2: *Battery power of node 8 (shown in Fig. 5.7) reduced below the predefined upper threshold value θ_1.*

First, the dominator 8 would send an RYS message to its dominatee 3. Node 3 after getting the RYS message would find that it is adjacent to another alternative dominator 6, so it would update its dominator from 8 to 6. In this case, node 8 is adjacent to more than two CDS nodes 5 and 6. So, node 8 would send the message CCID to node 5 and 6. These CDS nodes on receiving the CCID message would change their component-ID to their own node-ID and pass their updated component-ID to their other adjacent nodes through the message UCI.

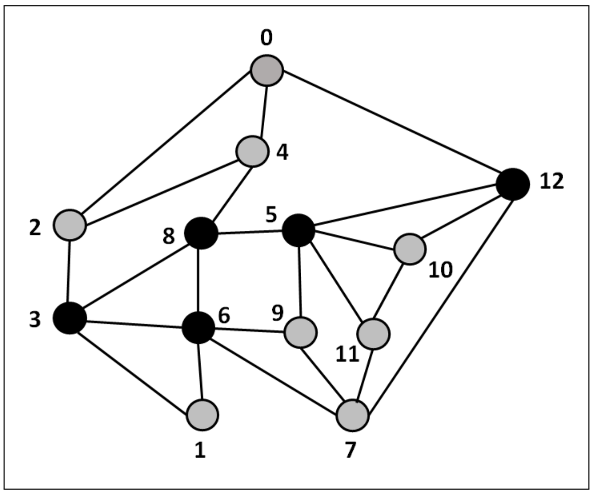


Figure 5.8: Updated CDS after the battery power of node 0 (shown in Fig. 5.7) reduced below the predefined upper threshold value θ_1

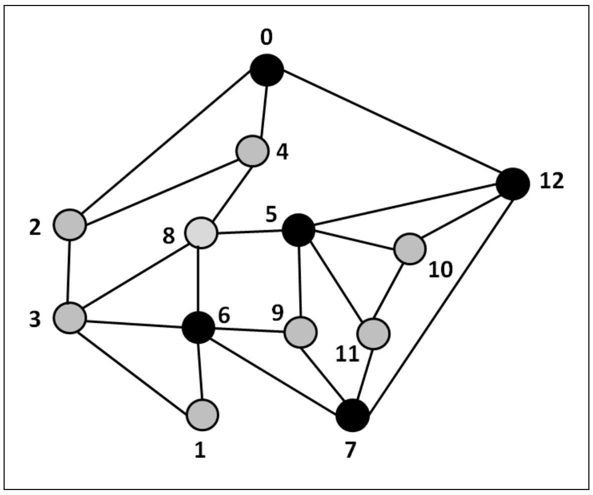


Figure 5.9: Updated CDS after the battery power of node 8 (shown in Fig. 5.7) reduced below the predefined upper threshold value θ_1

The component-ID of CDS node 5, 12 and 0 would be changed to 5 and the component-Id of CDS node 6 would be changed to 6. The dominatees would update their adjacent dominators' component-ID. The dominator 8 would become a dominatee. Anyone of its adjacent dominator would become its dominator (let it be 6). When the dominatee 7 would find that it is adjacent to two CDS nodes 6 and 12 with different component-IDs 6 and 5 respectively it would send a DPROM_REQ $\langle \{5, 6\}, bat_7 \rangle$ message to its adjacent CDS nodes 6 and 12 if its battery power bat_7 is more than θ_1. Similarly, dominatee 9 would also find that it is adjacent to two CDS nodes 6 and 5 with different component-IDs 6 and 5 respectively, it would send a DPROM_REQ $\langle \{5, 6\}, bat_9 \rangle$ message to its adjacent CDS nodes 6 and 5 if its battery power bat_9 is more than θ_1. The CDS nodes of each component would forward the DPROM_REQ messages to their adjacent CDS nodes and wait for any DPROM_REQ from them. Assuming bat_7 is more than bat_9, node 7 would receive CRTD message from both CDS nodes 6 and 12. Dominatee 7 on receiving the CRTD message from 6 and 12 would become the dominator and send the UCI message to pass the new component-ID 5 (minimum of 5 and 6) to all its neighbours. Finally, node 8 would become a dominatee of one of its adjacent dominator (let it be 5). It would send its new role through the DOMINATEE message. The resultant CDS is in Fig. 5.9.

Case 3: *Node 6 (shown in Fig. 5.7) failed without informing to its neighbours.*

When the dominatee neighbours 1, 3, 7, 9 of the failed CDS node 6 find that their neighbouring CDS node has already failed, they would first remove node 6 from their *1HopNebTable*. Next, all these dominatees would look for any other alternative dominators to which they can be connected. Dominatee 3, 7 and 9 would find their alternative dominators 8, 12 and 5 respectively. They would update their dominator information and forward their *1HopNebTable* to their neighbours. Dominatee 1 would find that it is not adjacent to any dominator other than 6 (which has already failed). However, it has a 2-hop neighbour 8 which is a dominator. Therefore, node 1 would send a CRTD message to node 3 which is the mutual neighbour of node 1 and 8. Node 3 after getting the CRTD message becomes a dominator and forwards the DOMINATOR message.

Node 1 would update its dominator information. When node 8 would come to know that its adjacent CDS node 6 has already failed and the only CDS neighbour of 6 was itself, it would remove node 6 from its *1HopNebTable* and would forward it to its neighbours to update their *2HopNebTable*. The resultant CDS is shown in Fig. 5.10.

Case 4: *Node 12 (shown in Fig. 5.7) failed without informing to its neighbours.*

When the dominatee neighbours 7 and 10 of the failed CDS node 12 find that their neighbouring CDS node has already failed, they would first remove node 12 from their *1HopNebTable*. Next, all these dominatees would look for any other alternative dominators to which they can be connected. Dominatee 7 and 10 would find their alternative dominators 6 and 5 respectively. They would update their dominator information and forward their *1HopNebTable* to their neighbours.

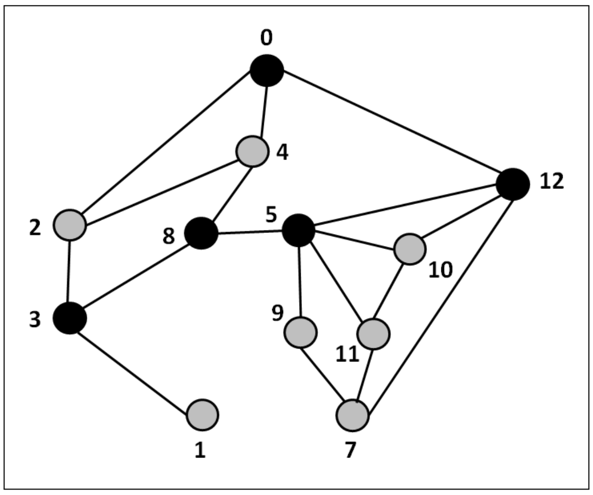


Figure 5.10: Updated CDS after the failure of node 6 shown in the Fig. 5.7

When dominator 0 and 5 would come to know that their CDS neighbour 0 has already failed, they would check their *2HopNebTable* to know the other adjacent CDS nodes (if any) of the failed node 12. Both nodes 0 and 5 would come to know that node 12 had more than one adjacent CDS nodes. Hence node 0 and 5 would change their component-ID to 0 and 5 respectively and forward their new component-ID to the other dominators through UCI message. After that node 0 would form its component with component-ID 0 and node 5, 6, 8 would form another component with component-ID 5. This new component information would also be circulated among the dominatees of these CDS nodes.

When dominatee 4 would find that it is adjacent to two different components it would send a DPROM_REQ $\langle\{0,5\}, bat_4\rangle$ message to both 0 and 8, if its battery power bat_4 is more than θ_1. The CDS node 0 and 5 would forward this message among their other adjacent CDS nodes and wait for the DPROM_REQ message from any other dominatees. As except node 4 there is no other dominatee, which is adjacent to more than one component, no more DPROM_REQ message would be received. So both 0 and 8 would send CRTD message to node 0. After receiving the CRTD message, dominatee 4 would become a dominator and send the UCI message to pass the new component-ID 0 (minimum of 0 and 5) to all its neighbours. When the adjacent dominatees 10 and 7 of the failed CDS node 12 would find that their adjacent dominator has failed, they would first remove node 12 from their *1HopNebTable*. Next, they would check for other alternative dominators. Node 10 would find its alternative dominator 5 and node 7 would find its alternative dominator 6. They would update their dominator information and forward their *1HopNebTable* to their neighbours. The resultant CDS is shown in Fig. 5.11.

Case 5: *Dominatee node 2 (shown in Fig. 5.7) finds that its battery power has reached below the predefined threshold value θ_2*

When dominatee node 2 would find that its battery power has reached below the lower threshold θ_2, it would be interested to enter into the sleep mode and during this mode, it would not sense any information. Before entering the sleep mode, it would send a message TOSLEEP to its dominator 0. After sending the TOSLEEP message, dominatee 2 would enter into the sleep mode. Dominator 0 after receiving the TOSLEEP message, would remove the dominatee 2 from its *1HopNebTable* and would pass it to its adjacent nodes. The dominatee list of node 0 after removing node 2 would not be empty since it contains node 4. Therefore, no more work would be done.

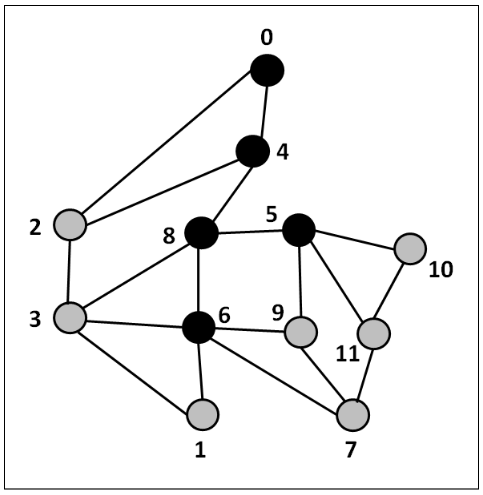


Figure 5.11: Updated CDS after the failure of node 12 shown in Fig. 5.7

Case 6: *Non-CDS node 1 (in Fig. 5.10) failed completely*

When the CDS node 3 would find that its dominatee 1 has failed, first it would remove its information from its *1HopNebTable*. After removing, node 3 would find that node 1 was its only dominatee and node 3 is adjacent to only one CDS node 8. Therefore, it would downgrade itself to a dominatee by sending the message DOMINATEE. The final CDS is shown in Fig. 5.12.

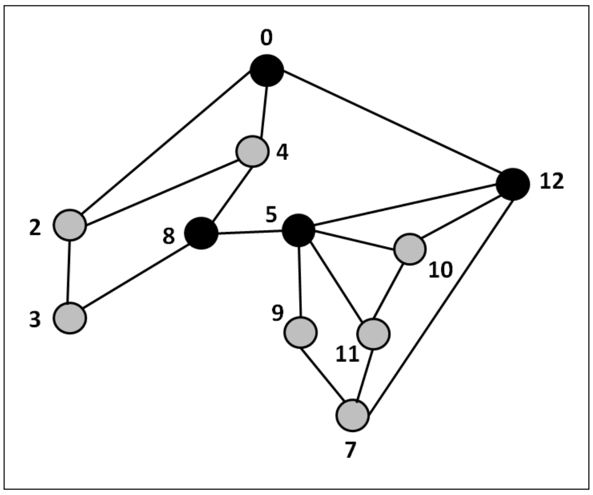


Figure 5.12: Updated CDS after the failure of non-CDS node 1 in the network shown in Fig. 5.10

Case 7: *Node 2 (in Fig. 5.7) currently in sleep mode finds that it has charged above the predefined upper threshold value θ_1*

If node 2 currently in sleep mode finds that it has charged above θ_1, it would become a dominatee by sending the message DOMINATEE. Node 2 after becoming a dominatee would find that one of its adjacent node 0 is a dominator, so it would be attached to it by updating its dominator information.

5.5 Algorithm Analysis

Analysis of the proposed distributed CDS maintenance algorithm is three fold. Initially, the performance ratio of the algorithm is calculated. Later the time and message complexities of the proposed scheme are presented. To do this, we use certain theorems. The detail proofs of all of these can be found in this section.

Theorem 5.1 *The distributed CDS maintenance scheme DMCDS changes the current CDS size by a factor of Δ, where Δ is the maximum degree of a node in the entire network.*

Proof. *Suppose initially D number of nodes formed the CDS. We need to apply the CDS maintenance algorithm due to one of the following cases arises: (1) One of the existing CDS node fails. (2) Battery power of one of the existing CDS node has reached below the upper threshold of θ_1 (3) A non-CDS node fails (4) Battery power of a non-CDS node has reached below the lower threshold of θ_2 (5) Battery power of a non-CDS node has increased above the upper threshold of θ_1 after recharging.*

Case 1: When a CDS node either completely failed or its battery power has reached below a predefined threshold value θ_1 *In this case, first the dominatees of the failed CDS node would try to find their alternative dominators. If a dominatee of the failed CDS nodes does not find any alternative dominator in its 1-hop neighbourhood but it finds at least one 2-hop dominator, then it sends the message to its mutual neighbour to become a dominator. So, for each dominatee a maximum of one number of CDS node would be added to the existing CDS. In the worst case, the failed dominatee would have Δ dominatees, where Δ is the maximum degree of each node in the network. So, the maximum increase in CDS size to attach the dominatees to some alternative dominators is $O(\Delta)$. The failed CDS node is adjacent to either one or more CDS nodes. If it is adjacent to one CDS node then there is no increase in CDS size after its failure. However, if it is adjacent to multiple CDS nodes, then a number of components would be formed. In the worst case, a maximum of Δ components would be formed. In that case, some of the dominatees would promote themselves to dominators to connect multiple components. According to the proposed mechanism, either one or two nodes (dominatees) are needed to connect each pair of components. So, in the worst case, $2(\Delta - 1)$ number of nodes would be needed to connect these components. Hence, the change in CDS size due to the failure (or about to fail) of CDS node is $O(\Delta + 2(\Delta - 1) - 1) = O(\Delta)$.*

Case 2: When a non-CDS node either completely failed or its battery power has reached below a predefined threshold value θ_2 *If the dominatee was the only dominatee of its dominator and the dominator was adja-*

cent to only one CDS node then the dominator downgrades itself to dominatee. This decreases the CDS size by 1.

Case 3: When a node (neither dominatee nor dominator) currently in sleep mode finds that it has charged above θ_1 and is ready to work as dominatee *In this case the node becomes a dominatee and searches for its dominator. If it finds a CDS node in its 1-hop neighbourhood then it becomes its dominatee. However, if there is no 1-hop neighbour of the new dominatee, but it has 2-hop CDS neighbour then their mutual neighbour would become the dominator which increases the CDS size only by 1.*

Considering all these three cases it can be concluded that in the case of CDS maintenance the CDS size would change at most by a factor of Δ. However, in practice, it is much less than that. ■

Theorem 5.2 *In a given a network already with a working CDS, DMCDS maintains the CDS in time $O(C)$ where C is the size of the largest component at the moment the CDS maintenance algorithm DMCDS is applied.*

Proof. *According to the discussion in the above Theorem 5.1 the CDS maintenance is needed due to any one of the above-mentioned cases. When a CDS node's battery power reduced below θ_1, first it would inform to its dominatees through RYS message in constant time. All its dominatees would check for their alternative dominator by looking their 1HopNebTable in the next unit of time. If a dominatee finds any alternative dominator it would update its dominator information in the 1HopNebTable in one unit of time. If it does not find any alternative dominator it looks for any 2-hop dominator in the 2HopNebTable in one unit of time. If it finds a 2-hop dominator it sends a CRTD message to the mutual neighbour to become a dominator. The mutual neighbour after getting the CRTD message becomes a dominator and sends a DOMINATOR message in one unit of time. Now the dominatee updates its dominator information in the next unit of time. So the total time needed to rescue all the dominatees of the CDS node whose battery power reduced below θ_1 is constant.*

If the CDS node whose battery power falls below the predefined threshold value θ_1 is adjacent to one CDS node then it sends a DOMINATEE message in one unit time. It becomes a dominatee and records its dominator information in the next *unit time. In the same time, its neighbours update their neighbour information. However, if the CDS node whose battery power falls below the predefined threshold value θ_1 is adjacent to more than one CDS node then it would send the CCID message to all its adjacent CDS nodes in one unit of time. The CDS nodes after receiving the CCID message would update their component-ID and pass their new component-ID among their neighbours of the same component in time O(C), where C is the size of the largest component. The dominator becomes dominatee and records its dominator information by sending the DOMINATEE message in one unit of time. The dominatees which would be interested to become the dominators send the DPROM-REQ in one unit of time. The CDS nodes of each component would circulate the message among their neighbours in O(C) time. The selected dominatees receives the CRTD message in the next unit of time. They become dominator in the next unit of time. Also, it sends the UCI message in one unit of time. If a dominatee does not find any 1-hop CDS neighbour then it tries to merge the components through 2-hop CDS neighbours. For that, it needs three extra units of time: one for GETPOW message, one for POW message and one for extra CRTD message. So, the total running time for maintaining the CDS in case of decrement of battery power below predefined threshold value θ_1 is O(C), where is the size of the largest component.*

If a CDS node fails without informing any of its neighbours then the adjacent dominatees and adjacent dominators would follow the above-discussed procedure after knowing that their neighbour failed. So, the running time would be the same as above. Now, we can conclude that the running time of CDS maintenance either in case of failure of CDS node or decrement of battery power of the CDS node below the predefined threshold value θ_1 is O(C), where C is the size of the largest component after the failure of the CDS node.

If a dominatee finds that its battery power has reached below the predefined lower threshold value θ_2 it first sends a TOSLEEP message in one unit of time. A dominator after receiving the TOSLEEP message first removes the node from its dominatee list in one unit of time. If the dominatee was its only dominatee, then the dominator becomes the dominatee and sends the DOMINATEE message in the next unit of time. Hence, the total CDS maintenance time, in this case, is constant.

Similarly, if a dominator finds that one of its dominatees has failed, first the dominator removes the dominatee from its dominatee list in one unit of time. If the dominatee was its only dominatee, then the dominator becomes the dominatee and sends the DOMINATEE message in the next unit of time. Hence, the total CDS maintenance time either due to the failure of a dominatee or decrement of battery power of a dominatee below the predefined threshold value θ_2 is constant.

When a non-CDS node after charging finds that its battery power has reached above the predefined threshold value θ_2, then first it sends a DOMINATEE message in one unit of time. If it finds a dominator in its 1-hop neighbourhood it updates its dominator information in one unit of time. Else if it has a 2-hop CDS neighbour, then it first sends a CRTD message to its mutual neighbour in one unit of time and the mutual neighbour becomes a dominator by sending the DOMINATOR message in next unit of time. The neighbours update their neighbour information in one more unit of time. Hence, the time for maintaining the CDS in case of a non-CDS node becomes a dominatee after recharging is also constant.

Therefore, the overall running time for maintaining the CDS either due to the failure of any node, or due to decrement of battery power below a predefined threshold or due to increment of battery power above a predefined threshold is $\mathsf{O}(\mathsf{C})$ *in the worst case, where* C *is the size of the largest component at the particular moment the CDS maintenance algorithm DMCDS is applied.* ■

Theorem 5.3 *DMCDS has message complexity of* $\mathsf{O}(\mathsf{n})$*, where* n *is the total number of nodes present in the network.*

Proof. *According to the discussion in the Theorem 5.1, the CDS maintenance is needed due to any one of the above-mentioned cases arises. In this theorem, the message complexity is analysed based on each of these cases.*

Let us consider the case in which a CDS node's battery power decreased below the upper threshold of θ_1. In that case, the dominator would send one RYS message to its dominatees. After receiving the RYS message, if the dominatee is attached to any other dominator, then it updates its dominator information and no further messages are sent. However, if the dominatee is not adjacent to any other dominator, but it has 2-hop CDS neighbours, then it sends a CRTD message to its mutual neighbour. The mutual neighbour sends the DOMINATOR message.

If the CDS node is adjacent to only one CDS node it would send a DOMINATEE message to all its neighbours. Its neighbours would send Δ *number of messages to notify their neighbours about the update in its 1-hop neighbourhood, where* Δ *is the maximum degree of each node. However, if the CDS node is adjacent to multiple CDS nodes, then it sends one CCID message. The CDS nodes on receiving the CCID message would send UCI message to the other member of the same component and they forward this to their neighbours. So, in total* $O(n)$ *number of UCI messages would be sent. The dominator then becomes a dominatee by sending a DOMINATEE message. When a dominatee finds that it is adjacent to multiple components with different component-ID, it sends a DPROM-REQ message. In the worst case,* $O(n-|D|)$ *number of DPROM-REQ messages would be sent, where D is the number of dominators. These DPROM-REQ messages would be forwarded among the CDS nodes for the final selection of dominatee to become a dominator. So O(n) number of DPROM-REQ messages would be sent. To the selected dominatees* $O(\Delta)$ *number of CRTD messages would be sent. These dominatees become dominators by sending* $O(\Delta)$ *DOMINATOR messages.*

If a dominatee finds that the component-ID of its 1-hop CDS neighbour and 2-hop CDS neighbour are different it sends GETPOW message and receives POW message in its reply. The total number of these two messages would be $O(n-|D|)$*, where D is the number of dominators. After this, the steps are very much similar to the case in which a dominatee has 1-hop adjacent dominators with different component-ID. So, the message complexity of this case is the same as the case in which a dominatee has 1-hop adjacent dominators with different component-ID. Hence, the total message complexity of the case when a CDS node's battery power decreased below the upper threshold of* θ_1 *is O(n).*

If a CDS node fails without informing any of its neighbours, then the adjacent dominatees and adjacent dominators would follow the same procedure as in the case of a CDS node's battery power decreased below the upper threshold of θ_1*. So no extra messages need to be sent in that case. Therefore, the message complexity of CDS maintenance either in case of failure of CDS node or decrement of battery power of the CDS node below the predefined threshold value* θ_1 *is O(n) where n is the total nodes present in the network.*

Now, let us calculate the message complexity of the case in which a domina-

tee's battery power is decreased below the lower threshold of θ_2. In this case, the dominatee would send a TOSLEEP message. A dominator on receiving this message, if it finds the dominatee is its only dominatee then it becomes a dominatee by sending a DOMINATEE message. Its neighbours would send their 1-hop information through $O(\Delta)$ messages. If a dominatee fails without information, and if its dominator finds the dominatee was its only dominatee, then it becomes a dominatee by sending a DOMINATEE message. Its neighbours after receiving the DOMINATEE message, would send their 1-hop information through $O(\Delta)$ messages. Therefore, the message complexity of CDS maintenance either in case of failure of a dominatee or decrement of battery power of the dominatee node below the predefined threshold value θ_2 is $O(\Delta)$.

When a non-CDS nodes battery power reaches above the predefined lower threshold value θ_2 it becomes a dominatee by sending the DOMINATEE message first. If the new dominatee finds that it is not adjacent to any dominator, but it has a 2-hop neighbour which is a dominator, then it sends a CRTD message to the mutual neighbour. The mutual neighbour sends the DOMINATOR message and its neighbours would send their 1-hop information through $O(\Delta)$ messages. Hence, the message complexity, in this case, is $O(\Delta)$ as well.

Therefore, the total message complexity of the proposed distributed CDS maintenance algorithm, DMCDS is $O(n)$ considering all the cases as mentioned in the theorem 5.1. ■

5.6 Simulation Results

In this section, we verify our algorithm through simulation. The results of simulations conducted for various situations of CDS maintenance. The wireless network is modelled in a fixed area of dimension 150×150 square units. Nodes are randomly placed by choosing the x-coordinate and y-coordinate of each node using a uniform random number generator. The size of the simulation network is 50 to 250 numbers in an increment of 25 nodes. In the network, two nodes are connected if their distance is less than or equal to the transmission range. The transmission range of each node is taken as 25 units. Initially, the CDS is constructed by using the CDS construction algorithms proposed in the previous chapter. On top of the constructed CDS, the proposed CDS maintenance algorithm is run for various situations of CDS maintenance. The proposed scheme is run for a number of times for various network sizes between 50 and 250. The average results are reported in the figure. The entire simulation is carried out in NS-2, a network simulator for the wireless network. The simulations conducted for analysing the proposed CDS maintenance scheme are (i) Performance comparison of the number of CDS nodes before and after CDS maintenance (ii) Performance comparison of the number of messages generated during the maintenance. During the analysis, we consider the various maintenance schemes like a failure of dominator/dominatee, reduction of battery power of dominator/dominatee below threshold and upgradation of a non-CDS node to dominatee after recharging.

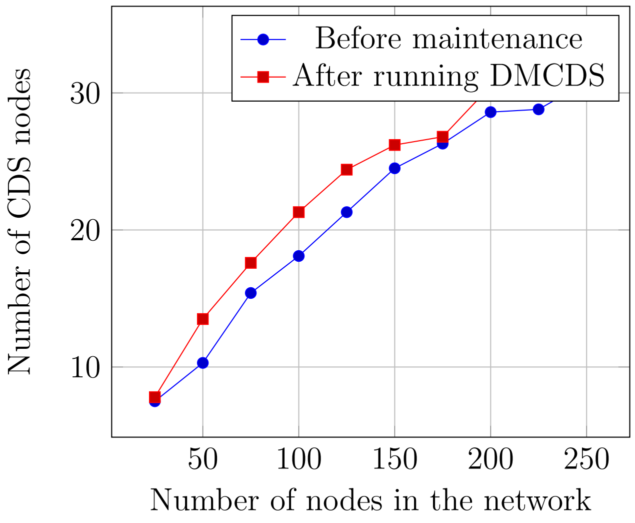


Figure 5.13: Performance comparison of CDS size due to failure of dominators

In the first experiment, we constructed the CDS and calculated the mean number of CDS nodes. Later, we experimented on current CDS nodes. We observed the effect of CDS after the failure of CDS nodes. The experiment also considered the effect of CDS due to the depletion of the battery power of the current CDS nodes. For both cases, the CDS is repaired using the proposed DMCDS algorithm and the total number of CDS node is calculated. The results are shown in Fig. 5.13. We found that the average number of CDS nodes does not vary too much, it remains almost the same as the number of CDS nodes before the maintenance. This is because if a dominator fails the maintenance algorithm needs to take care of two things, it has to rescue the dominatees of the failed dominator and also ensure that the remaining CDS nodes are connected. The proposed algorithm first tries to connect the dominatees of the failed dominator with some existing dominators. If it does not find suitable dominator, then it promotes some of the dominatees into dominators and attaches the dominatees to them. However, these promotions are very rare. Similarly, because of the failure of a dominator the CDS nodes may be disconnected forming different components. The proposed scheme connects these components by promoting some of the dominatees into dominators. According to the result shown in the figure, it is observed that although during maintenance due to the failure of a single CDS node, more than one dominatees need to be promoted. However, this case is rare. From the experiment, it is found that in some cases failure of a single dominator does not increase the CDS size at all. This happens in the case where the dominatees of the failed dominator can be attached to some existing dominators and the failed dominator does not disconnect the CDS nodes.

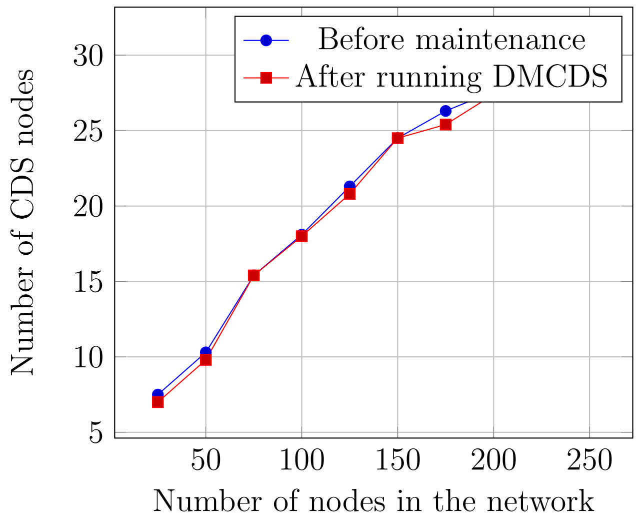


Figure 5.14: Performance comparison of CDS size due to failure of dominatees

In the next experiment, we found the effect of CDS after the failure of dominatees. The experiment also considered the effect of dominatees due to the decrement of their battery power. For these two cases, the CDS is repaired using the proposed DMCDS algorithm and the total number of CDS nodes is counted. The results are shown in Fig. 5.14. We found that the average number of CDS nodes after the CDS maintenance due to the dominatees is always less than or equal to the number of nodes present in the network. This is because in case failure of a dominatee or its battery power decremented below a level, the dominator has to just update this information in its *1HopNebTable*. If the dominatee was its only dominatee and the dominator has only one adjacent dominator then the dominator has to downgrades itself from dominator to dominatee. Therefore, if the failed dominatee was the only dominatee of its dominator and the dominator has only one adjacent dominator then the size of the CDS would decrease, otherwise, it would remain the same. Therefore, the CDS size after maintenance either remains the same or decreases by one, which is very rare. The figure shows the same result as expected.

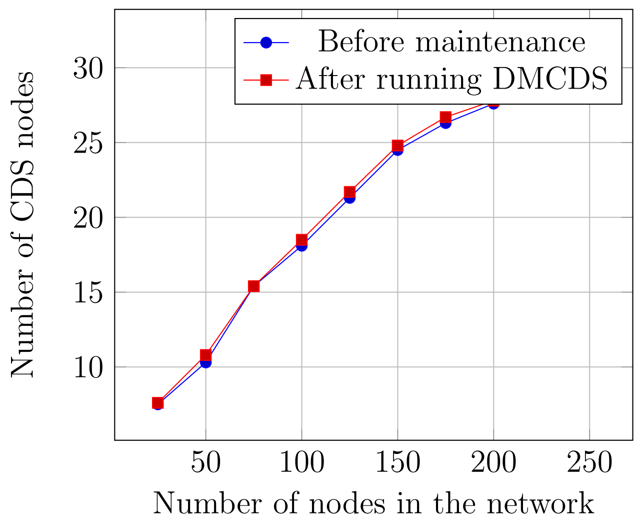


Figure 5.15: Performance comparison CDS size due to upgradation of normal nodes to dominatees

In the third experiment, we tried to find the effect of CDS when a non-CDS node after recharging would be interested to become a dominatee. In that case after the node becoming a dominatee it would look for at least one dominator within its communication range to which it would be attached. If it finds a dominator within its range, then it would become its dominatee and there is no change in the CDS. However, if it does not find a 1-hop dominator then it would look for

2-hop dominator if any. If it finds a 2-hop dominator, then the mutual neighbour becomes a dominator and it would be attached to it. So, in this case, there is an increase of CDS size by one. The simulation result is shown in Fig. 5.15 exactly shows the same thing. We find that the average number of CDS nodes after the CDS maintenance due to the non-CDS nodes become dominatees varies a little with the number nodes present in the original CDS before the maintenance.

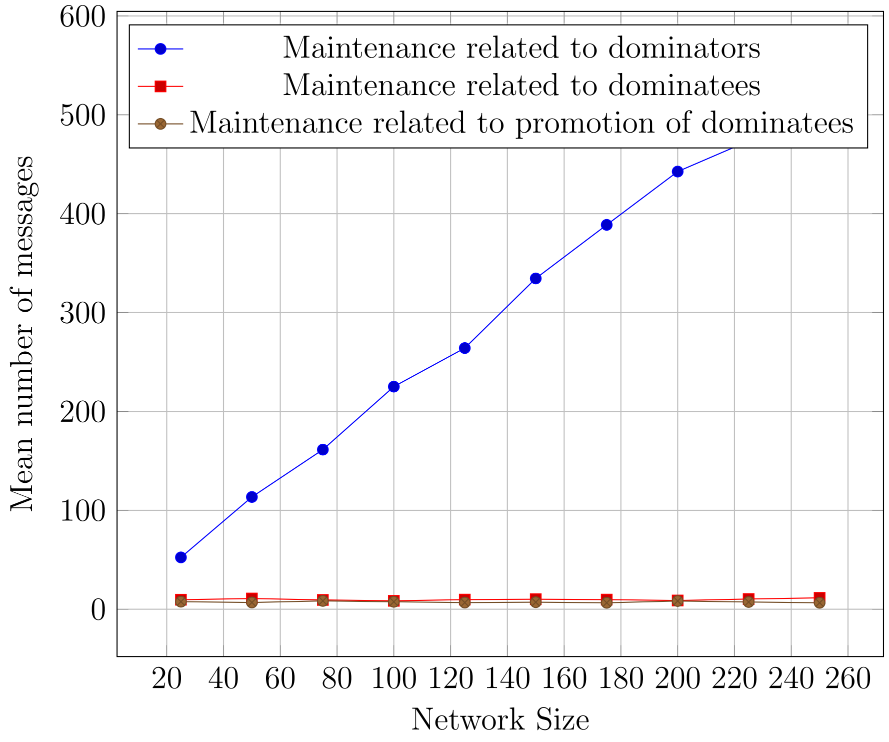


Figure 5.16: Performance analysis of DMCDS by comparing the number of messages generated

Next experiment compares the number messages generated by the proposed distributed CDS maintenance algorithm in each of the following cases: (1) Number of messages generated due to either the failure of dominators or the dominator battery power falls below the predefined level (2) Number of messages generated due to either the failure of dominatees or the dominatees battery power falls below the predefined level (3) Non-CDS node is interested to promote itself into dominatee. According to the discussion in Theorem 5.3 of Section 5.5, the message complexity of the first case is $O(n)$ and the second and third case is O(Δ). The simulation result shown in Fig. 5.16 exactly shows that. The number of messages generated in the first case is more whereas the number of messages generated in the second and third case is very less and almost the same.

Our last experiment is on comparison of CDS maintenance with fresh CDS construction. We compared the number of messages generated in the construction of CDS with the maintenance of CDS through the proposed algorithm. The result is shown in Fig. 5.17. From the figure, it is very much clear that in case of failure of a CDS node CDS maintenance is better than the construction of fresh CDS due to the number of messages generated in the case of fresh construction of CDS is much more.

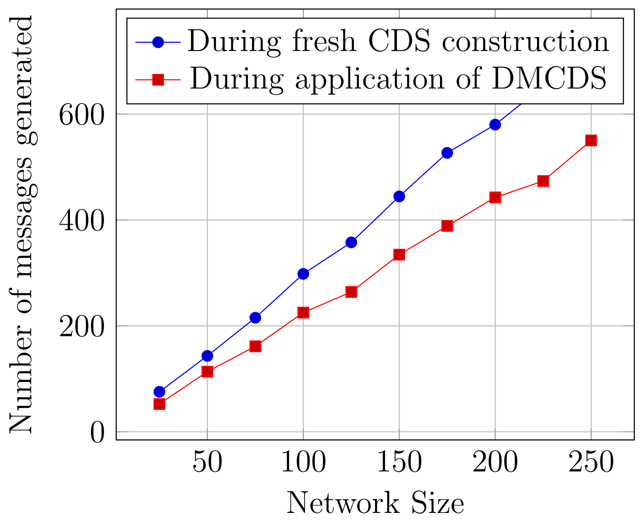


Figure 5.17: Performance comparison of CDS construction with CDS maintenance

5.7 Summary

In this chapter, it is observed that CDS maintenance is very much necessary like CDS construction. Later, a novel distributed CDS maintenance algorithm known as DMCDS is presented. The distributed CDS maintenance algorithm maintains the CDS successfully in the following situations: (1) A CDS/non-CDS node discharged below a predefined level, but not completely discharged. (2) A CDS/non-CDS node has completely failed and there is no chance that it would again be active. (3) A normal non-CDS node becomes ready to work as a dominatee again. The time and message complexities of the proposed algorithm are $\mathsf{O(C)}$ and $\mathsf{O(n)}$ respectively, where C is the size of the largest component of the network during the application of CDS maintenance algorithm and n is the size of the network. The distributed CDS maintenance scheme changes the current CDS by a factor of Δ, where Δ is the maximum degree of a node in the entire network. The simulation results show that the size of the CDS does not change too much in case of failure of CDS and non-CDS nodes. The proposed algorithm encourages both CDS and non-CDS nodes to downgrade themselves in case of lack of battery power. The non-CDS nodes can again work as dominatees after recharging whereas the dominatees can again work as dominators in case there is a need. Through simulation, it is also concluded that the maintenance of CDS is better than fresh construction of CDS. The proposed algorithm can be modified to maintain the CDS construction in the network of nodes with different transmission ranges.

CHAPTER 6

Conclusion

Connected Dominating Set is one of the effective constructs which solves a variety of problems related to wireless networks. Some of the major applications of the connected dominating set include the creation of a virtual network backbone for unicast and multicast routing, energy efficiency, media access coordination, etc. The CDS related problems found in the literature include:

- Construction of Minimum Connected Dominating Set: The workload of a CDS node is much more than other ordinary nodes in a network. Therefore, minimizing the CDS size can greatly reduce the generation and transmission of control messages. However, the construction of a minimum connected dominating set is an NP-Complete problem. Due to this reason, most of the researchers working in this area are focussed on reducing the CDS size further since the last two decade. Mainly these algorithms try to reduce the performance ratio since lower the performance ratio of the algorithm, the better is the CDS.

- Construction of Minimum sized Connected Dominating Set with additional parameters: Some of the CDS construction algorithms not only focussed on reducing CDS size but also tried to reduce the CDS diameter, average backbone path length, etc. Construction of Connected Dominating Set with fault tolerance and robustness: In wireless network node failure due to depletion of battery power or hardware is not rare. Some of the works have provided enough tolerance and robustness to the CDS nodes by constructing k-m-CDSs.

- Construction of Connected domatic partition: There are few works found in the literature which construct disjoint CDSs by dividing the entire network into a number of node-disjoint CDSs. These CDSs work in rotation to prolong the network lifetime.

The first two works presented in this dissertation comprise of centralized and distributed construction of size optimal connected dominating sets. The third work is on the maintenance of CDS due to failure of either CDS or non-CDS nodes. As all these problems are NP-Complete, we have proposed approximation algorithms. The results and findings from each of the work are summarized in the next section.

6.1 Contributions

The contributions of the thesis are described as follows:

- **Centralized construction of Minimum Connected Dominating Set** The first work of our thesis is a centralized algorithm for construction of minimum sized Connected Dominating Set. As construction of Minimum Connected Dominating Set is an NP-Complete problem, we have proposed a new centralized degree-based greedy approximation algorithm to construct smaller CDSs with the current best approximation ratio of $(4.8+\ln 5)|opt|+1.2$, where $|opt|$ is the size of an optimal CDS of the network. The algorithm has the best time complexity of $O(D)$, where D is the network diameter. To the best of our knowledge, this is the most time efficient and size-optimal CDS construction algorithm.

- **Distributed construction of Minimum Connected Dominating Set** Our second work is on the construction of minimum sized Connected Dominating set in a distributed manner. We have developed a distributed degree based algorithm (named as DCMCDS) for the minimum connected dominating set problem with the current best approximation factor of $(4.8 + ln5)opt + 1.2$, where $|\mathsf{opt}|$ is the size of an optimal CDS of the network. Simulation results show that DCMCDS is better than existing CDS construction algorithms in terms of CDS size and construction costs, using a slightly higher expense of a number of messages exchanged as compared to previous degree-based CDS construction techniques. Its time complexity is $\mathsf{O(D)}$, where D is the diameter of the network. It has a linear message complexity of $\mathsf{O(nR)}$, where n is the network size and R is the maximum of the number of rounds needed to construct the PDS and number of rounds needed to interconnect the PDS nodes. The distributed greedy algorithm DCMCDS does not depend on any specific initiating node. It identifies non-trivial CDSs of smaller size for both uniform and random distribution of nodes in a distributed manner. The algorithm constructs the CDS in lesser number of rounds in comparison to other degree-based algorithms.

- **Distributed maintenance of Minimum Connected Dominating Set** The third work is on CDS maintenance. In a wireless network, there is a chance of failure of nodes due to many reasons. The CDS nodes have a greater chance of failure due to depletion of battery because of more work-load. Failure of a single CDS node may disconnect the CDS. Similarly, sometime the failure of a non-CDS node may need some maintenance in the CDS to make it more effective. We have developed a new distributed CDS maintenance approach to handle the following situations: (1) a CDS or non-CDS node is about to fail due to the depletion of its battery power (2) a CDS or non-CDS node has already failed due to some reason (3) a non-CDS node becomes ready to sense the required data after recharging. The proposed distributed CDS maintenance scheme only changes the current CDS by a factor of Δ, where Δ is the maximum degree of a node in the entire network. Its time complexity is $\mathsf{O(C)}$, where C is the size of the largest component of the network during the application of the proposed CDS maintenance algorithm. It has a linear message complexity of $\mathsf{O(n)}$, where n is the number of nodes present in the network. It also promotes some nodes to improve the lifetime of CDS.

6.2 Scope for Future Work

In addition to the CDS related problems found in the literature, we should consider the following open problems in the future:

- **Construction of CDS using local information only:** Although distributed CDS construction algorithms are useful in a wireless network, we need to look for algorithms which can construct CDSs using local information only. Researchers should try to design approximated local CDS construction algorithms with better performance.

- **CDS construction in non-UDG graphs:** Most of the CDS construction algorithms proposed by various researchers are for UDG. However, the actual wireless networks may contain the nodes which are deployed in a three-dimensional space which is very practical. Also, the communication range of the nodes present in the network may be different. In this scenario, the network may be considered as a directed ball graph in which the nodes are deployed in a three-dimensional space and their radio ranges may not be unique.

- **Construction of CDSs with interference:** Most of the CDS construction algorithms assume that there is no interference in the network. However, in practice, there exists interference in the network. In the presence of interference, the data sent by a source node is either delayed to reach the destination node or may not reach the destination at all. This would affect the CDS construction process. Therefore, we should look for CDS construction algorithms which would work in the presence of interference.

- **CDS construction in mobile network:** In our proposed algorithms we have assumed that the network is static, that means once the nodes are deployed they do not change their position till the end of their life. However, the wireless network can be formed using the mobile nodes also. Suppose we have a CDS currently working in a wireless network. If some of the CDS nodes change their position, then we need a maintenance process to repair the CDS. Sometimes also after a certain duration, we need to reconstruct the CDS. Therefore, we should look for distributed CDS construction algorithms in the mobility model with a facility of maintenance in case of a change in the position of CDS nodes.

- **Maintenance of CDS in case of failure of multiple nodes at a time:** Our CDS maintenance algorithm would work properly when multiple nodes would not fail at a time. The proposed algorithm does not consider the situation where multiple nodes fail together. Although this situation is very rare, in some applications like military surveillance it happens. So, finding CDS maintenance algorithms which can work in case of failure of multiple nodes is essential.

References

[1] D. C. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, “Research challenges in environmental observation and forecasting systems,” in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking.* ACM, 2000, pp. 292–299.

[2] A. Salhieh, J. Weinmann, M. Kochhal, and L. Schwiebert, “Power efficient topologies for wireless sensor networks,” in *International Conference on Parallel Processing, 2001.* IEEE, 2001, pp. 156–163.

[3] J. M. Kahn, R. H. Katz, and K. S. Pister, “Next century challenges: mobile networking for smart dust,” in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking.* ACM, 1999, pp. 271–278.

[4] B. Warneke, M. Last, B. Liebowitz, and K. S. Pister, “Smart dust: Communicating with a cubic-millimeter computer,” *Computer*, vol. 34, no. 1, pp. 44–51, 2001.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[6] S. S. Pradhan and K. Ramchandran, “Distributed source coding: Symmetric rates and applications to sensor networks,” in *Proceedings DCC 2000. Data Compression Conference.* IEEE, 2000, pp. 363–372.

[7] S. S. Pradhan, J. Kusuma, and K. Ramchandran, “Distributed compression in a dense microsensor network,” *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, 2002.

[8] S. Soro and W. B. Heinzelman, “Cluster head election techniques for coverage preservation in wireless sensor networks,” *Ad Hoc Networks*, vol. 7, no. 5, pp. 955–972, 2009.

[9] S. Butenko, S. Kahruman-Anderoglu, and O. Ursulenko, “On connected domination in unit ball graphs,” *Optimization Letters*, vol. 5, no. 2, pp. 195–205, 2011.

[10] S. Andrew, “Tanenbaum computer networks,” *Computer Networks, Englewood Cliffs*, pp. 141–148, 1996.

[11] B. Han, “Zone-based virtual backbone formation in wireless ad hoc networks,” *Ad Hoc Networks*, vol. 7, no. 1, pp. 183–200, 2009.

[12] A. Ephremides, J. E. Wieselthier, and D. J. Baker, “A design concept for reliable mobile radio networks with frequency hopping signaling,” *Proceedings of the IEEE*, vol. 75, no. 1, pp. 56–73, 1987.

[13] D. Kim, Y. Wu, Y. Li, F. Zou, and D.-Z. Du, "Constructing minimum connected dominating sets with bounded diameters in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 2, pp. 147–157, 2009.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness.* WH Freeman and Company, New York, 1979.

[15] B. Han and W. Jia, "Clustering wireless ad hoc networks with weakly connected dominating set," *Journal of Parallel and Distributed Computing*, vol. 67, no. 6, pp. 727–737, 2007.

[16] J. Akbari Torkestani and M. R. Meybodi, "An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata," *Computer Networks*, vol. 54, no. 5, pp. 826–843, 2010.

[17] J. A. Torkestani and M. R. Meybodi, "Weighted steiner connected dominating set and its application to multicast routing in wireless manets," *Wireless Personal Communications*, vol. 60, no. 2, pp. 145–169, 2011.

[18] B. An and S. Papavassiliou, "A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks," *International Journal of Network Management*, vol. 11, no. 6, pp. 387–395, 2001.

[19] M. Gerla and J. T.-C. Tsai, "Multicluster, mobile, multimedia radio network," *Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.

[20] R. Sivakumar, P. Sinha, and V. Bharghavan, "Cedar: a core-extraction distributed ad hoc routing algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1454–1465, 1999.

[21] P. Sinha, R. Sivakumar, and V. Bharghavan, "Enhancing ad hoc routing with dynamic virtual infrastructures," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3. IEEE, 2001, pp. 1763–1772.

[22] R. De Gaudenzi, T. Garde, F. Giannetti, and M. Luise, "Ds-cdma techniques for mobile and personal satellite communications: An overview," in *IEEE Second Symposium on Communications and Vehicular Technology in the Benelux.* IEEE, 1994, pp. 113–127.

[23] S. R. Bevan Das and V. Bharghavan, "Routing in ad-hoc networks using a virtual backbone," in *Proceedings of the 6th International Conference on Computer Communications and Networks (IC3N'97)*, 1997, pp. 1–20.

[24] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proceedings of ICC'97-International Conference on Communications*, vol. 1. IEEE, 1997, pp. 376–380.

[25] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications.* Citeseer, 1999, pp. 7–14.

[26] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in ad hoc networks using a spine." in *ICCCN*, 1997, pp. 34–41.

[27] R. Sivakumar, B. Das, and V. Bharghavan, "An improved spine-based infrastructure for routing in ad hoc networks," in *IEEE Symposium on Computers and Communications*, vol. 98, 1998.

[28] Y.-L. Chang and C.-C. Hsu, "Routing in wireless/mobile ad-hoc networks via dynamic group construction," *Mobile Networks and Applications*, vol. 5, no. 1, pp. 27–37, 2004.

[29] A. B. McDonald, "A mobility-based framework for adaptive dynamic cluster-based hybrid routing in wireless ad-hoc networks," in *Wireless Ad Hoc Networks, Ph. D. Dissertation, Univ. of Pittsburgh.* Citeseer, 1999.

[30] S. Datta, I. Stojmenovic, and J. Wu, "Internal node and shortcut based routing with guaranteed delivery in wireless networks," *Cluster Computing*, vol. 5, no. 2, pp. 169–178, 2002.

[31] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, no. 2-3, pp. 153–167, 2002.

[32] X. Cheng, X. Huang, D. Li, and D.-z. Du, "On the construction of connected dominating set in ad hoc wireless networks," in *Wireless Communications and Mobile Computing.* Citeseer, 2006.

[33] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems.* ACM, 2000, pp. 61–68.

[34] J. Wu and F. Dai, "Broadcasting in ad hoc networks based on self-pruning," *International Journal of Foundations of Computer Science*, vol. 14, no. 02, pp. 201–221, 2003.

[35] J. Wu and B. Wu, "A transmission range reduction scheme for power-aware broadcasting in ad hoc networks using connected dominating sets," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*, vol. 5. IEEE, 2003, pp. 2906–2909.

[36] J. Wu, B. Wu, and I. Stojmenovic, "Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets," *Wireless Communications and Mobile Computing*, vol. 3, no. 4, pp. 425–438, 2003.

[37] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[38] M. Ding, X. Cheng, and G. Xue, "Aggregation tree construction in sensor networks," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*, vol. 4. IEEE, 2003, pp. 2168–2172.

[39] J. A. Shaikh, J. Solano, I. Stojmenovic, and J. Wu, "New metrics for dominating set based energy efficient activity scheduling in ad hoc networks," in *28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN'03. Proceedings.* IEEE, 2003, pp. 726–735.

[40] J. Wu, F. Dai, M. Gao, and I. Stojmenovic, "On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks," *Journal of Communications and Networks*, vol. 4, no. 1, pp. 59–70, 2002.

[41] J. Wu and B. Wu, "A transmission range reduction scheme for power-aware broadcasting in ad hoc networks using connected dominating sets," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*, vol. 5. IEEE, 2003, pp. 2906–2909.

[42] B. Deb, S. Bhatnagar, and B. Nath, "Multi-resolution state retrieval in sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.* IEEE, 2003, pp. 19–29.

[43] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in ad hoc networks using a spine," in *Computer Communications and Networks, 1997. Proceedings., Sixth International Conference on.* IEEE, 1997, pp. 34–39.

[44] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1-3, pp. 165–177, 1990.

[45] R. Misra and C. Mandal, "Rotation of cds via connected domatic partition in ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 4, pp. 488–499, 2009.

[46] ——, "Efficient clusterhead rotation via domatic partition in self-organizing sensor networks," *Wireless Communications & Mobile Computing*, vol. 9, no. 8, pp. 1040–1058, 2009.

[47] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks.* ACM, 2004, pp. 20–27.

[48] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime prediction for energy-aware computing," in *Proceedings of the 2002 International Symposium on Low Power Electronics and Design.* ACM, 2002, pp. 154–159.

[49] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 2, no. 3, pp. 277–324, 2003.

[50] C. Knight, J. Davidson, and S. Behrens, "Energy options for wireless sensor nodes," *Sensors*, vol. 8, no. 12, pp. 8037–8066, 2008.

[51] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking.* ACM, 2001, pp. 70–84.

[52] D. M. Blough and P. Santi, "Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks," in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking.* ACM, 2002, pp. 183–192.

[53] W. B. Heinzelman, A. P. Chandrakasan, H. Balakrishnan *et al.*, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[54] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences.* IEEE, 2000, pp. 10–pp.

[55] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.

[56] D. Zhou, M.-T. Sun, and T.-H. Lai, "A timer-based protocol for connected dominating set construction in ieee 802.11 multihop mobile ad hoc networks," in *The 2005 Symposium on Applications and the Internet.* IEEE, 2005, pp. 2–8.

[57] F. Dai and J. Wu, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks," *IEEE Transactions on Parallel & Distributed Systems*, no. 10, pp. 908–920, 2004.

[58] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[59] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K.-I. Ko, "A greedy approximation for minimum connected dominating sets," *Theoretical Computer Science*, vol. 329, no. 1, pp. 325–330, 2004.

[60] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-optimal connected dominating sets in mobile ad hoc networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing.* ACM, 2002, pp. 157–164.

[61] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 1, pp. 14–25, 2002.

[62] P.-J. Wan, K. M. Alzoubi, and O. Friede, "Distributed construction of connected dominating set in wireless ad hoc networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1597–1604.

[63] C. Adjih, P. Jacquet, and L. Viennot, "Computing connected dominated sets with multipoint relays," Ph.D. dissertation, INRIA, 2002.

[64] K. Islam, S. G. Akl, and H. Meijer, "A constant factor localized algorithm for computing connected dominating sets in wireless sensor networks," in *2008 14th IEEE International Conference on Parallel and Distributed Systems.* IEEE, 2008, pp. 559–566.

[65] M. Cardei, M. X. Cheng, X. Cheng, and D.-Z. Du, "Connected domination in multihop ad hoc wireless networks." *JCIS*, pp. 251–255, 2002.

[66] Y. L. S. Zhu and M. T. D.-Z. Du, "Localized construction of connected dominating set in wireless networks," in *Proceedings of US National Science Foundation International Workshop Theoritical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks*, 2004.

[67] X. Cheng, M. Ding, and D. Chen, "An approximation algorithm for connected dominating set in ad hoc networks," in *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks (TAWN)*, vol. 2, no. 4, 2004.

[68] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum cds in unit disk graphs," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 444–453, 2006.

[69] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences.* IEEE, 2002, pp. 3866–3875.

[70] K. Sakai, F. Shen, K. M. Kim, M.-T. Sun, and H. Okada, "Multi-initiator connected dominating set construction for mobile ad hoc networks," in *2008 IEEE International Conference on Communications.* IEEE, 2008, pp. 2431–2436.

[71] Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, and D.-Z. Du, "On greedy construction of connected dominating sets in wireless networks," *Wireless Communications and Mobile Computing*, vol. 5, no. 8, pp. 927–932, 2005.

[72] R. Misra and C. Mandal, “Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 292–302, pp. 292 – 302, 2010.

[73] H. Du, W. Wu, Q. Ye, D. Li, W. Lee, and X. Xu, “Cds-based virtual backbone construction with guaranteed routing cost in wireless sensor networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 652–661, 2013.

[74] Z. Yuanyuan, X. Jia, and H. Yanxiang, “Energy efficient distributed connected dominating sets construction in wireless sensor networks,” in *Proceedings of the 2006 international conference on Wireless communications and mobile computing.* ACM, 2006, pp. 797–802.

[75] N. Meghanathan, “An algorithm to determine energy-aware connected dominating set and data gathering tree for wireless sensor networks.” in *ICWN*, 2009, pp. 608–614.

[76] T. Nieberg and J. Hurink, “A ptas for the minimum dominating set problem in unit disk graphs,” in *International Workshop on Approximation and Online Algorithms.* Springer, 2005, pp. 296–306.

[77] F. Zou, X. Li, D. Kim, and W. Wu, “Construction of minimum connected dominating set in 3-dimensional wireless network,” in *International Conference on Wireless Algorithms, Systems, and Applications.* Springer, 2008, pp. 134–140.

[78] A. D. Amis, R. Prakash, T. H. Vuong, and D. T. Huynh, “Max-min d-cluster formation in wireless ad hoc networks,” in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 1. IEEE, 2000, pp. 32–41.

[79] F. Dai and J. Wu, “On constructing k-connected k-dominating set in wireless ad hoc and sensor networks,” *Journal of Parallel and Distributed Computing*, vol. 66, no. 7, pp. 947–958, 2006.

[80] Y. Wu and Y. Li, “Construction algorithms for k-connected m-dominating sets in wireless sensor networks,” in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing.* ACM, 2008, pp. 83–90.

[81] M. T. Thai, N. Zhang, R. Tiwari, and X. Xu, “On approximation algorithms of k-connected m-dominating sets in disk graphs,” *Theoretical Computer Science*, vol. 385, no. 1-3, pp. 49–59, 2007.

[82] Z. Zhang, J. Zhou, Y. Mo, and D.-Z. Du, “Performance-guaranteed approximation algorithm for fault-tolerant connected dominating set in wireless networks,” in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications.* IEEE, 2016, pp. 1–8.

[83] W. Wang, B. Liu, D. Kim, and D. Li, “A new constant factor approximation to construct highly fault-tolerant connected dominating set in unit disk graph,” *IEEE/ACM Transactions on Networking*, no. 99, p. 1, 2016.

[84] D. Kim, W. Wang, X. Li, Z. Zhang, and W. Wu, “A new constant factor approximation for computing 3-connected m-dominating sets in homogeneous wireless networks,” in *2010 Proceedings IEEE INFOCOM.* IEEE, 2010, pp. 1–9.

[85] F. Wang, M. T. Thai, and D.-Z. Du, “On the construction of 2-connected virtual backbone in wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1230–1237, 2009.

[86] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240)*, vol. 2. IEEE, 2001, pp. 472–476.

[87] M. Cardei, D. MacCallum, M. X. Cheng, M. Min, X. Jia, D. Li, and D.-Z. Du, "Wireless sensor networks with energy efficient organization," *Journal of Interconnection Networks*, vol. 3, no. 03n04, pp. 213–229, 2002.

[88] T. Shi, S. Cheng, Z. Cai, Y. Li, and J. Li, "Exploring connected dominating sets in energy harvest networks," *IEEE/ACM Transactions on Networking*, 2017.

[89] R. Misra and C. Mandal, "Ant-aggregation: ant colony algorithm for optimal data aggregation in wireless sensor networks." in *International Conference on Wireless and Optical Communications Networks, 2006 IFIP*. IEEE, 2006, p. 5.

[90] D. Du, L. Wang, and B. Xu, "The euclidean bottleneck steiner tree and steiner tree with minimum number of steiner points," in *International Computing and Combinatorics Conference*. Springer, 2001, pp. 509–518.

[91] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. ACM, 1987, pp. 230–240.

[92] W. Wu, H. Du, X. Jia, Y. Li, and S. C.-H. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs," *Theoretical Computer Science*, vol. 352, no. 1-3, pp. 1–7, 2006.

Publications from this Thesis

Journals

- **Jasaswi Prasad Mohanty**, Chittaranjan Mandal, and Chris Reade, ”Construction of minimum connected dominating set in wireless sensor networks using pseudo dominating set.” in *Ad Hoc Networks (Elsevier) 42*, (2016): 61-73.
- **Jasaswi Prasad Mohanty**, Chittaranjan Mandal, and Chris Reade, ””Distributed construction of minimum connected dominating set in wireless sensor network using two-hop information.” in *Computer Networks (Elsevier) 123*, (2017): 137-152.

Conferences

- **Jasaswi Prasad Mohanty**, and Chittaranjan Mandal, ”A distributed greedy algorithm for construction of minimum connected dominating set in wireless sensor network.” in 2014 Applications and Innovations in Mobile Computing (AIMoC). IEEE, 2014.

Book Chapters

- **Jasaswi Prasad Mohanty**, and Chittaranjan Mandal, ”Connected Dominating Set in Wireless Sensor Network.” Handbook of Research on Advanced Wireless Sensor Network Applications, Protocols, and Architectures. IGI Global, 2017. 62-85.

Under Preparation

- **Jasaswi Prasad Mohanty**, and Chittaranjan Mandal, ”Distributed Maintenance of Connected Dominating set as Virtual Backbone in Wireless Sensor Network.”

Author's Biography

Jasaswi Prasad Mohanty is a Ph.D. candidate from Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur. He completed his M. Tech. degree in Computer Science from Utkal University during 2004-06. He is currently working as an Sr. Assistant Professor in the department of Computer Science and Engineering, Silicon Institute of Technology, Bhubaneswar, Odisha, India.

Contact Information

Department of Computer Science and Engineering,
Silicon Institute of Technology,
Bhubaneswar, PIN–751024, India.
Mobile: +91-9437564198
Email: jasaswiprasad@gmail.com

Research Interests

Wireless Network, Distributed Systems.