

M.Sc. Engg. Thesis

**FAULT TOLERANT MULTIPLE DOMINATING
SET CONSTRUCTIONS FOR WIRELESS
AD-HOC NETWORKS**

By

Khaleda Akther Papry (ID: 1015052041)

Submitted to

Department of Computer Science and Engineering
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering





Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1205


09 November, 2021

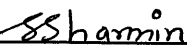
The thesis titled “**FAULT TOLERANT MULTIPLE DOMINATING SET CONSTRUCTIONS FOR WIRELESS AD-HOC NETWORKS**”, submitted by Khaleda Akther Papry, Roll No. 1015052041, Session October 2015, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on 09 November, 2021.

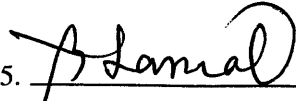
Board of Examiners

1. 

Dr. A.K.M. Ashikur Rahman
Professor
Department of CSE
BUET, Dhaka 1205.
Chairman
(Supervisor)
2. 

Head
Department of CSE
BUET, Dhaka 1205.
(Ex-Officio)
3. 

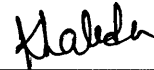
Dr. Rifat Shahriyar
Professor
Department of CSE
BUET, Dhaka 1205.
Member
4. 

Dr. Sadia Sharmin
Assistant Professor
Department of CSE
BUET, Dhaka 1205.
Member
5. 

Dr. Dr. Bilkis Jamal Ferdosi
Professor
Department of Computer Science & Engineering
University of Asia Pacific, Dhaka.
Member
(External)

Candidate's Declaration

This is to certify that the work presented in this thesis entitled “**FAULT TOLERANT MULTIPLE DOMINATING SET CONSTRUCTIONS FOR WIRELESS AD-HOC NETWORKS**” is the outcome of the investigation carried out by me under the supervision of Professor Dr. A.K.M. Ashikur Rahman in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka. It is also declared that neither this thesis nor any part thereof has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.



Khaleda Akther Papry
Candidate

Contents

<i>Board of Examiners</i>	i
<i>Candidate's Declaration</i>	ii
Acknowledgements	viii
Abstract	ix
1 Introduction	1
1.1 Connected dominating set (CDS) as virtual backbone	2
1.2 Motivation	3
1.3 Contribution	4
1.4 Thesis Organization	5
2 Related Work	6
2.1 Different broadcasting Algorithms for wireless Adhoc Networks	6
2.2 CDS based broadcasting Algorithms for wireless Adhoc Networks	7
2.3 Fault tolerant broadcasting algorithms	9
3 Preliminaries	11
3.1 Network Model	11
3.2 Assumptions and Notations	12
3.3 Definitions	13
3.3.1 Minimum Connected Dominating Sets (MCDSs)	14
3.3.2 Multiple MCDS (MMCDS)	14
3.3.3 Overlapping Boundary (K)	14
3.3.4 Dominant Pruning	15

4	Multiple Minimum Connected Dominating Sets (MMCDSs) Algorithm	16
4.1	Centralized Algorithm for MMCDSs	16
4.1.1	Complexity Analysis of Centralized MMCDS	20
4.2	Distributed Algorithm for MMCDSs	22
4.2.1	Complexity Analysis of Distributed MMCDS	27
4.3	Network lifetime	27
4.4	Fault tolerance	30
5	Experiments	36
5.1	Experimental Setup	36
5.2	Performance Measurements	37
5.2.1	Multiple MCDSs evaluation	37
5.2.2	Network lifetime	38
5.2.3	Average Forwarding Nodes	38
5.2.4	Network Fault Tolerance	39
5.3	Experiment Results	39
5.3.1	Performance based on Number of MMCDSs	40
5.3.2	Performance based on Network Lifetime	40
5.3.3	Performance based on Network Fault Tolerance	42
5.3.4	Performance based on Average Forwarding Nodes	45
5.3.5	Execution Time of MMCDSs Algorithm	46
6	Conclusion and Future Work	48

List of Figures

1.1	An ad hoc network with uncontrolled flooding	2
1.2	An Ad hoc network topology with 7 nodes	3
3.1	A randomly deployed Adhoc network with 10 nodes and their transmission range	12
3.2	Graph representation of the network with 10 nodes	13
3.3	Some important notations for dominant pruning	15
4.1	Basic Greedy Centralized Construction of MCDS (a) ~ (c) and multiple MCDS construction (d) ~ (i)	19
4.2	Basic Greedy Distributed Forwarding List Construction of MCDS (a) ~ (b) and multiple MCDS (c) ~ (e) construction for static scenario	24
4.3	New one-hop and two-hop uncovered sets of node A (a), new forwarding list generation (b) ~ (c) for dynamic scenario	25
4.4	A CDS (Green nodes) for communication over the network	31
4.5	A node (Red node) of the CDS becomes faulty	32
4.6	Another backup CDS (Green nodes) run for communication over the network	32
4.7	A forwarding node list (CDS) for communication over the network	33
4.8	A node (Red node) of the CDS becomes faulty	33
4.9	Another backup forwarding node list (CDS) run for communication over the network	34
5.1	Number of MMCDSs construction for different overlapping boundary varying network size	41
5.2	Results for size of MMCDS construction with different values of K	41
5.3	Number of MMCDSs construction for different overlapping boundary	42

5.4	Average network lifetime for different overlapping boundary varying network size	43
5.5	Results of average network lifetime for different overlapping boundary values on (a) Sparse graph and (b) Dense graph	43
5.6	Average Network Fault Tolerance for different overlapping boundary varying network size	44
5.7	Results of average Fault Tolerance for different overlapping boundary values on (a) Sparse graph and (b) Dense graph	45
5.8	Average Forwarding Nodes of different centralized algorithms along with our MMCDSs with $K=1$	46
5.9	Average Forwarding Nodes of distributed algorithms along with our MMCDSs with $K=1$ and $K=2$	46
5.10	Execution Time of our MMCDSs with $K=1$ and $K=2$	47

List of Tables

3.1	Table for the notations used in Centralized MMCDs algorithms	13
3.2	Table for the notations used in Distributed MMCDs algorithms	14
4.1	Table for the forwarding List of all nodes	23
5.1	Table for simulation setup parameters	37
5.2	Table for the average forwarding nodes of different algorithms	45

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Dr. A.K.M Ashikur Rahman for the continuous support of my M.Sc. study and research. His guidance, motivation and deep knowledge in research area helped me all along this research as well as writing and implementation of the thesis. He has always driven me in the right direction when I was in depression and confusion while selecting my research topic. I am really grateful to him, his patience and enthusiasm. I could not have imagined my M.Sc. study without his continuous support.

I am also grateful to all the board members of my M.Sc. thesis defense for their valuable feedback and deep observations in my thesis work.

Abstract

In wireless network, broadcasting is the most common communication method. To reduce redundancy, traffic and collision induced by broadcasting, different virtual backbones are used on top of the physical topology and Connected Dominating Set (CDS) is one of those. However, constructing minimum connected dominating set (MCDS) containing minimum number of nodes participating in packet forwarding is an NP-complete problem. Although some approximation algorithms are available, the CDS or its approximation has poor fault tolerance as in a CDS one vertex not in CDS is exactly connected with one vertex in CDS. In this work, we present two heuristics, one centralized and the other distributed for constructing multiple connected dominating sets providing enhanced fault tolerance of the network. Both algorithms are intended to maximize network lifetime involving minimal nodes. Moreover, both the algorithms also ensure load balancing over the network. Finally, we simulate our result to show the improvement of network lifetime and system fault tolerance.

Chapter 1

Introduction

Wireless Ad hoc networks consist of some wireless nodes that communicate over the network without the existence of any fixed infrastructure. They can directly communicate the neighbor nodes which are within their transmission ranges. The nodes use other intermediate nodes as relay nodes to communicate with those nodes that are no longer in their transmission range [1]. This communication can be of three types: unicast, multicast and broadcast. In wireless Ad hoc network, broadcasting is the most common communication method where each node over the network receives the message from a source node. There are many approaches for such communication method. Uncontrolled flooding is the easiest approach for broadcasting where each node unconditionally distributes its incoming packets to each of its neighbors. In this process no node prevents the re circulation of the same packet. Therefore, it causes too much traffic, contention and collision resulting into broadcast storm problem [2]. Figure 1.1 shows a scenario of five nodes where uncontrolled flooding is used. Each node forwards packets to all of its neighbors. We can see that each node receives the same packets multiple times, for example node C receives same packets 4 times. The situation is worse in the larger network. This problem can be minimized by creating a virtual backbone on physical topology and run any routing protocol over this backbone so that it can minimize the re-circulation of same packets. The most promising virtual backbone is approximated by Connected Dominating Set (CDS) [3].

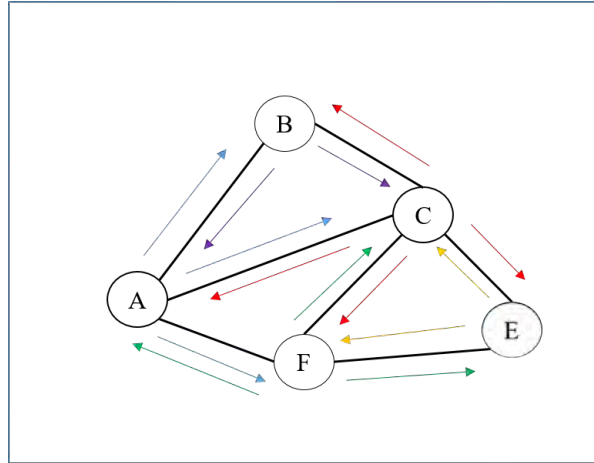


Figure 1.1: An ad hoc network with uncontrolled flooding

1.1 Connected dominating set (CDS) as virtual backbone

A CDS is the subset of a graph where all nodes within the set are connected and the other nodes are 1-hop neighbour of at least one of the member of CDS. A CDS in an ad hoc network can serve as a backbone for packet routing over the network. Figure 1.2 shows a network with seven nodes where connectivity means their transmission area. Here, some possible CDSs are: $\{A,C\}$, $\{A,B,F\}$, $\{A,B,C\}$, $\{C,B,E\}$, $\{A,B,G\}$, $\{A,C,E\}$, $\{B,C,D\}$, $\{A,B,C,D\}$, and so on. However, a small size CDS is desirable in many applications. The less nodes in a CDS, the more efficient a network is as along with routing redundancy, the number of forwarding packets are also decreased. For example, for the previous graph, CDS $\{A,C\}$ is more desirable than other CD sets. In this case, only A, B, and C can receive a duplicate message.

A CDS with minimum number of nodes is called minimum connected dominating set (MCDS). However, finding an MCDS of a graph is an NP-complete problem [4]. Therefore, we need to apply heuristics to find out most efficient MCDS. A lot of researches have already been performed in this regard. Moreover, nodes are usually battery operated. Therefore, load balancing among the nodes ensure proper utilisation of energy over the network and increase network lifetime as well. Another serious issue is fault tolerance ability of a network. There may contain some nodes in a network which may fail to forward packets or communicate due to power failures or other errors. Moreover, a faulty node may contain in the network misleading the routing by dropping it's upcoming packets. This type of failures may interrupt the whole network to perform properly. Therefore, if there is a backup CDS for routing, the system might work properly again. Moreover, there may occur some situations when some nodes may become un-

trusted and temporarily become unreachable [5]. Therefore, sometimes, the network requires multi-coverage or multiple times packets receiving. These failure problems and requirements in broadcasting requires fault-tolerant broadcasting algorithm. In recent years, ensuring system fault tolerance is another prominent issue in research area.

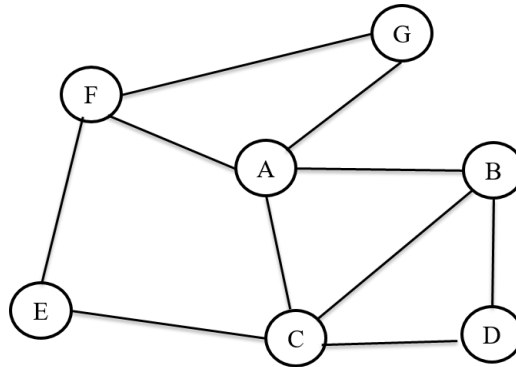


Figure 1.2: An Ad hoc network topology with 7 nodes

1.2 Motivation

To make broadcasting algorithms more fault tolerant and to increase the network lifetime, we need to improve the existing CDS based algorithms. One idea is to generate multiple connected dominating sets (CDS) and use them according to the network requirements. However, it is not always possible to generate all disjoint CDSs in a network. Hence, we plan to develop a fault tolerant broadcasting algorithm by generating multiple minimum connected dominating sets (MMCDSs) allowing overlaps within the sets to some extents [6]. Additionally, While generating multiple sets allowing overlaps we will try select node with minimum overlaps among the sets. However, it sometimes might add some redundant nodes in a set and hence we plan to apply some optimization techniques with proper modifications of the existing algorithms required to ensure fault-tolerance and network longevity.

Finding minimum connected dominating set (MCDS) is NP-complete problem. Moreover, constructing multiple minimum connected dominating sets are also NP-complete as they are harder problem. Finding out all possible CDSs are not useful in real environment as any node might be used in several sets. On the other hand, only disjoint CDSs might not be feasible for all the network. Therefore, we can consider generating multiple CDSs allowing some overlapping within the nodes which is similar to the work in wireless directional sensor networks proposed

in [6]. Hence, we introduce a variable K which limits the overlapping boundary of CDSs. The value of K indicates the possible presence of a node over K number of CD sets. In this work, we target to construct all possible minimum connected dominating sets but limiting the use of a single node K times. Therefore, one node can only be present up to K sets which will save energy ensuring fault tolerance at the same time.

1.3 Contribution

The main idea of this work is to find out multiple minimum connected dominating sets (MM-CDS) using minimal nodes. The construction of a MCDS can be divided into two categories based on the network topology they use. These are centralized and distributed algorithm. Centralised algorithm relies on entire or global topology and one node works as the main node that processes the entire network centrally. However, in wireless ad hoc networks the global topology is not always known. Most of the cases, we can not derive the global topology. In that case, distributed algorithms are applied where the routing are taken by the participating nodes and the decision process is decentralised. Here, we implement both types of algorithms for multiple MCDS construction: centralized and distributed. In our centralized algorithm, we use the basic greedy construction of MCDS [7] every time and consider minimum overlapping (up to a certain value K) among the sets. On the other hand, we use the concept of distributed multiple MCDS algorithm based on dominant pruning algorithm [8] allowing the same overlapping criteria as the centralized one.

Moreover, the wireless ad hoc network is not always static, the scenario might change abruptly due to the change of node's position. In the static scenario, the nodes do not change and if the packet forwarding request comes from the same previous node, it can choose another set from the already created multiple CD sets, instead of creating all sets once again. The node can keep on selecting dominating sets in a round robin fashion thus ensuring load balancing among the neighbor nodes. The static scenario is more prevalent in Wireless Sensor networks which is a subclass of Wireless Ad hoc Networks. Moreover, fault tolerance, load balancing, network longevity etc, makes much more sense in those networks. For example, suppose a node is using one dominating set and suddenly a node within the set starts malfunctioning, it can immediately replace the dominating set with another one from an already built CDSs list. On the other hand,

in mobile scenario, the situation is not that simple. Every time the topology can be changed. However, we can still apply the same strategy but in a little conservative way. For example, the node can keep track of which dominating sets were used last several times for forwarding. Then it can create new dominating set and choose the one which has minimal overlaps with the previous sets, no matter which node was its predecessor. Therefore, in distributed system we can apply our algorithm in both ways, just making some little changes.

The main objectives of our work are enumerated below:

- i. To develop a centralized algorithm that will find out multiple MCDS allowing overlapping of nodes at most K times.
- ii. To develop a distributed approach for MMCDS using 2-hop neighbor information and overlapping boundary K both for static and mobile scenarios.
- iii. Finally, to validate the algorithm using simulation results, compare the performance of the two algorithms with existing algorithms based on the performance criteria like network lifetime, fault tolerance etc.

The possible outcomes of the proposed research work are as follows:

- i. An approach that enhance the fault tolerance of the system by ensuring multiple CDS.
- ii. Make the network more energy efficient and enhance the network lifetime by balancing load among the nodes.
- iii. Make multi path routing for system fault tolerance.
- iv. A comprehensive comparison of the proposed two algorithms through extensive simulations.

1.4 Thesis Organization

The remainder of the thesis after introduction is organized as follows: chapter 2 discusses on related works on connected dominating set (CDS) and fault-tolerance in CDS. Two proposed multiple MCDS algorithms with example scenarios are described to understand their functions thoroughly are discussed in chapter 4. Simulation results are shown in chapter 5. Finally chapter 6 concludes the thesis with some possible future works.

Chapter 2

Related Work

In this chapter, we discuss broadcasting algorithms that focus on reducing broadcast storm problem. Several works have been done to reduce the problem. In section 2.1 we discuss some broadcasting algorithm regarding the problem. Section 2.2 presents connected dominating based different broadcasting algorithms. Works related to fault-tolerant optimized algorithm are presented in Section 2.3.

2.1 Different broadcasting Algorithms for wireless Adhoc Networks

Many broadcasting algorithms have been proposed over the decades to overcome the broadcast storm problem. H. Lim and C. Kim studied efficient routing mechanisms for multicast and broadcast in ad hoc wireless networks in paper [9]. If a packet is broadcast to all neighboring nodes, the optimality of the wireless network interrupts. Hence, the authors provided two new flooding methods for broadcasting, self pruning and dominant pruning in paper [8]. In self pruning algorithm, each node uses its one-hop neighbors information and prunes itself from rebroadcasting if all its neighbors have already received the packet. In dominant pruning algorithm, each node uses its 2-hop neighbor information to reduce redundant transmissions. Therefore, both methods utilize neighbor information to reduce redundant transmissions and especially, dominant pruning performs significantly better than blind flooding. W. Peng and X. lu provided a self pruning based mechanism in paper [10], where local topology and statistical

information are gathered to reduce duplicate broadcasts. When a node receives a message, it delays the rebroadcast for a random time and within this time all duplicates are discarded.

To reduce broadcast redundancy in wireless ad-hoc networks, W. Lou and J. Wu [11] proposed two improved algorithms based on dominant pruning: Total Dominant Pruning and Partial Dominant Pruning. In Total Dominant pruning, the rebroadcasting is reduced more, however, it requires 3-hop neighbor information which increases message redundancy and overhead in the network. On the other hand, Partial Dominant Pruning requires only 2-hop neighbor information as generic Dominant Pruning. Therefore, it requires less overhead than the first one. G. Calinescu et al. proposed location aware pruning methods [12] for reducing re broadcasting with minimum forwarding nodes based on the the two proposed heuristics presented in [8]. A. Rahman et al. proposed enhanced dominant pruning-based broadcasting in untrusted ad-hoc wireless networks in paper [13]. Moreover, the authors also provided enhanced partial dominant pruning (EPDP) based broadcasting in Ad hoc Wireless Networks [14]. Both of their methods provides more efficiency in reducing rebroadcasting than the generic ones.

Y. Kim and E.C. Park proposed relay-based broadcasting mechanism in wireless ad hoc networks for various emerging applications in the Internet of Things (IoT) in paper [15]. To handle the broadcast storm problem, the authors proposed a reasonable criterion called duplication ratio based on the number of adjacent nodes. Based on this ratio duplicate frames are discarded in a probabilistic manner to decrease the redundancy. However, there is a re-queuing scheme which provides a re-transmission opportunity for the delivery failure as well. Moreover, paper [16] is proposed by M.K. Goyal et al., where the number of forwarding nodes is derived from 1-hop neighbors to cover entire 2-hop neighbors by utilizing 2-hop region information. For this purpose, network coding is used to minimize repetitive communications.

2.2 CDS based broadcasting Algorithms for wireless Adhoc Networks

Ephremides et al. [17] first proposed the idea of using a CDS as virtual backbone for broadcasting. Das and Bharghavan [18] proposed a simple centralized algorithm for constructing CDS,

however, the size of the CDS was very large. Therefore, the idea of constructing a CDS with minimum nodes (MCDS) emerged. Constructing minimum connected dominating set (MCDS) is NP-complete problem. Several researches have been performed for approximating the solution. A lot of algorithms have already been proposed for constructing MCDS which can be of two types: centralized and distributed. In actual environment, distributed algorithms are more suitable than centralized one. Guha and Khullar [19] used first greedy approach to construct centralized MCDS. An approximation of distributed algorithms for constructing MCDS have been proposed in [7] by the authors as well.

In paper [20], Y.P. Chen et al. proposed an approximation algorithm for weakly connected dominating set (WCDS). L. Ruan et al. [21] proposed a new one-step greedy approximation to construct a Minimum Connected Dominating Set (MCDS). In paper [22], Stojmenovic et al. proposed a connected dominating set and neighbor elimination based broadcasting algorithm to reduce the communication overhead and to ensure reliable broadcasting. Paper [23] presents a new energy efficient distributed Connected Dominating Set (CDS) algorithm for mobile ad-hoc networks where a node is replaced with another one with activity scheduling when it lacks energy power.

In [24], the authors proposed a greedy algorithm for MCDS in unit-disk graphs based on Maximal Independent Set (MIS). Another work by Al-Nabhan et al. [25], proposed three centralized algorithms to construct CDS to minimize the size of the CDS.

Paper [26] proposed distributed greedy approximation algorithm for CDS construction which reduces the CDS size effectively in wireless sensor networks using two-hop information with lower construction cost.

T.N. Trant et al. [27] proposed an efficient Connected Dominating Set clustering based routing protocol in cognitive mobile ad-hoc Networks utilizing the dynamic channel allocation. Moreover, G. Omer et al. in paper [28] proposed distributed CDS algorithm for wireless sensor networks with solar energy harvester nodes for smart agriculture applications namely CDSSEHA. In this method, environmental features are gathered to optimize production parameters with lower battery power consumption. Additionally, two meta heuristic algorithms in paper [29] are proposed to construct minimum connected dominating set in wireless networks. The first algorithm is Memetic Algorithm for the MCDS (MA-MCDS) based on genetic algorithm and local search strategies. In the second one, simulated annealing and stochastic local

search is used for MCDS construction. Most of the algorithms are based on homogeneous networks and 2D networks. For 3D heterogeneous network where nodes sensing ranges are different, B. Xin et al. proposed the construction of minimum connected dominating set using Maximal Independent Set (MIS) [30].

2.3 Fault tolerant broadcasting algorithms

Fault tolerant is one of the major issues in broadcasting algorithm. A lot of researches have already been performed for fault tolerant mechanism for broadcasting in ad-hoc networks. In paper [31], the authors studied approximation algorithms for fault-tolerant clustering using k -fold dominating set of a graph where S is the subset of $G(V, E)$ such that every node $v \in V - S$ has at least k neighbors in S . In paper [32], J. R. Diaz et al., proposed a fault tolerant method for multimedia flows based on Fast Switching Paths. In the work, the authors showed how a node might fall while in routing and it recovers the routing mechanism using instant path switching instead of sending the fault message to the source. Paper [33, 34] provides an approximation algorithm to build a fault tolerant CDS model named as k -connected m -dominating set where the CDS is k -connected, and each node not in CDS is dominated (adjacent) by at least m nodes in CDS. However, to achieve this mechanism, each node must be connected with at least m nodes where $m \geq k$ and for larger values of m the number of rebroadcasting increases.

Paper [35, 36] proposed both centralized and distributed approach for contention aware connected dominating sets to minimize transmission over a shared channel. P. Johnson and C. Jones [37] first introduced the idea of secure dominating sets of graphs. A subset X of the vertex set of a graph G is a secure dominating set S of a graph G if for each vertex u not in S , there is a vertex v in S which is neighbour of u such that if we swap u and v from S is again a dominating set of G . A. P. Burger et al. [38] proposed minimum secure dominating sets of graphs. However, all those works were based on graphs and the dominating sets were not connected. The idea of connected minimum secure dominating sets in grids was first proposed by J. Barnett et al. in paper [39]. However, this secure connected dominated sets can be generated in such cases when G is a grid, and in the majority of cases when G is a cylindrical or toroidal grid. In case of ad-hoc wireless network the idea of using secure-CDS is not feasible. Additionally, although, this work ensures system fault tolerance by ensuring replacement of a failed node,

this may not work when a node and its replacement node fails (two node failures) at a time. Moreover, any graph of at least minimum degree 2 may possess a minimum secure dominating set which always might not be achieved in real scenarios.

J. Zhou et al. in paper [40] proposed first fault tolerant connected dominating set (CDS) construction for heterogeneous wireless sensor networks (WSNs). This is modeled as an approximation algorithm for $(3, m)$ -CDS in a heterogeneous WSN which is a special case of k -connected m -fold dominating sets. In paper [41], S. Farzana et al. derived multiple disjoint set covers for directional sensor networks. Here, each set cover is capable of monitoring all the targets using minimal number of sensors. The authors utilized set covers successively in a round robin fashion to increase the lifetime of the network as well as fault tolerance at the same time. Additionally, S. Saha et al. in paper [6], addressed the problem of fully disjoint set covers of the previous paper and proposed overlapping multiple set covers with minimum sensors for prolonging network lifetime and providing fault tolerance. However, both the methods are for wireless directional sensor networks which is a special case of wireless ad-hoc network. Moreover, as the structure of directional sensors are different than generic ad-hoc network and there are vast differences in the working principles between the directional sensors and ad-hoc networks, it requires different procedures for ad-hoc networks.

None of all the above algorithms considered fault tolerance and network longevity by constructing multiple connected dominating sets. In this paper, we are proposing a new approach to construct multiple connected dominating sets by round robin scheduling of the sets to communicate throughout the network. Additionally, using multiple CDSs one after another increases fault tolerance as well as the longevity of the network. In our work, we propose both centralized and distributed algorithm in this purpose.

Chapter 3

Preliminaries

In this chapter we discuss some basic terms, assumptions and notations used throughout the thesis. In Section 3.1, we present the network model we assume for the development and simulation of our algorithm. Section 3.2 presents some basic assumptions and notations that we assume for the scenario. Finally Section 3.3 describes the essential definitions used in our thesis.

3.1 Network Model

A wireless ad-hoc network can be represented as a graph like a wired network. We represent the connectivity between two nodes of a network if they are within transmission range. We represent the ad-hoc network with a graph $G(V, E)$ using the idea of unit disk graphs [42]. Here, V represents the set of nodes in the network and E represents the set of edges. An edge between two nodes represents connectivity between them. Moreover, it means they are within their transmission range. Figure 3.1 represents a wireless adhoc network with ten nodes with similar transmission range. Additionally, the circle around a node represents the transmission range of a node and all the nodes within this circle represents the neighbors of the node. On the other hand, Figure 3.2 is the graph representation of the network.

In this thesis, we consider two scenarios: one is that the global topology of the network is known. In this case, we apply centralized algorithm. On the other hand, we consider, no global topology is known. Here, we apply distributed algorithm. Moreover, we consider both static

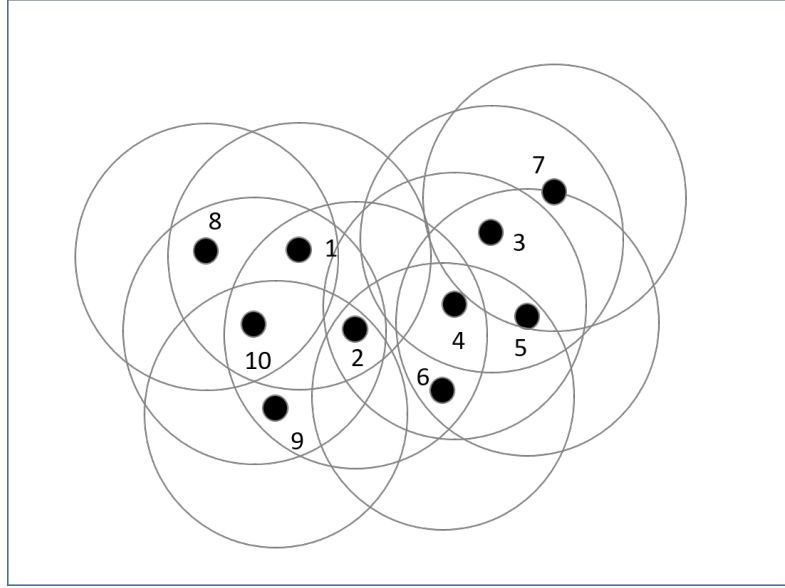


Figure 3.1: A randomly deployed Adhoc network with 10 nodes and their transmission range and dynamic situation for the distributed system. We apply our distributed algorithm for both cases but with some little changes.

3.2 Assumptions and Notations

For simulation, we made some assumptions on our network model. We assume a 2-dimensional ad-hoc network where nodes are randomly deployed into a 100×100 units area. We consider the network is not much dense. Therefore, we keep network size with 20 to 200 nodes. We assume battery power of each node same and fixed initially. The battery power of a node decreases linearly with time. Transmission range of each node is also assumed same and kept fixed throughout the whole simulation process. We assume no power loss of the nodes when the nodes of a CDS set are inactive.

We implement our algorithm both for centralized and distributed system. The centralized algorithm is based on basic centralized MCDS algorithm [3]. The notations used in our centralized algorithm is represented in Table 3.1. On the other hand, for distributed approach, each node creates its own forwarding list based on basic CDS. Therefore, based on the choice of CDS it creates its own forwarding list. Here, we apply the MCDS to create the forwarding list followed by the procedure of multiple dominant pruning [43]. The notations used in this algorithm along

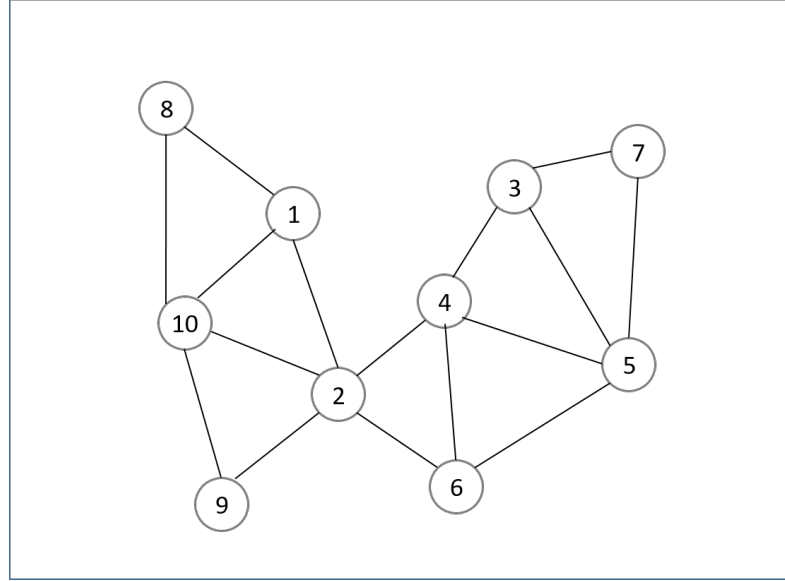


Figure 3.2: Graph representation of the network with 10 nodes

with the dominant pruning is presented in Table 3.2.

Notation	Description
V	Set of nodes
K	Overlapping boundary (A node can be present at maximum K CDSs)
BlackSet	Set of nodes that are selected for CDS
GraySet	Set of nodes that are one-hop neighbors of BlackSet nodes
WhiteSet	Set of uncovered or not explored nodes (Initially all nodes are in WhiteSet)
Degree(i)	Adjacent nodes of i th node
C_i	Cardinality of node i (If Cardinality of a node is k , it means a node is present in k different CDSs)
Min(C)	A function which returns the minimum cardinality value among the nodes with at least one adjacent white neighbor
W_i	Adjacent white neighbors count of i
$N(i)$	Adjacent Neighbors set of node i

Table 3.1: Table for the notations used in Centralized MMCDs algorithms

3.3 Definitions

In this section, we briefly describe some important terms used throughout our thesis. At first we describe some common terms that we consider for our both algorithms and finally some specific terms used for specific algorithms (centralized or distributed).

Notation	Description
u	Source node
v	Receiver node
$N(u)$	Set of one-hop neighbors of node u
$N(N(u))$	Set of two-hop neighbors of u
B_v	Set of one-hop neighbors that will be used for creating forwarding list of v
U_v	Set of one-hop neighbors that will be used for creating forwarding list of v
F_v	Forwarding list of v
F_{vp}	Previously created forwarding list of v
F_{vn}	New created forwarding list of v
K	Overlapping boundary of a node
C_i	Cardinality of node i in B_v

Table 3.2: Table for the notations used in Distributed MMCDs algorithms

3.3.1 Minimum Connected Dominating Sets (MCDSs)

A MCDS is a CDS which constructs using possible minimum number of Nodes. The MCDS of graph represented in Figure 1.2 is $\{A, C\}$ which uses only two nodes rather than other sets. However, constructing minimum connected dominating sets (MCDSs) is NP-complete problem. Therefore, we can use different heuristics to construct MCDS of a network. The most common approach is greedy method. In this approach, we select the node that covers maximum neighbors of the network. In the second iteration, it selects a neighbor of the previous node with maximum uncovered node. The process continues until there is no new node to cover.

3.3.2 Multiple MCDS (MMCDs)

Multiple MCDS refers to generating different possible MCDSs from a network. However, generating all possible CDSs are not necessary for a network. Moreover, there might exist a lot of overlapping among the CDSs. Therefore, in this thesis, we try to generate some CDSs with possible minimum nodes minimizing overlaps among the sets. As fully disjoint sets might not be achieved always, here we consider overlapping up to a certain amount. We denote this value as K which refers to the boundary value of a node's presence in multiple sets.

3.3.3 Overlapping Boundary (K)

For creating multiple MCDSs (MMCDs), we consider some overlapping among sets. However, we only consider overlapping among sets when there is no disjoint set possible to create.

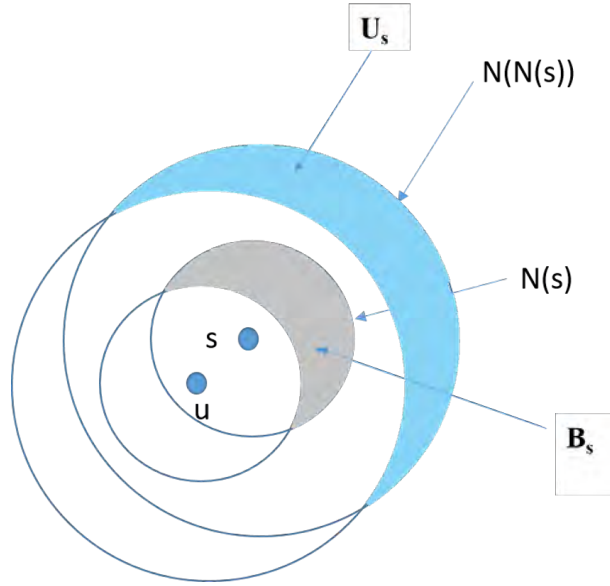


Figure 3.3: Some important notations for dominant pruning

Moreover, we try to create sets with minimum overlaps and allow up to a certain value. Hence, we introduce a user given upper bound K on overlapping among sets. The parameter denotes that no node can participate in CDS construction more than this given upper bound value. Therefore, we can say that no more than K sets from the MMCDs can contain a single node. For example, if the value of K is 1 then a node can only be present in only one CDS, thus generating only disjoint MMCDs. Additionally, if K is 2, then a node can be present up to two MCDSs among all MMCDs. Therefore, with the increase of K , the number of overlaps among sets will increase.

3.3.4 Dominant Pruning

For distributed system, we use the procedure of creating the forwarding list followed by dominant pruning [8]. Let us presume that, u sends a message to v . A forwarding list (F_u) is attached with the packet header. For a node v in forwarding list, it will create next forwarding list before rebroadcasting the packet. For constructing the new forwarding list, node u will need all the two-hop neighbours U_v that are not listed. Node v selects a neighbour $p \in B_v$ to cover the highest number of nodes in U_v , in other words, a node p is selected if the neighbor of p in U_v is maximum among all the nodes in B_v and added to F_v . The U_v is updated by subtracting the neighbors of p . The process terminates when there is no node in U_v . Figure 3.3 shows the notations for dominant pruning where u is the source and s is the receiver node who will create its forwarding list next.

Chapter 4

Multiple Minimum Connected Dominating Sets (MMCDSs) Algorithm

We represent the ad-hoc network with a graph $G(V, E)$. Here, V is the set of nodes in the network and E is the set of edges that indicate connectivity between two nodes. The method proposed in this thesis is multiple connected dominating set constructions for improved fault tolerance and increased network lifetime which will be presented in the following sections.

Firstly, Section 4.1 presents our centralized algorithm along with its time complexity analysis and set optimization procedure. Section 4.2 describes our distributed algorithms both for static and dynamic environment along with its time complexity analysis. Finally, a brief discussion about network lifetime and fault tolerance mechanisms for our both algorithms are presented in Section 4.3 and 4.4 respectively.

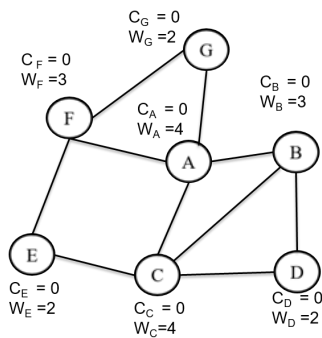
4.1 Centralized Algorithm for MMCDSs

The basic idea of the centralized algorithm is to generate multiple CDSs with minimal number of nodes avoiding node overlaps over sets as much as possible. The overlapping is controlled using a tunable parameter K which means one node can only participate in constructing (at most) K CDSs. In this approach, we run a centralized greedy algorithm to generate a new CDS with minimum nodes and minimum overlaps. This algorithm follows the basic MCDS construction algorithm [19] at each iteration. In MCDS, each node of the graph is given color

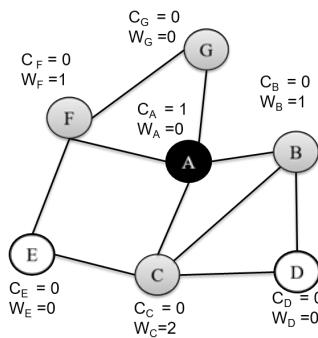
white at the beginning of this algorithm. Among all the white node, a node is selected which is connected to most of the white colored nodes and is coloured black. The adjacent nodes of that selected node is coloured gray. The next node of this process is selected from the gray ones. The selection is done on the basis of the highest quantity of white adjacent nodes. The selected one is added in the black node list. This selection procedure continues to run as long as any white node exists. The MCDS consists of all the black nodes. After this iteration, all the nodes again colored white and the cardinality of the black nodes in the previous iteration is increased by one. In the next iteration, the algorithm tries to select nodes from minimum cardinality (unused nodes of the previous iterations). If there is no new node, it can select a previously used node up to K times which is its maximum cardinality. The process continues until no new set can be generated. After the generation of each CDS set, another optimization algorithm is run to remove redundant nodes from the set. This step requires as while choosing a node we first consider candidates with minimum cardinality. Among the candidates we select the nodes with maximum white neighbors. Therefore, there might be a chance to select inefficient nodes first for coverage and then select another node with better coverage. Hence, to make overall efficient set we remove those redundant nodes without which nodes we can still reach all the nodes of the network. Figure 4.1 represents how our algorithm works. We keep the overlapping boundary value $K=2$. In Figure 4.1 (a) all the nodes are kept in white (In algorithm, all nodes are in *WhiteSEt*). The cardinality of each node is set to 0 and number of white neighbors (W) (initially one-hop neighbor count) are counted. 4.1 (b) illustrates how a node with minimum cardinality and maximum white count is selected for connected dominating set construction. The neighbors of node A are colored Gray (In algorithm the neighbors are added to *GraySet*) and the cardinality value of A is incremented by 1. The white count neighbor values (W_i) of each node i is updated as well. In the next iteration (Figure 4.1 (c)), node C is selected as it covers maximum white neighbors as well. After this step a minimum CDS is generated. For next set generation all the values are configured as initial values except the cardinality values of the nodes (4.1 (e)). Therefore, for constructing the next CDS set, we try to choose candidate from minimum cardinality nodes. Hence, we select node B from the candidates (B and F) with maximum white count neighbors. After that, there is no node in GraySet with cardinality 0. Therefore, we select candidates which has the minimum cardinality with at least one white neighbors. From the candidates set we can see that node A is selected again as there is no node

with 0 cardinality that covers the remained white neighbors. Finally, node F is selected to cover the remaining white neighbors and another CDS is constructed. (4.1 (f)). Finally, another another CDS set generated by selecting the nodes D, C, E and F (Figure 4.1 (h)). In this iteration, node D is selected first as it has minimum cardinality. After that nodes C, E and F are selected sequentially. If we see closely, we can see that there is a redundant node D, as if we discard the node, there will be no effect on the CDS and all the nodes of the network will be covered properly. This type of redundancy is eliminated by our *CDS_Optimization* function (similar to paper [6]). Therefore, another MCDS is constructed by the remaining nodes (Figure 4.1 (i)). This step is necessary, because it not only reduces the size of the CDS but also makes chance to use those redundant nodes to construct another new CDS as well. The process continues until no new set can be generated.

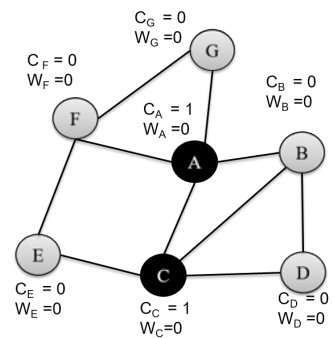
The centralized algorithm of MMCDSs is shown in Algorithm 1. Initially, we provide a graph representation of the network $G(V, E)$ and overlap limit K . The target is to generate multiple sets of MCDS. Initially, each node's cardinality is initialized to 0 (line 2 ~ 4). Then, we initialize a variable n which counts the number of MCDSs. The outer loop (line 6 ~ 44) generates the desired MCDS set and runs until no new set is found. It generates a set of MCDSs named $\{CDS_1, CDS_2, \dots, CDS_n\}$ where each value of n represents a new set of CDS. Within the while loop, we take three sets *BlackSet*, *WhiteSet* and *GraySet* (lines 7 ~ 9 of the algorithm) and all of them are initialized to empty sets except *WhiteSet*. All nodes of the graph are in the *WhiteSet* first and the inner loop (lines 10 ~ 37) runs until there is no remaining nodes in the *WhiteSet*. In the first iteration of the inner while loop, a node s from V with minimum cardinality and with maximum white neighbors(W_s) is selected (lines 14 ~ 15). This node is then added to *Blackset* and its cardinality is increased by 1. In *GraySet*, the neighbors of the selected node is added but those already in the *BlackSet* are discarded. Finally, *WhiteSet* is updated as well (lines 26 ~ 29). From the second iteration, a *CandidateSet* is generated from *GraySet* nodes with minimum cardinality. Among the nodes of *CandidateSet*, a node s with maximum white neighbor count us selected like before (lines 17 ~ 24). If it has at least one white neighbor, then the three sets are updated as before. This procedure runs until *WhiteSet* is empty or no new node can be selected with cardinality less than K . After a complete iteration, if *WhiteSet* is empty then a new set is generated. However, as we consider minimum cardinality for primary selection of candidates, there might be some irrelevant nodes in the generated



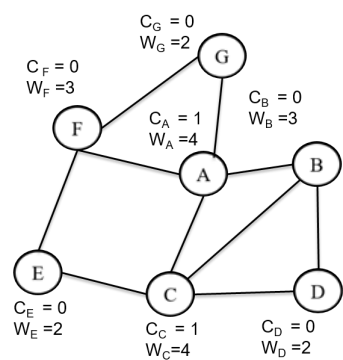
(a) All nodes are initially white



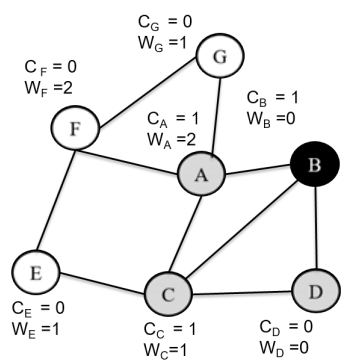
(b) Node A selected and colored black, all neighbors of A are colored grey



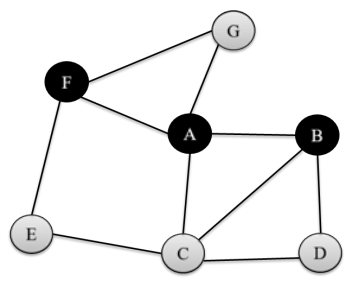
(c) Finally Node C selected and colored black, all neighbors are colored grey, hence A,C constructs a MCDS



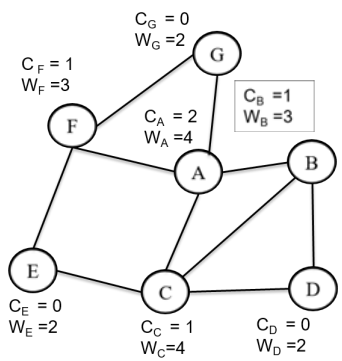
(d) All nodes are colored white again but with previous C_i



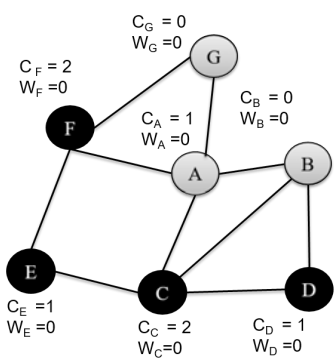
(e) Node B is chosen for next set generation, all neighbors of B are colored gray, cardinality of B is updated and values of white count neighbors of all nodes are updated



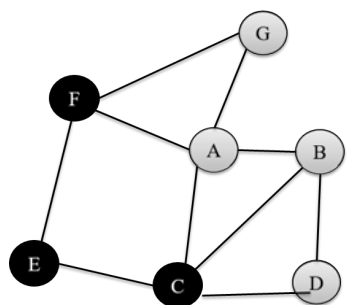
(f) Another MCDS is constructed by nodes F, A, B



(g) All nodes are colored white again but with previous C_i



(h) Another MCDS is constructed by nodes D, C, E and F



(i) After removing redundant node D, another MCDS with C,E,F nodes remains

Figure 4.1: Basic Greedy Centralized Construction of MCDS (a) ~ (c) and multiple MCDS construction (d) ~ (i)

CDS set. Therefore, a function named *CDS_Optimization* (similar to optimization function of paper [6]) is invoked to optimize the generated set and add it to CDS_n set (line 39). If there is any redundant nodes in *BlackSet* that is not in CDS_n , then its cardinality is decremented by 1, as the node is not really used in the set. The value of n is incremented after a successful creation of CDS.

Dominating Sets Optimization

For optimizing a newly generated CDS, C , we use *CDS_Optimization* function which is represented in Algorithm 2. In this function, we remove the redundant nodes within the CDS. In spite of adding optimal nodes as we consider minimum cardinality first, some nodes with highest white neighbors might be added later. Therefore, some redundant nodes might be present in the set. Hence, we use the function for optimization. Initially, it creates two empty set C_{opt} , *GraySet* and another set *WhiteSet* is initialized to V . The while loop runs until each white node is not covered. Initially, it selects a node from the constructed CDS C with maximum white neighbors. It is then added to C_{opt} and discarded from C . The *WhiteSet* and *GraySet* are updated as before. In the while loop, it selects node from $(C \cap GraySet)$ set as the selected node must be a neighbor of previously selected nodes and also must contain in C . The other procedures are same as before. After, completing the iteration, if there is any redundant node in C it will not be added to C_{opt} . Therefore, we can minimize number of nodes also from this function.

4.1.1 Complexity Analysis of Centralized MMCDS

In this algorithm we have two while loops. The outer while loop generates multiple sets which is $O(V)$ times in the worst case. The inner while loop generates a single CDS in each loop. This loop runs in $O(V)$ times as the size of the Whiteset is V initially. There are some loops within the inner while loop which can run at most $V-1$ times in worst case. Hence, $O(V^2)$ is the total run time of the while loop. There is an optimization function after the inner while loop and within the outer while loop which also costs $O(V^2)$ as the function has an outer while loop and a for loop within the while loop each of which costs $O(V)$ times. Therefore, the total time complexity of our Centralized MMCDS is $O(V^3)$ times.

Algorithm 1 Centralized Multiple Dominating Sets construction Algorithm

Require: $G(V, E), K$ **Ensure:** A Set of MCDSs CDS

```
1:  $CDS = \emptyset, GraySet = \emptyset, n \leftarrow 0$ 
2: for  $i \leftarrow 1$  to  $V$  do
3:    $C_i \leftarrow 0$ 
4:    $W_i \leftarrow Degree(i)$ 
5: end for
6: while no new set generated do
7:    $BlackSet = \emptyset$ 
8:    $GraySet = \emptyset$ 
9:    $WhiteSet = V$ 
10:  while  $WhiteSet \neq Null$  do
11:     $CandidateSet \leftarrow \emptyset$ 
12:     $Minc \leftarrow Min(C)$ 
13:    if  $Minc < K$  then
14:      if  $\|WhiteSet\| == \|V\|$  then
15:        Select a node  $s$  from  $V$  with maximum  $W_s$  and  $C_s = Minc$ 
16:      else
17:        for  $k \in Grayset$  do
18:          if  $C_k == Minc$  then
19:             $CandidateSet \leftarrow CandidateSet \cup \{k\}$ 
20:          end if
21:        end for
22:        if  $CandidateSet \neq \emptyset$  then
23:          Find the node  $s \in CandidateSet$  with maximum  $W_s$ ;
24:        end if
25:        if  $W_s > 0$  then
26:           $BlackSet = BlackSet \cup s$ 
27:           $C_s = C_s + 1$ 
28:           $Grayset = Grayset \cup (N(s) - BlackSet)$ 
29:           $WhiteSet = WhiteSet - N(s) - \{s\}$ 
30:        else
31:          break;
32:        end if
33:      end if
34:    else
35:      break;
36:    end if
37:  end while
38:  if  $WhiteSet == Null$  then
39:     $n = n + 1$ 
40:     $CDS_n = CDS\_Optimization(BlackSet, G)$ 
41:    for  $k \in (BlackSet - CDS_n)$  do
42:       $C_k = C_k - 1$ 
43:    end for
44:  else
45:    break;
46:  end if
47: end while
```

Algorithm 2 Dominating Sets Optimization Algorithm

Require: $G(V, E), C$

Ensure: An optimized set C_{opt}

```
1: function CDS_Optimization(C, G)
2:    $C_{opt} = \emptyset$ 
3:    $WhiteSet = V$ 
4:   Find the node  $s \in C$  with maximum  $W_s$ 
5:    $C_{opt} = C_{opt} \cup \{s\}$ 
6:    $GraySet = GraySet \cup N(s) - \{s\}$ 
7:    $Whiteset = Whiteset - N(s) - \{s\}$ 
8:    $C = C - \{s\}$ 
9:   while  $WhiteSet \neq Null$  do
10:     Find the node  $s \in (C \cap GraySet)$  with maximum  $W_s$ 
11:      $C_{opt} = C_{opt} \cup \{s\}$ 
12:      $C = C - \{s\}$ 
13:      $GraySet = GraySet \cup N(s) - \{s\}$ 
14:      $Whiteset = Whiteset - N(s) - \{s\}$ 
15:   end while
16: return  $C_{opt}$ 
```

4.2 Distributed Algorithm for MMCDSs

In distributed algorithm, for creating multiple MCDSs, we use the dominant pruning based MCDS construction multiple times. Each node tries to contribute to multiple CDS with minimum overlapping over sets assuming that, every node is provided with its 2-hop neighbour information. As mentioned before, we consider both static and dynamic scenarios for the system. Here, we describe both types of algorithms.

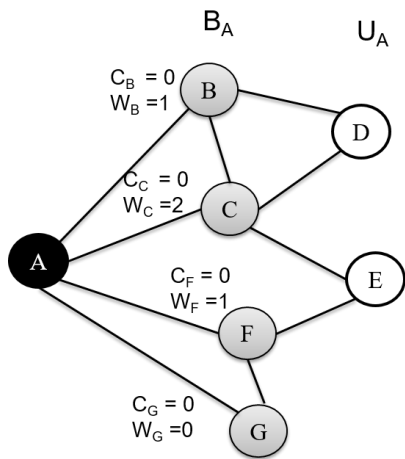
Firstly, we consider a static ad-hoc network. Therefore, we need to generate only multiple MCDSs and use all these sets according to the system criteria. In this algorithm, each node has an additional count of cardinality. If a node is selected for a CDS, its cardinality is increased by one. A node can be used up to K as like the centralized algorithm. When a node receives a packet, it creates its own forwarding list. To create the forward list, it discovers all of its undiscovered two-hop neighbours by using its one-hop neighbours. For this algorithm, a node from its one-hop neighbor list is selected for forwarding that has minimum cardinality and maximum number of nodes coverage of its uncovered two-hop neighbors. The process is repeated until all uncovered two-hop neighbors are covered. The outer process is repeated until there is no new set generated. Figure 4.2 represents how our distributed algorithm for static system works. Suppose, A receives a packet forwarding from a source node. Now, A needs

to generate its own forwarding list. Figure (4.2 (a)) illustrates its one-hop (B_A) neighbors for creating forwarding list and two-hop uncovered neighbors (U_A) which should be covered by the created list. From B_A , node C is selected first as it covers maximum white nodes (all nodes in this case) from U_A . Therefore, only C constructs the first forwarding list (4.2 (b)). As, we aim to create multiple forwarding lists, after first iteration, the values of cardinality (C) of each node is updated. For next iteration, configuration are initialized as initial values except the cardinality values (4.2 (c)). Again, another new forwarding list is created using the same procedure and each time of node selection it checks if the node has less cardinality than K . Finally, node B and F from B_A constructs the second CDS list (4.2(d) ~ (e)). The procedure stops as no new set can be generated further. The whole process is run each time on each node when it receives packets for forwarding. Table 4.1 shows forwarding list creation for other nodes of the scenario.

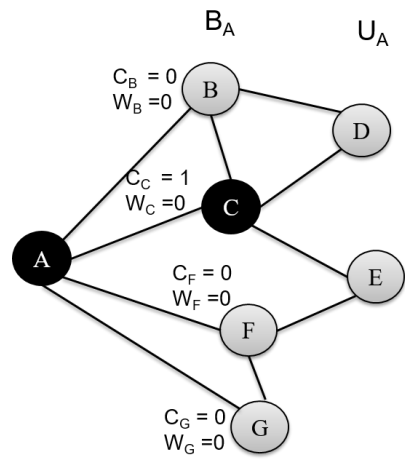
Previous Node	Current Node	B_v	U_v	F_v
—	A	{B,C,F,G}	{D,E}	{C}, {B, F}
A	B	{C,D}	{E}	{C}
A	F	{G,E}	×	×
A	C	{D, E}	×	×

Table 4.1: Table for the forwarding List of all nodes

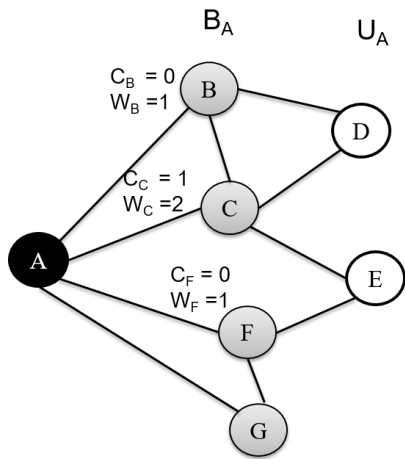
For dynamic scenario, as the environment changes, initially generating all multiple nodes might not give any benefit. In that case, we generate only a set for first time and store the set with cardinality of the used nodes. Suppose the overlapping boundary is K . Therefore, we store last $K - 1$ sets. For the next time, if the forwarding list comes from same node it creates its new forwarding list with minimal overlap with before $K - 1$ sets allowing only overlapping criteria up to K . This is how, for dynamic scenario, we try to load balance among the network. We can consider the same scenario as Figure 4.2 and consider that $K=2$. Here, we only construct the first forwarding list {C} and store it. Suppose that, a node G has changed its position and got out of the transmission range of A. Figure 4.3 represents the new alignment of the network. However, C node is still in the forwarding list of A. Its cardinality has been initialized to 1 as it is in the previous list. When the node receives packets from same source, it tries to select nodes that are not in the previous list that means it tries to select from minimum cardinality values. From the new figure 4.3 (a), we can see both C and F covers maximum nodes from U_A . It selects F instead of C as it has minimum cardinality values than C. Finally, node B is



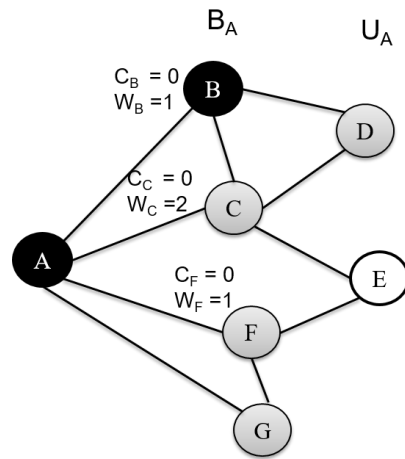
(a) Connection of node A with B_A and U_A



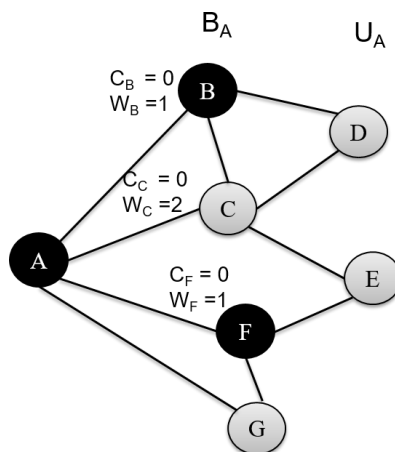
(b) Node C is chosen for forwarding with highest W



(c) After first iteration all nodes are in initial condition except C



(d) Node B is chosen for second forwarding list



(e) Finally, node F is chosen for second forwarding

Figure 4.2: Basic Greedy Distributed Forwarding List Construction of MCDS (a) ~ (b) and multiple MCDS (c) ~ (e) construction for static scenario

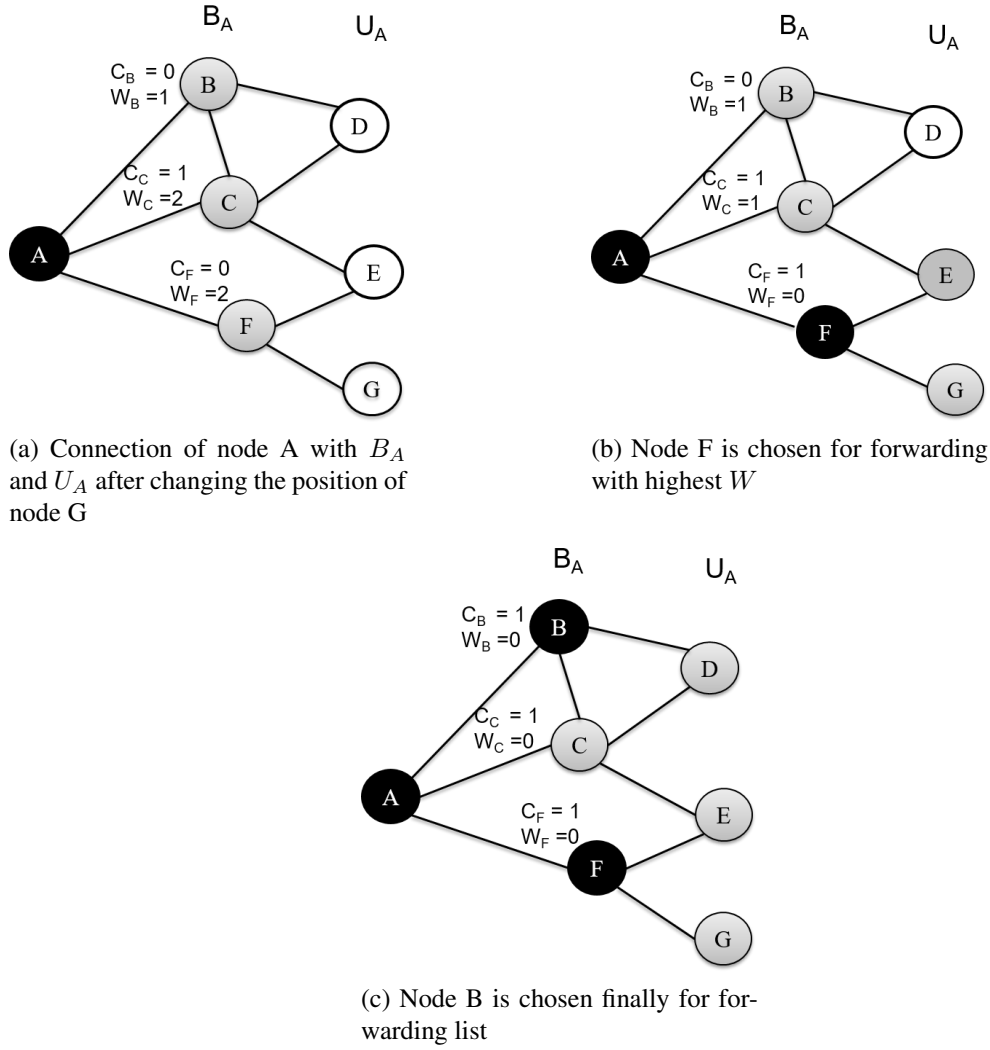


Figure 4.3: New one-hop and two-hop uncovered sets of node A (a), new forwarding list generation (b) ~ (c) for dynamic scenario

selected for forwarding list (4.3 (b) ~ (c)) . One important issue is that, for dynamic scenario, the number of stored forward lists depend on the boundary value. If we use $K=2$, we have only one set in the previous list. Hence, for the above scenario we will discard the list $\{C\}$ and add $\{ B, F \}$ to the previous list.

As before stated that the algorithm is designed using 2-hop neighbours information of each node. If node v receives a packet from node u and if it is already in the forwarding list of u , it creates its own forwarding list. To create the forward list (F_v), it discovers all of its undiscovered two-hop neighbours (U_v) by using its one-hop neighbours (B_v). We calculate U_v and B_v from the following formulas [8]:

$$U_v = N(N(v)) - N(v) - N(u)$$

$$B_v = N(v) - N(u)$$

Hence, we create a set of n forward lists (F_{vn}) for static network. If node v receives packets from same source then it selects a forward list from the sets of lists in round robin fashion. Algorithm 3 represents the multiple forwarding lists creation algorithm for static scenario. The selection procedure of a node for forwarding list is same as our centralized algorithm. Here, initialize the cardinality value of all nodes in B_v to 0 (line 3 ~ 4). A new forwarding set F_{v1} is taken initially to null and a new set U is initialized to U_v . The outer while loop generates multiple forwarding lists (line 7 ~ 39). In each iteration it selects candidate sets from B_v with minimum cardinality (line 9 ~ 14). Then it selects the most efficient node that covers maximum uncovered nodes from U (line 15-20). If we can select a node, then it is added to the current forwarding list F_{vn} and all neighbors of the selected node are discarded from U (line 21 ~ 32). The cardinality of the selected node is incremented by 1 and if it's cardinality reaches to K then it is discarded from B_v set (line 26 ~ 29). This procedure runs until there is no node remaining in the list of U . Whenever, U becomes null, it is again initialized to U_v to create next forwarding list along with necessary changes. In this process, we generate as much lists as possible and use them according to the criteria of the network.

However, if the scenario is dynamic, we generate only one suitable forwarding list and store the nodes cardinality of used sensors. For, next iteration, even if the node v receives packets from same source, it needs to generate its own forwarding list again. In that case, while generating new set it considers minimum overlapping with the previous used forwarding list. Suppose that F_{vK-1} is the previously created $K - 1$ forward lists. As we consider up to K overlapping among sets, hence we store last $K - 1$ sets to check overlapping. Now, if we want to generate a new updated forwarding list F_{vu} , it tries to select a node from B_v which does not contain in the previous set of lists F_{vK-1} . If there is no node to select from B_v that does not contain in F_{vK-1} , then it selects node with minimum cardinality nodes from the previous lists. The algorithm for new set generation is presented in algorithm 4. Initially, it counts the cardinality values of nodes containing in B_v that were in the previous forwarding lists F_{vK-1} whereas other nodes get cardinality value 0 in the B_v (line 3 ~ 11). The other procedure is same as the algorithm 3. However, in this case we generate only one set avoiding the nodes of higher cardinality (line 12 ~ 34). Finally, we delete the earliest (that is created first) forward list from the F_{vK-1} list and add the new list to it. Therefore, the overall stored forward lists remain same as $K - 1$ always and hence we reduce the redundancy of multiple calculation of forward lists

for dynamic situation.

4.2.1 Complexity Analysis of Distributed MMCDS

In a graph representation of a network, each node or vertex is connected with another multiple nodes or vertices. The number of vertices that is associated with a vertex is called the degree of the vertex. Suppose that Δ is the maximum degree of the graph. In distributed algorithm, the forwarding list is created from the one-hop neighbor set B_v that contains all the adjacent nodes of v . The size of B_v can be at most Δ . The two-hop neighbor set U_v contains all the adjacent nodes of B_v which can be at most Δ^2 . Therefore, the run time complexity of creating a single forwarding list of a node is Δ^3 . In our distributed algorithm, for static network we generate multiple forwarding lists using the outer while loop which could be at most Δ . Hence, total time complexity for constructing multiple forwarding lists is Δ^4 . On the other hand, instead of generating all lists we generate a single set just minimizing the overlaps with previous K lists. Hence, the time complexity does not increase and remains same as constructing a single forwarding list which is $K\Delta^3$.

4.3 Network lifetime

The network lifetime of a system means how long the network remains operative. The longer lifetime is required for any system. Generally, the network lifetime for an ad hoc network indicates battery power of the nodes. As we consider here all the nodes have similar battery power, after a certain time (full power consumption) all nodes will go out of power and the system will die. However, if we load balance among the nodes and only consider power consumption while these nodes are active, we can increase the lifetime. As we propose multiple minimum connected dominating set covers, if we schedule them in a round robin fashion for communication and control power consumption of the whole system, the lifetime can be increased tremendously. Generally, we consider network lifetime without considering any fault or errors. Therefore, for a longer lifetime creating multiple sets would be sufficient for the network.

Algorithm 5 shows the algorithm for calculation of total network lifetime for our algorithm. We can apply this algorithm both for centralized and distributed algorithm. The key difference is

Algorithm 3 Forward Lists Creation of a node v for Distributed Algorithm

Require: B_v, U_v **Ensure:** A Set of forwarding lists

```
1:  $F_{v1} = \emptyset, U = U_v$ 
2:  $Size\_F_{v1} = 0$ 
3: for all node  $i \in B_v$  do
4:    $C_i \leftarrow 0$ 
5: end for
6:  $n = 1$ 
7: while no new set generated or  $B_v$  remains unchanged do
8:    $minimum \leftarrow ||V||, Candidate\_Set = \emptyset, max = 0$ 
9:    $Minc \leftarrow Min(C)$ 
10:  for all node  $r \in B_v$  do
11:    if  $C_r == Minc$  then
12:       $Candidate\_Set = Candidate\_Set \cup r$ 
13:    end if
14:  end for
15:  for all node  $s \in Candidate\_Set$  do
16:    if  $(N(s) \cap U) > max$  then
17:       $max = N(s) \cap U$ 
18:       $selected = s$ 
19:    end if
20:  end for
21:  if  $max > 0$  then
22:     $F_{vn}[size ++] = selected$ 
23:    for all node  $y \in N(selected)$  do
24:       $U = U - y$ 
25:    end for
26:     $C_{selected} ++$ 
27:    if  $C_{selected} == K$  then
28:       $B_v = B_v - selected$ 
29:    end if
30:  else
31:    break
32:  end if
33:  if  $U$  is  $NULL$  then
34:     $U = U_v$ 
35:     $n ++$ 
36:     $F_{vn} = \emptyset$ 
37:     $Size\_F_{vn} = 0$ 
38:  end if
39: end while
```

Algorithm 4 New forward List Creation for dynamic network of a node v for Distributed Algorithm

Require: B_v, U_v, F_{vK-1}, K

Ensure: A new forwarding List, F_{vu}

```

1:  $F_v = 0, U = U_v$ 
2:  $F_{vu} = \emptyset, Z = \emptyset, size = 0$ 
3: for each node  $i \in B_v$  do
4:   for each value  $j \in K - 1$  do
5:     for each node  $s \in F_j$  do
6:       if  $i == s$  then
7:          $C_i ++$ 
8:       else
9:          $C_i \leftarrow 0$ 
10:      end if
11:    end for
12:  end for
13: end for
14: while  $U \neq NULL$  or  $B_v$  remains unchanged do
15:    $minimum \leftarrow ||V||, Candidate\_Set = \emptyset, max = 0$ 
16:    $Minc \leftarrow Min(C)$ 
17:   if  $Minc < K$  then
18:     for each node  $r \in B_v$  do
19:       if  $C_r == Minc$  then
20:          $Candidate\_Set = Candidate\_Set \cup r$ 
21:       end if
22:     end for
23:     for each node  $s \in Candidate\_Set$  do
24:       if  $(N(s) \cap U) > max$  then
25:          $max = N(s) \cap U$ 
26:          $selected = s$ 
27:       end if
28:     end for
29:     if  $max > 0$  then
30:        $F_{vu}[size ++] = selected$ 
31:       for each node  $y \in N(selected)$  do
32:          $U = U - y$ 
33:          $C_{selected} ++$ 
34:          $B_v = B_v - selected$ 
35:       end for
36:     else
37:       break
38:     end if
39:   end if
40: end while
41: if  $U == NULL$  then
42:   Delete the earliest forward list from  $F_{vK-1}$ 
43:   Add the new forward list  $F_{vu}$  to  $F_{vK-1}$ 
44: end if

```

that for centralized algorithm the whole network lifetime can be calculated centrally or globally. On the other hand, for distributed system, each node will calculate its lifetime on the basis of its number of forwarding lists. The algorithm is very simple. It basically calculates the network lifetime or activation time (t) of each CDS (line 4 ~ 13). To calculate each individual sets activation time, we find out the node with maximum cardinality value within the set (line 6 ~ 10). Then, we divide the activation time of a single node T by this maximum cardinality value and find out t . This is because a set can survive up to t times as the set will die as soon as its one of the nodes with highest cardinality will die. A node with highest cardinality will contribute to more sets rather than other nodes of the set and it will die out first due to lack of power than the others one. Finally, the value t of each set is added with the total network life (NL) (line 14) which gives total network lifetime after the final execution of the outer *for* loop.

Algorithm 5 Network Lifetime Calculation Algorithm

Require: CDS (set of $CDSs$), C (Cardinality set), T (activation time of a single node)

Ensure: Total Network Lifetime, NL

```

1: function Lifetime_Calculation (CDS, C, T)
2:    $NL = 0$ 
3:   for each set  $S \in CDS$  do
4:      $t \leftarrow 0$ 
5:      $max \leftarrow 0$ 
6:     for each node  $i \in S$  do
7:       if  $C_i > max$  then
8:          $max \leftarrow C_i$ 
9:       end if
10:    end for
11:    if  $max > 0$  then
12:       $t \leftarrow \frac{T}{max}$ 
13:    end if
14:     $NL \leftarrow NL + t$ 
15:  end for
16:  return  $NL$ 

```

4.4 Fault tolerance

A system or a network is called fault tolerant if any node of the system fails but the system or network still remains operative. Our system is fault tolerant also. When any node of the CDS fails to operate another CDS is run alternatively for communication. We have two topology: centralized and distributed. In order to explain how our algorithm works, we show examples

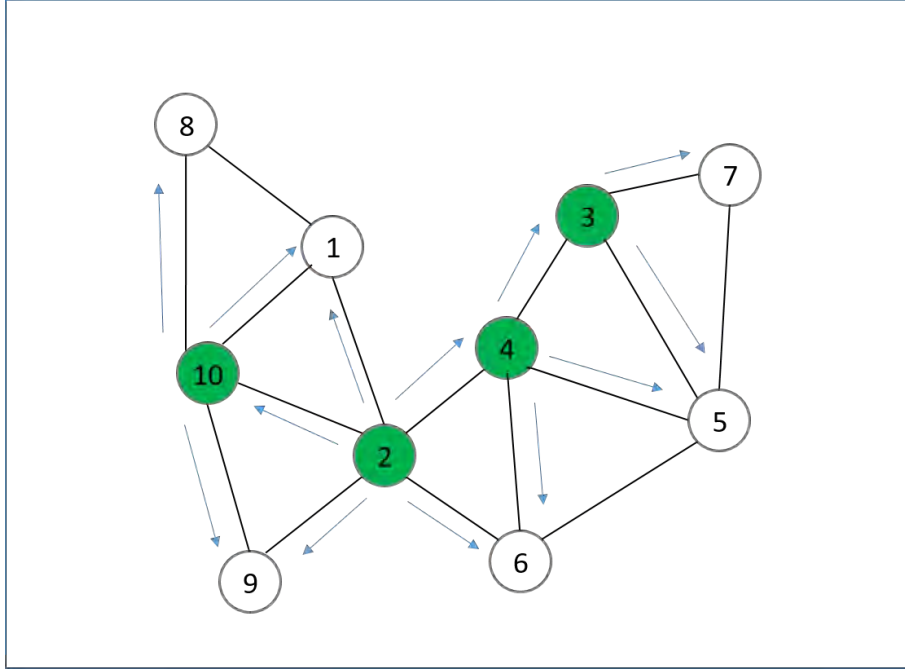


Figure 4.4: A CDS (Green nodes) for communication over the network

for both topology. Firstly, we show the fault tolerant mechanism of our centralized algorithm. For better understanding we consider the graph of figure 3.2. The possible MCDSs for $K=2$ are : $\{ 2, 3, 4, 10 \}$ and $\{ 1, 2, 5, 6 \}$. Suppose, the messaging starts from node 2 and we choose the route $\{ 2, 3, 4, 10 \}$ for communication. Message forwarding will be as like figure 4.4. However, the node 3 becomes faulty and could not continue message passing. Therefore, an error message is sent to its corresponding source node and finally when node 2 knows about the fault, it selects another route for routing which does not contain the error node 3 without any extra overhead of path calculation. Therefore, the network recover from its failure ultimately. Though it requires more time than [32], it requires no extra overhead.

For distributed algorithm, the case is more simple. When a node, forwards message, it selects one of its forwarding list. If any node of its forwarding list becomes faulty, it immediately switches to its another forwarding list. Figure 4.7 ~ 4.9 show how it changes forwarding node list when a node of previous forwarded list fails.

For calculating system fault tolerance we provide an algorithm 6. It is easy to calculate fault tolerance when a system generates n disjoint sets. The value of fault tolerance of that system is $n-1$ as it can tolerate up to $n-1$ node failures [44]. When we choose a set from n disjoint MCDSs, if any node fails, it instantly provide another set for active service. Hence, up to $n-1$ node failures, it can provide a backup CDS. However, when we consider overlapping boundary

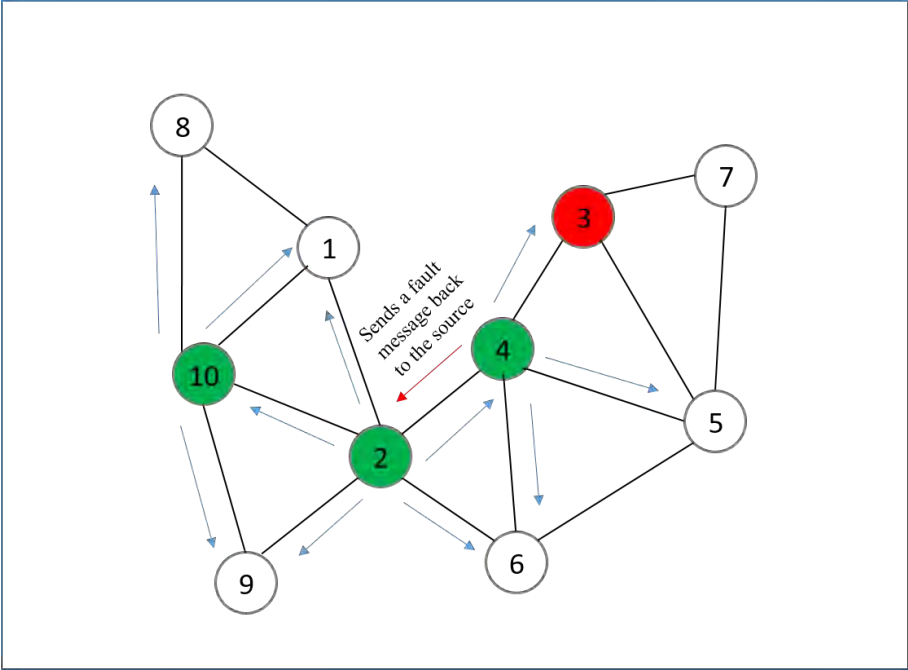


Figure 4.5: A node (Red node) of the CDS becomes faulty

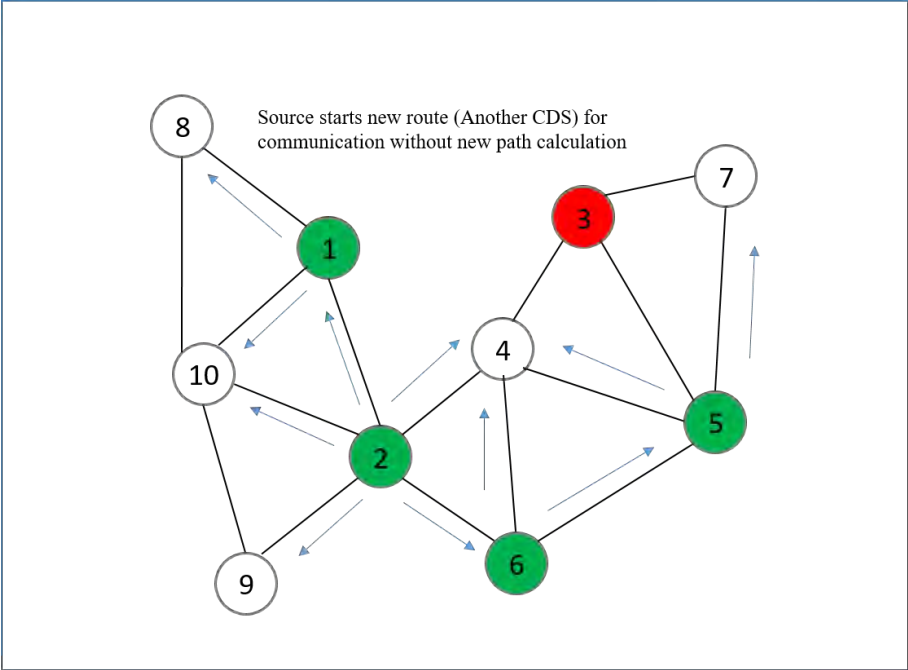


Figure 4.6: Another backup CDS (Green nodes) run for communication over the network

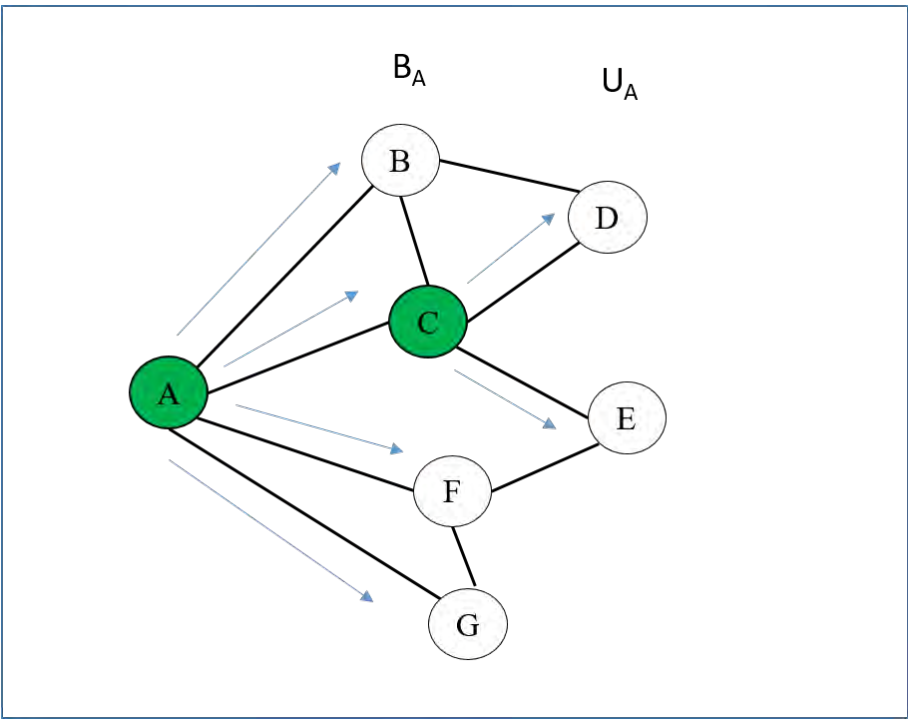


Figure 4.7: A forwarding node list (CDS) for communication over the network

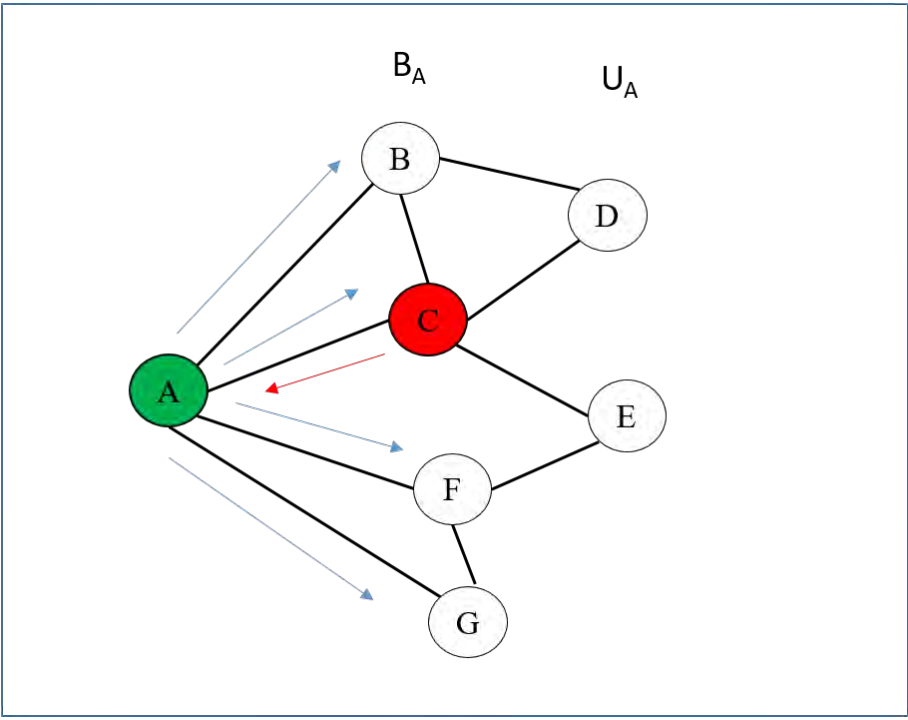


Figure 4.8: A node (Red node) of the CDS becomes faulty

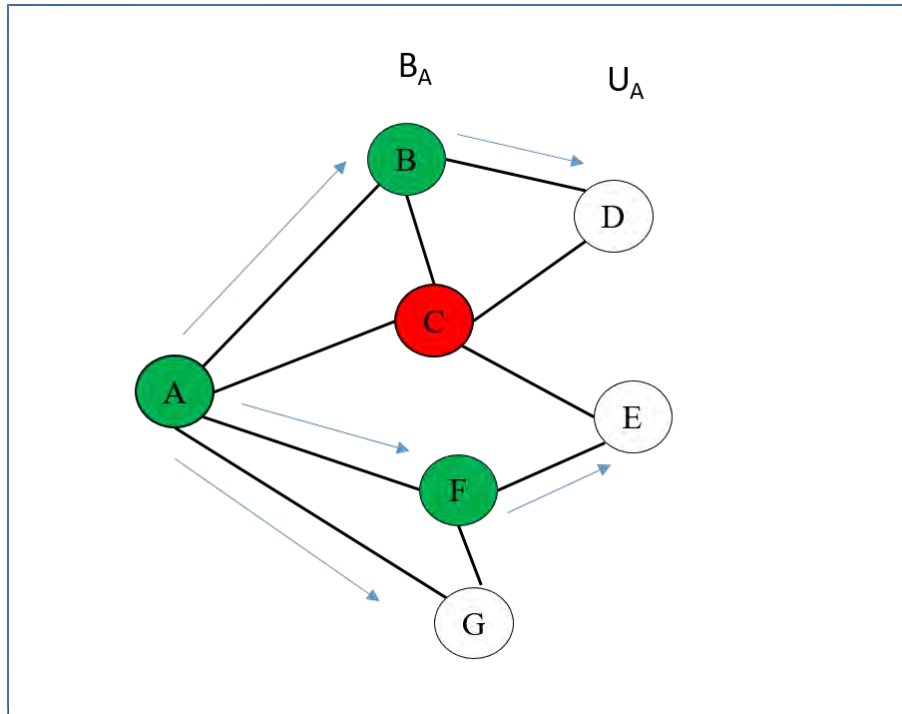


Figure 4.9: Another backup forwarding node list (CDS) run for communication over the network

up to K , the fault tolerance might not be so straightforward.

To calculate fault tolerance for overlapping system we use the formula that, we count total number of MMCDs and consider the node with maximum cardinality fails first and subtract those sets which contain the faulty nodes. We do the same process until there is no set remaining in the MMCDs. From the procedure, we count how much node failures it can tolerate (it has at least one remaining sets for backup). It should be noted that, this is the minimum fault tolerance value of the network as we always consider that the node with highest cardinality value fails. However, this could be improved if we consider all possible node failures and so on.

Algorithm 6 presents the algorithm for calculating fault tolerance of our algorithm. Initially, we introduce fault tolerance F by 0. In each iteration we find out the node with maximum cardinality (line 5 ~ 10). Hence, we subtract those sets which contains the selected node and update the cardinality of each node from those subtracted sets by decreasing 1 (line 13 ~ 20). Finally, if there still remains any set we update the fault tolerance value by increasing 1. The loop continues until there remains any set in the CDS sets, otherwise breaks (line 21 ~ 25).

Algorithm 6 Fault Tolerance Calculation Algorithm

Require: CDS (set of $CDSs$), n (total number of $CDSs$), C (Cardinality set), V (set of nodes)

Ensure: Fault Tolerance, F_T

```
1: function FaultTolerance_Calculation (CDS, n, C, V)
2:  $F = 0$ 
3:  $max = 0, index = -1, Total = n$ 
4: while true do
5:   for each node  $i \in V$  do
6:     if  $C_i > max$  then
7:        $max \leftarrow C_i$ 
8:        $index \leftarrow i$ 
9:     end if
10:  end for
11:  if  $index > -1$  then
12:     $Total \leftarrow Total - max$ 
13:    for each set  $S \in CDS$  do
14:      if  $index \in S$  then
15:        for each node  $j \in S$  do
16:           $C_j --$ 
17:        end for
18:        Delete set  $S$  from  $CDS$ 
19:      end if
20:    end for
21:    if  $Total > 0$  then
22:       $F ++$ 
23:    else
24:      break
25:    end if
26:  end if
27: end while
28: return  $F$ 
```

Chapter 5

Experiments

Finally, in this chapter, simulation results of the algorithms and the improvement of system fault tolerance will be shown in performance metrics.

We describe the experimental setup and results in different sections of this chapter. Section 5.1 discusses about the experimental setup that has been used to carry out the experiments. Section 5.2 describes the performance metrics to evaluate the performance of our proposed algorithms. Finally, the simulation results are presented in Section 5.3 of our fault tolerant MMCDs algorithms along with other algorithms.

5.1 Experimental Setup

We simulate our fault tolerant MMCDs algorithms along with basic MCDS and other algorithms. The parameters considered here are summarized in Table 5.1. To simulate our algorithms along with other algorithms, we consider a scenario of a network over an area of 100×100 units followed by N number of wireless nodes. Initial battery power of each node is kept fixed at 100 time unit. Transmission range of a node is kept fixed at 25 units. Overlapping boundary K is set 1 to 20 for different simulations. Additionally, to observe the impact of K on different performance we vary K 1 to 20 values. In our simulations, we vary N from 20 to 200 with an increment of 20 to simulate the network performance. We simulate all the algorithms in Java programming language in Netbeans IDE.

Parameters	values
Network area	100×100 units
Number of Nodes, (N)	20 - 200
Battery power, (T)	100 units
Transmission range (fixed)	25 units
Overlapping boundary (K)	1 - 20
Node distribution	Random
Number of simulations	100

Table 5.1: Table for simulation setup parameters

5.2 Performance Measurements

In this section, we present the performance measurements of our algorithms. For centralized algorithm, we consider number of MMCDSs evaluation, size of MMCDSs with and without optimization function, average packet passing, network life time and fault tolerance of the network. On the other hand, for distributed algorithm, average packet passing are considered. Finally, we compare average required time to calculate forwarding nodes for both algorithms.

5.2.1 Multiple MCDSs evaluation

An individual CDS is capable of communicating whole network or keep the network active. However, generating Multiple MCDS indicates that the network gets more options to choose for communication. The network might be faulty or error prone. If there is only one CDS and any node of the CDS fails, the whole system will fail to communicate properly. Therefore, for longer activity of the network constructing multiple CDSs might be a prominent way. If we run multiple CDSs in round robin way for a centralized network, lifetime of the network will increase with respect to a single CDS. Moreover, we can also provide a CDS as backup of another CDS, if it fails to work. Therefore, fault tolerance of the network will also increase. However, generating multiple disjoint CDSs always might not be feasible. As a result, we allow some overlapping among the CDSs while generating them. We introduce a overlapping boundary K which is a tenable parameter of our algorithm. If $K = 1$, only disjoint CDSs are generated. When $K = 2$, we allow any node to be present at maximum two CDSs. Therefore, with the increase of K , the number of MCDSs increase also. However, it reaches to a saturated value or does not increase any set numbers after a certain value of K . For simulation, we use $K = 1$ to 20 with a increment of 1 whereas we vary N from 20 to 200 with an increment of 20.

In this paper, we find out a suitable value of K for upper boundary of overlaps with the value of N . We evaluate 100 random scenarios to calculate the MMCDSs and take average values of the results.

5.2.2 Network lifetime

The network lifetime of a system means how long the network remains active. The long network lifetime indicates the system is long operative. Generating multiple CDSs, would increase network longevity. Consequently the lifetime of a network depends on the lifetime of the individual CDS set. We consider that all nodes have initially same battery power. Additionally, we assume that available power runs a node T time unit. If only a CDS is generated and all the nodes are activated for whole time to communicate over the network, then the lifetime of the network will become $NL = T$. Therefore, maximizing the number of CDSs will also maximizes the network lifetime as each CDS is scheduled in different time periods. If there are n CDSs in the network and we keep active each CDS for T time unit for communication, then the network lifetime of the system can be defined as, $NL = nT$ [6]. However, if there exists some overlapping CDSs up to K boundary overlap, then network lifetime becomes as follows [6]:

$$NL = \sum_{i=1}^n t_i \quad (5.1)$$

Here, t_i is the activation time for i th CDS which can be defined as :

$$t_i = \frac{T}{\max(C_{ij} : j = 1, 2, \dots, \|CDS_i\|)} \quad (5.2)$$

Here, C_{ij} is the cardinality of node j in i -th CDS.

5.2.3 Average Forwarding Nodes

The number of forwarding nodes can be defined as the total number of nodes (forward nodes) who forward or rebroadcast the broadcast packet by adding 1 (for source node). The equation of number of nodes forwarding can be defined as [5]:

NFN = Number of nodes forwarding + 1 (source node)

As, for our algorithm, we run multiple CDSs, here, we consider average packet forwarding up to a certain time, for example T . Suppose, our algorithm runs n CDSs in round robin fashion for total T times. Hence, The equation can be defined as :

$$\text{Total Forwarding Nodes(TFN)} = \sum_{i=1}^n NFN_i$$

$$\text{Average Forwarding Nodes (AFN)} = \frac{TFN}{n}$$

5.2.4 Network Fault Tolerance

Fault tolerance of a network can be calculated as up to how many node failures it can tolerate or handle. The more number of node failures it can tolerate, the more value of the network fault tolerance gains. For example, if any node of the system fails but the network still remains operative , then it has fault tolerance of 1. Moreover, if any two nodes of the system fails and the system still works, it has fault tolerance value of 2. Therefore, it is a very important factor of any network system. For our system, fault tolerance is very high than a single MCDS network or secure-CDS. If our system generates n disjoint sets, then it can tolerate up to $n-1$ node failures. Therefore, the fault tolerance value is $n-1$. However, as we consider some overlapping up to K , the fault tolerance value decreases. Although, the value of fault tolerance for overlapping system depend on total number of sets creation and the overlapping boundary K , with the increase of K , number of sets also increases. Hence, fault tolerance of the system increases. To calculate fault tolerance for overlapping system we use the formula that, we count total number of MMCDSs and consider the most used sensors fails first and subtract those sets which contain the faulty nodes. We do the same process until there is no set remaining in the CDS set. From the procedure, we count up to how much node failures it can survive (it has at least one remaining sets). It should be noted that, this is the minimum fault tolerance value of the network as we always consider that the node with highest cardinality value fails. However, this could be improved if we consider all possible node failures and so on.

5.3 Experiment Results

In this section, we present the results of our algorithms based on the performance measurement parameters along with other algorithms.

5.3.1 Performance based on Number of MMCDSs

We evaluate total number of CDSs varying network size with $N=20$ to $N=200$ with an increment of 20. Figure 5.1 illustrates the results of CDSs construction. Here, we apply optimization step for set optimizations and find that total numbers of MMCDSs with minimum nodes by applying optimization. The result shows that with the increase of density of nodes the total number of MCDSs increase. Moreover, with the increment of overlapping boundary it also increases, as each node contributes to more new sets. For our MMCDSs construction, we use a step to minimize redundant nodes, hence the size of the CDSs also decrease. Figure 5.2 illustrates how the sizes of MCDSs construction decreases for different network size with optimization than without optimization respectively. Figure shows that if we apply optimization step number of nodes used to construct MMCDSs decrease for each scenario. This happens because when we apply optimization some unnecessary nodes are removed from a set which basically reduces the sizes of MCDSs. Furthermore, when optimization are used the number of MCDSs also might increase. As those removed redundant nodes might be used for further set constructions. Figure 5.3 shows how total number of MCDSs change with the increase of overlapping boundary K . It is clear that with the increase of overlapping boundary number of MMCDSs will increase. However, it will move to a saturated point when there is no new node to generate a new MCDS. For example, when we consider network size $N=20$, for $k=4$ the set construction is almost in saturated condition, whereas, for $N=30$, the point moves to $K=18$. Hence, we can conclude that number of MMCDSs will increase up to a certain point. The point also changes with the increase of network size.

5.3.2 Performance based on Network Lifetime

Here, we present how network lifetime increases if we apply our algorithms in a ad hoc network. We calculate life time of a network varying network size (number of nodes n). As we already have seen that with the increase of network size number of MCDSs increase also. Therefore, the network lifetime increase also. Figure 5.4 illustrates the network lifetime varying network size 20 to 200 with an increment of 5 for overlapping boundary $K=1$ to 5 with an increment of 1 and keeping other parameters fixed. From figure, we can see that network lifetime increases for $K \geq 2$ than $K=1$. That is the average network life time increases when we consider

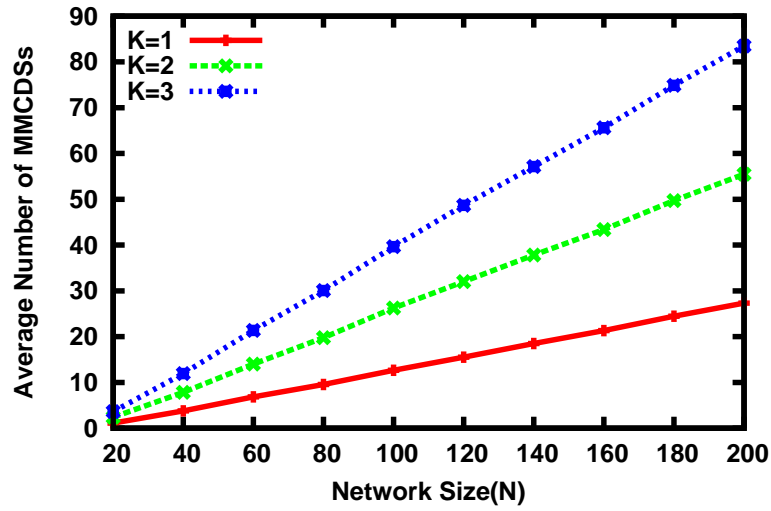
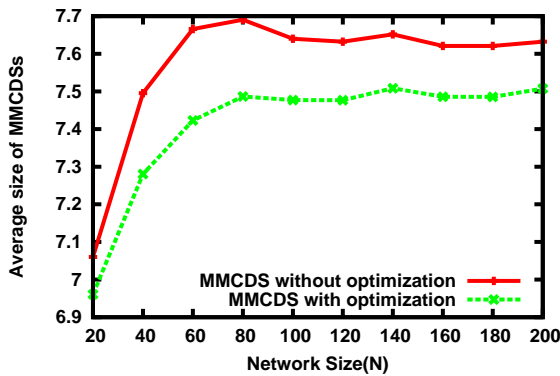
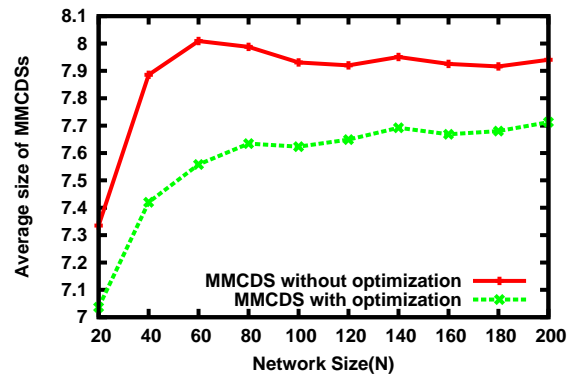


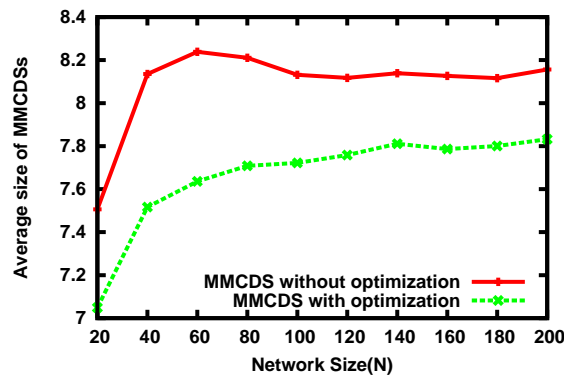
Figure 5.1: Number of MMCDSSs construction for different overlapping boundary varying network size



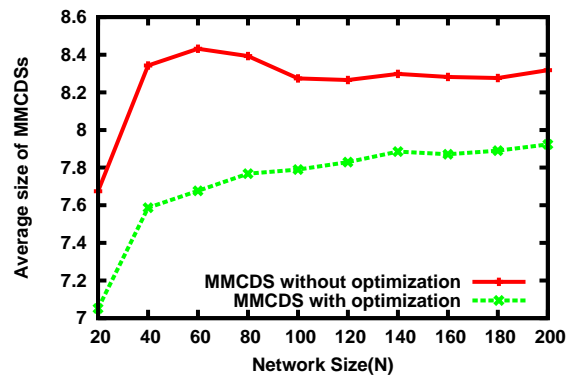
(a) K=2



(b) K=3



(c) K=4



(d) K=5

Figure 5.2: Results for size of MMCDSS construction with different values of K

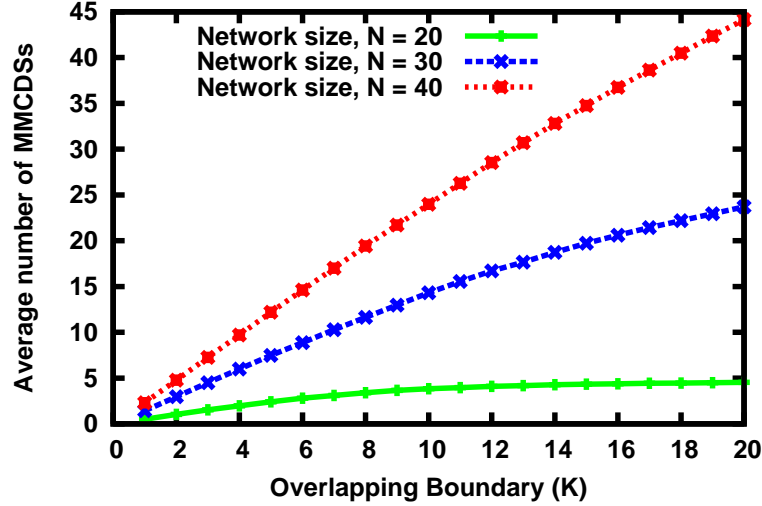


Figure 5.3: Number of MMCDs construction for different overlapping boundary

some overlapping. However, the lifetime remains almost same for higher values of K . Therefore, if we use overlapping boundary $K=2$, then we can achieve almost saturated network life. Additionally, we represent here in Figure 5.5, the relation of network lifetime with K more elaborately from two types of graph: sparse (number of nodes are minimum) and dense graph (number of nodes are maximum). Here, we represent the values of average network lifetime for overlapping boundary $K=1$ to 20. From figure, we can see that for both types of graph network lifetime increases for $K=2$ than $K=1$. However, for upper boundary values it is not stable. Sometimes, it falls and sometimes increases although the values are always higher than disjoint CDSs ($K=1$). Therefore, we can conclude that for overlapping cases network lifetime increases, but gives higher values for all types of network for $K=2$ or near values of 2.

5.3.3 Performance based on Network Fault Tolerance

Here, we present how network fault tolerance changes if we apply our algorithms in an ad hoc network varying the overlapping boundary K . We calculate network fault tolerance for different K varying the network size (number of nodes) from 20 to 200 with an increment of 20. We simulate the algorithm for 100 different random networks. As we have seen that with the increase of network size along with the number of MCDSs, the network fault tolerance increases also. Figure 5.6 illustrates the network fault tolerance for overlapping boundary $K=1$ to 5 with an increment of 1. Other parameters remain fixed here also. From figure, we can see that network fault tolerance for $K=1$ increases with the increase of network side. For, network size

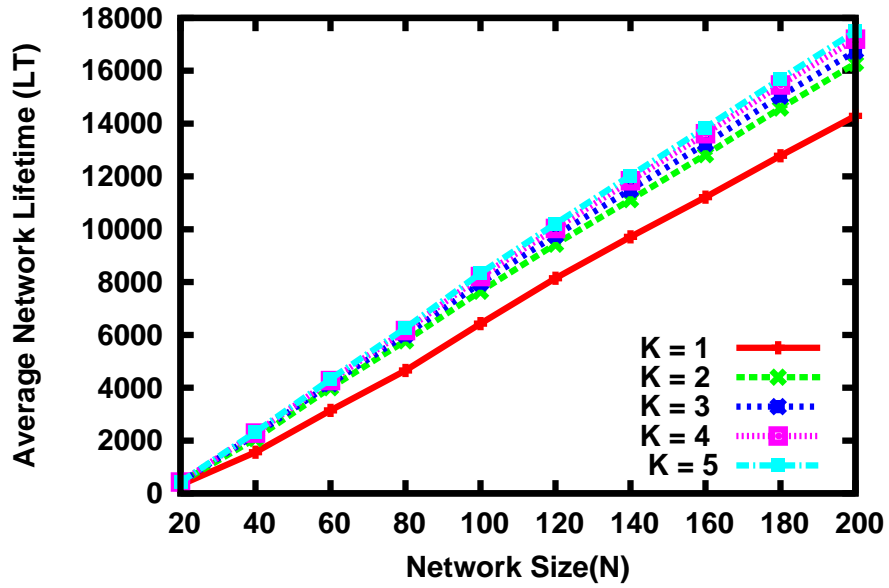
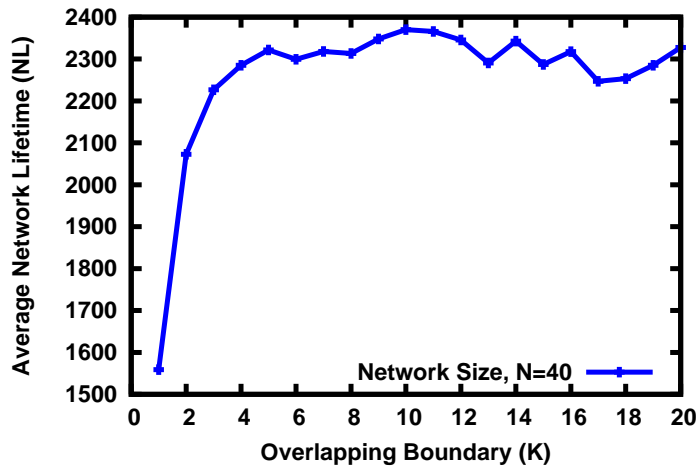
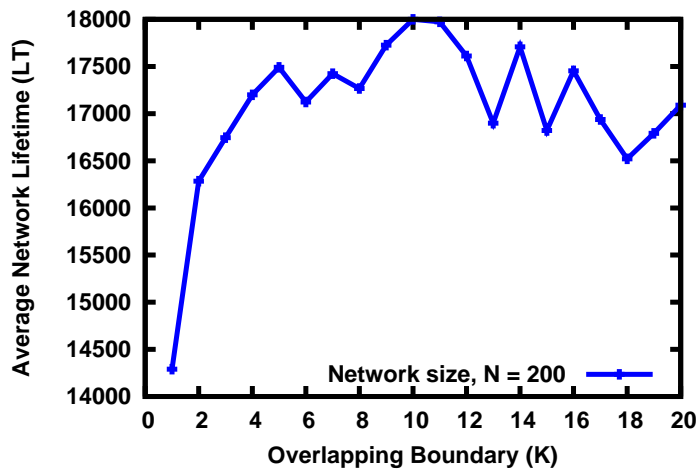


Figure 5.4: Average network lifetime for different overlapping boundary varying network size



(a) Sparse graph (N=40)



(b) Dense graph (N=200)

Figure 5.5: Results of average network lifetime for different overlapping boundary values on (a) Sparse graph and (b) Dense graph

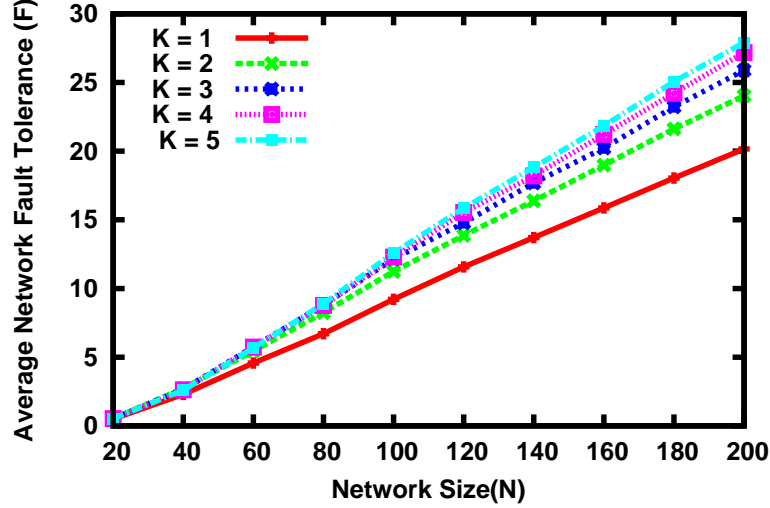


Figure 5.6: Average Network Fault Tolerance for different overlapping boundary varying network size

of 200, it can tolerate approximately 20 faults on average. However, this is basically the value of total numbers of disjoint MMCDS subtracting by 1. Although the number of MMCDSs increase gradually for other values of $K \geq 1$ (Figure 5.1), the fault tolerance values don't increase with that similar proportion. For example, for network size 200, when $K=2$, the network can handle almost 24 node failures. However, for same network size, the network can handle nearly 27 node failures for $K=5$. Therefore, we can say that in spite of having overlapping boundaries and more MCDSs, the fault tolerance doesn't improve that much. This is because when we consider more overlapping, the network tends to be more faulty. When any node with k' cardinality value fails it will fail k' sets from total n sets of MMCDSs. Therefore, with the increase of K , the fault tolerance doesn't increase comparing with the number of MMCDSs. Moreover, from the figure we can also notice that the fault tolerance increases more for $K=2$ than $K=1$. However, the values don't change that much for overlapping boundary $K > 2$. Hence, we can conclude that to get a better fault tolerance system the overlapping boundary $K=2$ is more desirable.

To clarify the impact of K on fault tolerance we represent Figure 5.7 also. We show the impact of K for sparse graph ($N=40$) and dense graph ($N=200$) separately. Here, we vary K from 1 to 20. From these both graph it is clearly visible that the fault tolerance increase abruptly when K value increase 1 to 2. Although, the fault tolerance increases to some point for $K > 2$, however it is not always true. Sometimes, it might fall (In Figure 5.7 (a) when $K=6$), when there is more overlapping among the sets. Therefore, we can conclude that when $K=2$ or nearly 2, we can get overall better results.

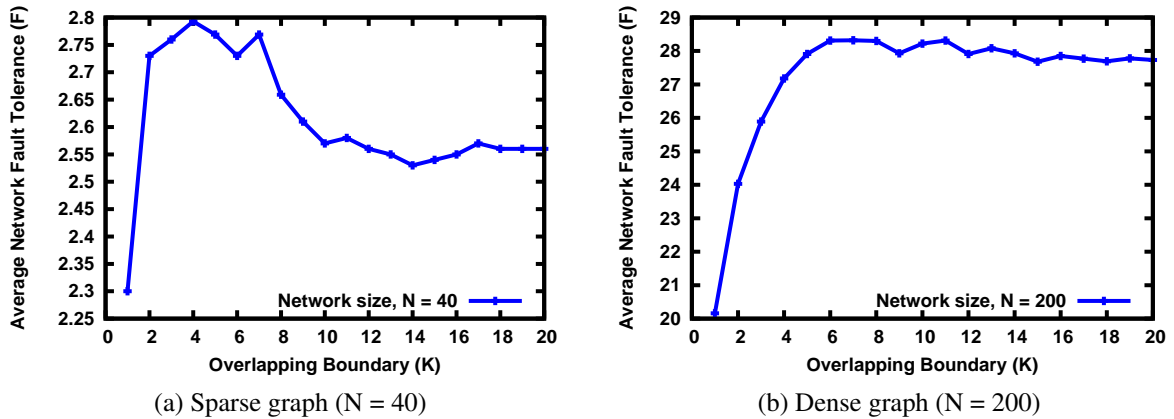


Figure 5.7: Results of average Fault Tolerance for different overlapping boundary values on (a) Sparse graph and (b) Dense graph

5.3.4 Performance based on Average Forwarding Nodes

Initially, we simulate our algorithms along with other algorithms for the network shown in Figure 1.2. The results of using average forwarding nodes of different algorithms is represented in Table 5.2. From the table we can see our centralized algorithm performs better than 1-2 CDS (a general case of k -connected- m -dominating sets where, $k=1$ and $m=2$) [33] in spite of having multiple sets. For our distributed algorithm, the number of forwarding nodes are not much higher than single Dominant Pruning (DP).

Algorithm	MCDS	MMCDS	1-2 CDS	DP	Distributed MMCDS
Avg forwarding nodes	2	2.667	4	3	3.5

Table 5.2: Table for the average forwarding nodes of different algorithms

Figure 5.8 represents average forwarding nodes for different network sizes for different algorithms. Here, we compare our centralized MMCDSs algorithm with $K=1$. From, figure we can see that, average forwarding nodes increase almost linearly for each algorithm. It is obvious that a single MCDS will perform better than any other algorithms. However, our algorithm also has better result than 1-2 CDS. This is because we always use MCDS to choose a new set and doesn't active more than one set at a single time. On the other hand, 1-2 CDS ensures at least two CDS nodes will be connected with other nodes which are not in CDS. Therefore, it needs more nodes to be active at a time and thus requires extra forwarding nodes. Additionally, Figure 5.9 shows the average node forwarding values of our algorithm along with the basic dominant pruning algorithm for packet forwarding. Although we consider here multiple sets construction our algorithm performs nearly the other one. However, if we consider any improved pruning

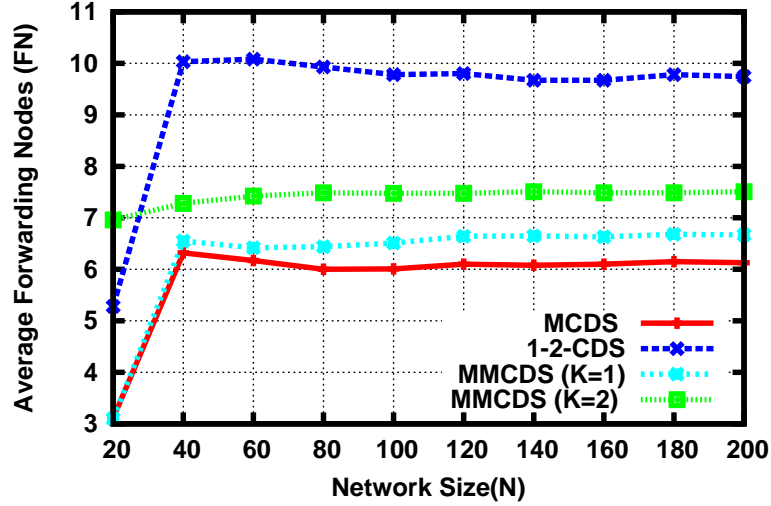


Figure 5.8: Average Forwarding Nodes of different centralized algorithms along with our MM-CDSs with $K=1$

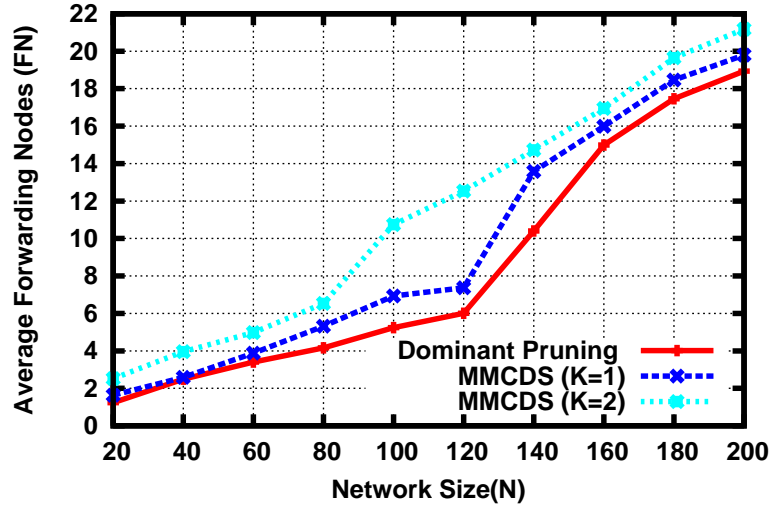


Figure 5.9: Average Forwarding Nodes of distributed algorithms along with our MMCDSs with $K=1$ and $K=2$

like total dominant or partial dominant pruning, our algorithm would give less values than the current one. To avoid extra data overhead and calculation overhead, we use here basic dominant pruning here.

5.3.5 Execution Time of MMCDSs Algorithm

In this subsection, we finally compare the execution time our algorithms. As stated earlier, in centralized algorithm it takes $O(V^3)$, where V is the number of nodes of the network. On the other hand, distributed algorithm takes $O(\Delta^4)$, where Δ is the maximum neighborhood size of a node. Therefore, the time need to execute the algorithm requires more time when the network

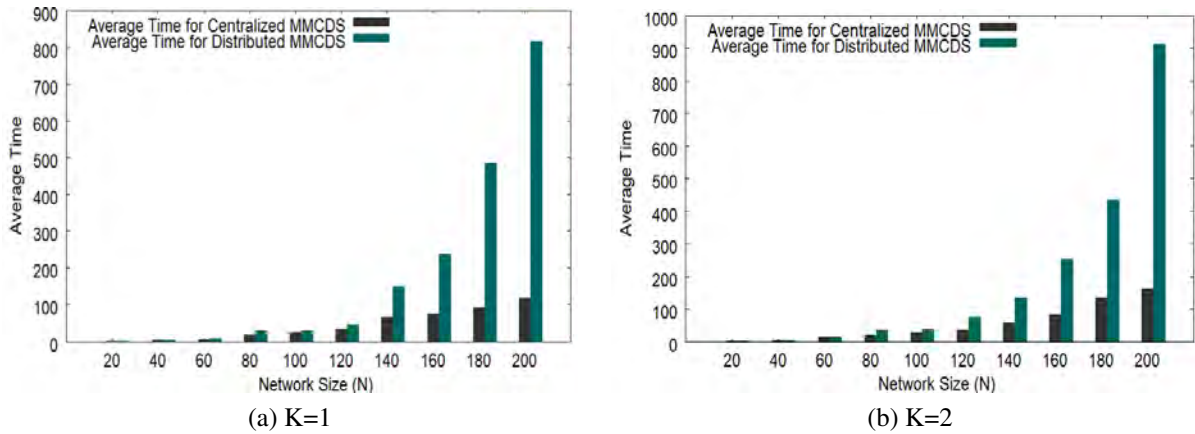


Figure 5.10: Execution Time of our MMCDs with $K=1$ and $K=2$

size increases and when it is a dense graph as well. Figure 5.10 the execution time of our derived algorithms. To experiment the execution time we run the algorithms with CPU of 4GB RAM, Intel Core i3 processor, 1.7 GHz, and 64 bit processor. From figure, we see that the overall performances are similar for both values of K , however, it requires some extra time when $K=2$. For both algorithms, it requires approximately similar time when the network size is smaller than 140. After that, it requires more time for distributed algorithm than the centralized one. This is because of their time complexity mentioned earlier.

Chapter 6

Conclusion and Future Work

In this paper, we concern about network lifetime and fault tolerance of wireless ad hoc networks. Therefore, for efficient communication among nodes over the network, we construct multiple connected dominating sets using possible minimum nodes. We can use those sets in round robin fashion to enhance network lifetime or keep as back up of active sets to handle system fault tolerance. However, always disjoint sets constructions might not be possible. Therefore, we introduce a user defined overlapping boundary which indicates in how much sets a node can be present. Moreover, we find out through simulation an appropriate boundary value for overlapping in multiple sets. We apply the strategy both in centralized and distributed version of our algorithm. A comprehensive simulation results is presented to analyse the behaviour of the developed algorithms. However, when we consider overlapping boundary $K \geq 2$, we only consider the worst case for calculating average fault tolerance. If we could consider all possible cases of node failures, the values of fault tolerance would improve more than our calculated values. Therefore, our future challenge is to provide a mathematical probabilistic model for analyzing system fault tolerance for all possible node failures. Our future work also includes to develop analytical model for finding out overlapping boundary based on network pattern.

Bibliography

- [1] A. K. Yadav, R. S. Yadav, R. Singh, and A. K. Singh, “Connected dominating set for wireless ad hoc networks: a survey,” *International Journal of Engineering Systems Modelling and Simulation*, vol. 7, no. 1, pp. 22–34, 2015.
- [2] 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, pp. 153–167, 2002.
- [3] S. Butenko, X. Cheng, D.-Z. Du, and P. M. Pardalos, “On the construction of virtual backbone for ad hoc wireless network,” in *Cooperative control: Models, applications and algorithms*, pp. 43–54, Springer, 2003.
- [4] J. Blum, M. Ding, A. Thaeler, and X. Cheng, “Connected dominating set in sensor networks and manets,” in *Handbook of combinatorial optimization*, pp. 329–369, Springer, 2004.
- [5] M. Akter, A. Islam, and A. Rahman, “Fault tolerant optimized broadcast for wireless ad-hoc networks,” in *2016 International Conference on Networking Systems and Security (NSysS)*, pp. 1–9, IEEE, 2016.
- [6] S. Saha, A. A. Zishan, and A. Rahman, “On target monitoring in directional sensor networks by jointly considering network lifetime and fault tolerance,” in *Proceedings of the 6th International Conference on Networking, Systems and Security*, pp. 68–76, 2019.
- [7] S. Butenko, X. Cheng, C. A. Oliveira, and P. M. Pardalos, “A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks,” in *Recent developments in cooperative control and optimization*, pp. 61–73, Springer, 2004.

- [8] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3, pp. 353–363, 2001.
- [9] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *International workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWiM) Boston, MA, USA August 20-22, 2000*.
- [10] W. Peng and X.-C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pp. 129–130, IEEE Press, 2000.
- [11] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Trans. Mobile Computing*, vol. 1, no. 2, 2002.
- [12] G. Călinescu, I. I. Măndoiu, P.-J. Wan, and A. Z. Zelikovsky, "Selecting forwarding neighbors in wireless ad hoc networks," *Mobile Networks and Applications*, vol. 9, no. 2, pp. 101–111, 2004.
- [13] A. Rahman, P. Gburzynski, and B. Kaminska, "Enhanced dominant pruning-based broadcasting in untrusted ad-hoc wireless networks," in *2007 IEEE International Conference on Communications*, pp. 3389–3394, IEEE, 2007.
- [14] A. Rahman, M. E. Hoque, F. Rahman, S. K. Kundu, and P. Gburzynski, "Enhanced partial dominant pruning (epdp) based broadcasting in ad hoc wireless networks.," *Journal of Networks*, vol. 4, no. 9, pp. 895–904, 2009.
- [15] Y. Kim and E.-C. Park, "An efficient relayed broadcasting based on the duplication estimation model for iot applications," *Sensors*, vol. 19, no. 9, p. 2038, 2019.
- [16] M. K. Goyal, S. P. Ghrrera, and J. P. Gupta, "Reducing the number of forward nodes from 1-hop nodes to cover 2-hop nodes with network coding," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 3-6, pp. 13–17, 2017.
- [17] 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.

- [18] 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, pp. 376–380, IEEE, 1997.
- [19] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.
- [20] Y. P. Chen and A. L. Liestman, "Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 165–172, 2002.
- [21] 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. 1-3, pp. 325–330, 2004.
- [22] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Transactions on parallel and distributed systems*, vol. 13, no. 1, pp. 14–25, 2002.
- [23] C. Penumalli and Y. Palanichamy, "An optimal cds construction algorithm with activity scheduling in ad hoc networks," *The Scientific World Journal*, vol. 2015, 2015.
- [24] 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, 2004.
- [25] N. Al-Nabhan, B. Zhang, X. Cheng, M. Al-Rodhaan, and A. Al-Dhelaan, "Three connected dominating set algorithms for wireless sensor networks," *International Journal of Sensor Networks*, vol. 21, no. 1, pp. 53–66, 2016.
- [26] J. P. Mohanty, C. Mandal, and C. Reade, "Distributed construction of minimum connected dominating set in wireless sensor network using two-hop information," *Computer Networks*, vol. 123, pp. 137–152, 2017.
- [27] T. N. Tran, T.-V. Nguyen, and B. An, "An efficient connected dominating set clustering based routing protocol with dynamic channel selection in cognitive mobile ad hoc networks," *Electronics*, vol. 8, no. 11, p. 1332, 2019.

- [28] O. Gulec, E. Haytaoglu, and S. Tokat, "A novel distributed cds algorithm for extending lifetime of wsns with solar energy harvester nodes for smart agriculture applications," *IEEE Access*, vol. 8, pp. 58859–58873, 2020.
- [29] A.-R. Hedar, R. Ismail, G. A. El-Sayed, and K. M. J. Khayyat, "Two meta-heuristics designed to solve the minimum connected dominating set problem for wireless networks design and management," *Journal of Network and Systems Management*, vol. 27, no. 3, pp. 647–687, 2019.
- [30] X. Bai, D. Zhao, S. Bai, Q. Wang, W. Li, and D. Mu, "Minimum connected dominating sets in heterogeneous 3d wireless ad hoc networks," *Ad Hoc Networks*, vol. 97, p. 102023, 2020.
- [31] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Fault-tolerant clustering in ad hoc and sensor networks," in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pp. 68–68, IEEE, 2006.
- [32] J. R. Diaz, J. Lloret, J. M. Jimenez, S. Sendra, and J. J. Rodrigues, "Fault tolerant mechanism for multimedia flows in wireless ad hoc networks based on fast switching paths," *Mathematical Problems in Engineering*, vol. 2014, 2014.
- [33] Y. Shi, Y. Zhang, Z. Zhang, and W. Wu, "A greedy algorithm for the minimum k-connected m-fold dominating set problem," *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 136–151, 2016.
- [34] 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.
- [35] M. S. R. Sohel, C. N. Ferdous, A. Rahman, A. J. Nafis, S. A. Shushmi, and R. Rab, "On constructing contention aware connected dominating sets for inter-connectivity among internet of things devices," *International Journal of Multimedia Intelligence and Security*, vol. 3, no. 3, pp. 244–270, 2019.
- [36] C. N. Ferdous and A. Rahman, "A contention aware connected dominating set construction algorithm for wireless ad-hoc networks," in *2018 14th International Conference on Wire-*

less and Mobile Computing, Networking and Communications (WiMob), pp. 1–8, IEEE, 2018.

- [37] P. Johnson and C. Jones, “Secure dominating sets in graphs,” *Advances in Domination Theory II*, pp. 1–9, 2013.
- [38] A. Burger, A. De Villiers, and J. Van Vuuren, “On minimum secure dominating sets of graphs,” *Quaestiones Mathematicae*, vol. 39, no. 2, pp. 189–202, 2016.
- [39] J. Barnett, A. Blumenthal, P. Johnson, C. Jones, R. Matzke, and E. Mujuni, “Connected minimum secure-dominating sets in grids,” *AKCE International Journal of Graphs and Combinatorics*, vol. 14, no. 3, pp. 216–223, 2017.
- [40] J. Zhou, Z. Zhang, S. Tang, X. Huang, Y. Mo, and D.-Z. Du, “Fault-tolerant virtual backbone in heterogeneous wireless sensor network,” *IEEE/Acm Transactions on Networking*, vol. 25, no. 6, pp. 3487–3499, 2017.
- [41] S. Farzana, K. A. Papry, A. Rahman, and R. Rab, “Maximally pair-wise disjoint set covers for directional sensors in visual sensor networks,” in *2016 Wireless Days (WD)*, pp. 1–7, IEEE, 2016.
- [42] B. N. Clark, C. J. Colbourn, and D. S. Johnson, “Unit disk graphs,” *Discrete mathematics*, vol. 86, no. 1-3, pp. 165–177, 1990.
- [43] M. Akter, A. Islam, and A. Rahman, “Fault tolerant optimized broadcast for wireless ad-hoc networks,” in *2016 International Conference on Networking Systems and Security (NSysS)*, pp. 1–9, 2016.
- [44] M. B. K. Dhir, “A survey on fault tolerant multipath routing protocols in wireless sensor networks,” *Global Journal of Computer Science and Technology*, 2016.