Fault Tolerant Multiple Dominating Set Constructions for Wireless Ad-hoc Networks

Khaleda Akther Papry and Ashikur Rahman

Abstract In wireless ad-hoc networks, 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 is an NP-complete problem. Although some approximation algorithms are available, the CDS or its approximation has poor fault tolerance. 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 heuristics to show the improvement of network lifetime and system fault tolerance.

1 Introduction

Wireless ad-hoc networks consist of some wireless nodes that communicate over the networks without the existence of any fixed infrastructure. Broadcasting is the most common communication method in such networks where each node over the network receives the message from a source node. Among different approaches, uncontrolled flooding is the easiest approach for broadcasting where each node unconditionally distributes its incoming packets to each of its neighbors. Therefore, it causes too much traffic, contention and collision resulting into broadcast storm problem [1]. This problem can be minimized by creating a virtual backbone and Connected Dominating Set (CDS) is one of them [2].

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 members of CDS. A CDS in an ad-hoc network can serve as a backbone for packet routing over the network. Figure 1 shows a network with seven nodes where connectivity means their transmission range. Here, some possible CDSs are: {A,C}, {A,B,F}, {A,B,C}, {A,C,E}, {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

Khaleda Akther Papry
Bangladesh University of Engineering and Technology, e-mail: papry.05084@gmail.com

Ashikur Rahman
Bangladesh University of Engineering and Technology e-mail: ashikur@buet.ac.bd

with routing redundancy, the number of forwarding packets are also decreased. For example, {A,C} is the most desirable CDS for the graph.

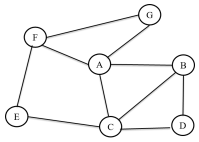


Fig. 1: A random ad-hoc network topology with 7 nodes

A CDS with minimum number of nodes is called minimum connected dominating set (MCDS). However, finding an MCDS is an NP-complete problem [3]. Different heuristics have already been proposed 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. If there is a backup CDS for routing, the system might work properly again which is another prominent research area in recent years.

In this paper, we aim to find out multiple minimum connected dominating sets (MMCDS) using minimal nodes to make system more fault tolerant and to increase the network longevity both for centralized and distributed algorithms. In our both algorithm, we allow some minimum overlaps when fully disjoint sets are not possible to generate. Moreover, we can apply our algorithm in both static and dynamic scenario, just making some little changes.

The main contributions of this work are enumerated below:

1. We develop a centralized and a distributed approach for MMCDS construction using overlapping boundary K.
2. Finally, we validate the algorithms using simulation results, and compare the performance of the two algorithms with existing state-of-the-art algorithms.

2 Related Work

Many broadcasting algorithms have been proposed over the decades to overcome the broadcast storm problem. Lim and Kim provide a new flooding methods [4] for efficient broadcasting named dominant pruning. In dominant pruning algorithm, each node uses its 2-hop neighbor information to reduce redundant transmissions. To reduce broadcast redundancy in ad-hoc wireless networks, Lou and Wu [5] proposed two improved algorithms based on dominant pruning. Rahman et al. proposed enhanced dominant pruning based [6] and partial dominant pruning based [7] broadcasting in untrusted ad-hoc wireless networks.

Ephremides et al. [8] first proposed the idea of using a CDS as virtual backbone for broadcasting. Butenko et al. [9, 2] proposed both distributed and centralized greedy algorithms for constructing minimum CDS (MCDS). In paper [10], the authors propose a greedy algorithm for MCDS in unit-disk graphs based on Maximal Independent Set (MIS) which has been used in 3D heterogeneous network [11] later.

Fault tolerance 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. Papers [12, 13] provide fault tolerant CDS models 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. The idea of connected minimum secure dominating sets has been first proposed by Barnett et al. in [14]. However, this works in cylindrical or toroidal grid networks only. In [15, 16], the authors derive multiple set covers for directional sensor networks which is a special case of wireless ad-hoc networks.

None of all the above algorithms have considered fault tolerance and network longevity jointly for wireless ad-hoc networks which is our main focus in this paper.

3 Preliminaries

Here, we discuss network model and some definitions used throughout the paper.

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. Figure 1 represents a wireless ad-hoc network with seven nodes with similar transmission range.

3.2 Important definitions

- **Multiple MCDS (MMCDS)**: MMCDS refers to generating different possible MCDSs from a network. As fully disjoint sets might not be achieved always, here we consider overlapping up to a certain amount denoted as K which refers to the boundary value of a node's presence in multiple sets.
- **Overlapping Boundary (K)**: For creating multiple MCDSs (MMCDSs), we consider overlapping among sets bounded by K. The parameter denotes that no node can participate in MCDSs more than this given upper bound value. For example, if K is 2, then a node can be present up to two MCDSs among MMCDSs.
- **Cardinality** (C): It is the measure of participation in different CDS of node i. For each node the value is bounded by K, that means $C_i \leq K$.
- **Dominant Pruning**: For distributed system, we use the procedure of creating the forwarding list followed by dominant pruning [4]. Suppose, u sends a message to v. Now, v will create its forwarding list from it's one-hop neighbor B_v that are not in $N(u)$. For constructing the new forwarding list, node v needs all uncovered two-hop neighbours U_v. Node v selects nodes from B_v to cover all the nodes in U_v and added to forward list F_v. To create (F_v), we calculate U_v and B_v from the following formulas [4]:
  $U_v = N(N(v))$ - $N(v)$ - $N(u)$ and $B_v = N(v)$ - $N(u)$

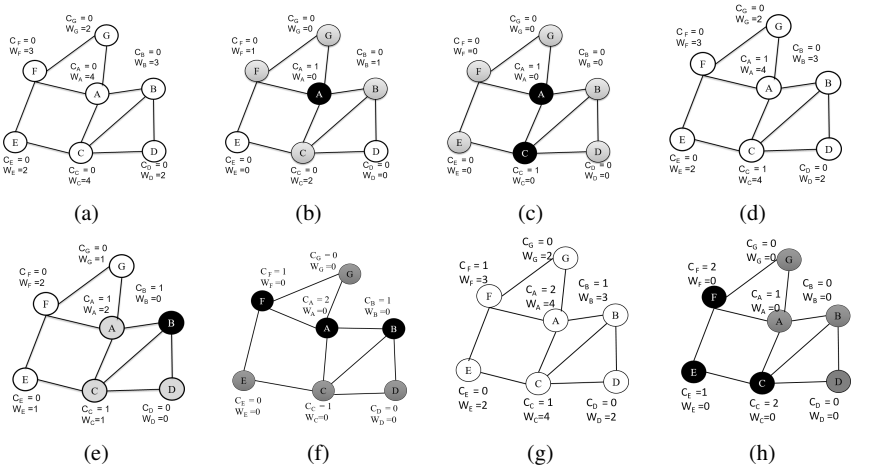


Fig. 2: Basic Greedy Centralized Construction of MCDS (a) $\sim$ (c) and multiple MCDS construction (d) $\sim$ (h)

4 Multiple Minimum Connected Dominating Sets (MMCDSs)

In this section, we represent our both centralized and distributed multiple connected dominating sets (MMCDSs) constructions respectively.

4.1 Centralized Algorithm for MMCDSs

In this approach, we run a centralized greedy algorithm to generate multiple CDSs with minimum nodes and minimum overlaps (up to K). It follows the basic MCDS construction [2] at each iteration. After a complete iteration a CDS is constructed with minimum nodes. In the next generation of MCDS, 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 sets can be generated. After the generation of each CDS set, another optimization algorithm is run to remove redundant nodes from the set. Figure 2 represents how our algorithm works. We keep the overlapping boundary value K=2. We can see that 2 (a) $\sim$ (c) constructs the first MCDS. Then all the nodes are initialized again except their cardinality values (2 (d)). Another new set is generated by node F, A and B (2 (e) $\sim$ (f)). At third iteration (2 (g) $\sim$ (h)), C,E, F constructs a new set although node D was selected first. D is discarded using optimization function as redundant node.

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 outer loop (line 3 $\sim$ 23) generates the desired MMCDSs and runs until no new set is found. The inner loop (lines 5 $\sim$ 17) creates a single MCDS after a complete iteration. In the inner while loop, a node s from V (first iteration) or *GraySet* (other iteration) with minimum cardinality and maximum white neighbors is selected (line 6 $\sim$ 10). The

selected node is added to CDS set and this procedure runs until $WhiteSet$ is empty or no new node can be selected with cardinality less than K. After a complete iteration, the set is optimized by removing redundant nodes using optimization algorithm in [16], if there is any and added it to CDS_n set (line 20). If there is any redundant nodes in CDS that is not in CDS_n, then its cardinality is decremented by 1, as the node is not really used in the set (line 21). The value of n is incremented after a successful creation of CDS.

Algorithm 1 Centralized Multiple Dominating Sets construction Algorithm

Require: $G(V,E),K$
Ensure: A Set of MCDSs CDS
1: $CDS = \emptyset$,n $\leftarrow 0$
2: Set $C_i \leftarrow 0$, for all i $\in V$
3: **while** a new set can be generated **do**
4: $CDS = \emptyset$, $WhiteSet = V$, $GraySet = \emptyset$
5: **while** $WhiteSet \neq Null$ **do**
6: **if** $CDS == \emptyset$ **then**
7: Find the node $s \in V$ with maximum W_s and minimum C_s
8: **else**
9: Find the node $s \in GraySet$ with maximum W_s and minimum C_s
10: **end if**
11: **if** $C_s < K$ and $W_s > 0$ **then**
12: $CDS = CDS \cup \{s\}$, $C_s = C_s + 1$
13: $GraySet = GraySet \cup N(s) - \{s\}$, $WhiteSet = WhiteSet - N(s) - \{s\}$
14: **end if**
15: **end while**
16: **if** $CDS \neq Null$ **then**
17: $n = n + 1$
18: Optimize CDS by removing redundant nodes and add to CDS_n
19: Update C_k by $C_k - 1$ for all $k \in (CDS - CDS_n)$
20: **end if**
21: **end while**

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 $O(V^2)$. There is an optimization function after the inner while loop and within the outer while loop which also costs $O(V^2)$ also. Therefore, the total time complexity of our Centralized MMCDS is $O(V^3)$ times.

4.2 Distributed Algorithm for MMCDSs

In distributed algorithm, for creating multiple MCDSs, we use the dominant prunning 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.

Initially, we consider the algorithm for static ad-hoc network and generate multiple forwarding lists for a node receiving packets. Here, a node can be maximum used

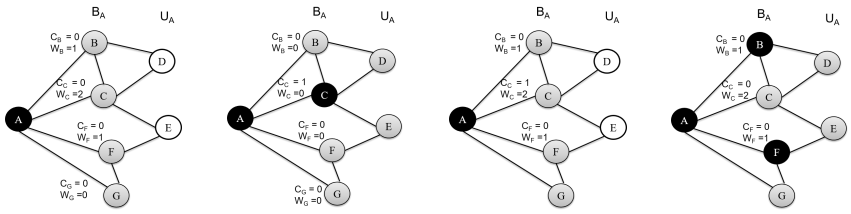


Fig. 3: Basic Greedy Distributed Forwarding List Construction of MCDS (a) $\sim$ (b) and multiple MCDS (c) $\sim$ (d) constructions for static scenario

up to K forwarding lists as like the centralized one. For this algorithm, a node from its one-hop neighbor list is selected for forwarding that has minimum cardinality and maximum number of node coverage of its uncovered two-hop neighbors. The process is repeated until all uncovered two-hop neighbors are covered. To generate multiple sets, the above process is repeated until no new set is generated. Figure 3 represents how node A creates its forwarding lists for static system. Figure (3(a)) illustrates one-hop (B_A) neighbors and two-hop uncovered neighbors (U_A) of node A. From B_A, node C is selected first with maximum white neighbors from U_A. Here, only C constructs the first forwarding list as it covers all nodes from U_A (3(b)). As, we aim to create multiple forwarding lists, after first iteration, configuration are initialized as initial values except the cardinality values of C (3(c)). Again, another new forwarding list (B and F) is created using the same procedure (3(d) $\sim$ (e)).

For dynamic scenario, as the environment changes,we generate only a set for each time and store last $K-1$ used sets. For the next time, it creates its new forwarding list with minimal overlap with last $K-1$ sets.

Algorithm 2 represents a set of n forward lists (F_{vn}) creation algorithm for static scenario of a node v. The outer while loop generates multiple forwarding lists (line 3 $\sim$ 14). In each iteration it selects node s from B_v with minimum cardinality and maximum uncovered 2-hop neighbours (line 4). All neighbors of s are discarded from U, the cardinality of s is incremented by 1, and s is added to F_{vn} (line 6). If it's cardinality reaches to K then it is discarded from B_v (line 7 $\sim$ 8). This procedure runs until there is no node remaining in U. Whenever, U becomes null, it is again initialized to U_v to create next forwarding list along with necessary changes.

Suppose, Δ is the maximum degree of the graph. Hence, the size of B_v and U_v can be at most Δ and Δ^2 respectively. Therefore, the run time complexity of creating a forwarding list is Δ^3. For static network, multiple forwarding lists could be at most Δ. Hence, total time complexity is Δ^4. On the other hand, for dynamic scenario, a single set is generated comparing with previous K-1 lists with $K\Delta^3$ complexity .

5 Experiments

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

Algorithm 2 Forward Lists Creation of a node v for Distributed Algorithm

Require: B_v, U_v
Ensure: A Set of forwarding lists
1: $F_{v1} = \emptyset$, $U = U_v$, $n = 1$
2: Set $C_i \leftarrow 0$ for all node $i \in B_v$
3: **while** a new set can be generated **do**
4: Select a node $s \in B_v$ with minimum C_s and maximum $N(s) \cap U$
5: **if** $(N(s) \cap U) > 0$ **then**
6: $F_{vn} = F_{vn} \cup s$, $U = U - N(s)$, $C_s = C_s + 1$
7: **if** $C_s == K$ **then**
8: $B_v = B_v - s$
9: **end if**
10: **end if**
11: **if** U is $NULL$ **then**
12: $U = U_v$, $n = n + 1$, $F_{vn} = \emptyset$
13: **end if**
14: **end while**

5.1 Experimental Setup

We simulate our fault tolerant MMCDS algorithms along with basic MCDS and other algorithms. Here, we consider a network over an area of 100 × 100 units followed by N wireless nodes. Initial battery power is kept fixed at 100 unit and transmission range is kept fixed at 25 units. Overlapping boundary K is set 1 to 5 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.

5.2 Performance Measurements

Here, we present the performance our algorithms considering number of MMCDSs evaluation, size of MMCDSs with and without optimization function, average packet passing, network life time and fault tolerance of the network.

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. However, generating multiple disjoint CDSs always might not be feasible. Therefore, we allow some overlapping among the CDSs bounded by K. 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 after a certain value of K as no new set can be generated. For simulation, we use K = 1 to 20 with a increment of 1 and find out a suitable value of K for upper boundary of overlapping among generated MCDSs.

5.2.2 Network lifetime

The network lifetime means how long the network remains active. Generating multiple CDSs can increase network longevity by using all the sets in round robin scheduling. For calculating network life time of our algorithm, we assume that all the nodes have similar battery power with 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 becomes $NL = T$. On the other hand, with the increase of the number of CDSs, network lifetime increases also as each CDS is scheduled in different time periods. If there are n disjoint CDSs, then the network lifetime of the system becomes, $NL = nT$ [16]. However, if there exists some overlapping CDSs up to K boundary overlap, then network lifetime becomes as follows [16]:

$$NL = \sum_{i=1}^{n} t_i \qquad where, \qquad t_i = \frac{T}{\max(C_{ij} : j = 1, 2, ..., \|CDS_i\|)} \tag{1}$$

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

5.2.3 Network Fault Tolerance

Fault tolerance of a network can be calculated as up to how many node failures it can tolerate or handle. 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. For our system, fault tolerance is very higher than a single MCDS network. If our system generates n disjoint sets, then it can tolerate up to n-1 node failures. However, as we consider some overlapping up to K, the fault tolerance value decreases and it depends on total number of sets creation and the overlapping boundary K. We calculate fault tolerance F for the system which satisfies the following equation:

$$n - \sum_{i=1}^{F} \max(C_i) = 0 \tag{2}$$

Where, C_i is cardinality of i-th node failure.

5.2.4 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 [17]:

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

As, our algorithm runs multiple CDSs, we consider average forwarding nodes (AFN) where it is the average of total forwarding nodes (TFN). Suppose, our algorithm runs n CDSs in round robin fashion. Hence, The equation can be defined as:

$$AFN = \frac{TFN}{n}; \qquad where, \qquad TFN = \sum_{i=1}^{n} NFN_i \tag{3}$$

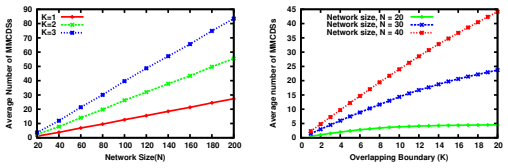
5.3 Experiment Results

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

5.3.1 Number of MMCDSs

We evaluate total number of CDSs varying network size with N=20 to 200 with an increment of 20. Figure 4 (a) illustrates the results of CDSs construction applying optimization step. 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 4 (b) 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 increase. However, it moves to a saturated point when there is no new node to generate a new set. 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.

Finally, Figure 5 illustrates how the sizes of MCDSs construction decreases for different network size with optimization step. This happens because when we apply optimization, unnecessary nodes are removed from a set which reduces the sizes of MCDSs. Therefore, the total number of MCDSs also might increase as removed nodes might be used for further set construction.



(a) (b)

Fig. 4: Number of MMCDSs construction (a) for different overlapping boundary varying network size, (b) for different network size varying overlapping boundary

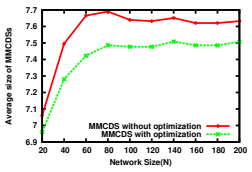


Fig. 5: Size of MMCDS construction for K=2 with optimization and without optimization

5.4 Network Lifetime

Here, we present how our algorithms increases network lifetime in an ad-hoc network. Figure 6 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. However, the lifetime remains almost same for higher

values of K. Therefore, if we use overlapping boundary K=2 or close to 2, then we can achieve maximum network life. Additionally, we represent here in Figure 7, the relation of network lifetime with K more elaborately for 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 observe that for both types of graph give higher values for K=2 or near values of 2.

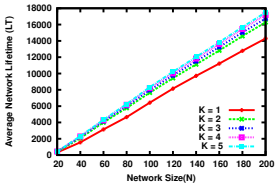
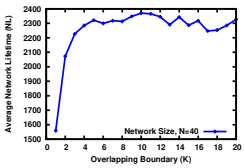
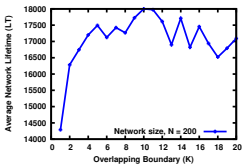


Fig. 6: Average network lifetime for different overlapping boundary varying network size



(a) Sparse graph (N=40)



(b) Dense graph (N=200)

Fig. 7: Results of average network lifetime for different overlapping boundary values on (a) Sparse graph and (b) Dense graph

5.5 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. Figure 8 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 of 200, it can tolerate approximately 20 faults on average. Although the number of MMCDSs increase gradually for other values of $K \geq 1$ (Figure 4), 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 more overlapping boundaries although generate more MCDSs, the fault tolerance doesn't improve that much. Figure 9 shows how K changes the fault tolerance for different network. It can be observed that, K has similar effects for all types of network.

5.6 Average Forwarding Nodes

Figure 10 represents average forwarding nodes for different network sizes for different algorithms. Here, in 10 (a) we compare our centralized MMCDSs algorithm with K=1 and K=2. From, figure we can see that, average forwarding nodes increase almost linearly for each algorithm. It is obvious that a single MCDS has less nodes forwarding than any other algorithms. However, our algorithm has better result than

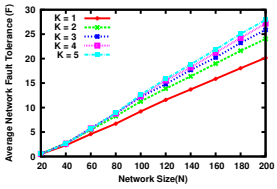
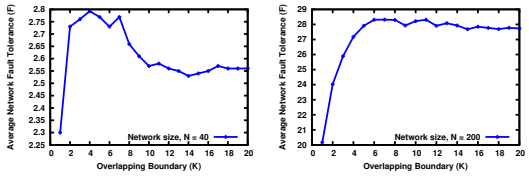


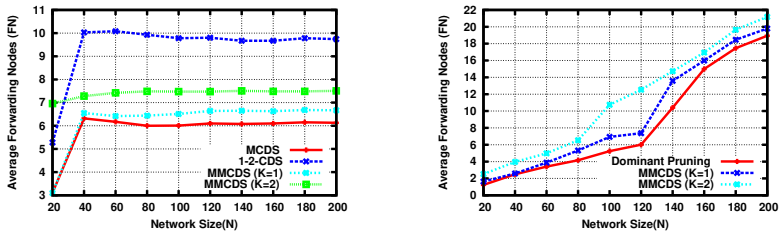
Fig. 8: Average Network Fault Tolerance for different overlapping boundary varying network size



(a) Sparse graph (N = 40) (b) Dense graph (N = 200)

Fig. 9: Results of average Fault Tolerance for different overlapping boundary values on (a) Sparse graph and (b) Dense graph

1-2 CDS which is a general case of k-m CDS. Additionally, Figure 10 (b) 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 basic one.



(a) (b)

Fig. 10: Average Forwarding Nodes of (a) centralized and (b) distributed algorithms along with our MMCDSs with K=1 and K=2

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. 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 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.

References

1. 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.
2. 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.
3. 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.
4. H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3, pp. 353–363, 2001.
5. W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Trans. Mobile Computing*, vol. 1, no. 2, 2002.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. S. Farzana, K. A. Papry, A. Rahman, and R. Rab, "Maximally pair-wise disjoint set covers for directional sensors in visual sensor networks," in *Wireless Days (WD)*, pp. 1–7, IEEE, 2016.
16. 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.
17. 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.