

# A SIMULATION STUDY ON THE STRONG NEIGHBORHOOD-BASED STABLE CONNECTED DOMINATING SETS FOR MOBILE AD HOC NETWORKS

Natarajan Meghanathan and Michael Terrell

Jackson State University, Jackson, MS, USA

Corresponding Author E-mail: natarajan.meghanathan@jsums.edu

## **ABSTRACT**

*We present a detailed simulation based performance study of our recently proposed strong-neighborhood based connected dominating sets (SN-CDS) for mobile ad hoc networks (MANETs). We compare the performance of SN-CDS with that of the well-known minimum connected dominating sets (MCDS) and the ID-based connected dominating sets (ID-CDS) for MANETs. We show that by computing the SN-CDS involving links that are only within 90% of the maximum transmission range of the nodes, we are able to significantly improve the stability of the CDS (as large as 250%), compared to the MCDS strategy of choosing the minimum number of nodes to cover the entire network. An SN-CDS requires only at most 20% more nodes and has the potential to consume only at most 25% more energy compared to that of a MCDS. On the other hand, the ID-CDS requires 40-70% more nodes and has the potential to consume as large as 140% more energy compared to that of a MCDS. The diameter of an SN-CDS is at most 10% more than that of a MCDS, and the hop count per source-destination path (between any two randomly chosen network nodes) involving the CDS nodes as intermediate nodes is about the same for both the SN-CDS and MCDS. Thus, the SN-CDS effectively balances the stability-node size tradeoff and is significantly more stable than that of a MCDS, only at the cost of a minimal to moderate increase in the other performance metrics.*

## **KEYWORDS**

*Broadcast, Connected Dominating Sets, Mobile Ad hoc Networks, Neighborhood, Node Size, Simulations, Stability, Tradeoffs*

## **1. INTRODUCTION**

A mobile ad hoc network (MANET) is a dynamically changing distributed system of arbitrarily moving wireless nodes that operate under resource constraints such as limited battery charge, memory and processing capacity. In addition, the network operates under limited bandwidth, necessitating limited transmission range for the nodes to prevent collisions when packets are transmitted over long-distance. A node within the transmission range of another node is said to be a 'neighbor' of the latter, and the set of nodes within the transmission range of a node are said to constitute the 'neighborhood' of the node. Due to the limited transmission range of the nodes, MANET routes are generally multi-hop in nature, and these routes are discovered through an expensive broadcast query-reply cycle involving participation from all the nodes in the network. Due to the dynamically changing topology, the routes fail over time, leading to the frequent route discoveries.

Traditionally, MANET broadcast route discoveries have been implemented using flooding under which the broadcast message initiated at one node is forwarded exactly once by every other node. Even though flooding ensures that every node gets the broadcast message, it incurs significant overhead due to multiple redundant transmissions and the resulting energy and  
DOI : 10.5121/ijcnc.2012.4401

bandwidth consumption is quite high [1]. As a result, the motivation has been to use efficient communication topologies to broadcast the route discovery packets and other network-wide packets. In this context, researchers have leveraged the idea of ‘connected dominating sets’ (CDS) from graph theory, and advocated the use of CDS for network-wide broadcasting. A CDS of a graph is a sub graph comprising of a subset of the vertices in the original graph, such that any vertex in the graph is either in the CDS or connected (through an edge) to a vertex in the CDS [2]. In a unit-disk graph modeling a MANET, all the nodes in the network are represented as vertices; there exists an edge between any two nodes in the graph only if the two nodes corresponding to the end vertices of the edge are neighbors in the MANET [3]. We say a network is covered if every node in the network is either in the CDS or is a neighbor of a node in the CDS.

As obviously, the entire graph could be a CDS, the primary objective of researchers has been to determine a CDS of the smallest size (i.e., CDS with the minimum number of nodes) that can cover the entire network. Such as CDS is hereafter referred to as the Minimum Connected Dominating Sets (MCDS). Unfortunately, the problem of determining a CDS of the smallest size is NP-complete [2]. Several heuristics had been proposed to approximate a CDS that will have the minimum number of constituent nodes (i.e. CDS Node Size), and yet cover all the nodes in the network. A common thread among all of the MCDS heuristics is to give preference to high-density nodes (i.e. nodes with several neighbors) for inclusion into the CDS so that the number of nodes in the CDS can be minimized. In an earlier research [4][5], and also vindicated through simulations in this paper, we observe that a MCDS is quite unstable in the presence of a dynamically changing network topology, characteristic of MANETs. Since a MCDS involves fewer nodes, the physical Euclidean distance (i.e. distance between the two end nodes of an edge) of the edges are closer to the transmission range of the nodes in order to span over a larger distance and cover multiple nodes in the network. This is also supported by an observation (in our simulations) that the edge distance ratio (the ratio of the physical Euclidean distance to that of the transmission range) of the MCDS edges at the time of CDS formation is 0.8 or above, on average. As a result, the two end nodes constituting an MCDS edge are highly likely to move out of their transmission ranges any time in the near future. Thus, there are very few edges in a MCDS and these edges are vulnerable to break any time.

All of the above observations lead us to develop an algorithm to determine connected dominating sets that will exist for a longer time. In the conference version [6] leading to this journal paper, we had proposed an algorithm to determine stable CDS using the notion of a strong neighborhood of the MANET nodes. We introduced a term called the ‘Threshold Neighborhood Ratio’ ( $0 \leq \text{TNDR} \leq 1$ ) that defines the strong neighborhood of a node. A node  $j$  at a physical Euclidean distance of  $r$  from node  $i$  is said to be in the strong neighborhood of node  $i$  if the Edge Distance Ratio  $r/R \leq \text{TNDR}$  where  $R$  is the fixed transmission range of all nodes in the network (i.e., we assume a homogeneous MANET). Note that like MCDS, an SN-CDS also prefers to include nodes with a larger number of uncovered neighbors as part of the CDS. However, the use of TNDR imposes the Strong Neighborhood criteria for defining the unit disk graph (i.e. MANET topology) and limits the physical Euclidean distance between the CDS nodes at the time of formation of the SN-CDS, and this largely contributes to the stability of the SN-CDS, as observed in our simulations. On the other hand, the Open Neighborhood policy adopted at the time of constructing a MCDS leads to the selection of unstable edges. Since the SN-CDS, when operated with a TNDR value of 0.9, showed significant connectivity close to that of a MCDS – all of our simulation results and analysis for SN-CDS in this paper would be based on a TNDR value of 0.9.

The contributions of this extended conference paper, compared to its original version are: (i) We study the performance of SN-CDS with respect to another well-known MANET CDS algorithm, referred to as ID-CDS [7], that gives preference to nodes with larger IDs to be part of the CDS.

We show that SN-CDS performs well against ID-CDS with respect to all the performance metrics evaluated in the simulations. (ii) We conduct an exhaustive simulation study of the SN-CDS, MCDS and ID-CDS under diverse conditions of network density and node mobility (low, medium and high scenarios for both density and mobility) and compare them with respect to several performance metrics – such as the Effective CDS lifetime (measured as the product of the absolute CDS lifetime and the percentage connectivity of the CDS); CDS Node Size and Edge Size; CDS Diameter (the maximum of the minimum distance between any two CDS nodes); Hop Count per source-destination path (the average number of hops in the paths between any two nodes in the network, with the CDS nodes serving as intermediate nodes, if needed); CDS Energy Index (a measure of the potential energy consumption that could result from the CDS; and the metric is computed as a product of the CDS Edge Size and the Edge Distance Ratio). (iii) The extensive simulation study enabled us to analyze the tradeoffs between the different CDS formation algorithms with respect to the various performance metrics, and conclude that the SN-CDS effectively balances the tradeoff between stability and the other performance metrics. In addition to the above, we also provide extensive details of the three CDS formation algorithms analyzed in this paper, and also explain their working through example graphs.

The rest of the paper is organized as follows: Section 2 reviews related work in the area of connected dominating sets for MANETs. Section 3 describes the SN-CDS algorithm in detail and illustrates its working through an example. The section also describes the similarity of the SN-CDS and the MCDS algorithms and illustrates the construction of a MCDS through an example. Section 4 reviews the ID-CDS algorithm and illustrates its working through an example. Section 5 describes the simulation environment, introduces the different performance metrics, presents the performance results of the three connected dominating sets evaluated in this paper, and interprets them. Section 6 concludes the paper with a summary of the simulation results observed and outlines future work. Note that the terms ‘node’ and ‘vertex’, ‘link’ and ‘edge’, ‘packet’ and ‘message’ are used interchangeably throughout the paper. They mean the same.

## 2. RELATED WORK

In this section, we briefly review the very few CDS-related algorithms that have been proposed in the MANET literature. The MCMIS (Maximal Independent Set with Multiple Initiators) algorithm [8] forms a complete virtual backbone by first developing a set of disjoint dominating trees, each kick started by an initiator node selected on the basis of the tuple <stability, effective degree and node ID>, and then connecting these trees through overlapping edges. In [9], the authors propose a signal-strength based CDS algorithm according to which a node categorizes a link to a neighbor node as a non-weak link if the number of beacons received from the neighbor is at least a threshold; the CDS algorithm prefers to include nodes that have a larger number of non-weak links.

In [10], the author had proposed a minimum node velocity-based CDS (MinV-CDS) formation algorithm that prefers to include slow moving nodes as part of the CDS. The MinV-CDS required a significantly larger number of nodes to be part of the CDS in order to sustain stability. In [11], the authors had proposed an algorithm to construct stable CDS by preferring to include the end nodes of the links (to be part of the CDS) that have a larger predicted link expiration time (LET). Referred to as LET-CDS, it is the only algorithm that constructs a CDS by directly selecting links rather than nodes. Like the MinV-CDS, the LET-CDS also existed for a significantly longer lifetime; albeit, at the cost of a larger CDS Node Size.

In [12], the author had proposed a benchmarking algorithm to determine the sequence of stable connected dominating sets that would exist for the longest amount of time such that the number of CDS transitions is the global minimum. Referred to as the *OptCDSTrans*, the algorithm

assumes the availability of the mobility profile of the nodes and the entire sequence of future topology changes to obtain a connected intersection graph spanning over multiple time instants since the current time instant. A CDS determined on the intersection graph is bound to exist in all the time instants spanned by the graph. Since the intersection graph spans over the longest sequence of static graphs as long as connectivity is maintained, the CDS determined on the intersection graph is the most stable. The above procedure is repeated for the rest of the session.

### 3. ALGORITHM TO CONSTRUCT STRONG NEIGHBORHOOD-BASED CONNECTED DOMINATING SET

#### 3.1 Definitions and Assumptions

The network model used in this research is described as follows:

- We assume a homogeneous network of wireless nodes, each operating at a fixed transmission range,  $R$ .
- We use the unit-disk graph model [13] according to which there exists a link between any two nodes  $i$  and  $j$  at  $(X_i, Y_i)$  and  $(X_j, Y_j)$  in the network as long as the Euclidean distance  $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$  is less than or equal to the transmission range per node.
- The set of neighbors of a node  $i$ ,  $Neighbors(i)$ , comprises of nodes that are connected to vertex  $i$  in the unit-disk graph model.
- A node learns about its own location through location service schemes such as the Global Positioning System (GPS) [14] or any other scheme (e.g. [15]).
- A node learns the location and mobility parameters (velocity and direction of movement – measured as the angle subscribed with respect to the positive X-axis) of its neighbor nodes through the beacon messages periodically broadcast by their nodes in the neighborhood.

#### 3.2 Auxiliary Variables and Functions

The SN-CDS algorithm uses the following auxiliary variables and functions:

- *SN-CDS-Node-List* (initialized to  $\Phi$ ) – consists of nodes in the strong neighborhood based CDS.
- *Covered-Nodes-List* (initialized to  $\Phi$ ) – consists of nodes that are either in the *SN-CDS-Node-List* or is covered by (i.e. is a neighbor of) a node in the *SN-CDS-Node-List*.
- *Uncovered-Nodes-List* (initialized to the Vertex set of the input static graph) – composed of nodes that are not covered by any node in the *SN-CDS-Node-List*.
- *Priority-Queue* (initialized to  $\Phi$ ) – consists of nodes that are in the *Covered-Nodes-List* and not in the *SN-CDS-Node-List*. The nodes in the priority queue are probable candidates for inclusion to the *SN-CDS-Node-List* and are considered in the decreasing order of the number of uncovered strong neighbors. A dequeue operation returns the node with the largest number of uncovered strong neighbors; if there is a tie, then a node is randomly chosen from among the contending nodes.
- *MaxUncoveredStrongNeighbors* – the maximum value for the number of uncovered strong neighbors of any node in the network at the beginning of the algorithm; initialized to  $-\infty$  to start with and is computed while determining the strong neighbors of the nodes in the network.
- *UncoveredStrongNeighbors(u)* – the set of all strong neighbors of node  $u$  that are not yet covered. Initially,  $\forall u, uncoveredStrongNeighbors(u) = \Phi$ .
- *Start Node* – the first node to be included into the SN-CDS. The *Start Node* is the node that has the maximum number of uncovered strong neighbors, to start with. Any tie is broken arbitrarily.

### 3.3 Description of the SN-CDS Algorithm

The SN-CDS construction algorithm (pseudo code in Figure 3) primarily works as follows: The algorithm inputs a snapshot of the network (referred to as the static graph  $G = (V, E)$  with vertex set  $V$  and edge set  $E$ ) at the time instant during which we want to determine the SN-CDS. The algorithm also inputs the Threshold Neighborhood Distance Ratio ( $TNDR$ ) and transmission range ( $R$ ) for the network.

When the *Start Node* is added to the *SN-CDS-Node-List*, all of its strong neighbors are said to be covered; these nodes are removed from the *Uncovered-Nodes-List* and added to the *Covered-Nodes-List* and to the *Priority-Queue*. If both the *Uncovered-Nodes-List* and the *Priority-Queue* are not empty, we dequeue the *Priority-Queue* to extract a node  $s$  that is not yet in the *SN-CDS-Node-List* and has the largest number of uncovered strong neighbor nodes. All the uncovered strong neighbor nodes of  $s$  are now removed from the *Uncovered-Nodes-List* and added to the *Covered-Nodes-List* as well as to the *Priority-Queue*. The number of uncovered strong neighbors of each node in the network is then updated based on the additional node coverage obtained during the iteration and accordingly, the *Priority-Queue* is re-sorted in the decreasing order of the number of uncovered strong neighbors of the nodes in the queue. The above procedure is repeated for several iterations until the *Uncovered-Nodes-List* becomes empty or the *Priority-Queue* becomes empty.

**Input:** Static Network Graph  $G = (V, E)$ , where  $V$  is the set of vertices and  $E$  is the set of edges  
 $TNDR$  and  $R$  (the transmission range of the nodes in the network)

**Output:** *SN-CDS-Node-List* // contains list of nodes part of the strong neighborhood based CDS.

**Begin** SN-CDS Algorithm

```

maxUncoveredStrongNeighbors = Compute-Strong-Neighborhood( $V$ ) // takes  $O(|V|^2)$  time
Start Node = Choose-Start-Node( $V$ , maxUncoveredStrongNeighbors) // takes  $O(|V|)$  time
Uncovered-Nodes-List = Uncovered-Nodes-List - {Start Node}
Covered-Nodes-List = Covered-Nodes-List  $\cup$  {Start Node}
Priority-Queue = Priority-Queue  $\cup$  {Start Node}

```

$$\forall u \in V, \text{uncoveredStrongNeighbors}(u) = \{v \mid v \in SN_u\}$$

**while** (*Uncovered-Nodes-List*  $\neq \Phi$  and *Priority-Queue*  $\neq \Phi$ ) **do** //  $O(|E|+|V|^2 \log |V|)$  time

```

node  $s$  = Dequeue(Priority-Queue)
if (uncoveredStrongNeighbors( $s$ ) =  $\Phi$ ) then
    return NULL; // the underlying network is not connected
end if

```

*SN-CDS-Node-List* = *SN-CDS-Node-List*  $\cup$   $\{s\}$

```

for all node  $u \in$  Neighbors( $s$ ) do
    if ( $u \in$  Uncovered-Nodes-List) then
        Uncovered-Nodes-List = Uncovered-Nodes-List -  $\{u\}$ 
        Covered-Nodes-List = Covered-Nodes-List  $\cup$   $\{u\}$ 
        Priority-Queue = Priority-Queue  $\cup$   $\{u\}$ 
    end if

```

**end for**

$$\forall u \in V, \text{uncoveredStrongNeighbors}(u) = \{v \mid v \in SN_u, v \in \text{Uncovered-Nodes-List} \}$$

Re-sort the entries in *Priority-Queue* in the decreasing order of the number of uncovered strong neighbors for the nodes in the queue

**end while**

**if** (*Uncovered-Nodes-List*  $\neq \Phi$  and *Priority-Queue* =  $\Phi$ ) **then**  
     **return** NULL; // the underlying network is not connected  
**end if**

**return** *SN-CDS-Node-List*

**End** SN-CDS Algorithm

**Figure 1:** Pseudo Code for the SN-CDS Construction Algorithm

Note that during a particular iteration, if the node  $s$  extracted from the *Priority-Queue* has all its strong neighbor nodes already covered, then it implies that all the other nodes, if any, in the *Priority-Queue* also have “zero” uncovered strong neighbor nodes. However, we have not yet broken from the while loop (i.e. the *Uncovered-Nodes-List* is not yet empty), indicating that the underlying network based on the strong neighborhood of the nodes is not connected and hence the algorithm returns NULL (i.e. a SN-CDS for the entire network does not exist). Also, even after exiting from the while loop, if the *Priority-Queue* becomes empty and the *Uncovered-Nodes-List* has at least one node, then the underlying network is considered to be disconnected (based on the strong neighborhood of the nodes) and the algorithm returns NULL. If the underlying network is connected based on the strong neighborhood of the nodes, then the algorithm does not return NULL and returns the *SN-CDS-Node-List* after all the nodes in the network are included to the *Covered-Nodes-List*.

### 3.4 Algorithm to Construct the Minimum Connected Dominating Set (MCDS)

The algorithm to construct the minimum connected dominating set (MCDS) [4] is quite similar to that of the SN-CDS – the main difference being that the MCDS algorithm is run on a graph topology determined based on the open neighborhood (fixed transmission range) of the nodes; whereas, the SN-CDS algorithm is run on a graph topology based on the strong neighborhood of the nodes. Accordingly, the MCDS and SN-CDS algorithms prefer to include nodes with the largest number of uncovered neighbors in their open and strong neighborhoods respectively. The above principle is followed even in the subsequent iterations of the two algorithms while selecting a node for inclusion (among the covered nodes) into the CDS.

### 3.5 Analysis of the Run-time Complexity of the SN-CDS and MCDS Algorithms

The initial two steps, viz., computing the strong neighborhood of each node and selecting the Start Node, can be conducted in  $O(|V|^2)$  and  $O(|V|)$  time respectively. The set of strong neighbors for each node can be determined by exploring every pair of nodes in the network ( $O(|V|^2)$  time). The Start Node – the node with the largest number of uncovered strong neighbors – can be selected in  $O(|V|)$  time by merely traversing the vertex set.

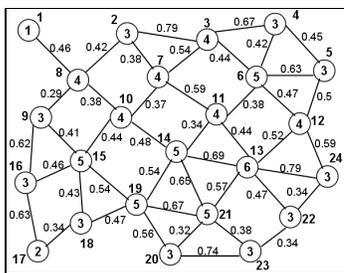
We assume the *Priority-Queue* used in the algorithm is implemented as a binary heap, as is also done in our simulations. The nodes in the *Priority-Queue* need to be always arranged in the decreasing order of the number of uncovered neighbors. A careful analysis of the pseudo code

of the SN-CDS algorithm (Figure 1) indicates that all the  $O(|E|)$  edges in the underlying network are to be explored for inclusion into the CDS. Every time we traverse an edge connecting a covered node to an uncovered node, we include the uncovered node into the *Priority-Queue* as well as in the *Covered-Nodes-List* and remove it from the *Uncovered-Nodes-List*. This would happen exactly  $|V|-1$  times if the underlying network is connected, as the whole algorithm is about moving the  $|V|-1$  nodes (i.e., all nodes other than the *Start Node*) from the *Uncovered-Nodes-List* to the *Covered-Nodes-List*. Each such change would require rearrangement of the *Priority-queue*, and this could be done in  $O(\log|V|)$  time. As there would be  $|V|-1$  such changes to the contents of the *Priority-Queue*, this particular aspect of the algorithm would require  $O(|V|\log|V|)$  time.

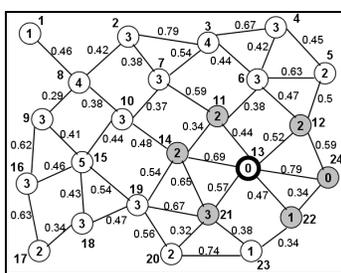
Due to the above such possible changes, we are required to re-sort the entries in the *Priority-Queue* at the end of each iteration. A re-sort of the *Priority-Queue* of  $|V|$  entries can be efficiently conducted in  $O(|V|*\log|V|)$  time, if the *Priority-Queue* is implemented as a binary heap. The loop (outer while loop in Figure 1) could run for at most  $|V|$  times if every node in the graph ends being part of the CDS. Hence, the time complexity to re-sort the *Priority-Queue* would be  $|V|*O(|V|*\log|V|) = O(|V|^2*\log|V|)$ . Summing up, the overall time-complexity of the SN-CDS algorithm is  $O(|V|^2 + |E| + |V|\log|V| + |V|^2*\log|V|) = O(|E| + |V|^2*\log|V|)$ . Since,  $|E| = O(|V|^2)$ , we could simply say that the time complexity of the SN-CDS algorithm is  $O(|V|^2*\log|V|)$ . Due to the similarity of the SN-CDS and MCDS algorithms (as explained in Section 3.4), the run-time complexity is the same for both of them. The only difference being the algorithm is executed on the open neighborhood, rather than the strong neighborhood.

### 3.6 Examples to Illustrate the Construction of the SN-CDS and MCDS

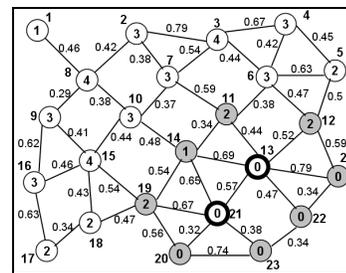
Figures 2 and 3 respectively illustrate the construction of the MCDS and SN-CDS through graph snapshots capturing the initial few iterations and the final iteration of the appropriate CDS construction algorithms. Figure 2.1 illustrates the initial unit disk graph representing the open neighborhood for the execution of the MCDS algorithm. In the case of SN-CDS (Figure 3), we apply a TNDR of 0.5 on the initial graph of Figure 2.1, and derive the graph of Figure 3.1 representing the strong neighborhood. The circles represent nodes, and the integer outside the circle for a node represents the Node ID. The real number on an edge indicates the EDR value for the edge. The integer inside a circle of Figure 2 indicates the number of uncovered neighbors of the node in the open neighborhood. Whereas, the integer inside a circle of Figure 3 indicates the number of uncovered strong neighbors of the node (determined after the application of the TNDR = 0.5 criterion). The CDS nodes are shown in bold circles; the covered (but non-CDS) nodes are shown in shaded circles; the circles representing uncovered nodes are neither made bold nor shaded.



**Figure 2.1:** Initial Network (based on fixed transmission range)



**Figure 2.2:** Iteration # 1



**Figure 2.3:** Iteration # 2

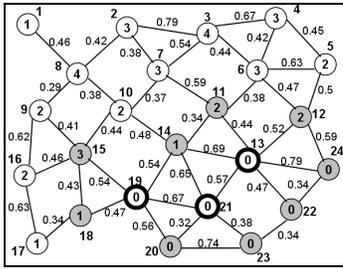


Figure 2.4: Iteration # 3

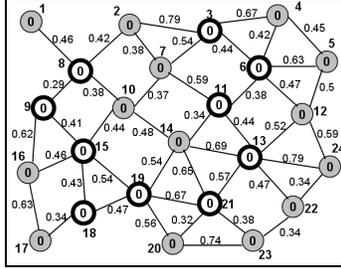


Figure 2.5: Final MCDS (at the end of Iteration # 10)

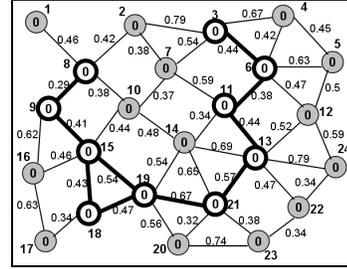


Figure 2.6: Edge List of the Final MCDS

Figure 2: Example to Illustrate the Construction of the Minimum Connected Dominating Sets (MCDS)

The number of iterations of both the MCDS and SN-CDS algorithms of Figures 2 and 3 directly correspond to the number of nodes that are part of their corresponding connected dominating sets. We observe that the SN-CDS has slightly more nodes and edges (12 nodes and 11 edges) compared to the MCDS (10 nodes and 10 edges). The strength of the SN-CDS lies in the selection of the edges that have an  $EDR \leq TNR$ . Even though this necessitates slightly more nodes and edges as part of the CDS, we observe that the SN-CDS sustains for a significantly longer lifetime compared to that of the MCDS, as also evidenced in our simulations.

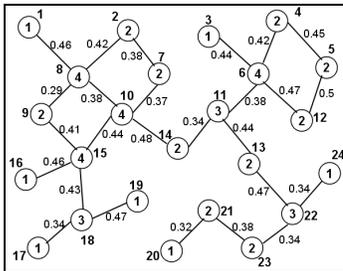


Figure 3.1: Initial Network (based on Strong Neighborhood  $TNR=0.5$  applied on Figure 2.1)

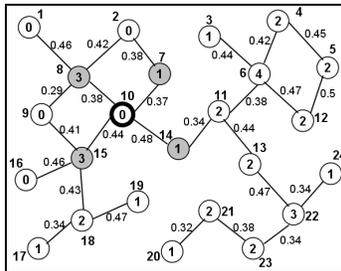


Figure 3.2: Iteration # 1

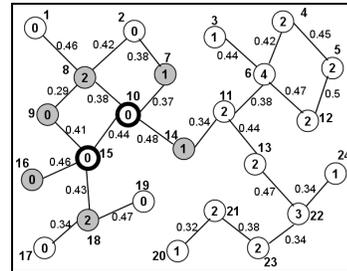


Figure 3.3: Iteration # 2

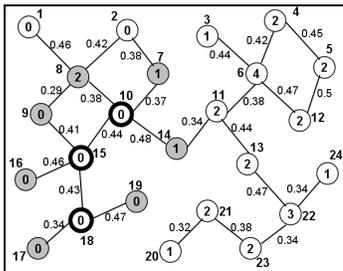


Figure 3.4: Iteration # 3

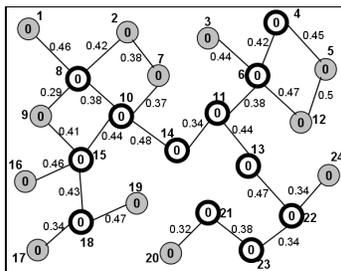


Figure 3.5: Final SN-CDS (at the end of Iteration # 12)

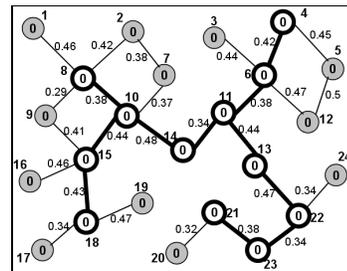


Figure 3.6: Edge List of the Final SN-CDS

Figure 3: Example to Illustrate Construction of Strong Neighborhood-based CDS (SN-CDS)

#### 4. ALGORITHM TO CONSTRUCT ID-BASED CONNECTED DOMINATING SET

The ID-based CDS algorithm prefers to include nodes in the decreasing order of their node IDs (i.e. nodes with larger IDs are given preference for inclusion) as long as a node has at least one

uncovered neighbor node. Figure 4 shows the pseudo code of an algorithm that outputs a CDS-Node-List comprising of nodes that are part of the ID-CDS by taking as inputs a static network graph representing a snapshot of the MANET at any particular time instant and a starting vertex  $s$ , the vertex with the larger node ID.

---

**Input:** Graph  $G = (V, E)$ ;  $V$  – vertex set,  $E$  – edge set  
 Start vertex,  $s$  – vertex with the largest ID in  $V$

**Auxiliary Variables and Functions:**  
 $CDS\text{-}Node\text{-}List$ ,  $Covered\text{-}Nodes\text{-}List$ ,  $Neighbors(v)$  for every  $v$  in  $V$

**Output:**  $CDS\text{-}Node\text{-}List$

**Initialization:**  $Covered\text{-}Nodes\text{-}List = \{s\}$ ;  $CDS\text{-}Node\text{-}List = \Phi$

**Begin** Construction of  $ID\text{-}CDS$

**while** ( $|Covered\text{-}Nodes\text{-}List| < |V|$ ) **do**

    Select a vertex  $r \in Covered\text{-}Nodes\text{-}List$  and  $r \notin CDS\text{-}Node\text{-}List$  such that  $r$  has the largest ID and has at least one neighbor that is not in  $Covered\text{-}Nodes\text{-}List$

$CDS\text{-}Node\text{-}List = CDS\text{-}Node\text{-}List \cup \{r\}$

**for** all  $u \in Neighbors(r)$  and  $u \notin Covered\text{-}Nodes\text{-}List$

$Covered\text{-}Nodes\text{-}List = Covered\text{-}Nodes\text{-}List \cup \{u\}$

**end for**

**end while**

**return**  $CDS\text{-}Node\text{-}List$

**End** Construction of  $ID\text{-}CDS$

---

**Figure 4:** Pseudo Code for the ID-CDS Algorithm

The ID-CDS algorithm was as follows: The Start vertex is the first node to be included in the CDS-Node-List; and as a result, all its neighbor nodes are said to be covered and are added to the Covered-Nodes-List. In the subsequent iterations, we select the node with the largest ID among the nodes that meets the following two criteria: (1) the node cannot be already in the CDS-Node-List and (2) the node should have at least one uncovered neighbor. The above process is repeated until all the nodes in the network are included in the Covered-Nodes-List, and the nodes selected to the CDS-Node-List by then constitute the ID-CDS.

#### 4.1 Analysis of the Run-time Complexity of the ID-CDS Algorithm

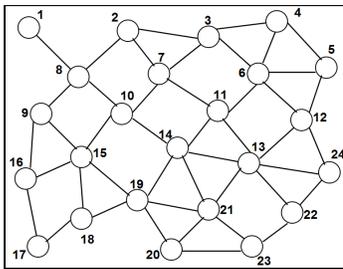
The  $Covered\text{-}Nodes\text{-}List$  is implemented as a *Priority-Queue* (binary heap) in our simulations. Accordingly, the run-time complexity of the ID-CDS algorithm is given by  $O(|E| + |V| \cdot \log|V|)$ ; there could be at most  $|V|$  iterations run to form the CDS, and as part of these iterations we would be traversing all the edges in the network exactly once. The  $Covered\text{-}Nodes\text{-}List$  will be updated  $O(|V|)$  times with membership about nodes that are newly covered, and each time a node is added to the  $Covered\text{-}Nodes\text{-}List$ , the list needs to be rearranged in the decreasing order of node IDs. Further, there would be at most  $O(|V|)$  times that the  $Covered\text{-}Nodes\text{-}List$  will be dequeued to populate the  $CDS\text{-}Node\text{-}List$ . After each such dequeue operation, we would again require rearranging the  $Covered\text{-}Nodes\text{-}List$  in the decreasing of their node IDs. Hence, there would be  $2 \cdot O(|V|)$  instances that would require rearrangement of the  $Covered\text{-}Nodes\text{-}List$  (a *Priority-Queue*), and when implemented as a binary heap, each such rearrangement can be completed in  $O(\log|V|)$  time.

Note that the ID-CDS algorithm is theoretically more faster compared to the MCDS and SN-CDS algorithms: because, each time a covered node is added to the CDS-Node-List and new nodes are brought into the  $Covered\text{-}Nodes\text{-}List$  (*Priority-Queue*), the MCDS and SN-CDS algorithms require us to first re-compute the number of uncovered neighbors of every covered

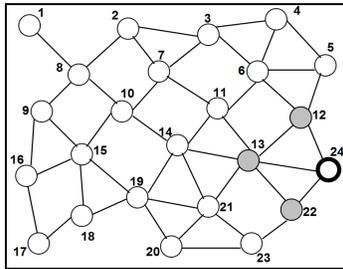
node in the *Priority-Queue* and then use that information to re-sort the entries in the *Priority-Queue*. On the other hand, the *Priority-Queue* for an ID-CDS algorithm can be rearranged (based on the node IDs) as soon as a new node is added to the *Priority-Queue*; there is no need to re-compute the number of uncovered neighbors of all the nodes in the network before rearranging the *Priority-Queue*.

#### 4.2 Randomization of the Node IDs for the ID-CDS Algorithm

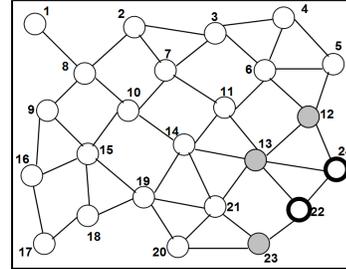
In our simulations, to be fair to all nodes in the network, the node IDs were randomized before every reconfiguration of the ID-CDS. In other words, before we reinitiate the ID-CDS algorithm after the failure of the currently known ID-CDS, we randomly redistribute the node IDs and ran the ID-CDS algorithm based on the network graph formed based on the new node IDs. In each such reconfiguration, we maintain the appropriate mapping between the new randomized node IDs and the original node IDs (that will never change throughout the simulation). While the original node IDs are used to determine the location and mobility of the nodes (from the mobility profile) at any time instant; the randomized node IDs are used to decide the inclusion of nodes from the Covered-Nodes-List to the CDS-Node-List as part of the ID-CDS algorithm. Note that the neighbors of a node that is recently added to the CDS-Node-List are considered covered based on their original node IDs; however they are listed in the Covered-Nodes-List based on their randomized node IDs.



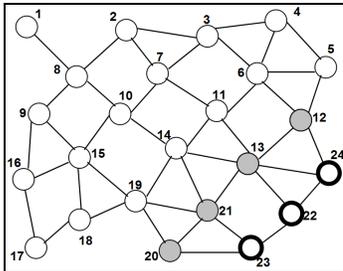
**Figure 5.1:** Initial Network (based on Strong Neighborhood TNR=0.5 applied on Figure 2.1)



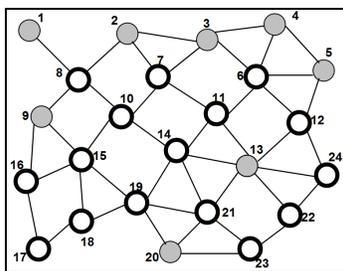
**Figure 5.2:** Iteration # 1



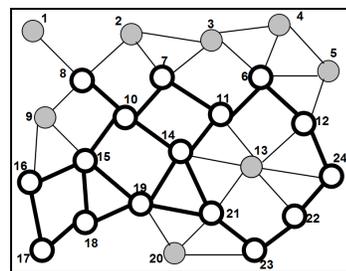
**Figure 5.3:** Iteration # 2



**Figure 5.4:** Iteration # 3



**Figure 5.5:** Final ID-CDS (at the end of Iteration # 16)



**Figure 5.6:** Edge List of the Final ID-CDS

**Figure 5:** Example to Illustrate Construction of ID-based Connected Dominating Set (ID-CDS)

#### 4.3 Example to Illustrate the Construction of an ID-CDS

Figure 5 illustrates the construction of the ID-CDS on the same network graph topology of Figure 2 that we had used for the MCDS, and from which the SN-CDS topology graph of Figure 3 was also derived. The circles represent nodes and the integer outside a circle represents the node ID. We intend to represent the graph topology with only values that are needed to illustrate the construction of the ID-CDS. Since the ID-CDS algorithm does not require the re-

computation of the number of uncovered neighbors of a node at the end of each iteration (the ID-CDS algorithm includes a covered node into the CDS-Node-List as long as the node has at least one uncovered neighbor; the exact number of uncovered neighbors is not needed to make a decision), we do not show the number of uncovered neighbors inside the circles of the nodes. Also, since the EDR values do not influence CDS node selection, we do not show these values associated with the graph edges. As in the case of Figures 2 and 3, the CDS nodes are shown in bold circles; the covered (but non-CDS) nodes are shown in shaded circles; the circles representing uncovered nodes are neither made bold nor shaded.

As can be noticed from Figure 5, the ID-CDS algorithm consumed 16 iterations (16 CDS nodes and 21 edges) on the network graph of Figure 2 in which the MCDS and SN-CDS algorithms required only 10 iterations (10 CDS nodes and 10 edges) and 12 iterations (12 CDS nodes and 11 edges) respectively. The presence of more nodes and edges in the ID-CDS gives it the perceived stability, as observed in our simulations, compared to a MCDS. However, the total negligence of the number of uncovered neighbors of a node to decide on the inclusion of a covered node into the CDS-Node-List contributes to the significantly high value for the ID-CDS Node Size and Edge Size, also leading to much higher energy consumption. For example in Figure 5.3 (Iteration 2), node 22 was selected ahead of node 13 for inclusion into the ID-CDS (because 22 was a higher ID than 13) even though node 13 had three uncovered neighbors and node 22 had only uncovered neighbor at the beginning of Iteration-2. Eventually, at the end of Iteration-16, we observe that node 13 did not get included at all into the ID-CDS even though it had 6 neighbor nodes in the initial network graph, whereas node 17 that had only 2 neighbors in the initial network graph gets included in the ID-CDS. As observed in our simulations and also in the examples of Figures 3 and 5, the SN-CDS (with much fewer CDS Node Size and Edge Size) is more stable than an ID-CDS, and there is no need to include a plethora of nodes and edges for the sake of stability.

## 5. SIMULATIONS

We conduct our simulations in a discrete-event simulator developed in Java. The network dimensions are 1000m x 1000m. The fixed transmission range per node is 250m. We vary the network density by conducting simulations with 50, 100 and 150 nodes to represent networks of low, moderate and high density respectively that also correspond to an average neighborhood size of 10, 20 and 30 nodes for a transmission range of 250m. We use the Random Waypoint model [13] as the mobility model for our simulations. According to this model, each node starts from an arbitrary location, and chooses to move to a randomly chosen destination location (within the network boundary) at a velocity uniform-randomly chosen from the range  $[0, \dots, v_{max}]$ . After reaching the targeted destination location, the node continues to move by randomly choosing another location to move, with a velocity again uniform-randomly chosen from the above range. The new location and velocity values are chosen independent of the mobility history. Each node continues to move like this for the simulation time of 1000 seconds. The movement of one node is independent of the other nodes in the network. The  $v_{max}$  values chosen for our simulations are 5, 25 and 50 m/s representing scenarios of low, moderate and high mobility respectively. All of the simulation results presented in Figures 6 through 13 are average values of the performance metrics obtained by running the simulations with 5 different mobility profiles generated for each combination of network density and node mobility.

### 5.1 Simulation Methodology

Our simulation methodology is as follows: We construct snapshots (static graphs) of the network topology for every 0.25 seconds, starting from time 0 to the simulation time of 1000 seconds. If a CDS is not known at a particular time instant, we run the appropriate CDS construction algorithm on the network snapshot. The CDS determined at a particular time instant is used in the subsequent time instants until the CDS ceases to exist. For a CDS to be

considered to exist at a particular time instant, two conditions have to hold good: (i) All the CDS nodes have to stay connected – i.e. reachable from one another directly or through multi-hop paths; and (ii) Every non-CDS node should have at least one CDS node as its neighbor. If a CDS ceases to exist at a particular time instant, we again run the appropriate CDS construction algorithm and continue to use the new CDS as explained above. This procedure is continued for the duration of the simulation time.

### 5.2 Connectivity of SN-CDS

We define the connectivity of a CDS as the ratio of the number of time instants there exists a CDS for the underlying network topology divided by the total number of time instants considered over the duration of the simulation time. Since SN-CDS works on a reduced set of edges (due to the imposition of the strong neighborhood criterion), the connectivity of SN-CDS for a given unit disk graph (defined according to the fixed transmission range corresponding to the open neighborhood) is heavily dependent on the TNDR value used. The larger the value of TNDR, the larger is the connectivity of SN-CDS, as the number of strong neighbors of a node increases with increase in TNDR. Obviously, if TNDR = 1.0, SN-CDS reverts to MCDS. Figure 6 illustrates the connectivity of SN-CDS for different network density conditions, observed for  $v_{max}$  values of 5 and 50 m/s. As observed in these figures, the connectivity of SN-CDS matches close to that of the MCDS only when TNDR = 0.9. Hence, in order to be fair to MCDS in our comparisons, and also evaluate the performance of SN-CDS in the presence of appreciable CDS connectivity, we will only report the performance of SN-CDS for TNDR = 0.9 for the rest of the metrics evaluated in this section.

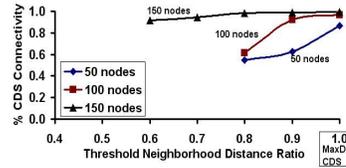


Figure 6.1:  $v_{max} = 5$  m/s

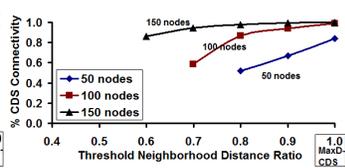


Figure 6.2:  $v_{max} = 25$  m/s

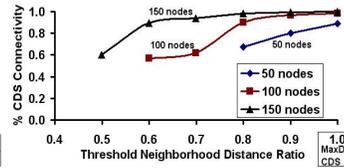


Figure 6.3:  $v_{max} = 50$  m/s

Figure 6: SN-CDS Connectivity vs. Threshold Neighborhood Distance Ratio

### 5.3 Performance Metrics

We measure the following performance metrics in our simulations:

- **Effective CDS Lifetime:** We measure the duration of time each instance of a CDS actually exists, and compute the average of the CDS lifetimes observed across the entire simulation time period (for all the mobility profiles representing a particular combination of network density and node mobility). We refer to these average values of the actually observed CDS lifetimes as the Absolute CDS Lifetime. However, since the connectivity of the different CDSs are less than 1.0 for most of the scenarios evaluated, we introduce a better representative metric called the Effective CDS Lifetime – defined as the product of the Absolute CDS Lifetime and the CDS Connectivity – to effectively capture the stability of a CDS taking into consideration the chances of determining the CDS. For example, if an SN-CDS has an average absolute lifetime of 10 seconds; but its connectivity is only 0.8 – it is more prudent to use the effective lifetime of  $10 \times 0.8 = 8$  seconds as a more realistic measure of the stability (lifetime) of the CDS.
- **CDS Node Size:** This is a time-averaged value for the number of nodes that are part of the CDS used for every time instant over the entire simulation. For example, if there exists a sequence of three CDS of size 30 nodes, 40 nodes and 20 nodes in a network for 6, 10 and 4 seconds respectively, then the average CDS Node Size for a total of 20 seconds is  $(30 \times 6 + 40 \times 10 + 20 \times 4) / (6 + 10 + 4) = 33.0$  and not simply the average of 30, 40 and 20 nodes = 30.

- **CDS Edge Size:** This is a time-averaged value of the number of edges that exist between any two CDS nodes, measured for every time instant a CDS existed during the simulation time period.
- **Hop Count per  $s-d$  Path:** The hop count per source-destination ( $s-d$ ) path is the average of the number of hops (edges) on the shortest path between any two nodes on the CDS-induced sub graphs, considered over the entire simulation time and all the 15  $s-d$  pairs. Note that the intermediate nodes, if any are needed, for such  $s-d$  paths must be the CDS nodes. To determine the hop count of the shortest path for every  $s-d$  pair, we run the Breadth First Search (BFS) algorithm [2], starting from the source node  $s$ , on the CDS-induced sub graphs containing edges between any two CDS nodes and between a CDS node and a regular non-CDS node.
- **CDS Diameter:** The CDS diameter is the maximum of the minimum number of hops between any two CDS nodes in the CDS-induced sub graphs containing only the edges between any two CDS nodes. To determine the CDS diameter, we run the BFS algorithm starting from an arbitrarily selected CDS node, and reach every other CDS node on the shortest minimum-hop path. We run five instances of BFS (each time starting from a different arbitrarily selected CDS node) on every CDS-induced graph and compute the overall average diameter (the values reported in Figure 12) across the entire simulation period.
- **CDS Energy Index:** The CDS Energy Index is a measure of the potential energy consumption that will be incurred if a CDS is used for network-wide broadcasting. As all the CDS edges will be used in a network-wide broadcast and energy consumption in ad hoc networks is modeled on the fourth power of the physical Euclidean distance for an edge [14], we compute the CDS Energy Index as the product of the average CDS Edge Size and the average Edge Distance Ratio (EDR). The average EDR is evaluated by computing the average of all the EDRs of the CDS edges (i.e., edges between any two CDS nodes) that are part of the different CDSs used across the entire simulation time period. The lower the value of the CDS Energy Index, the lower is the energy consumption that is likely to be incurred if the CDS is used for network-wide broadcasting.

#### 5.4 Effective CDS Lifetime

We observe the SN-CDS to have the largest effective lifetime compared to the MCDS and ID-CDS. For a given network density, the difference between the effective lifetime of the SN-CDS and that of the MCDS and ID-CDS decreases with increase in node mobility. This is attributable to the dynamic changes in the network topology with increase in node mobility, and the effectiveness of the strong neighborhood criterion in determining a stable CDS gets marginally diminished in networks of moderate and high mobility. However still, the effective lifetime of the SN-CDS is as large as 250% and 185% more than that of the MCDS in networks of low mobility and moderate/high mobility respectively. The poor stability of the MCDS can be attributed to the lack of scope for robustness of the CDS-induced sub graph comprising of only fewer CDS nodes and edges connecting them. As observed in Sections 5.5 and 5.6, the MCDS incurs the lowest values for CDS Node Size and CDS Edge Size. Even though this could contribute to reducing the energy consumption overheard during network-wide broadcasting, the fact that the EDRs of the CDS edges are above 0.8, accompanied by the presence of very few CDS edges, contributes to the poor stability of the MCDS. Even the failure of one or two MCDS edges could lead to the disintegration of the whole MCDS. On the other hand, even though the SN-CDS does not involve substantially more edges between the CDS nodes (unlike an ID-CDS that has significantly more edges), the SN-CDS edges are part of the strong neighborhood (at least at the time of discovery of the SN-CDS) and hence are likely to exist for a longer time. The effective lifetime of the SN-CDS is about 25-35% and 15-25% more than that of the ID-CDS in networks of low and moderate/high node mobility respectively. However,

the stability of the ID-CDS comes at the cost of including a significantly large number of nodes as part of the CDS (explained more in Section 5.5).

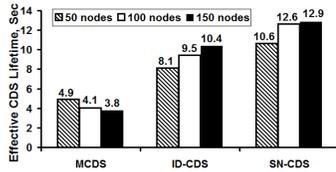


Figure 7.1:  $v_{max} = 5$  m/s

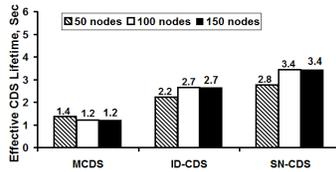


Figure 7.2:  $v_{max} = 25$  m/s

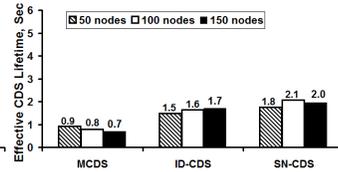


Figure 7.3:  $v_{max} = 50$  m/s

Figure 7: Effective CDS Lifetime

### 5.5 CDS Node Size

As expected, the MCDS incurs the lowest value for the CDS Node Size. The MCDS is constructed with the objective of including only the least number of nodes into the CDS to cover the rest of the nodes in the network. The SN-CDS incurs about 20% more nodes compared to the MCDS. This is attributable to the primary consideration given to stability. With the imposition of TNDR (of value  $< 1.0$ ), the number of nodes that can be covered by a node in its stronger neighborhood is likely to be less than the number of nodes that can be covered by the node in its open neighborhood (TNDR = 1.0). However, the moderate increase in the number of CDS nodes and the relatively lower EDR value for the accompanying edges helps the SN-CDS to be much more stable, compared to the MCDS. The ID-CDS incurs the largest value for the CDS Node Size, about 40 – 70% more nodes compared to that of the MCDS. This could be attributed to the “zero” consideration given to the CDS Node Size at the time of formation of the ID-CDS. As observed in the example in Section 4.3, several nodes at the network periphery are vulnerable for inclusion into the ID-CDS, just because they have a larger node ID.

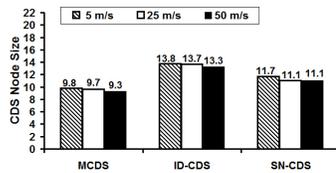


Figure 8.1: 50 nodes

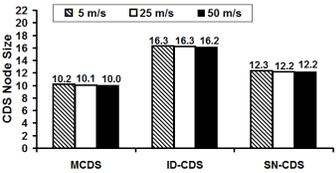


Figure 8.2: 100 nodes

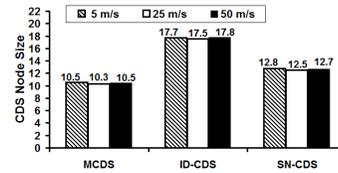
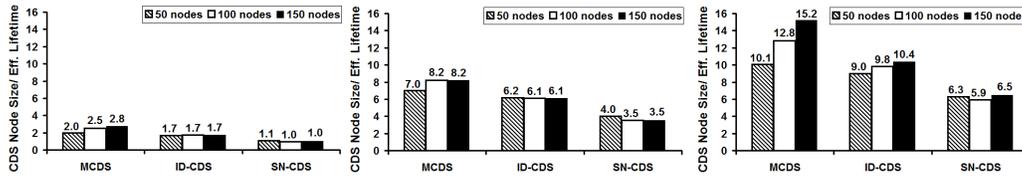


Figure 8.3: 150 nodes

Figure 8: CDS Node Size

### 5.6 CDS Node Size – Effective CDS Lifetime Tradeoff

Since we observe an inverse relationship between the CDS Node Size and the Effective CDS Lifetime, we are interested in identifying which CDS effectively balances the CDS Node Size to Lifetime tradeoff. The ratio has the CDS Node Size in the numerator and the Effective CDS Lifetime in the denominator. Since we prefer to incur a larger CDS lifetime and a lower CDS Node Size, the ratio is ideally expected to be closer to 0. In other words, the CDS that incurs the least value for the tradeoff ratio is said to be effectively balancing the CDS Node Size and the Lifetime. In this context, we observe (in Figure 9) that the SN-CDS incurs the lowest tradeoff ratio values for all combinations of network density and node mobility. The MCDS incurs the largest values for the tradeoff ratio, implying that even the CDS Node Size is lower, lower values for the CDS Lifetime shoots up the values for the ratio. The ID-CDS incurs a tradeoff ratio that is relatively lower, but much closer to the values observed for the MCDS. In the case of the ID-CDS, the significantly larger CDS Node Size contributes to the larger tradeoff ratio, even though a relatively larger value for the CDS Lifetime helps in preventing the tradeoff ratio from shooting up.


**Figure 9.1:**  $v_{max} = 5$  m/s

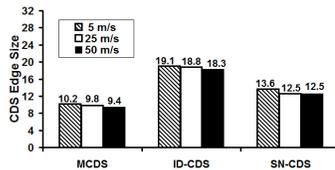
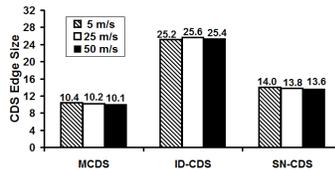
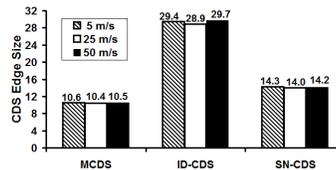
**Figure 9.2:**  $v_{max} = 25$  m/s

**Figure 9.3:**  $v_{max} = 50$  m/s

**Figure 9:** Tradeoff Ratio: Effective CDS Lifetime / CDS Node Size

### 5.7 CDS Edge Size

Since the MCDS uses the lowest possible number of CDS nodes to cover the rest of the nodes in the network, the MCDS nodes are highly likely to be far away from each other, as evidenced in the EDR ratio (average EDR of MCDS edges is above 0.8). As a result, there are likely to be very few edges connecting the MCDS nodes, leading to the lowest value for the CDS Edge Size. The SN-CDS incurs about 35% more edges (compared to the MCDS) and this could be attributed to the 20% increase in the CDS Node Size. The ID-CDS incurs the largest values for the CDS Edge Size, about 80% (low density) to 180% (high density) more than that of the MCDS. The significantly larger CDS Edge Size contributes to a certain extent in the robustness of the ID-CDS to node movements, as the ID-CDS incurs a relatively larger lifetime than that of a MCDS. However, the SN-CDS incurs a much larger lifetime compared to the ID-CDS by requiring only at most 35% more edges than that of a MCDS. The reason for the relatively poor stability of ID-CDS (compared to the SN-CDS that incurs a lower CDS Edge Size) is the likely presence of several peripheral nodes (nodes that are in the network boundary) as part of the ID-CDS; these nodes are bound to break off from the ID-CDS even with the failure of one or more edges. So, even though there might be several CDS edges at the core of the ID-CDS, the presence of nodes with fewer edges at the periphery of the ID-CDS contributes to its lower lifetime, compared to the SN-CDS.


**Figure 10.1:** 50 nodes

**Figure 10.2:** 100 nodes

**Figure 10.3:** 150 nodes

**Figure 10:** CDS Edge Size

### 5.8 Hop Count per $s$ - $d$ Path

We determine the average hop count per  $s$ - $d$  path on a CDS-induced sub graph comprising of all the nodes in the network, and edges between any two CDS nodes and those between a CDS node and non-CDS node, all based on the open neighborhood (even in the case of SN-CDS, since the EDR of the edges is not guaranteed to be less than TNR on network snapshots other than those in which the SN-CDS was determined). We conduct simulations with randomly selected 15  $s$ - $d$  pairs, for each of which we run the BFS algorithm on the CDS-induced sub graphs during every simulation time instant for which there is a CDS available, and then average out the results across all the time instants and  $s$ - $d$  pairs. We observe all the three connected dominating sets (MCDS, ID-CDS and SN-CDS) to sustain about the same hop count per  $s$ - $d$  path, with the MCDS recording the lowest and the ID-CDS recording the highest (in terms of the absolute magnitude) for most of the cases. We had expected a lower hop count per

path for the ID-CDS as it involves several CDS edges and there would be lot of best choices for the BFS algorithm to find close to a minimum hop path for any two nodes  $s$  and  $d$ . This illustrates that the ID-CDS has a lean backbone (core center of the ID-CDS) rather a denser one that one would expect given the significantly larger CDS Edge Size compared to a MCDS or SN-CDS. However, as explained earlier, a considerable portion of these CDS edges involve the peripheral nodes of the ID-CDS.

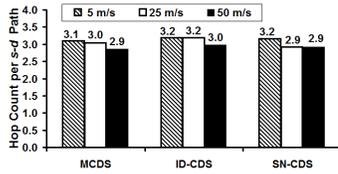


Figure 11.1: 50 nodes

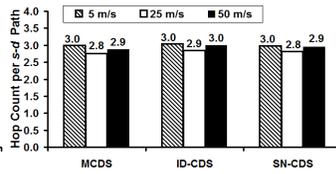


Figure 11.2: 100 nodes

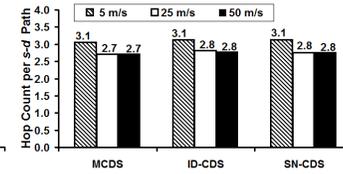


Figure 11.3: 150 nodes

Figure 11: Hop Count per Source-Destination ( $s-d$ ) Path

For a given network density, we observe the hop count per  $s-d$  path to be independent of the node mobility. However, for a given mobility, we observe the hop count per  $s-d$  path for MCDS and ID-CDS to decrease by about 5-10%, and this could be attributed to the presence of more constituent nodes and edges in the corresponding CDS at higher network density, virtually increasing the number of  $s-d$  paths from which BFS can choose the shortest one. SN-CDS shows the least sensitivity in the hop count per  $s-d$  path with increase in network density. This could be attributed to the imposition of the strong neighborhood criterion on the network graphs and the resulting possibility of not a substantial increase in the number of alternate paths between  $s-d$  pairs.

### 5.9 CDS Diameter

The CDS diameter is a measure of the maximum delay incurred for a message to reach all the nodes in the network when broadcast through the CDS-induced sub graph, comprising of edges that exist between the CDS nodes at the particular time instant. We ran the BFS algorithm starting from a randomly chosen starting vertex (repeated the execution of the algorithm with five different starting vertices) on the CDS-induced sub graph for the particular time instant, and counted the maximum depth (as a measure of the CDS diameter) of the BFS tree rooted at the starting vertex. We ran the BFS algorithm (five instances, each instance with a different starting vertex) for every time instant for which a CDS existed across the entire simulation time period, and then averaged the results across all the time instants and starting vertices.

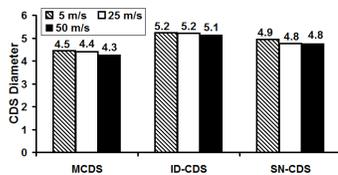


Figure 12.1: 50 nodes

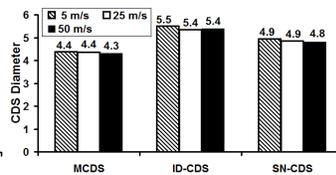


Figure 12.2: 100 nodes

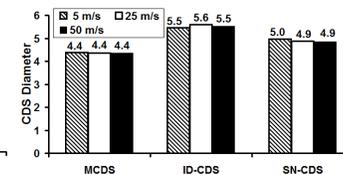


Figure 12.3: 150 nodes

Figure 12: CDS Diameter

The MCDS incurred the lowest CDS diameter for all combinations of network density and node mobility. We observe the CDS diameter to increase with increase in the number of CDS nodes – not proportionately though. The SN-CDS incurred about 20% more nodes compared to ID-CDS; but the SN-CDS diameter is only 10-12% more. Similarly, the ID-CDS has about 40-70% more nodes compared to that of the MCDS; however, the ID-CDS diameter is only 17-27%

more. For a given network density, we observe the CDS diameter to remain almost the same with variations in network density. For a given node mobility, while the MCDS and SN-CDS are almost insensitive to increase in network density, the ID-CDS incurs about 7-10% increase in the diameter with increase in network density, and this could be attributed to the 20-30% relative increase in the ID-CDS Node Size with increase in network density.

### 5.10 CDS Energy Index

The CDS Energy Index (results shown in Figure 13) is a measure of the energy consumption that could be possibly incurred when a message is broadcast on the CDS-induced sub graph consisting of the edges between the CDS nodes. For a given CDS (at a particular combination of network density and node mobility), we compute the average EDR of the CDS edges on the CDS-induced sub graphs spanning over all the time instants for which there exist a CDS of the particular type across the entire simulation time period. The CDS Energy Index is the product of the average EDR and the average CDS Edge Size. The reasoning behind such a computation model for the CDS Energy Index is that energy consumption for broadcasting on a sub graph depends on the physical Euclidean distance of the constituent edges and the number of edges. We observe the MCDS to incur the lowest CDS Energy Index, apparently due to the lower number of constituent CDS Edges. The Energy Index for the SN-CDS is only at most 25% more than that of the MCDS. Note that the effective lifetime of the SN-CDS can be as large as 250% more than that of the MCDS. Also, to be noted is that for a given node mobility, the Energy Index of the SN-CDS does not vary too much with variations in the network density, and vice-versa. All of these observations indicate that the SN-CDS can effectively balance the stability-energy consumption tradeoff incurring a significantly longer lifetime by accommodating a relatively fewer more nodes and edges and incurring only a very modest increase in the energy consumption. On the other hand, we observe the Energy Index of the ID-CDS to shoot up with increase in network density. The Energy Index of the ID-CDS is about 70% (low density) to 140% (high density) more than that of the MCDS. Note that the average EDR of the MCDS, SN-CDS and ID-CDS were about 0.82, 0.75 and 0.70 respectively. So, even though the ID-CDS incurs a lower EDR, the plethora of edges that are part of the ID-CDS contributes to its significantly high Energy Index. On the other hand, the MCDS and SN-CDS effectively tradeoff for their larger EDRs by incurring relatively very few edges.

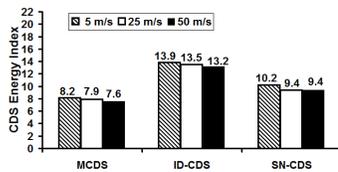


Figure 13.1: 50 nodes

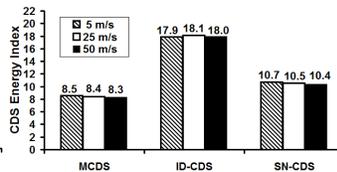


Figure 13.2: 100 nodes

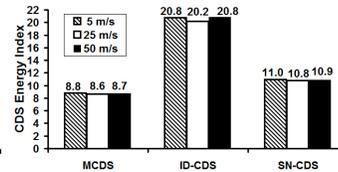


Figure 13.3: 150 nodes

Figure 13: CDS Energy Index

## 6. CONCLUSIONS AND FUTURE WORK

The high-level contribution of this paper is to demonstrate that the strong-neighborhood based connected dominating set (SN-CDS) can effectively balance the stability-CDS node size tradeoff, and perform well with respect to critical performance metrics like the CDS Lifetime (a measure of stability), Hop count per s-d path (a measure of delay in unicast transmissions), CDS Diameter (a measure of delay in network-wide broadcasting) and CDS Energy Index (a measure of the energy consumption that would be incurred in network-wide broadcasts). We compared the performance of the SN-CDS with that of two widely used CDS algorithms – the minimum connected dominating set (MCDS) designed to minimize the CDS Node Size, and the ID-based

connected dominating set (ID-CDS) – often used as an alternative to MCDS. We observe that the SN-CDS incurs only 20% additional nodes (compared to the MCDS) and sustain for a lifetime that could be as large as 250% more than that of the MCDS. The CDS Energy Index and the CDS Diameter of SN-CDS are at most only 25% and 10-12% more than that of the MCDS. Both the MCDS and SN-CDS incur almost the same values for the hop count per s-d path.

The ID-CDS is not better than the MCDS and SN-CDS with respect to any performance metric. Though the ID-CDS is relatively more stable than the MCDS, this advantage comes at the cost of accommodating a substantially larger number of nodes and edges as part of the CDS that also contribute to larger values for the CDS diameter and Energy Index. We attribute the relatively poor performance of the ID-CDS to the “zero” consideration given to the number of uncovered neighbors of a covered node before deciding whether the node should be part of the CDS. A covered node with higher ID (but with only one uncovered neighbor) could end up being in the ID-CDS and a covered node with a lower ID may not even be part of the ID-CDS. As a result, the ID-CDS tends to contain several nodes, and some of these nodes are likely to be peripheral nodes whose inclusion into the CDS does not really cover multiple nodes (just one or two nodes may be covered). Moreover, the inclusion of peripheral nodes contributes as part of the CDS contributes to the instability of the ID-CDS – because, these nodes at the network boundary could easily break away from the rest of the CDS nodes.

When we factor in the energy lost due to frequent reconfiguration of the CDS, we conjecture that the energy consumption of SN-CDS will be almost equal to that of the MCDS, if not lower. This is part of the future work we plan to do – i.e. to implement the actual energy consumption models for MANETs and rank the different CDS algorithms, including the above three, according to the overall energy consumption (that includes the energy lost for discovering/reconfiguring the CDS and using the CDS for broadcasting).

## 7. ACKNOWLEDGMENTS

The work leading to this paper was partly funded through the U.S. National Science Foundation (NSF) grant CNS-0851646. The views and conclusions contained in this paper are those of the authors and do not represent the official policies, either expressed or implied, of the funding agency.

## 8. REFERENCES

- [1] S. Ni, Y. Tseng, Y. Chen and J. Sheu: The Broadcast Storm Problem in a Mobile Ad Hoc Network, *Proceedings of the 5th ACM International Conference on Mobile Computing and Networking*, pp.151-162, 1999.
- [2] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 3<sup>rd</sup> Edition, MIT Press, September 2009.
- [3] F. Kuhn, T. Moscibroda and R. Wattenhofer, “Unit Disk Graph Approximation,” *Proceedings of the Workshop on the Foundations of Mobile Computing (DIALM-POMC)*, pp. 17-23, October 2004.
- [4] N. Meghanathan, “An Algorithm to Determine the Sequence of Stable Connected Dominating Sets in Mobile Ad Hoc Networks,” *Proceedings of 2nd Advanced International Conference on Telecommunications*, Guadeloupe, French Caribbean, February 2006.
- [5] N. Meghanathan, "A Simulation-based Performance Comparison of the Minimum Node Size and Stability-based Connected Dominating Sets for Mobile Ad hoc Networks," *International Journal of Computers and Network Communications (IJCNC)*, vol. 4, no. 2, pp. 169-184, March 2012.

- [6] N. Meghanathan and M. Terrell, "Strong Neighborhood based Stable Connected Dominating Sets for Mobile Ad hoc Networks," *Proceedings of the 4th International Conference on Wireless, Mobile Network and Applications*, WiMoA, Delhi, India, pp. 415-424, May 25-27, 2012.
- [7] N. Velummylum and N. Meghanathan, "On the Utilization of ID-based Connected Dominating Sets for Mobile Ad hoc Networks," *International Journal of Advanced Research in Computer Science*, vol. 1, no. 3, pp. 36-43, September-October 2010.
- [8] F. Wang, M. Min, Y. Li and D. Du, "On the Construction of Stable Virtual Backbones in Mobile Ad hoc Networks," *Proceedings of the 24<sup>th</sup> IEEE International Conference on Performance, Computing and Communications*, pp. 355-362, April 2005.
- [9] P-R. Sheu, H-Y, Tsai, Y-P. Lee and J. Y. Cheng, "On Calculating Stable Connected Dominating Sets based on Link Stability for Mobile Ad hoc Networks," *Tamkang Journal of Science and Engineering*, vol. 12, no. 4, pp. 417-428, 2009.
- [10] N. Meghanathan, "Use of Minimum Node Velocity Based Stable Connected Dominating Sets for Mobile Ad hoc Networks," *International Journal of Computer Applications: Special Issue on Recent Advancements in Mobile Ad hoc Networks*, vol. 2, pp. 89-96, September 2010
- [11] P. Fly and N. Meghanathan, "Predicted Link Expiration Time Based Connected Dominating Sets for Mobile Ad hoc Networks," *International Journal of Computer Science and Engineering*, vol. 2, no. 6, pp. 2096-2103, September 2010.
- [12] N. Meghanathan and A. Farago, "On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad Hoc Networks," *Ad hoc Networks*, vol. 6, no. 5, pp. 744-769, July 2008.
- [13] C. Bettstetter, H. Hartenstein and X. Perez-Costa, "Stochastic Properties of the Random-Way Point Mobility Model," *Wireless Networks*, vol. 10, no. 5, pp. 555-567, September 2004.
- [14] L. M. Feeney, "An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad hoc Networks," *Journal of Mobile Networks and Applications* Vol. 3, No. 6, pp. 239 – 249, June 2001.

#### Authors

Dr. Natarajan Meghanathan is currently an Associate Professor of Computer Science at Jackson State University in Mississippi, USA. He graduated with a PhD in Computer Science from The University of Texas at Dallas (UTD) in 2005. His primary areas of research interests are Wireless Ad hoc Networks, Mobile Sensor Networks, Software Security, Graph Theory and Computational Biology. His research is funded by grants from US National Science Foundation and Army Research Lab. He was a Summer Faculty Fellow at the Air Force Research Lab in 2012 at Dayton, OH. In February 2011, he was awarded the Best Faculty Honoree from Jackson State University by the Mississippi State Legislature. He has published in more than 125 peer-reviewed international journals and conferences. He serves as the Editor-in-Chief of three international journals in the area of computer networks and also in the editorial boards and review panels of several international journals. He has also organized several international workshops and conference sessions in several sub-areas of computer networks and security.



Michael Terrell is currently a Graduate Research Assistant in the Computer Networks and Systems Security (CNSS) Lab at Jackson State University. He is pursuing his MS in Computer Science at Jackson State University since Spring 2012. He graduated with a BS in Computer Science from Grambling State University in Fall 2011. He had attended two summer internships at the Air Force Research Lab and one REU research internship at Jackson State University. His areas of research interests are computer networks, operating systems and security.

