Network Protocols and Algorithms
ISSN 1943-3581
2015, Vol. 7, No. 2

Centrality-based Connected Dominating Sets for Mobile Ad hoc Networks

Natarajan Meghanathan

Department of Computer Science, Jackson State University

1400 John R. Lynch Street, Jackson, MS 39217, USA

Tel: 1-601-979-3661 E-mail: natarajan.meghanathan@jsums.edu

Received: April 9, 2015 Accepted: July 21, 2015 Published: July 31, 2015

DOI: 10.5296/npa.v7i2.7414 URL: http://dx.doi.org/10.5296/npa.v7i2.7414

Abstract

Centrality measures capture the topological importance of a node in a network graph and they have been currently used to find the most influential nodes or the key nodes in large-scale complex network networks such as the social networks, Internet, transportation network, disease network, etc. A connected dominating set (CDS) for a graph is the set of nodes such that every node in the graph is either in this set or directly connected to a node in this set. CDSs form the backbone of a network and have been typically used for all broadcast communication within the network. In this paper, we investigate the use of centrality measures to determine CDSs for mobile ad hoc networks (MANETs) - a category of wireless networks whose topology changes dynamically with time. CDSs for MANETs have been traditionally determined using the degree centrality measure, but such CDSs have been observed to be quite unstable in the presence of node mobility. In this paper, we explore the use of more complex centrality measures such as the eigenvector centrality (EVC), betweenness centrality (BWC) and closeness centrality (ClC) to determine CDSs for MANETs, and evaluate the stability of these centrality-based CDSs with that of a degree-based CDS and the maximum stable CDS determined using a benchmarking algorithm.

Keywords: Centrality; Connected Dominating Set; Mobile Ad hoc Networks; Simulations; Stability

1. Introduction

A mobile ad hoc network (MANET) is a dynamic distributed system of wireless devices (nodes) moving arbitrarily in a network without any centralized control [1]. The nodes move independent of each other and the network topology changes dynamically with time. The transmission medium is shared and transmissions are prone to collisions and interference. Nodes operate with a limited battery charge. To conserve node lifetime and reduce collisions due to long-range transmissions, nodes in MANET operate in a limited transmission range for direct communication. As a result, communication between any two nodes is typically in a multi-hop fashion involving one or more intermediate nodes. Various protocols for unicast [2-3], multicast [4-5] and broadcast communication [6-7] have been proposed for MANETs.

In recent years, we have also seen works focusing on developing benchmarking algorithms [8-9] that model the underlying topology for communication as a graph-theoretic construct such as paths for unicast source-destination routes, trees for multicast communications and connected dominating sets (CDS) for broadcast communication and target at determining the best possible values that could be accomplished for the performance metrics associated with these topologies (for e.g., path hop count, path lifetime, tree height, CDS node size, etc). The benchmarking algorithms typically assume a centralized environment where the entire sequence of network topology changes are known ahead of time as well as assume ideal physical and medium access control layers for the communication protocols at the network layer to extract the best possible performance. We take a similar approach in this paper and focus on arriving at certain benchmarks for connected dominating sets and the use of centrality measures [10] to determine them.

A Connected Dominating Set (CDS) for a network graph (nodes modeled as vertices and links as edges) is a subset of the vertices such that every vertex in the set is either in the CDS or is a neighbor (in the adjacency list) of a node in the CDS [11]. A CDS is typically considered the backbone of a network and the constituent nodes of a CDS (i.e., the nodes that are part of a CDS) are used for facilitating broadcast communication initiated from any node in the network [12]. In MANETs, network-wide broadcasts are viewed as an expensive communication operation in terms of both energy consumption as well as the volume of the traffic generated and the resulting collisions in the neighborhood of the nodes. If network-wide broadcasts are implemented in the form of flooding wherein each node broadcasts the message once in its neighborhood, it results in a node receiving the same message once from each of its neighbors and losing a significant amount of energy due to multiple receptions of duplicate messages [13]. Also, the network medium is prone to collisions due to redundant retransmissions. Thus, the motivation is to use a CDS wherein only the nodes that are part of the CDS retransmit the message and the rest of the nodes (that are not part of the CDS, but connected to at least one of the nodes in the CDS) merely receive the message and not retransmit. The fewer the nodes that are part of the CDS (i.e., lower the CDS Node Size), the fewer the number of redundant messages being received at the nodes while still ensuring that all nodes receive at least one copy of the message being broadcast. Unfortunately, the problem of determining a minimum node size CDS (abbreviated as MCDS) is an NP-hard problem [11]. Hence, the focus of research has been traditionally on developing heuristics (e.g., [14-16]) that determine CDSs with an approximation ratio as low as possible. A common thread among these heuristics is to give preference to nodes that are in the neighborhood of a larger number of nodes (i.e., nodes having a larger degree) to be part of the CDS so that all the nodes in the network can be eventually covered with a fewer number of nodes. Thus, several maximum degree-based MCDS heuristics (e.g., [17-18]) were developed with the idea of minimizing the CDS Node Size as much as possible.

In [19], a benchmarking algorithm to determine maximum lifetime CDS was developed for MANETs; assuming the entire sequence of future topology changes is known a priori, the algorithm determines a sequence of long-living stable CDSs as follows: Whenever a CDS is required at time instant *t*, we determine a connected graph of the network (called a mobile graph) whose constituent edges exist for the longest possible time starting from time instant *t*; we then simply run a CDS algorithm/heuristic on the mobile graph and use the CDS for the duration of the mobile graph and repeat the above procedure for the entire network session. The algorithm can be used to arrive at a sequence of long-living stable CDSs such that the number of transitions (changes from one CDS to another) required during a network session is the global minimum and the average CDS lifetime for the network session is the global maximum (benchmarks). In [19] and other related works [20-22], it has been observed that maximum-degree and minimum node size-based CDS is quite unstable (i.e., have a lower lifetime) when compared to the maximum possible CDS lifetime that could be obtained under identical conditions.

With the recent splurge of research interests in the area of Network Science, there has been renewed focus on the use of graph-theoretic centrality measures for quantitatively analyzing the importance of nodes in complex networks. The centrality measures could be of two types [10]: degree-based and shortest path-based. The degree centrality (DegC) and eigenvector centrality (EVC) are two common degree-based centrality measures, while the betweenness-centrality (BWC) and closeness centrality (ClC) are two common shortest path-based centrality measures. The degree centrality of a node is simply the number of neighbors for the node, whereas, the eigenvector centrality of a node is a measure of the degree of the node as well as the degree of its neighbors. If two nodes have identical degree, but one of them has neighbors with relatively lower degree than the other node, then the node in a high-degree neighborhood will have a relatively larger eigenvector centrality. The betweenness-centrality of a node is a measure of how significant a node is in facilitating shortest-path communication between any two nodes in the network; the betweenness centrality of a node is the ratio of the number of shortest paths a node is part of for any source-destination node pair in the network, summed over all possible source-destination pairs that do not involve the particular node. The closeness centrality of a node is a measure of how close is the node to the rest of the nodes in the network; it is simply the inverse of the sum of the minimum number of hops from the node to every other node in the network.

One common thread among all the above four centrality measures is that the larger the centrality score for a node (according to any of these four measures), the larger the importance of the node in the network. Assuming that we have a centralized view of the MANET topology at any time, we propose to determine a sequence of connected dominating sets using each of these centrality measures and evaluate their average CDS Lifetime and average CDS Node Size over the duration of the MANET session. We intend to explore whether any of the three centrality measures (EVC, BWC, ClC) could contribute to finding a relatively stable CDS than the one obtained with the traditionally used degree centrality measure as well as compare the average lifetime of the CDSs obtained with these four centrality measures with that of the maximum possible CDS lifetime obtainable through our benchmarking algorithm for Maximum Stable CDS run under identical operating conditions. To the best of our knowledge, we have not come across any paper that uses centrality measures (other than degree centrality) as the basis to determine CDSs for MANETs. Note that our focus in this paper is not to determine a stable CDS [20-21]; our focus is rather to investigate whether any of the other commonly studied centrality measures could be used to determine a CDS whose lifetime is larger than that of the degree centrality-based CDS for MANETs without any substantial increase in the CDS Node Size.

The rest of the paper is organized as follows: Section 2 introduces the four centrality measures investigated in this research along with an example. Section 3 presents a generic algorithm that is used to determine a CDS based on any of the centrality measures and also presents an algorithm used to validate the existence of a CDS at any time. Section 4 presents the benchmarking algorithm for determining Maximum Stable CDS and the proof of correctness that it finds the sequence of maximum lifetime CDSs for a MANET session. Section 5 presents the simulation results obtained for the four centrality measures-based CDSs as well as the Maximum Stable CDS under various scenarios of node mobility and analyzes the tradeoffs. Section 6 concludes the paper. Throughout the paper, the terms 'node' and 'vertex', 'edge' and 'link' are used interchangeably. They mean the same.

2. Centrality Measures

We now discuss the four centrality measures studied in this paper to quantitatively assess the importance of vertices and illustrate their computation with an example.

2.1 Degree Centrality

The degree centrality of a node is the number of edges incident on the node. The larger the degree of the node, the higher its rank is. Figure 1 illustrates an example to compute the degree centrality of the vertices. As noticed, the degree centrality measure is likely to lead to tie among one or more vertices and may not be an accurate measure to unambiguously rank the vertices.

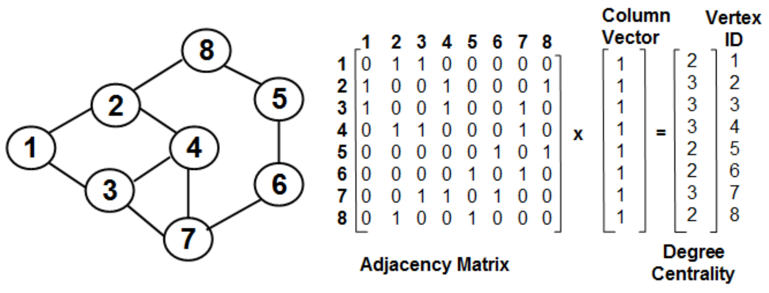


Figure 1: Example to Illustrate the Computation of Degree Centrality

2.2 *Eigenvector Centrality*

Eigenvector centrality of a vertex in an undirected graph is a measure of the degree of the vertex as well as the degree of its adjacent vertices. The Eigenvector centrality (EVC) of the vertices in a network graph is the principal eigenvector of the adjacency matrix of the graph. The principal eigenvector has an entry for each of the *n*-vertices of the graph. The larger the value of this entry for a vertex, the higher is its ranking with respect to Eigenvector centrality. In Figure 2, we illustrate the use of the Power Iiteration method [23] to efficiently calculate the principal eigenvector for the adjacency matrix of a graph. As can be seen in the example of Figure 2, Eigenvector centrality of a vertex is a function of both its degree as well as the degree of its neighbors. For instance, we see that both vertices 2 and 4 have the same degree (3); however, vertex 4 is connected to three vertices that have a high degree (3); whereas vertex 2 is connected to two vertices that have a relatively low degree (of degree 2); hence, the EVC of vertex 4 is larger than that of vertex 2. As can be seen in the example of Figure 2, the Eigenvector centrality values of the vertices are more likely to be distinct and could be a better measure for unambiguously ranking the vertices of a network graph.

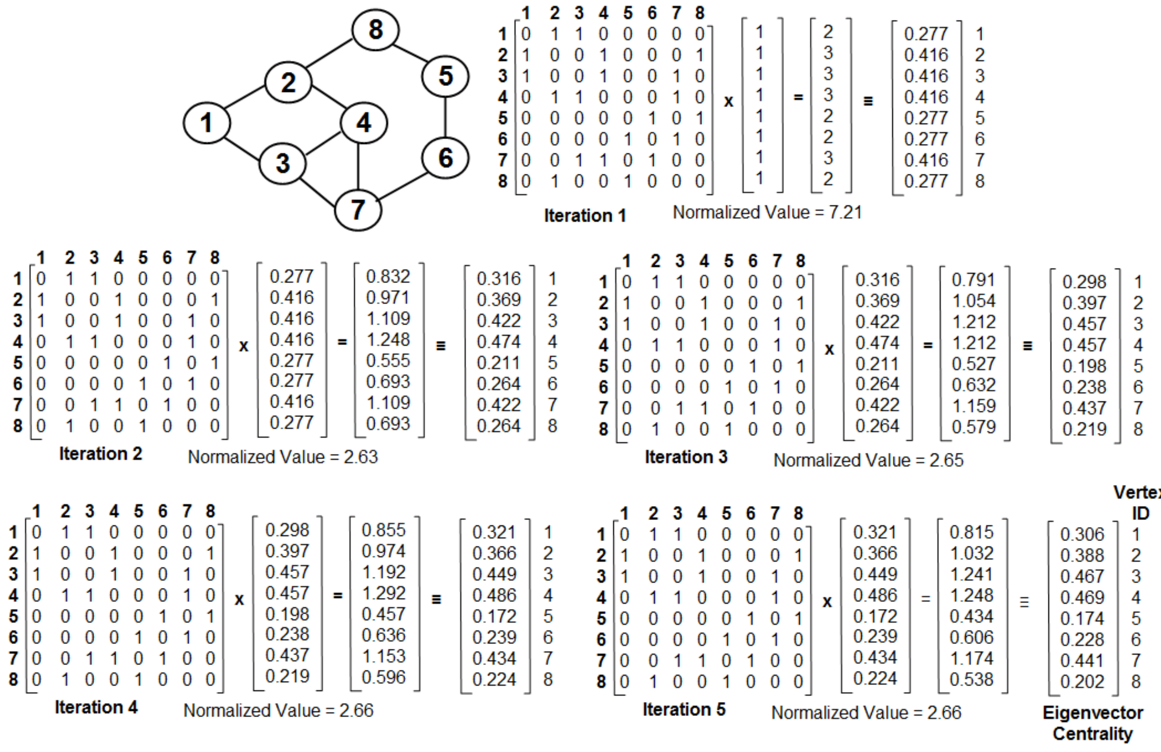
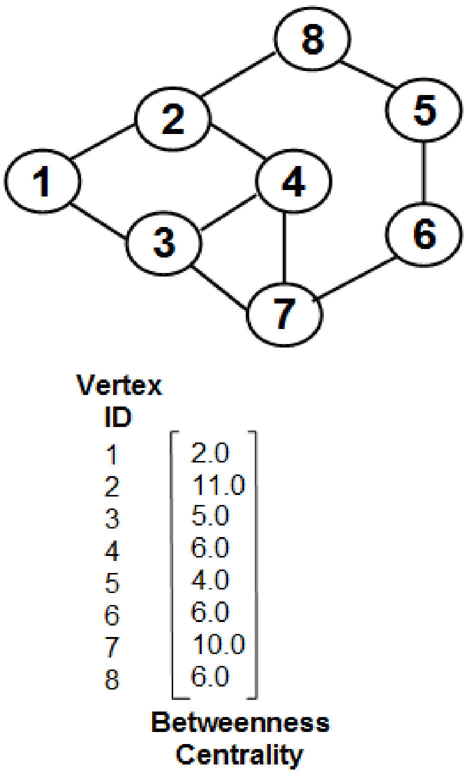
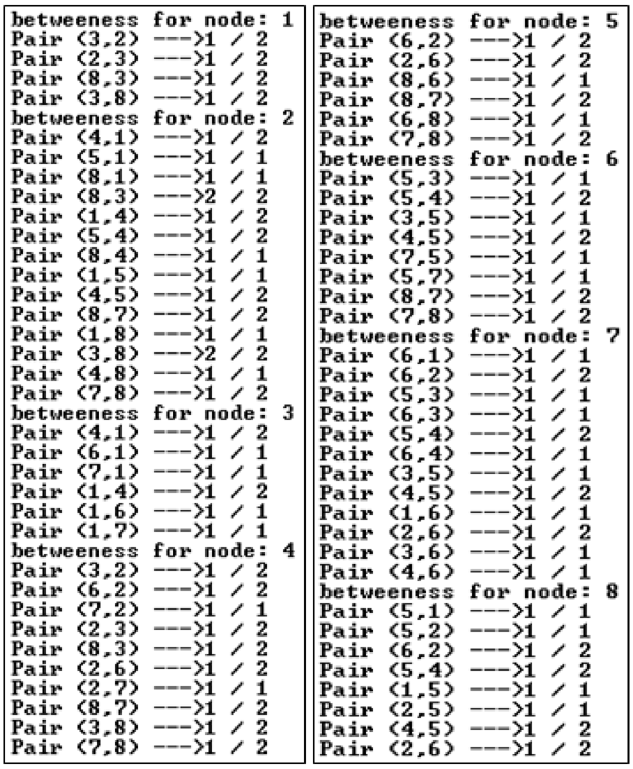


Figure 2: Example to Illustrate the Calculation of Eigenvector Centrality using Power Iteration Method

2.3 *Betweenness Centrality*

Betweenness centrality (BWC) is a measure of how significant a node is in facilitating communication between any two nodes in the network. Betweenness centrality for a node is the ratio of the number of shortest paths a node is part of for any source-destination node pair in the network, summed over all possible source-destination pairs that do not involve the particular node. If the number of shortest paths between two nodes *j* and *k* that go through node *i* as the intermediate node is denoted as $\mathbf{sp}_{jk}(i)$ and the total number of shortest paths between the two nodes *j* and *k* is denoted as $\mathbf{sp}_{jk}$, then the Betweenness centrality for node *i* is given by: $BWC(i) = \sum_{j \neq k \neq i} \frac{sp_{jk}(i)}{sp_{jk}}$.





```
betweeness for node: 1
Pair (3,2) --->1 / 2
Pair (2,3) --->1 / 2
Pair (8,3) --->1 / 2
Pair (3,8) --->1 / 2
betweeness for node: 2
Pair (4,1) --->1 / 2
Pair (5,1) --->1 / 1
Pair (8,1) --->1 / 1
Pair (8,3) --->2 / 2
Pair (1,4) --->1 / 2
Pair (5,4) --->1 / 2
Pair (8,4) --->1 / 1
Pair (1,5) --->1 / 1
Pair (4,5) --->1 / 2
Pair (8,7) --->1 / 2
Pair (1,8) --->1 / 1
Pair (3,8) --->2 / 2
Pair (4,8) --->1 / 1
Pair (7,8) --->1 / 2
betweeness for node: 3
Pair (4,1) --->1 / 2
Pair (6,1) --->1 / 1
Pair (7,1) --->1 / 1
Pair (1,4) --->1 / 2
Pair (1,6) --->1 / 1
Pair (1,7) --->1 / 1
betweeness for node: 4
Pair (3,2) --->1 / 2
Pair (6,2) --->1 / 2
Pair (7,2) --->1 / 1
Pair (2,3) --->1 / 2
Pair (8,3) --->1 / 2
Pair (2,6) --->1 / 2
Pair (2,7) --->1 / 1
Pair (8,7) --->1 / 2
Pair (3,8) --->1 / 2
Pair (7,8) --->1 / 2
```

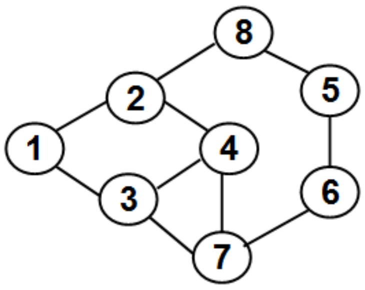
```
betweeness for node: 5
Pair (6,2) --->1 / 2
Pair (2,6) --->1 / 2
Pair (8,6) --->1 / 1
Pair (8,7) --->1 / 2
Pair (6,8) --->1 / 1
Pair (7,8) --->1 / 2
betweeness for node: 6
Pair (5,3) --->1 / 1
Pair (5,4) --->1 / 2
Pair (3,5) --->1 / 1
Pair (4,5) --->1 / 2
Pair (7,5) --->1 / 1
Pair (5,7) --->1 / 1
Pair (8,7) --->1 / 2
Pair (7,8) --->1 / 2
betweeness for node: 7
Pair (6,1) --->1 / 1
Pair (6,2) --->1 / 2
Pair (5,3) --->1 / 1
Pair (6,3) --->1 / 1
Pair (5,4) --->1 / 2
Pair (6,4) --->1 / 1
Pair (3,5) --->1 / 1
Pair (4,5) --->1 / 2
Pair (1,6) --->1 / 1
Pair (2,6) --->1 / 2
Pair (3,6) --->1 / 1
Pair (4,6) --->1 / 1
betweeness for node: 8
Pair (5,1) --->1 / 1
Pair (5,2) --->1 / 1
Pair (6,2) --->1 / 2
Pair (5,4) --->1 / 2
Pair (1,5) --->1 / 1
Pair (2,5) --->1 / 1
Pair (4,5) --->1 / 2
Pair (2,6) --->1 / 2
```

Figure 3: Example to Illustrate the Calculation of Betweenness Centrality

Figure 3 illustrates an example to compute the Betweenness centrality of the vertices in the example graph that is also used in Figures 1 and 2. We notice that Betweenness centrality-based ranking of the vertices is different from the Degree centrality and Eigenvector centrality-based ranking of the vertices. The Degree and Eigenvector centralities consider only the degree of the vertices and they are positively correlated . However, the BWC centrality takes into consideration the contribution of a vertex in facilitating communication between any two vertices in the network on the shortest path; such vertices are likely to be more central to the network and form the backbone or the core of the network. As can be seen in the example presented in Figure 3, vertices 2, 4 and 7 all have degree 3 each; however vertices 2 and 7 facilitate shortest path communication between a majority of the vertex pairs, especially from one community (vertices 5, 6 and 8) to another community (vertices 1, 3 and 4). Though vertex 4 has a high degree, it does not lie on the shortest path for several pairs of vertices, and hence has a low BWC value.

2.4 Closeness Centrality

Closeness Centrality of a vertex is the inverse of the sum of the shortest path distances to all the other vertices in the graph and hence is a measure of the proximity of the vertex to the rest of the vertices in the graph. The larger the value of the inverted sum of shortest path distances on the BFS tree of a vertex, the higher is its rank based on Closeness centrality. Figure 4 illustrates an example to compute Closeness centrality. Note that vertices 2, 4 and 7 that lie in the center of the network and have the lowest sum of the shortest path distances (12) to every other vertex (and hence have the largest value for the inverse of the sum) are ranked to have the largest Closeness centrality; though vertices 2 and 7 also had the highest Betweenness centrality, vertex 4 had a lower Betweenness centrality.



$$\begin{array}{c|cccccccc|} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} \\ \hline \mathbf{1} & 0 & 1 & 1 & 2 & 3 & 3 & 2 & 2 \\ \mathbf{2} & 1 & 0 & 2 & 1 & 2 & 3 & 2 & 1 \\ \mathbf{3} & 1 & 2 & 0 & 1 & 3 & 2 & 1 & 3 \\ \mathbf{4} & 2 & 1 & 1 & 0 & 3 & 2 & 1 & 2 \\ \mathbf{5} & 3 & 2 & 3 & 3 & 0 & 1 & 2 & 1 \\ \mathbf{6} & 3 & 3 & 2 & 2 & 1 & 0 & 1 & 2 \\ \mathbf{7} & 2 & 2 & 1 & 1 & 2 & 1 & 0 & 3 \\ \mathbf{8} & 2 & 1 & 3 & 2 & 1 & 2 & 3 & 0 \end{array} \quad \begin{array}{c} \textbf{Sum} \\ \begin{bmatrix} 14 \\ 12 \\ 13 \\ 12 \\ 15 \\ 14 \\ 12 \\ 14 \end{bmatrix} \end{array} \equiv \begin{array}{c} \textbf{Inverse of the Sum} \\ \begin{bmatrix} 0.071 \\ 0.083 \\ 0.077 \\ 0.083 \\ 0.067 \\ 0.071 \\ 0.083 \\ 0.071 \end{bmatrix} \\ \textbf{Closeness Centrality} \end{array} \quad \begin{array}{c} \textbf{Vertex ID} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}$$

Figure 4: Example to Illustrate the Calculation of Closeness Centrality

3. Connected Dominating Set

A connected dominating set (CDS) is a subset of the vertices of a graph such that each vertex in the graph is either in the subset or is a neighbor of a vertex in the subset [11]. In this section, we first describe a generic heuristic that could be used to determine the CDS of a graph based on a quantitative score (like the centrality values) for the vertices in the graph. We then illustrate examples to show the execution of the generic CDS construction algorithm for each of the four centrality measures explained in Section 2. We finally describe an algorithm that can be used to validate the existence of a CDS at any time instant.

3.1 Generic Heuristic to Determine a CDS

The overall idea is to give preference (for inclusion to the CDS) for vertices that have a larger value for the centrality measure. As indicated in the pseudo code of Figure 5, we maintain three lists: *CDS Node List* - vertices that are part of the CDS; *Uncovered Node List* - vertices that are yet to be covered by a node in the CDS Node List; *Candidate Node List* - vertices that are covered by a node in the CDS Node List and are not yet considered for inclusion to the CDS Node List. Initially, the CDS Node List is empty and the Uncovered Node List is the set of all the vertices in the graph; to start with, the Candidate Node List has a single entry corresponding to the vertex that has the largest centrality score. The Candidate

Node List is implemented as a priority queue and the vertices in the Candidate Node List are stored in the decreasing order of their centrality score (the vertex with the largest centrality score is the first vertex to be removed from this list). In each iteration, we remove a vertex from the Candidate Node List and if it has one or more uncovered neighbor nodes, then the vertex is added to the CDS Node List and the newly covered neighbor nodes are removed from the Uncovered Node List and included in the Candidate Node List. If a vertex removed from the Candidate Node List has no uncovered neighbors, then the vertex is not included in the CDS Node List.

```
Input: Graph G = ($V$, $E$); the Centrality Score Vector $C(V)$ for all the vertices
Output: CDS Node List
Initialization: CDS Node List = $\phi$; $\forall x \in V$, Uncovered Node List = {$x$; $C(x)$}
                Candidate Node List = {$s$; $C(s)$} where $s \in V$ and $C(s) = \underset{i \in V}{Max}\{C(i)\}$
Begin CDS Construction
    while (Candidate Node List $\neq \phi$ AND Uncovered Node List $\neq \phi$) do
        Vertex candidateNode = Dequeue(Candidate Node List)
        uncoveredNodes = {$x$ | $x \in$ Neighbors(candidateNode) AND $x \in$ Uncovered Node List}
        if (uncoveredNodes $\neq \phi$) then
            CDS Node List = CDS Node List $\cup$ {candidateNode}
            $\forall x \in$ uncoveredNodes,
                Uncovered Node List = Uncovered Node List - {$x$; $C(x)$}
                Candidate Node List = Candidate Node List $\cup$ {$x$; $C(x)$}
        end if
    end while
    if (Uncovered Node List $\neq \phi$) then
        return NULL // indicating the graph is not connected
    end if
    return CDS Node List // indicating the graph is connected
End CDS Construction
```

Figure 5: Pseudo Code for a Generic CDS Construction Heuristic based on a Centrality Measure

We repeat the above procedure until the Uncovered Node List gets empty or there are no more vertices in the Candidate Node List. If the Candidate Node List gets empty while the Uncovered Node List is not yet empty, then it implies the underlying graph is not connected (i.e., all the vertices are not in one component). If the underlying graph is connected, the iterations stop when there are no more vertices in the Uncovered Node List (this implies, all the vertices in the graph are covered by at least one node in the CDS Node List). Note that the centrality score for a node is determined offline (prior to the execution of the heuristic) and is not updated during the execution of the heuristic. In other words, the ordering of the nodes in the Candidate Node List in each iteration is based on the initial (static) centrality scores input to the heuristic. Thus, for example, if two nodes *u* and *v* with degree centrality scores of 4 and 6 respectively are in the Candidate Node List, but node *v* has only two uncovered neighbors whereas node *u* has three uncovered neighbors, node *v* with a relatively larger degree centrality score would still be ahead of node *u* in the Candidate Node List. If two or more vertices have the same initial centrality score and each of them have at least one uncovered neighbor during any iteration, then the tie is broken in favor of the vertex with the smaller ID for the examples presented in Figures 6-9 (Section 3.2), whereas the tie is broken arbitrarily among the contending vertices in the simulations (Section 5).

The time complexity for the generic CDS construction heuristic is O($V\log V + E\log V$), where V and E are respectively the number of vertices and edges in the network graph; O($\log V$) is the time complexity for a dequeue operation in a priority queue (Candidate Node List) maintained as a maximum heap [11] as well as the time complexity to include a vertex in the Candidate Node List by visiting the neighbors of the vertex that is added to the CDS Node List. We could run the *while* loop at most V times, once for each vertex, and across all such iterations, the entire set of edges are traversed and the end vertices of the edges are considered for inclusion to the Candidate Node List. Hence, the overall time complexity of the generic heuristic to construct a CDS is O(($V+E$) $\log V$). Note that the centrality scores of the vertices are assumed to be obtained offline and the time complexity to determine the centrality scores for the vertices is not accounted in evaluating the time complexity of the generic CDS construction heuristic.

3.2 Examples to Construct Centrality-based CDS

In this section, we show the execution of the generic heuristic (explained in Section 3.1) on the network graph that was used in Section 2 to introduce the four centrality measures. For each vertex in Figures 6-9, the ID is indicated inside the circle and the centrality score of the vertex is indicated outside the circle. The last graph in each of these figures is a CDS graph comprising of only edges between two CDS nodes (indicated as solid lines) and edges between a non-CDS node and a CDS node (indicated as dotted lines).

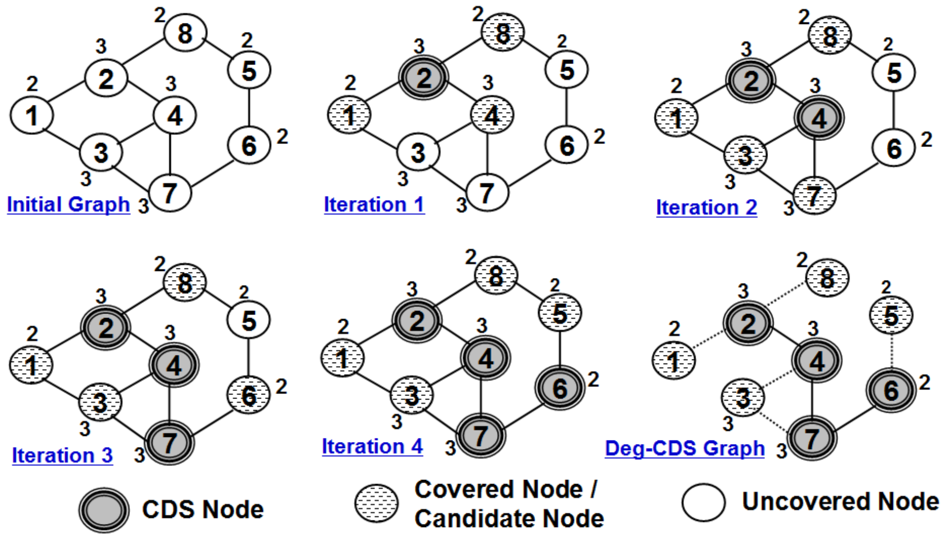


Figure 6: Example to Illustrate the Construction of a Degree Centrality-based CDS

Figure 6 illustrates the execution of the heuristic to determine degree centrality-based CDS: in iteration 3, even though vertices 3 and 7 have the same larger degree (3), vertex 3 is not considered for inclusion to the CDS because all of its three neighbors are already covered (i.e., none of the neighbors of vertex 3 are in the Uncovered Node List); on the other hand vertex 7 has one uncovered neighbor and is hence included to the CDS Node List. The sequence of vertices included in the DegC-CDS are 2, 4, 7 and 6. Figure 7 illustrates the execution of the generic heuristic to construct Eigenvector Centrality-based CDS (EVC-CDS) with the sequence of vertices included being 4, 3, 7, 2, and 6. Since nodes with a high EVC are considered for inclusion as long as they have at least one uncovered node, we observe a relatively larger number of nodes to be part of the EVC-CDS - a similar observation is also made in our simulation results - this contributes to the stability of the EVC-CDS, as can be also observed through a relatively dense EVC-CDS graph.

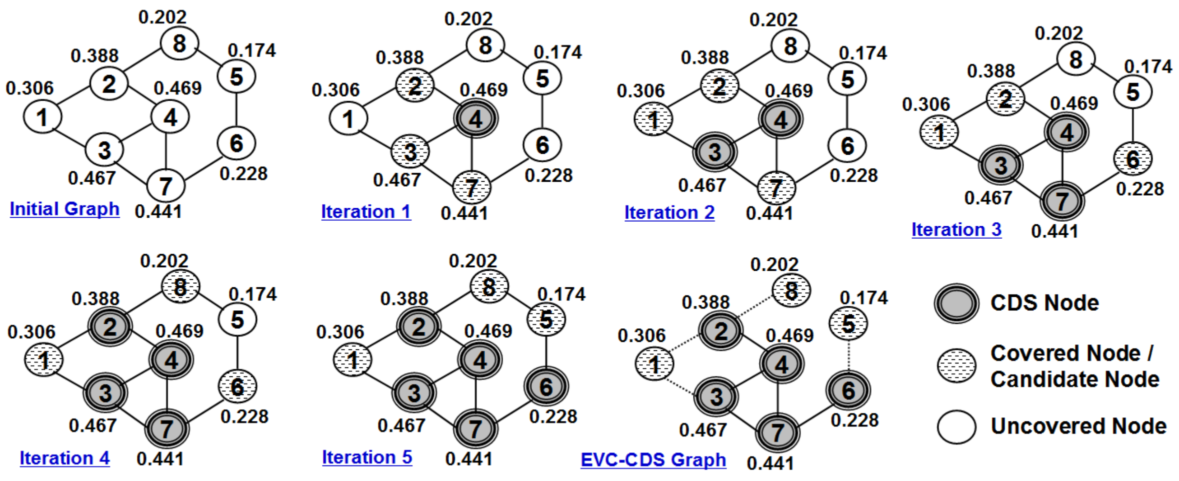


Figure 7: Example to Illustrate the Construction of a Eigenvector Centrality-based CDS

Figures 8 and 9 respectively show the execution of the generic heuristic to determine a Betweenness Centrality-based CDS (BWC-CDS) and Closeness Centrality-based CDS (ClC-CDS). As can be seen, the DegC-CDS, BWC-CDS and ClC-CDS tend to be of comparable size (in the toy example shown in Figures 6, 8 and 9 - the constituent nodes of all the three CDSs are the same: vertices 2, 4, 6 and 7) - a pattern that is also observed in the simulations to a certain extent. One significant observation could be made in the toy example and the centrality-based CDSs shown in Figures 6-9: Vertex 3 (having a high EVC score) with relatively a higher degree (3) is connected to two high-degree vertices (vertices 4 and 7), but does not contribute much in terms of covering nodes that are yet in the Uncovered Nodes List once vertices 4 and 7 get into the CDS Node List. Thus, vertex 3 is a highly enclosed vertex that is not exposed much to vertices outside its own community. On the other hand, when we consider vertices 2 and 7 that are also of the same degree as vertex 3, we notice that even if vertex 2 gets into the CDS Node List, vertex 7 could also get into the CDS Node List, because the two vertices (vertices 2 and 7) are connected to two different vertices/communities (vertices 8 and 6) that only either of them could cover, but not both.

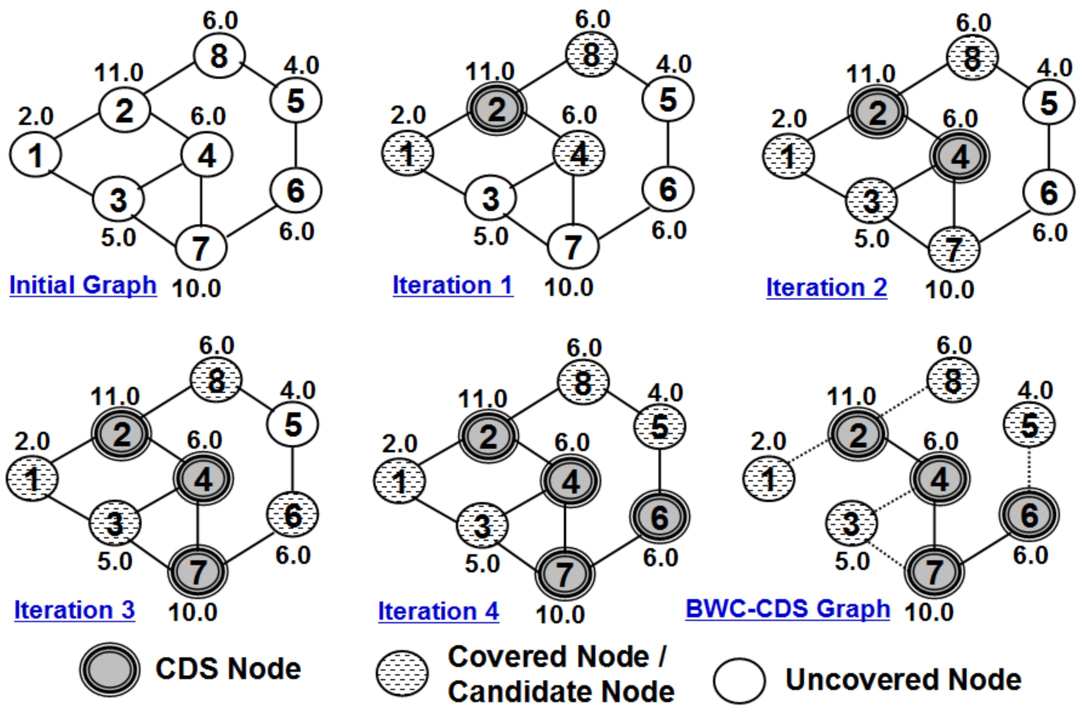


Figure 8: Example to Illustrate the Construction of a Betweenness Centrality-based CDS

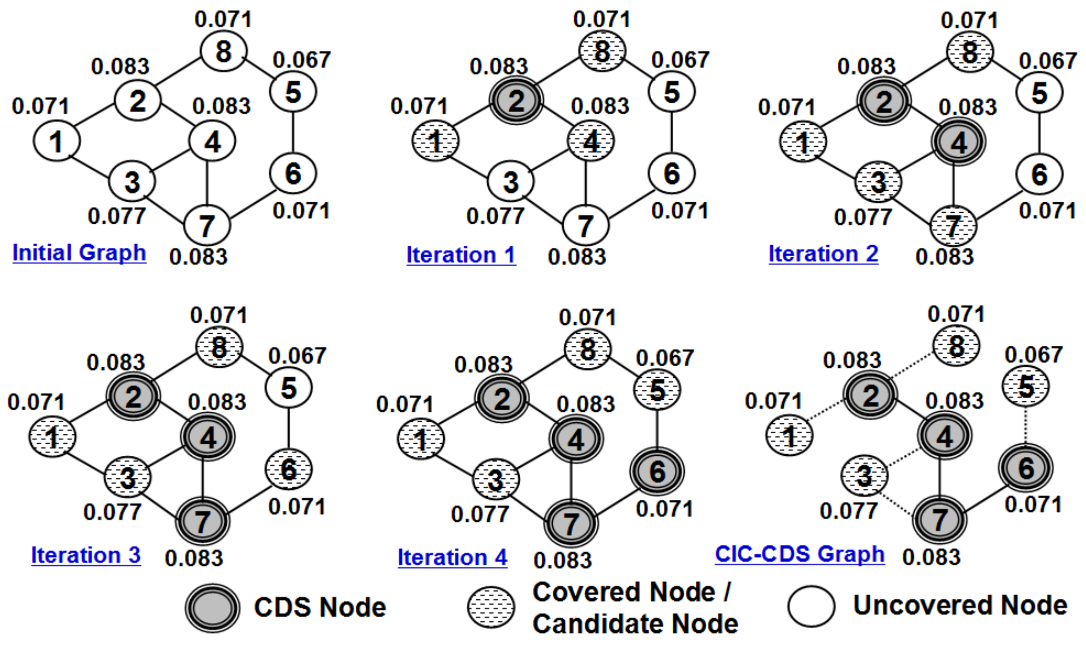


Figure 9: Example to Illustrate the Construction of a Closeness Centrality-based CDS

3.3 Algorithm to Validate the Existence of a CDS

We now present the algorithm that we use to validate the existence of a CDS at any time instant. This algorithm would be very useful in discrete-event simulations when we tend to use a CDS that existed at an earlier time instant (say, *t*-1) also at a later time instant (*t*). That is, given a CDS Node List that was connected (i.e., the CDS nodes are reachable from one another directly or through one or more intermediate CDS nodes) and covered the non-CDS nodes in the network graph at time instant *t*-1, we want to validate whether the same holds true at time instant *t*. We do so by first constructing a CDS graph based on the CDS Node List at time instant *t*. All the vertices in the network are part of the CDS graph and the edges in the CDS graph are those that exist between any two CDS nodes as well as between a CDS node and a non-CDS node at time instant *t* (like the CDS graphs shown in Figures 6-9). We check if the CDS Node List is connected at time instant *t* by running the Breadth First Search (BFS) [11] algorithm, starting from an arbitrarily chosen CDS node and see if every other CDS node could be reached as part of the BFS traversal only on the CDS-CDS edges. If all the CDS nodes could be visited, the CDS Node List is considered to be connected. We then check whether every non-CDS node has an edge to at least one CDS node at time *t*. If both the validations (connectivity of the CDS Node List and coverage of every non-CDS node by at least one CDS node) are true, then we consider the CDS to exist at time instant *t*.

4. Benchmarking Algorithm for Maximum Stability Connected Dominating Set

In this section, we describe a benchmarking algorithm [19] proposed to determine a sequence of longest-living connected dominating sets for MANETs such that the number of transitions is the global minimum and the average lifetime of the CDS is the maximum. We refer to the algorithm as the Maximum Stable CDS algorithm; it works on the basis of graph intersections and assumes the a priori availability of information about topology changes for the entire duration of the network session. The algorithm introduces the notion of a mobile graph $G(t, ..., t+k) = G(t) \cap G(t+1) \cap G(t+2) \cap ... \cap G(t+k)$, wherein $G(t)$, $G(t+1)$, $G(t+2)$, ..., $G(t+k)$ are static graphs (snapshots) of the network at time instants t, $t+1$, $t+2$, ..., $t+k$ respectively. A mobile graph $G(t, ..., t+k)$ is thus a graph that essentially captures the vertices and edges that exist in the network for all the time instants t, $t+1$, $t+2$, ..., $t+k$. Note that even when two nodes u and v are mobile, an edge (u, v) is considered to exist at time instants t, $t+1$, $t+2$, ..., $t+k$ if the two nodes u and v are within the transmission range of each other during each of these time instants; the actual locations of the end nodes of the edge during each of these time instants does not matter from the point of view of the Maximum Stable CDS benchmarking algorithm.

The Maximum Stable CDS algorithm works as follows: Whenever a CDS is required at time instant t, we determine a mobile graph $G(t, ..., t+k)$ such that $G(t, ..., t+k)$ is connected (i.e., all the vertices are in one single component, reachable from each other) and $G(t, ..., t+k+1)$ is not connected. We determine a CDS on such a longest-living connected mobile graph $G(t, ..., t+k)$ and repeat the above procedure for the duration of the network session to obtain a sequence of longest-living CDSs such that the number of transitions needed to change from one CDS to another is the optimum (minimum). We say the optimum number of transitions incurred is the global minimum because the algorithm assumes the knowledge of the entire topology changes and always chooses the longest-living mobile graph since the time instant starting from which a CDS is needed. Later, we also prove that the number of transitions accomplished by any other CDS construction algorithm cannot be below the value obtained for the Maximum Stable CDS algorithm. Accordingly, the average of the lifetimes of the longest-living connected dominating sets in the sequence determined by the Maximum Stable CDS algorithm for the duration of the network session is the global maximum and can serve as a benchmark to compare the average lifetime incurred with any other sequence of CDSs (like each of the centrality-based CDSs) under identical conditions.

```
------------------------------------------------------------------------------------------------
Input: Static graphs $G_1, G_2, ..., G_T$, where $T$ is the duration of the network session
Output: Maximum Stable CDS
Auxiliary Variables: $i, j$
Initialization: Maximum Stable CDS = $\phi$; $i = 1$; $j = 1$
Begin Maximum Stable CDS Construction
    while ($i \le T$) do
        while $G(i, ..., j)$ is connected AND $j \le T$ do
            $j \leftarrow j + 1$
        end while
        if ($i < j$) then
```

```
            $j \leftarrow j - 1$
            Maximum Stable CDS = Maximum Stable CDS $\cup$ {CDS in G($i$, ..., $j$) }
            $i \leftarrow j + 1$
        end if
        else
            $i \leftarrow i + 1$
        end else
        $j \leftarrow i$
    end while
return Maximum Stable CDS
End Maximum Stable CDS Construction
```

Figure 10: Pseudo Code for the Benchmarking Algorithm to Determine Maximum Stable CDS

As seen in the pseudo code for the Maximum Stable CDS algorithm (Figure 10), it does not matter what CDS construction heuristic we use to determine a CDS in each of the longest-living mobile graphs G(i, ..., j), because the CDS would not exist in the mobile graph G(i, ..., j+1) as the latter would not be a connected graph. In this research, we determine the connected dominating sets in the longest-living mobile graph G(i, ..., j) based on the degree of the vertices in the mobile graph G(i, ..., j). The average lifetime of the Maximum Stable CDS would not be affected because of this; however, the size of the Maximum Stable CDS is likely to be affected. As degree centrality-based CDSs are likely to incur a lower CDS Node Size, it would be only appropriate to determine the connected dominating sets in each of the longest-living mobile graphs G(i, ..., j) using the degree of the vertices in the mobile graph so that the Maximum Stable CDS would incur a lower CDS Node Size and at the same time incur the maximum lifetime.

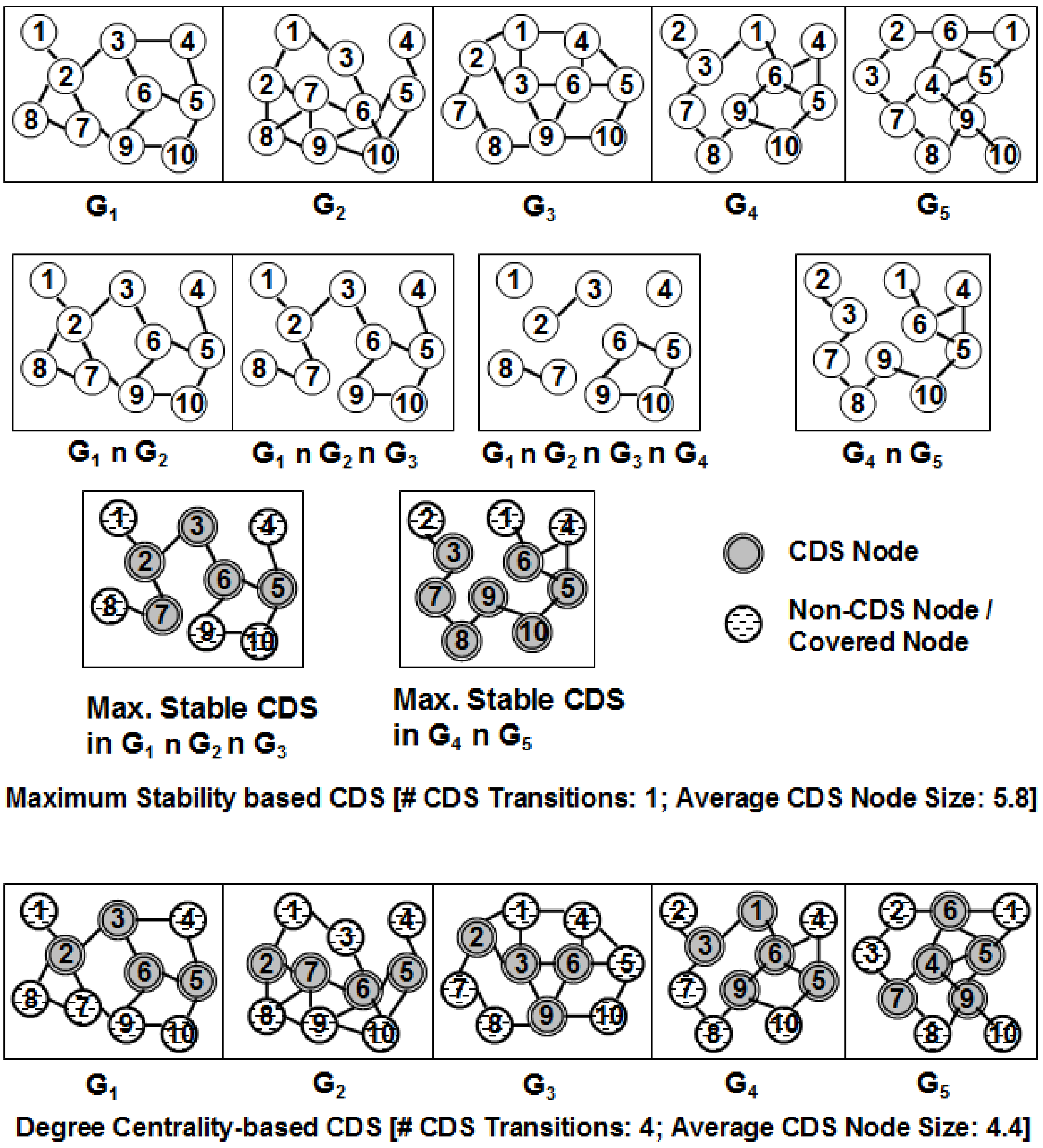

G1
G2
G3
G4
G5
G1 n G2
G1 n G2 n G3
G1 n G2 n G3 n G4
G4 n G5
Max. Stable CDS in G1 n G2 n G3
Max. Stable CDS in G4 n G5
CDS Node
Non-CDS Node / Covered Node
Maximum Stability based CDS [# CDS Transitions: 1; Average CDS Node Size: 5.8]
Degree Centrality-based CDS [# CDS Transitions: 4; Average CDS Node Size: 4.4]

Figure 11: Example to Illustrate the Execution of the Maximum Stable CDS Benchmarking Algorithm and a Comparison of Maximum Stable CDS vs. Degree Centrality-based CDS

The time complexity of the Maximum Stable CDS algorithm depends on the time complexity of the underlying heuristic used to determine the CDSs in each of the mobile graphs. As one can see in the pseudo code (Figure 10), there could be at most T mobile graphs (where T is the number of static graph snapshots, number of graph samples) on which a CDS heuristic needs to be run; if we use a $\Theta(V^2)$ heuristic to determine degree-based CDSs, then the overall time complexity of the Maximum Stable CDS algorithm is $O(V^2T)$.

4.1 Example

Figure 11 presents an example to illustrate the execution of the Maximum Stable CDS algorithm on a sequence of five static graphs G_1, ..., G_5 and a comparison of the number of transitions (changes from one CDS to another) and CDS Node Size incurred for Maximum Stable CDS vis-a-vis a Degree Centrality-based CDS on the same sequence of static graphs. We see that there exists a connected mobile graph G(1, 2, 3) and a connected mobile graph G(4, 5). Hence, it would suffice to use one CDS determined on the mobile graph G(1, 2, 3) and another CDS determined on the mobile graph G(4, 5) - resulting in only one CDS transition across the five static graphs. On the other hand, the degree centrality-based CDS determined on an individual static graph does not exist in the subsequent static graph. Hence, we end up determining a new degree centrality-based CDS on each of the five static graphs - resulting in a total of four CDS transitions across the five static graphs. In Figure 11, we use a degree-based heuristic to determine a CDS on each of the two mobile graphs. We do observe a tradeoff between CDS Node Size and the number of CDS transitions. The average CDS Node Size incurred for the Maximum Stable CDS is (3*5 + 2*7) / 5 = 5.8, whereas the average CDS Node Size incurred for Degree Centrality-based CDS is (3*4 + 2*5) / 5 = 4.4. The mobile graphs G(1, 2, 3) and G(4, 5) have relatively fewer links than the individual static graphs. Hence, it is not possible to cover a larger number of nodes with the inclusion of few nodes in the CDS. As a result, the Maximum Stable CDS incurs a relatively larger CDS Node Size compared to that of the Degree Centrality-based CDS. Such tradeoffs are also observed in the simulation results presented in Section 5.

4.2 Proof of Correctness

In this section, we prove that the Maximum Stable CDS algorithm determines a sequence of stable CDSs such that the number of CDS transitions is the global minimum. We will use the technique of "Proof by Contradiction." To start with, we will assume there exists a hypothetical CDS algorithm that determines a sequence of CDSs such that the number of CDS transitions is less than that incurred by the Maximum Stable CDS algorithm. As part of the proof, we will explore whether such a hypothetical CDS algorithm can exist and conclude that it cannot exist. We will first present an informal proof of correctness, followed by a formal proof. For more details about the proof, the interested reader is referred to [19].

Informal Proof: We present an informal proof through the example (hypothesis) illustrated in Figure 12. We show two timelines in Figure 12: one timeline for the Maximum Stable CDS algorithm and another timeline for the hypothetical CDS algorithm. Both the algorithms start at time instant t_1 and end at time instant t_T. As shown in the timelines, let the Maximum Stable CDS algorithm make *two* transitions (one transition at time instant t_a and another transition at time instant t_b) and let the hypothetical CDS algorithm just make *one* transition (at time instant t_c). We will now argue whether this would be feasible. Note that t_c cannot be greater than t_a, as the Maximum Stable CDS algorithm made a transition at t_a because it could not find a connected mobile graph that spans from time instant t_1 to time instants t_a or above. Hence, $t_c \leq t_a$. In that case, if the hypothetical CDS algorithm has to make no further transitions beyond t_c, there has to be a connected mobile graph that spanned from time instant t_c (with $t_c \leq t_a$) to the ending time instant t_T. However, the fact that the Maximum Stable CDS algorithm made a transition at t_b implies that there was no connected mobile graph spanning from time instant t_a to time instants t_b or above. If there existed a connected mobile graph that spanned from t_a to t_T, our Maximum Stable CDS algorithm would have also found it and used it (as it seeks to find a connected mobile graph that would exist for the longest time instant starting from time instant t_a). Hence, it would not be possible for the hypothetical CDS algorithm to find a connected mobile graph (and hence a CDS) that spanned from time instants t_c (with $t_c \leq t_a$) to t_T. Thus, the hypothetical CDS algorithm also has to make a transition at either time instant t_b or before t_b. In other words, it is evident that the hypothetical CDS algorithm has to undergo at least the same number of transitions as that of the Maximum Stable CDS algorithm. This is a contradiction to our hypothesis that the Maximum Stable CDS algorithm made two transitions and the hypothetical CDS algorithm made one transition, as illustrated in Figure 12.

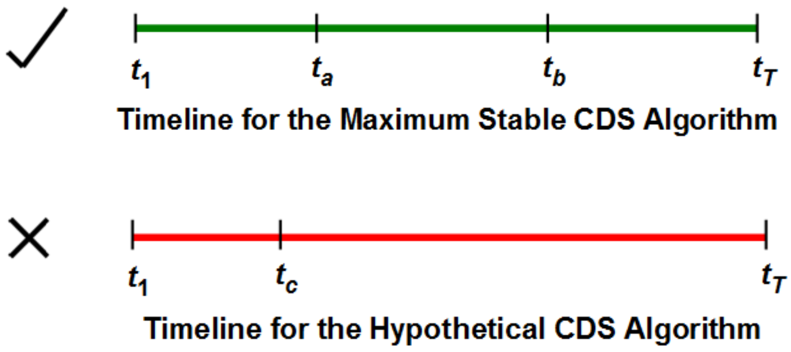

✓
t_1
t_a
t_b
t_T
Timeline for the Maximum Stable CDS Algorithm
✗
t_1
t_c
t_T
Timeline for the Hypothetical CDS Algorithm

Figure 12: Hypothesis to Illustrate Proof of Correctness for the Maximum Stable CDS Algorithm [Proof Technique: Proof by Contradiction]

Basis for the Formal Proof: The only way the hypothetical CDS algorithm (in Figure 12) can undergo fewer transitions than the Maximum Stable CDS algorithm is when there existed an epoch of time instants [like t_c... t_T in Figure 12, where $t_c \leq t_a$] during which the hypothetical CDS algorithm made no transitions and the Maximum Stable CDS algorithm made at least one transition. However, based on the above discussion it is evident that such a scenario is not possible. This observation forms the basis for the formal proof of correctness discussed below.

Formal Proof: We now present a formal proof of correctness for the Maximum Stable CDS algorithm. To prove that the Maximum Stable CDS algorithm finds a sequence of long-living CDSs that incur the minimum number of transitions (say m), assume the contrary - i.e., there exists a hypothetical CDS algorithm that determines a sequence of CDSs that undergo n transitions such that $n < m$. We will now explore whether this is possible. For $n < m$ to exist, there has to be an epoch of time instants [p, ..., s] that is a superset of an epoch of time instants [q, .., r] such that $p < q < r < s$ and that the hypothetical CDS algorithm was able to find a CDS that existed during time instants [p, ..., s], but the Maximum Stable CDS algorithm can only find a CDS that existed during time instants [q, ..., r] and had to go a transition at time instant r + 1. This implies the Maximum Stable CDS algorithm could not find a connected mobile graph G(q, ..., r+1) and could only find a connected mobile graph G(q, ..., r). This means there existed no connected mobile graph from time instants [q, ..., s]

and hence there is no connected mobile graph from time instants [*p*, ..., *s*]. If that is the case, it would not be possible to find a CDS that exists from time instants [*p*, ..., *s*]; thus, the sequence of CDSs determined by the hypothetical CDS algorithm has to undergo at least as many transitions as those determined by the Maximum Stable CDS algorithm, which is a contradiction to our initial assumption that $n < m$. Hence, the number of transitions (m) incurred by the sequence of long-living CDSs determined by the Maximum Stable CDS algorithm serves as a lower bound for the number of transitions (n) incurred by any other algorithm/heuristic to determine a sequence of CDSs.

5. Simulations

We conducted the simulations in a discrete-event simulator implemented in Java. We assume a network of dimensions 1000m x 1000m; the number of nodes is fixed as 100 and the transmission range of the nodes is fixed at 250m; for these values, on average, there are $\pi*250*250*100/(1000*1000) = 20$ neighbors per node. The overall connectivity of the network for the above conditions is observed to be more than 99.5%. Initially, the nodes are uniform-randomly distributed throughout the network. The number of static nodes in the network is varied with values of 0, 20, 40, 60 and 80. A node designated to be static in the beginning of the simulation does not move at all for the duration of the simulation; likewise, a node designated as mobile - keeps moving for the entire simulation.

The node mobility model used is the Random Waypoint Model [24], wherein the maximum velocity of each mobile node is v_{max} and it is varied with values of 5 m/s (low mobility), 25 m/s (moderate mobility) and 50 m/s (high mobility). According to the Random Waypoint Model, each mobile node starts from its initial location and chooses to move to an arbitrary location (located within the network boundaries) with a velocity that is uniform-randomly selected from the range [0, ..., v_{max}]; after moving to the chosen location, the node chooses another arbitrary location (within the network) and moves to the chosen location with a new velocity that is uniform-randomly chosen from the range [0, ..., v_{max}]. In other words, a node moves in a straight line from one location to another chosen location with a particular velocity and after reaching the targeted location, the node chooses another target location to move with a velocity that could be different from the one used before. A mobile node moves like this for the duration of the simulation session and generates a mobility profile for a particular combination of values for the number of static nodes and maximum node velocity. The collection of the mobility profiles of all the nodes is stored in a mobility profile file and is feed in to the heuristics/algorithms for determining a centrality-based CDS and the Maximum Stable CDS. We generate 100 such mobility profile files for each of the combinations of the values for the number of static nodes (0, 20, 40, 60 and 80) and the maximum node velocity (5 m/s, 25 m/s and 50 m/s). The decision of whether a node will be static or mobile is made prior to creating each of the mobility profile files.

We run the simulations for each mobility profile file and average the results observed for the CDS Lifetime and CDS Node Size under each of the centrality measures as well as the Maximum Stable CDS algorithm. We start the simulations at time 0 sec (based on the initial distribution of the nodes in a particular mobility profile file) and run the simulations for a period of 1000 seconds, sampling the network for every 0.25 seconds of the simulation. We take a snapshot of the network at each of these sampling time instants (referred to as the static graphs). We use the following approach to run the simulations for a particular centrality measure: Whenever a CDS is needed at a particular time instant *t*, we determine the centrality scores of the nodes based on the static graph snapshot of the network at time instant *t* and feed in these scores to the CDS construction heuristic (described in Section 3.1). The CDS determined at time instant *t* is used for the subsequent sampling time instants as long as it exists (validated using the algorithm presented in Section 3.3). When the currently known CDS is observed to no longer exist at a particular time instant, we repeat the above procedure. We continue like this for the duration of the simulation and measure the lifetime and node size for the sequence of centrality-based connected dominating sets used for the particular mobility profile file. We average the results for these two metrics observed for all the mobility profile files generated for a particular combination of values for the number of static nodes and v_{max} (see Figures 13, 14 and 15 for the results). In the case of the Maximum Stable CDS algorithm, for a particular simulation run under a mobility profile file, we determine a sequence of connected mobile graphs for the duration of the simulation session and run the degree-based CDS construction heuristic to determine a sequence of connected dominating sets. We repeat this procedure for all the mobility profile files generated and determine the average of the lifetime and node size for the sequence of Maximum Stable CDSs for the particular combination of values of the number of static nodes and v_{max}. The results for the Maximum Stable CDS are shown along with the centrality-based CDSs in Figures 13, 14 and 15.

5.1 Average CDS Lifetime

Among the four centrality measures, we observe the Eigenvector Centrality-based CDS (EVC-CDS) to be consistently sustaining for the longest lifetime for all the mobility scenarios, closely followed by the Closeness Centrality-based CDS (ClC-CDS). For a given number of static nodes, the percentage difference between the lifetimes of the EVC-CDS and ClC-CDS is lower for v_{max} values of 25 and 50 m/s (see Figures 14-a and 15-a) compared to v_{max} values of 5 m/s (see Figure 13-a). For all the mobility scenarios, the Betweenness Centrality-based CDS (BWC-CDS) incurs the lowest lifetime among all the centrality measures-based CDSs; the Degree Centrality-based CDS (DegC-CDS) incurs a longer lifetime than the BWC-CDS (about 30% larger) but a lower lifetime compared to that of the ClC-CDS and EVC-CDS. For a given v_{max}, the difference between the lifetimes of the DegC-CDS and the EVC-CDS (or the ClC-CDS) increases from 5% to 40% with increase in the number of static nodes.

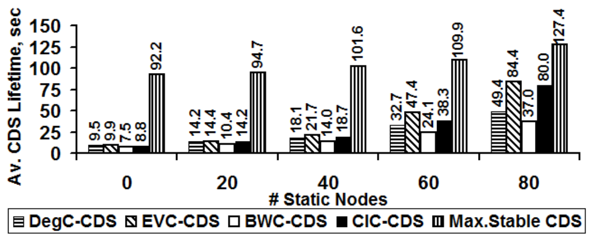


Figure 13-a: Average CDS Lifetime

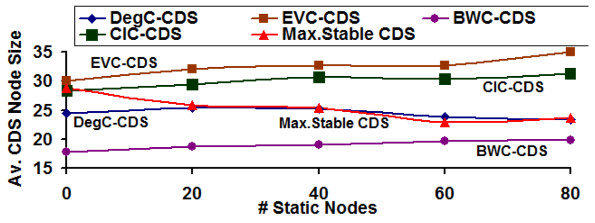


Figure 13-b: Average CDS Node Size

Figure 13: Average CDS Lifetime and Average CDS Node Size for Centrality-based CDSs and Maximum Stable CDS [*Maximum Node Velocity*, v_{max} = 5 m/s]

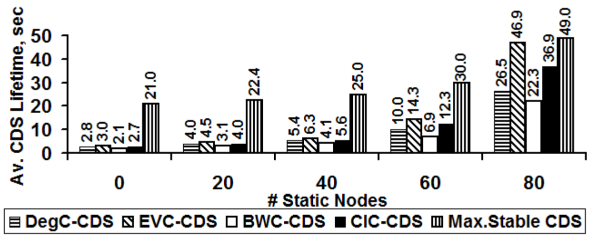


Figure 14-a: Average CDS Lifetime

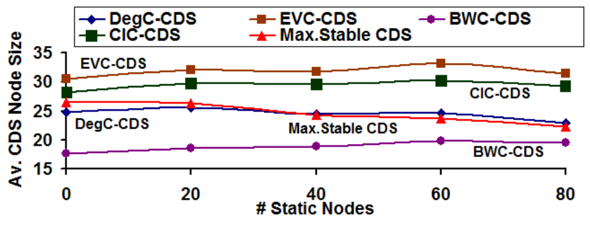


Figure 14-a: Average CDS Node Size

Figure 14: Average CDS Lifetime and Average CDS Node Size for Centrality-based CDSs and Maximum Stable CDS [*Maximum Node Velocity*, v_{max} = 25 m/s]

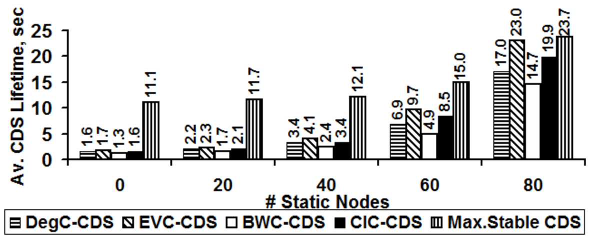


Figure 15-a: Average CDS Lifetime

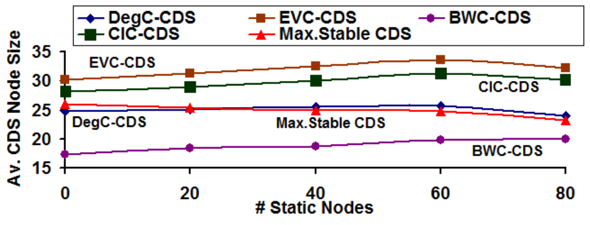


Figure 15-b: Average CDS Node Size

Figure 15: Average CDS Lifetime and Average CDS Node Size for Centrality-based CDSs and Maximum Stable CDS [*Maximum Node Velocity*, v_{max} = 50 m/s]

For all v_{max} values, when the network is dynamically changing (0 static nodes), the values for the average CDS lifetime incurred with all the four centrality measures are comparable; the percentage difference increases as we increase the number of static nodes (for a given v_{max}), especially when the number of static nodes gets to 60 and 80. For a fixed number of static nodes, the percentage difference between the lifetime of the CDSs determined under the four centrality measures decreases with increase in the maximum node velocity. In other words, as we increase the node velocity (for a fixed number of static nodes), the CDS lifetimes with respect to all the four centrality measures converges; nevertheless, the EVC-CDS lifetime is substantially larger than that of the BWC-CDS and DegC-CDS lifetime for all the mobility scenarios. For a scenario of 80 static nodes, the average EVC-CDS lifetime is about 120% more than that of the BWC-CDS when v_{max} = 5 m/s (see Figure 13-a) and is about 60% more than that of BWC-CDS when v_{max} = 50 m/s (see Figure 15-a). The larger lifetime obtained for the EVC-CDS could be attributed to the relatively larger number of nodes constituting the EVC-CDS, compared to those incurred for the BWC-CDS (that incurs the smallest CDS Node Size) and the other centrality measures. The EVC-CDS Node Size is about 70% larger than that incurred for the BWC-CDS for all the mobility scenarios. This illustrates a CDS Lifetime - CDS Node Size tradeoff (explored further in Section 5.3).

When compared to the benchmark CDS lifetimes obtained from the Maximum Stable CDS algorithm run with the same set of mobility profile files, we observe the stability of the connected dominating sets determined under all the four centrality measures to be substantially lower (compared to the Maximum Stable CDS) when the number of static nodes is 0, 20 and 40. However, when more than half of the nodes are static, the lifetimes incurred with the centrality-based connected dominating sets swiftly approach the lifetime incurred for the Maximum Stable CDS. For v_{max} values of 25 m/s and 50 m/s, and 80 static nodes (see Figures 14-a and 15-a), we observe the EVC-CDS lifetime to be only at most 5% lower than that of the lifetime incurred for Maximum Stable CDS. Under the same set of conditions, the BWC-CDS lifetime is observed to be about 50-60% of the lifetime incurred for the Maximum Stable CDS.

A significant observation is that for a given v_{max}, the lifetime of the Maximum Stable CDS does not increase substantially with increase in the number of static nodes. Likewise, as we increase v_{max} from 5 m/s to 50 m/s, the average lifetime incurred for the Maximum Stable CDS increases by a factor of 1.4-2.3 as we increase the number of static nodes from 0 to 80. This could be attributed to the fact that the mobility of certain percentage of nodes is sufficient to disrupt the connectivity of the mobile graphs spanning over several time instants. On the other hand, as we increase v_{max} from 5 m/s to 50 m/s, the CDS lifetime under all the four centrality measures increases substantially (by factors as large as 15 for EVC-CDS) with increase in the number of static nodes from 0 to 80. Thus, even if a certain fraction of nodes move faster, the presence of a significant fraction of static nodes helps to boost the lifetime of the CDSs determined under the four centrality measures.

5.2 Average CDS Node Size

With regards to the CDS Node Size (see Figures 13-b, 14-b, 15-b), for a given centrality measure, we do not observe any significant impact of the mobility scenarios. For a given v_{max}, the DegC-CDS and the Maximum Stable CDS (both are degree based) display a marginal decrease (by a factor of about 20%) in the CDS Node Size with increase in the number of static nodes. Other than this, the CDS Node Size incurred for CDSs determined with respect to each of the centrality measures remains almost at the same value, independent of the v_{max} values and the number of static nodes. The EVC-CDS and BWC-CDS incur respectively the largest and smallest CDS Node Size values for all the mobility scenarios (the percentage difference is about 70%). The CDS Node Size incurred with the DegC-CDS and ClC-CDS are respectively about 30% and 55% more than that incurred with the BWC-CDS. The relatively lower CDS Node Size incurred with the BWC-CDS (in comparison with the DegC-CDS) is an interesting and significant observation. Until now in the literature, the degree-based criterion for forming CDS has been considered to return the smallest possible CDS Node Size for MANETs [16-22]; however, as we observe in the simulation results, for all the conditions tested, the BWC-CDS incurs an appreciably smaller CDS Node Size compared to that of the DegC-CDS, EVC-CDS and ClC-CDS. This could be attributed to the fact that though the shortest path-based CDSs (like BWC-CDS) do not take the degree of the nodes into consideration while forming the CDS, a node that has a higher degree is more likely to have a shorter path to several other nodes as well as more likely to be on the shortest path between several node pairs [25]. Thus, the inclusion of high degree nodes that lie on the shortest path for several other node pairs (i.e., high degree vertices having high BWC) could contribute to a lower CDS Node Size.

5.3 CDS Node Size - CDS Lifetime Tradeoff

An interesting observation is that the Maximum Stable CDS could incur a substantial longer CDS Lifetime without any significant increase in the CDS Node Size. On the contrary, among the four centrality measures, the EVC-CDS incurs a longer CDS Lifetime at the expense of a larger CDS Node Size. The CDS Node Size incurred for the DegC-CDS and the Maximum Stable CDS are very much comparable and are about the same for most of the mobility scenarios. The knowledge of the future topology changes is the key factor for the Maximum Stable CDS benchmarking algorithm and the resulting stability. Given the lack of such a global network-wide and future knowledge, we have to choose an appropriate centrality measure to determine stable CDSs that exist for a longer lifetime without substantial increase in the CDS Node Size. In this pursuit, we analyze the CDS Node Size-CDS Lifetime tradeoff as the value of the ratio CDS Node Size / CDS Lifetime. The smaller the value of this ratio (closer it is to zero), the better the centrality measure with respect to minimizing the tradeoff.

Figure 16 displays the CDS Node Size / CDS Lifetime tradeoff ratio for v_{max} values of 5, 25 and 50 m/s. For any of the four centrality-based CDSs, the tradeoff ratio increases with increase in the dynamicity of the network. For a fixed number of static nodes, the tradeoff ratio increases with increase in the maximum node velocity. For a fixed v_{max}, the tradeoff ratio decreases with increase in the number of static nodes. Among the four centrality measures, for v_{max} values of 50 m/s (see Figure 16-c), we observe the BWC-CDS to incur a relatively lower tradeoff ratio (compared to the other centrality-based CDSs) when the number of static nodes is typically on the low end (0 and 20). This implies that by accommodating relatively fewer nodes as part of the CDS, the BWC-CDS incurs a proportionally larger CDS Lifetime when compared to that of the EVC-CDS incurring a larger CDS Lifetime by accommodating more CDS nodes. On the other hand, when v_{max} = 5 m/s or 25 m/s (see Figures 16-a and 16-b) and when the number of static nodes is on the higher end (40, 60 and 80), the tradeoff ratio values incurred with all the four centrality-measures based CDSs are comparable, with the EVC-CDS and ClC-CDS incurring a slightly lower tradeoff ratio (but not much different from that of the DegC-CDS and BWC-CDS).

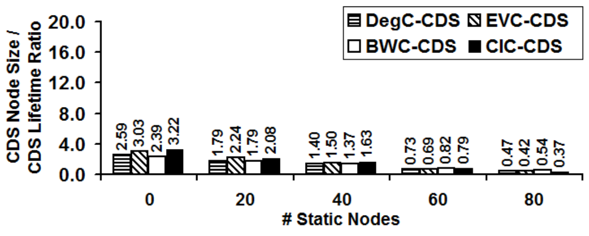


Figure 16-a: v_{max} = 5 m/s

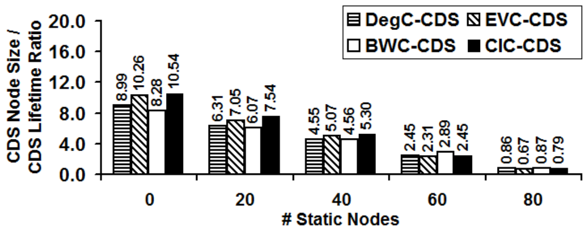


Figure 16-b: v_{max} = 25 m/s

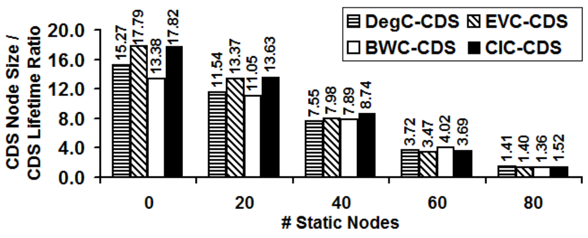


Figure 16-c: v_{max} = 50 m/s

Figure 16: CDS Node Size - CDS Lifetime Tradeoff Ratio for Centrality Measures-based CDSs

6. Conclusions

The simulation studies return interesting results that form the key revelations of this paper with regards to the use of centrality measures for mobile ad hoc networks (MANETs). We have identified a centrality measure (eigenvector centrality, EVC) that could be used to determine CDSs that are more stable than those traditionally determined using the degree centrality measure. We have also identified a centrality measure (betweenness centrality, BWC) that can be used to determine CDSs whose constituent node size could be smaller than that of the CDSs determined using the degree centrality measure. Nevertheless, we witness a CDS Node Size - CDS Lifetime tradeoff: the EVC-CDS is the most stable, but incurs the largest values for the CDS Node Size; whereas, the BWC-CDS incurs the smallest values for the CDS Node Size, but it is the least stable.

When the network topology changes dynamically, we observe the CDS Lifetimes obtained under all the four centrality measures to be very much comparable to each other and are far lower than the maximum possible CDS lifetime obtained for the Maximum Stable CDS. On the other hand, as we increase the fraction of static nodes in the network, the lifetimes of all the four centrality-based CDSs increase significantly (in a concave-up fashion) and approach the benchmarking lifetimes obtained with the Maximum Stable CDS algorithm run under identical operating conditions (especially in the case of EVC-CDS). We anticipate the results observed in this paper to open further avenues of research on the use of centrality measures-based connected dominating sets for mobile ad hoc networks.

Acknowledgment

The research is financed by the NASA EPSCoR sub award (#: NNX14AN38A) from University of Mississippi.

References

[1] Murthy, C. S. R., Manoj, B. S., *Ad Hoc Wireless Networks: Architectures and Protocols*, 1st ed., Prentice Hall, June 2004.

[2] Johnson, D. B., Maltz, D. A., "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing - The Kluwer International Series in Engineering and Computer Science, Vol. 353, pp. 153-181, 1996. http://dx.doi.org/10.1007/978-0-585-29603-6_5

[3] Perkins, C. E., Royer, E. M., "Ad Hoc On-demand Distance Vector Routing," *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, New Orleans, LA, USA, February 25-26, 1999. http://dx.doi.org/10.1109/MCSA.1999.749281

[4] Wu, C. W., Tay, Y. C., "AMRIS: A Multicast Protocol for Ad hoc Wireless Networks," *IEEE Military Communications Conference*, Vol. 1, pp. 25-29, Atlantic City, NJ, USA, October 31-November 3, 1999. http://dx.doi.org/10.1109/MILCOM.1999.822636

[5] Mnaouer, A. B., Chen, L., Foh, C. H., Tantra, J. W., "OPHMR: An Optimized Polymorphic Hybrid Multicast Routing Protocol for MANET," IEEE Transactions on Mobile Computing, Vol. 6, No. 5, pp. 551-562, May 2007. http://dx.doi.org/10.1109/TMC.2007.1030

[6] Dai, F., Wu, J., "Performance Analysis of Broadcast Protocols in Ad Hoc Networks Based on Self-Pruning," IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 11, pp. 1-13, November 2004. http://dx.doi.org/10.1109/TPDS.2004.69

[7] Saha, S., Hussain, S. R., Ashikur Rahman, A. K. M., "RBP: Reliable Broadcast Protocol in Large Scale Mobile Ad Hoc Networks," *24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 526-532, Perth, WA, USA, April 20-23, 2010. http://dx.doi.org/10.1109/AINA.2010.91

[8] Meghanathan, N., "Graph Theory Algorithms for Mobile Ad hoc Networks", Informatica - An International Journal of Computing and Informatics, Vol. 36, No. 2, pp. 185-200, June 2012.

[9] Meghanathan, N., Mumford, P., "A Benchmarking Algorithm to Determine the Sequence of Stable Data Gathering Trees for Wireless Mobile Sensor Networks," Informatica - An International Journal of Computing and Informatics, Vol. 37, No. 3, pp. 315-338, Oct. 2013.

[10] Newman, M., *Networks*: *An Introduction*, 1st ed., Oxford University Press, May 2010.

[11] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., *Introduction to Algorithms*, 3rd ed., MIT Press, July 2009.

[12] Wan, P-J., Alzoubi, K. M., Frieder, O., "Distributed Construction of Connected Dominating Set in Wireless Ad hoc Networks," *21st IEEE International Conference on Computer and Communications*, Vol. 3, pp. 1597-1604, New York City, NY, USA, June 23-27, 2002. http://dx.doi.org/10.1109/INFCOM.2002.1019411

[13] Ni, S-Y., Tseng, Y-C., Chen, Y-S., Sheu, J-P., "The Broadcast Storm Problem in a Mobile Ad hoc Network," *5th Annual IEEE/ACM International Conference on Mobile Computing and Networking*, pp. 151-162, 1999, Seattle, WA, USA, August 15-20, 1999. http://dx.doi.org/10.1145/313451.313525

[14] Butenko, S., Cheng, X., Oliveira, C. A., Pardalos, P. M., "A New Heuristic for the Minimum Connected Dominating Set Problem on Ad Hoc Wireless Networks," Recent

Developments in Cooperative Control and Optimization Cooperative Systems, Vol. 3, pp. 61-73, 2004. http://dx.doi.org/10.1007/978-1-4613-0219-3_4

[15] Misra, R., Mandal, C., “Minimum Connected Dominating Set using a Collaborative Cover Heuristic for Ad hoc Sensor Networks,” IEEE Transactions on Parallel and Distributed Systems, Vol. 21, No. 3, pp. 292-302, March 2010. http://dx.doi.org/10.1109/TPDS.2009.78

[16] Guha, S., Khuller, S., “Approximation Algorithms for Connected Dominating Sets,” Algorithmica, Vol. 20, No. 4, pp. 374-387, April 1998. http://dx.doi.org/10.1007/PL00009201

[17] King, L., Meghanathan, N., “A Weighted-Density Connected Dominating Set Data Gathering Algorithm for Wireless Sensor Networks,” Journal of Computer and Information Science, Vol. 2, No. 4, pp. 3-13, November 2009. http://dx.doi.org/10.5539/cis.v2n4p3

[18] Das, A., Mandal, C., Reade, C., Aasawat, M., “An Improved Greedy Construction of Minimum Connected Dominating Sets in Wireless Networks,” *IEEE Wireless Communications and Networking Conference*, pp. 790-795, March 28-31, 2011, Cancun, Mexico. http://dx.doi.org/10.1109/WCNC.2011.5779233

[19] Meghanathan, N., Farago, A., “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. http://dx.doi.org/10.1016/j.adhoc.2007.06.005

[20] Meghanathan, N., “Node Stability Index: A Stability Metric and an Algorithm to Determine Long-Living Connected Dominating Sets for Mobile Ad hoc Networks,” International Journal of Interdisciplinary Telecommunications and Networking, Vol. 4, No. 1, pp. 31-46, January-March 2012. http://dx.doi.org/10.4018/jitn.2012010102

[21] Meghanathan, N., Terrell, M., “A Simulation Study on the Strong Neighborhood-based Stable Connected Dominating Sets for Mobile Ad hoc Networks,” International Journal of Computer Networks and Communications, Vol. 4, No. 4, pp. 1-19, July 2012. http://dx.doi.org/10.5121/ijcnc.2012.4401

[22] Meghanathan, N., “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, Vol. 4, No. 2, pp. 169-184, March 2012. http://dx.doi.org/10.5121/ijcnc.2012.4211

[23] Strang, G., *Linear Algebra and its Applications*, 4th ed., Cengage Learning, July 2005.

[24] Bettstetter, C., Hartenstein, H., Perez-Costa, X., “Stochastic Properties of the Random-Way Point Mobility Model,” Wireless Networks, Vol. 10, No. 5, pp. 555-567, September 2004. http://dx.doi.org/10.1023/B:WINE.0000036458.88990.e5

[25] N. Meghanathan, “Correlation Coefficient Analysis of Centrality Metrics for Complex Network Graphs,” *4th Computer Science Online Conference, Intelligent Systems in Cybernetics and Automation Theory*: *Advances in Intelligent Systems and Computing*, Vol. 348, pp. 11-20, April 27-30, 2015. http://dx.doi.org/10.1007/978-3-319-18503-3_2