Use of Minimum Node Velocity Based Stable Connected Dominating Sets for Mobile Ad hoc Networks

Natarajan Meghanathan
Jackson State University
P. O. Box 18839, 1400 J. Lynch St
Jackson, MS 39217, USA

ABSTRACT
We propose an algorithm to determine stable connected dominating sets (CDS), based on node velocities, for mobile ad hoc networks (MANETs). The proposed minimum velocity-based CDS (MinV-CDS) algorithm prefers slow-moving nodes with lower velocity, rather than the usual approach of preferring nodes with a larger number of uncovered neighbors, i.e., larger density (referred to as MaxD-CDS). The construction of the MinV-CDS starts with the inclusion of the node having the lowest velocity, into the CDS. Once a node is added to the CDS, all its neighbors are said to be covered. The covered nodes are considered in the increasing order of their velocity, for inclusion in the CDS. If a node has lower velocity and is the next candidate node to be considered for inclusion in the CDS, it is added to the CDS if it has at least one neighbor that is yet to be covered. This procedure is repeated until all the nodes in the network are covered. Simulation results illustrate that the MinV-CDS has a significantly longer lifetime compared to MaxD-CDS. MinV-CDS also has a larger number of nodes and edges compared to MaxD-CDS and this helps to reduce the hop count as well as the end-to-end delay and improves the fairness of node usage.

General Terms
Algorithms

Keywords
Connected Dominating Set, Mobile Ad hoc Network, Node Velocity, Density

1. INTRODUCTION
A mobile Ad hoc network (MANET) is a dynamic distributed system of arbitrarily moving wireless nodes that operate on a limited battery charge. The network operates on a limited bandwidth and the transmission range of each node is limited. As a result, multi-hop communication is very common in MANETs. Route discovery in MANETs has been traditionally accomplished through a flooding-based Route-Request-Reply cycle in which all the wireless nodes are responsible for forwarding the Route-Request (RREQ) messages from the source towards the destination and propagating the Route-Reply (RREP) messages on the discovered path from the destination back to the source. Recent studies (e.g., [1][2][3][4][5]) demonstrate the use of connected dominating set (CDS)-based virtual backbones to propagate the RREQ and RREP messages so that the routing control messages are exchanged only among the nodes in the CDS instead of being broadcast by all the nodes in the network, thus reducing the number of unnecessary retransmissions.

Ad hoc networks are often represented as a unit disk graph [6], in which vertices represent wireless nodes and a bi-directional edge exists between two vertices if the corresponding nodes are within the transmission range of each other. A CDS is a sub graph of the undirected graph such that all nodes in the graph are included in the CDS or directly attached to a node (i.e., covered by the node) in the CDS. A minimum connected dominating set (MCDS) is the smallest CDS (in terms of number of nodes in the CDS) for the entire graph. For a virtual backbone-based route discovery, the smaller the size of the CDS, the smaller is the number of unnecessary retransmissions. If the RREQ packets are forwarded only by the nodes in the MCDS, we will have the minimum number of retransmissions. Unfortunately, the problem of determining the MCDS in an undirected graph like that of the unit disk graph is NP-complete. Efficient heuristics (e.g., [7][8][9]) that give preference to nodes with high neighborhood density (i.e., a larger number of uncovered neighbors) for inclusion in the MCDS have been proposed for wireless ad hoc networks. A common thread among these heuristics is to. The MaxD-CDS algorithm [10] studied in this paper is one such density-based heuristic earlier proposed by us.

In this paper, we show that aiming for the minimum number of nodes for the CDS in MANETs, results in CDSs that are highly unstable. The CDS itself has to be frequently rediscovered and this adds considerable overhead to the resource-constrained network. Our contribution is a minimum-velocity based CDS construction algorithm that gives preference to include slow-moving nodes (i.e., nodes with lower velocity) in the CDS rather than nodes that have high neighborhood density. The proposed algorithm, referred to as MinV-CDS, starts with the inclusion of the node having the lowest velocity, into the CDS. Once a node is added to the CDS, all its neighbors are said to be covered. The covered nodes are considered in the increasing order of their velocity, for inclusion in the CDS. If a node has lower velocity and is the next candidate node to be considered for inclusion in the CDS, it is added to the CDS if it has at least one neighbor that is yet to be covered. This procedure is repeated until all the nodes in the network are covered. The overall time complexity of the MinV-CDS algorithm is $O(|E| + |V|\log|V|)$ where $|V|$ and $|E|$ are the number of nodes and edges in the underlying ad hoc network graph, which could be a snapshot of the network at a particular time instant. A CDS is used as long as it exists. We outline an $O(|CDS\text{-}Node\text{-}List|^2 + |V|)$ algorithm to check the existence of a CDS at any particular time instant, where $|CDS\text{-}Node\text{-}List|$ is the number of nodes that are part of the CDS. Upon failure of the existing CDS, we again initiate the MinV-CDS algorithm to determine a new CDS.

We compare the performance of MinV-CDS with a maximum-density (MaxD-CDS) based algorithm that gives preference to nodes that have a larger number of uncovered neighbors for inclusion in the CDS. Simulation results illustrate that MinV-CDS has a significantly longer lifetime than MaxD-CDS. The tradeoff is an increase in the number of nodes and number of edges that are part of the MinV-CDS vis-à-vis MaxD-CDS. However, this helps the MinV-CDS to support a relatively lower hop count per source-destination path compared to MaxD-CDS.

The rest of the paper is organized as follows: Section 2 reviews related work in the literature on stable CDSs. Section 3 describes our MinV-CDS algorithm and also the MaxD-CDS algorithm with which the former is compared to. In addition, we outline an algorithm to check the existence of a CDS at any time instant and also show an example to illustrate the working of the MinV-CDS and MaxD-CDS. Section 4 presents the simulation environment and describes the simulation results comparing the performance of MinV-CDS with that of MaxD-CDS. Section 5 concludes the paper and discusses future work.

2. RELATED WORK

Very few algorithms are proposed in the literature to determine a stable connected dominating set for MANETs. In [2], the authors propose a localized algorithm, called maximal independent set with multiple initiators (MCMIS), to construct stable virtual backbones. MCMIS consists of two phases: In the first phase, a forest consisting of multiple dominating trees rooted at multiple initiators is constructed. A dominating tree, rooted at an initiator node, comprises of a subset of the nodes in the network topology. Multiple dominating trees, each started by its initiator, are constructed in parallel. In the second phase, dominating trees, with overlapping branches are interconnected to form a complete virtual backbone. Nodes are ranked according to the tuple (stability, effective degree, ID) and are considered as candidate nodes to be initiators, in decreasing order of importance.

A novel mobility handling algorithm proposed in [3] shortens the recovery time of CDS (i.e., CDS membership changes) in the presence of node mobility and also maintains a lower CDS size. In [4], the authors describe an algorithm to calculate stable CDS based on link-stability for MANETs. According to this algorithm, a link is said to be non-weak if the strength of the beacon signals received on that link is above a threshold. For inclusion in the stable CDS, nodes are considered in the decreasing order of the number of non-weak links associated with the node.

In [5], the authors propose a distributed topology management algorithm that constructs and maintains a minimal dominating set (MDS) of the network. MDS members connect to form a CDS, used as the backbone infrastructure for network communication. Each node self-decides the membership of itself and its neighbors in the MDS based on the two-hop neighborhood information disseminated among neighboring nodes.

In [10], we had proposed a centralized algorithm, referred to as *OptCDSTrans*, to determine a sequence of stable static connected dominating sets for MANETs. Algorithm *OptCDSTrans* operates according to a simple greedy principle, described as follows: whenever a new CDS is required at time instant *t*, we choose the longest-living CDS from time *t*. The above strategy when repeated over the duration of the simulation session yields a sequence of long-living stable static connected dominating sets such that the number of CDS transitions (change from one CDS to another) is the global minimum. Some of the distinguishing characteristics of *OptCDSTrans* are that the optimal number of CDS transitions does not depend on the underlying algorithm or heuristic used to determine the static CDSs and the greedy principle behind *OptCDSTrans* is very generic such that it can be applied to determine the stable sequence of any communication structure (for example, paths or trees) as long as there is a heuristic or algorithm to determine that particular communication structure in a given network graph [11].

3. ALGORITHMS TO DETERMINE MinV-CDS AND MaxD-CDS

3.1 Data Structures

We maintain four principal data structures:

(i) *MinV-CDS-Node-List* – includes all the nodes that are part of the minimum-velocity based CDS.

(ii) *Covered-Nodes-List* – includes nodes that either in the *MinV-CDS-Node-List* or covered by a node in the *MinV-CDS-Node-List*

(iii) *Uncovered-Nodes-List* – includes all the nodes that are not covered by a node in the *MinV-CDS-Node-List*

(iv) *Priority-Queue* – includes nodes that are in the *Covered-Nodes-List* and are probable candidates for addition to the *MinV-CDS-Node-List*. This list is sorted in the decreasing order of the velocity of the nodes. A dequeue operation returns the node with the lowest velocity.

3.2 Algorithm to Determine the Minimum Velocity-based Connected Dominating Set (MinV-CDS)

The MinV-CDS (pseudo code in Figure 1) is primarily constructed as follows: The *Start Node* is the first node to be added to the *MinV-CDS-Node-List*. As a result of this, all the neighbors of the *Start Node* are said to be covered: 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 has the lowest velocity and is not yet in the *MinV-CDS-Node-List*. If there is at least one neighbor node *u* of node *s* that is yet to be covered, all such nodes *u* are removed from the *Uncovered-Nodes-List* and added to the *Covered-Nodes-List* and to the *Priority-Queue*; node *s* is also added to the *MinV-CDS-Node-List*. If all neighbors of node *s* are already covered, then node *s* is not added to the *MinV-CDS-Node-List*. The above procedure is repeated until the *Uncovered-Nodes-List* becomes empty or the *Priority-Queue* becomes empty. If the *Uncovered-Nodes-List* becomes empty, then all the nodes in the network are covered. 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. During a dequeue operation, if two or more nodes have the same lowest velocity, we choose the node with the larger number of uncovered neighbors. If the tie cannot be still broken, we randomly choose to dequeue one of these candidate nodes.

Input: Snapshot of the Network Graph G = (*V*, *E*), where *V* is the set of vertices and *E* is the set of edges

Auxiliary Variables and Functions:

MinV-CDS-Node-List, Covered-Nodes-List, Uncovered-Nodes-List, Priority-Queue, minVelocity

Dequeue(*Priority-Queue*) – Extracts the node with the minimum velocity from the queue – if two or more nodes have the same minimum velocity, then a node is randomly chosen and extracted from the queue.

Neighbors(*s*) – List of neighbors of node *s* in graph *G*

velocity(*u*) – the velocity (in m/s) of node *u*

startNode – the first node to be added to *MinV-CDS-Node-List*

Output: *MinV-CDS-Node-List* // contains the list of nodes part of the minimum velocity – based CDS.

Initialization:

MinV-CDS-Node-List = Φ; *Covered-Nodes-List* = Φ; *Priority-Queue* = Φ; *Uncovered-Nodes-List* = *V*

Begin Construction of MinV-CDS
  // To Determine the *Start Node*
  minVelocity = ∞
  for every vertex *u* ∈ *V* **do**
    if (*minVelocity* > *velocity*(*u*)) **then**
      minVelocity = *velocity*(*u*)
      startNode = *u*
    end if
  end for
  // Initializing the data structures
  MinV-CDS-Node-List = {*startNode*}
  Priority-Queue = {*startNode*}
  Covered-Nodes-List = {*startNode*}
  Uncovered-Nodes-List = *Uncovered-Nodes-List* – {*startNode*}
  // Constructing the MinV-CDS-Node-List
  while (*Uncovered-Nodes-List* ≠ Φ and *Priority-Queue* ≠ Φ) **do**
    node *s* = Dequeue(*Priority-Queue*)
    alreadyCovered = true // to test whether all neighbors of node *s* have already been covered or not
    for all node *u* ∈ Neighbors(*s*) **do**
      if (*u* ∈ *Uncovered-Nodes-List*) **then**
        alreadyCovered = false
        Uncovered-Nodes-List = *Uncovered-Nodes-List* – {*u*}
        Covered-Nodes-List = *Covered-Nodes-List* U {*u*}
        Priority-Queue = *Priority-Queue* U {*u*}
      end if
    end for
    if (*alreadyCovered* = false) **then**
      MinV-CDS-Node-List = *MinV-CDS-Node-List* U {*s*}
    end if
  end while
  return *MinV-CDS-Node-List*
End Construction of MinV-CDS

Figure 1. Pseudo Code for MinV-CDS Construction Algorithm

3.3 Algorithm to Determine the Maximum Density-based Connected Dominating Set (MaxD-CDS)

The MaxD-CDS algorithm works similar to that of the MinV-CDS algorithm. The major difference is that the criterion for including nodes in the CDS is the number of uncovered neighbors and not the node velocity. The *Start Node* is the node with the maximum number of uncovered neighbors. In subsequent iterations, we dequeue the node with the maximum number of uncovered neighbors from the *Priority-Queue*. Ties are broken arbitrarily. The procedures to update the *Covered-Nodes-List* and the *Uncovered-Nodes-List* are the same as in MinV-CDS.

3.4 Time Complexity of MinV-CDS and MaxD-CDS

If we use a binary heap for maintaining the *Priority-Queue* of |*V*| nodes, each dequeue and enqueue operation can be completed in O(log|*V*|) time; otherwise if the *Priority-Queue* is simply maintained as an array, each dequeue and enqueue operation takes O(|*V*|) time. Overall, all the |*V*| nodes and their associated |*E*| edges in the underlying network have to be explored for inclusion in the CDS. Assuming the *Priority-Queue* is implemented as a binary heap (as in our simulations), the overall time complexity of both the MinV-CDS and MaxD-CDS algorithms is O(|*E*| + |*V*|*log |*V*|).

3.5 Algorithm to Check the Existence of a CDS at any Time Instant

The algorithm to check the existence of a CDS (applicable for both MinV-CDS and MaxD-CDS) at a particular time instant *t* works as follows: Given the currently known list of nodes in the CDS (referred to as *CDS-Node-List*), we first construct the list of edges (referred to as *CDS-Edge-List*) that may exist at time instant *t* between any pair of nodes in the *CDS-Node-List*. An edge exists between any two nodes if and only if the Euclidean distance between the co-ordinates of these two nodes is less than or equal to the transmission range per node. We run the well-known Breadth First Search (BFS) algorithm [12] on the *CDS-Node-List* and *CDS-Edge-List* and examine whether the underlying CDS is connected or not. If the CDS is not connected, the algorithm returns false and a new run of the CDS construction algorithm is

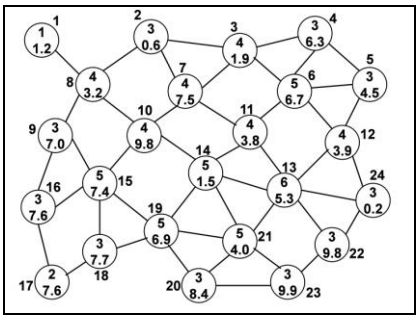

Figure 2.1. Initial Network

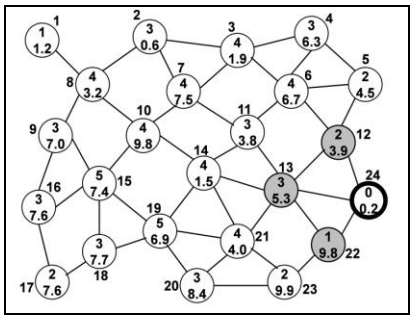

Figure 2.2. Iteration # 1

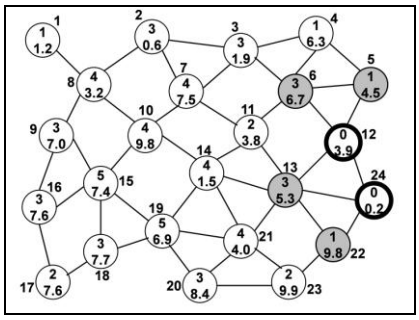

Figure 2.3. Iteration # 2

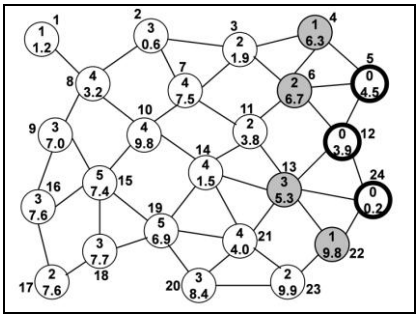

Figure 2.4. Iteration # 3

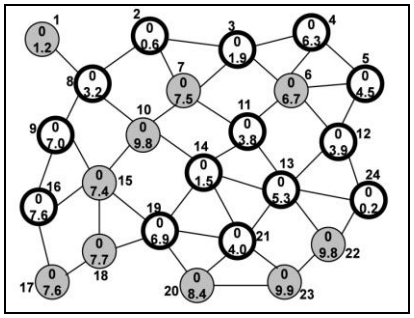

Figure 2.5. Final MinV-CDS (at the end of Iteration # 13)

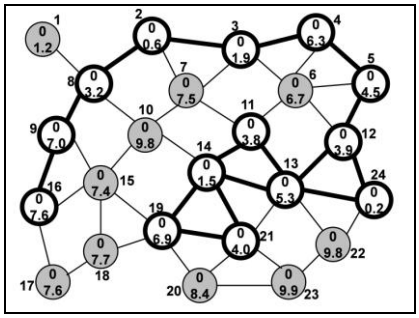

Figure 2.6. Edge List of the Final MinV-CDS

Figure 2. Example to Illustrate the Construction of the Minimum Velocity-based CDS (MinV-CDS)

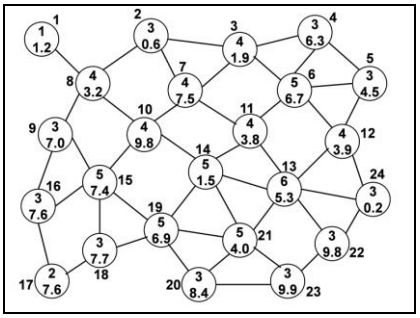

Figure 3.1. Initial Network

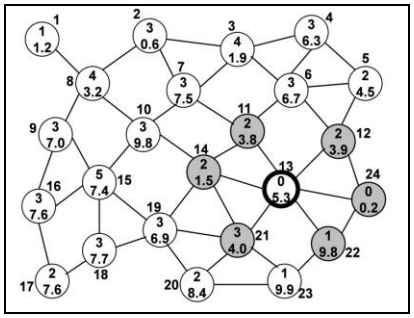

Figure 3.2. Iteration # 1

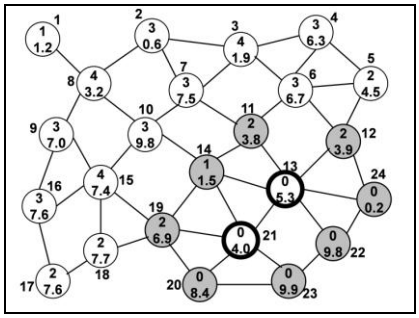

Figure 3.3. Iteration # 2

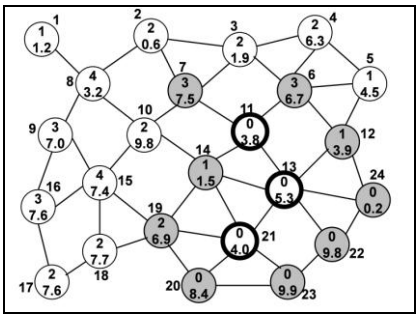

Figure 3.4. Iteration # 3

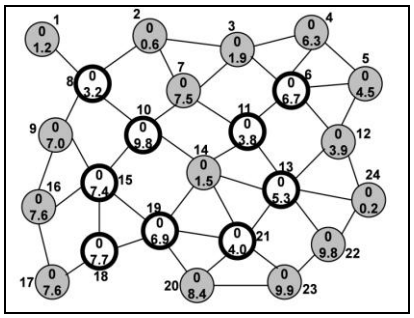

Figure 3.5. Final MinV-CDS (at the end of Iteration # 9)

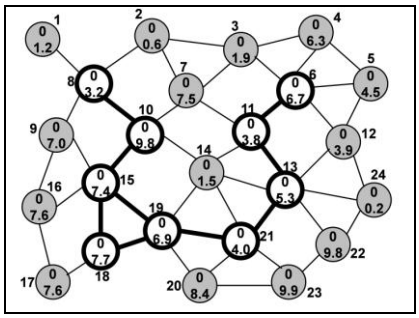

Figure 3.6. Edge List of the Final MaxD-CDS

Figure 3. Example to Illustrate the Construction of the Maximum Density-based CDS (MaxD-CDS)

initiated. If the CDS is connected, we then test whether every non-CDS node in the network is a neighbor of at least one CDS node. If there exists at least one non-CDS node that is not a neighbor of any CDS node at time *t*, the algorithm returns false – necessitating the instantiation of the appropriate CDS construction algorithm. If every non-CDS node has at least one CDS node as neighbor, the algorithm returns true – the current CDS covers the entire network and there is no need to determine a new CDS.

3.6 Example to Illustrate the Construction of MinV-CDS and MaxD-CDS

Figures 2 and 3 illustrate examples to demonstrate the working of the MinV-CDS and MaxD-CDS algorithms respectively. In these figures, each circle represents a node. The integer outside the circle represents the node ID and the integer inside the circle represents the number of uncovered neighbors of the corresponding node. The real-number inside the circle represents the velocity (in m/s) for the particular node. The nodes that are part of the CDS have their circles bold. We shade the circles of nodes that are covered, but are not part of the CDS. The circles of nodes that are not yet covered are neither shaded nor made bold.

On the 24-node network example considered in Figures 2 and 3, it takes respectively 13 and 9 iterations for the MinV-CDS and MaxD-CDS algorithms to find the CDS. The MinV-CDS includes 14 nodes and 16 edges; whereas the MaxD-CDS includes 9 nodes and 9 edges. Similar results have also been observed in our simulations. The MinV-CDS includes a relatively larger number of nodes and edges compared to the MaxD-CDS and this helps the former to sustain for a relatively longer lifetime as well as a lower hop count per source-destination path.

4. SIMULATIONS

All of the simulations are conducted in a discrete-event simulator developed by the author in Java. This simulator has also been successfully used in recent studies (e.g., [13][14][15]). The network topology is of dimensions 1000m x 1000m. The network density is represented as a measure of the average neighborhood size, which is calculated as follows: $N*\pi R^2/A$, where N is the number of nodes in the network, R is the transmission range of a node and A is the network area. The transmission range per node used in all of our simulations is 250 m. With a fixed transmission range and network area, the network density is varied from low to moderate and high by altering the number of nodes. We employ 50, 100 and 150 nodes to represent networks of low (average of 9.8 neighbors per node), moderate (average of 19.6 neighbors per node) and high (average of 29.4 neighbors per node) respectively. The network connectivity observed for these three networks at different conditions of node mobility is illustrated in Figure 4.

We use the Random Waypoint mobility model [16], according to which each node starts moving from an arbitrary location to a randomly selected destination with a randomly chosen speed in the range [v_{min} .. v_{max}]. Once the destination is reached, the node stays there for a pause time and then continues to move to another randomly selected destination with a different speed. We use v_{min} = 0 and pause time of a node is also set to 0. The values of v_{max} used are 5, 25 and 50 m/s representing low mobility, moderate mobility and high mobility levels respectively.

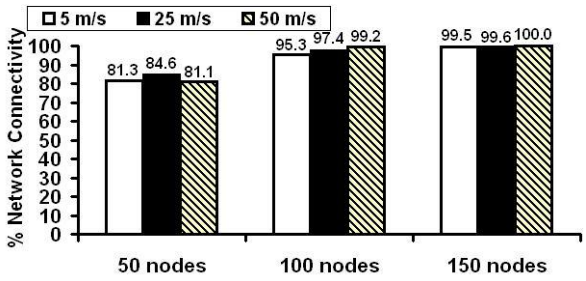

Figure 4. Average Percentage Network Connectivity

We obtain a centralized view of the network topology by generating mobility trace files for the simulation time (1000 seconds) under each of the above conditions. We sample the network topology for every 0.25 seconds. Two nodes *a* and *b* are assumed to have a bi-directional link at time *t*, if the Euclidean distance between them at time *t* (derived using the locations of the nodes from the mobility trace file) is within the wireless transmission range of the nodes. If a CDS does not exist for a particular time instant, we take a snapshot of the network topology at that time instant and run the appropriate CDS algorithm.

4.1 Performance Metrics

We measure the following performance metrics. Each data point in Figures 4 – 8 is an average computed over 10 mobility trace files and 15 *s-d* pairs from each of the mobility trace files. The starting time for each *s-d* session is uniformly distributed between 1 to 20 seconds.

- *CDS Node Size*: This is a time-averaged value of the number of nodes that are part of the CDS, determined by the MaxD-CDS and MinV-CDS algorithms. For example, if there exists a CDS of size 20 nodes, 23 nodes and 18 nodes in the network for 5, 10 and 5 seconds respectively, then the average CDS Node Size is (20*5 + 23*10 + 18*5)/(5 + 10 + 5) = 21.0 and not (20 + 23 + 18)/3 = 20.3.
- *CDS Edge Size*: This is a time-averaged value of the number of edges connecting the nodes that are part of the CDS, determined by the MaxD-CDS and MinV-CDS algorithms.
- *CDS Lifetime*: This is the time elapsed between the discovery of a CDS and its disconnection, averaged over the entire duration of the simulation.
- *Hop Count per Path*: This is the time-averaged hop count of individual source-destination (*s-d*) paths involving the CDS nodes as source, intermediate and destination nodes, averaged across all *s-d* paths over the entire simulation time.

4.2 CDS Node Size

The MinV-CDS, based on node velocity, includes more nodes (refer Figure 5) compared to the MaxD-CDS, based on node density. The maximum density-based CDS attempts to minimize

the number of nodes that are part of the CDS as it gives preference to nodes that have a larger number of uncovered neighbors over nodes that have a smaller number of uncovered neighbors. But, the minimum velocity-based CDS does not give much importance to the number of uncovered neighbors of a node before including the node in the *CDS-Node-List*.

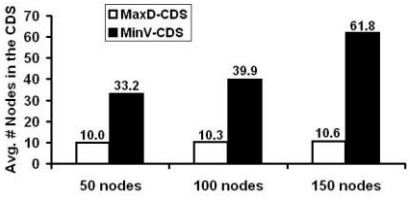

Figure 5.1. v_{max} = 5 m/s

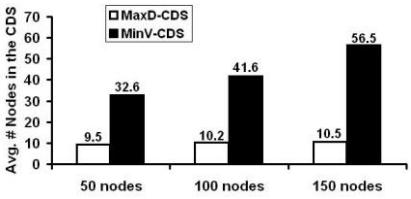

Figure 5.2. v_{max} = 25 m/s

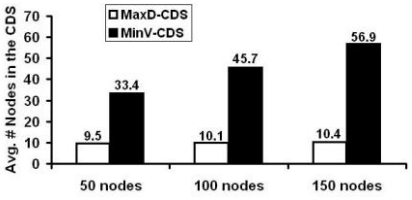

Figure 5.3. v_{max} = 50 m/s

Figure 5. CDS Node Size – Average Number of Nodes per MaxD-CDS and MinV-CDS

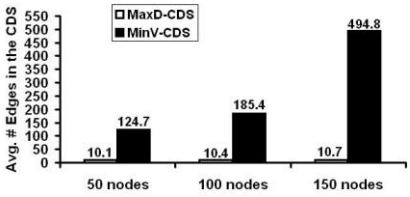

Figure 6.1. v_{max} = 5 m/s

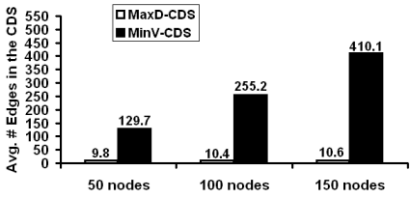

Figure 6.2. v_{max} = 25 m/s

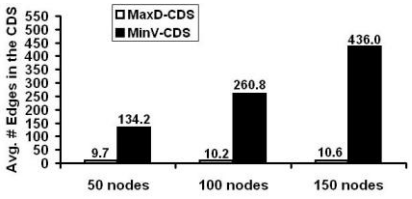

Figure 6.3. v_{max} = 50 m/s

Figure 6. CDS Edge Size – Average Number of Edges per MaxD-CDS and MinV-CDS

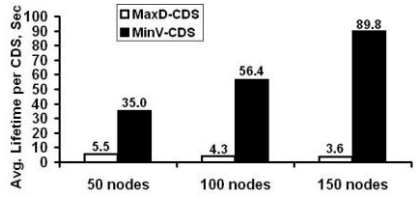

Figure 7.1. v_{max} = 5 m/s

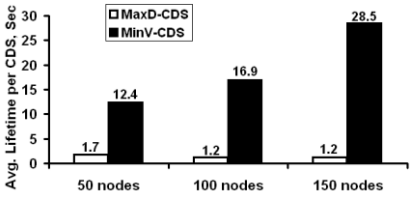

Figure 7.2. v_{max} = 25 m/s

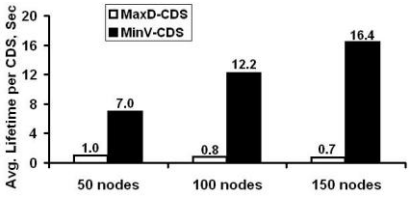

Figure 7.3. v_{max} = 50 m/s

Figure 7. Average Lifetime per MaxD-CDS and MinV-CDS

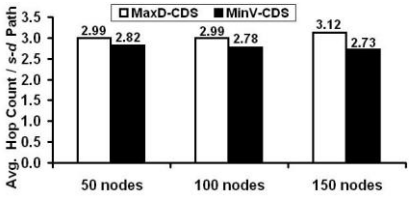

Figure 8.1. v_{max} = 5 m/s

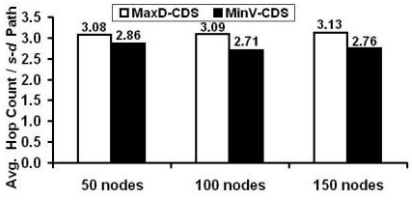

Figure 8.2. v_{max} = 25 m/s

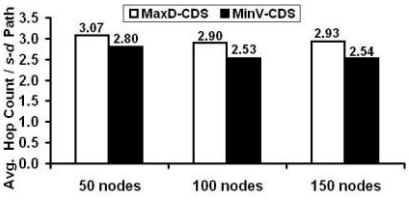

Figure 8.3. v_{max} = 50 m/s

Figure 8. Average Hop Count per Path in a MaxD-CDS and MinV-CDS

If a node has a lower velocity and is the next candidate node to be considered for inclusion (when the already covered nodes are considered in the increasing order of their velocity) in the *CDS-Node-List*, the low velocity node is added to the *CDS-Node-List* if it has at least one neighbor that is yet to be covered. As a result, the number of nodes in the *CDS-Node-List* is relatively high for the CDS based on minimum velocity.

With respect to the magnitude of the difference in the number of nodes in the *CDS-Node-List*, we observe that the Node Size for a MinV-CDS is 3.3 (low network density) to 5.8 (high network density) times larger than that of the Node Size for a MaxD-CDS. In the case of a MaxD-CDS, for fixed node mobility, as we increase node density from low to high, there is only at most a 10% increase in the Node Size. On the other hand, for the MinV-CDS, for fixed node mobility, as we increase the node density from low to high, the Node Size can increase as large as by 190%. This can be attributed to the relative insensitivity of the MinV-CDS based algorithm to consider the number of uncovered neighbors of a node before including the node in the CDS. A long-living stable CDS is eventually formed by including more nodes to be part of the CDS. While, even if the network density

is tripled, the MaxD-CDS algorithm manages to cover all the nodes in the high-density network by incurring only at most a 10% increase in the CDS Node Size, compared to that for a low-density network. For a given node density, as we increase the node mobility from low to high, the Node Size for a MaxD-CDS does not change appreciably, whereas the Node Size for a MinV-CDS changes by at most 15%.

4.3 CDS Edge Size

The MaxD-CDS algorithm, in its attempt to minimize the CDS Node Size, chooses CDS nodes that are far away from each other such that each node covers as many uncovered neighbors as possible. As the CDS nodes are more likely to be away from each other, spanning the entire network, the number of edges (Edge Size) between the MaxD-CDS nodes is very low. On the other hand, since the MinV-CDS algorithm incurs a larger Node Size because of its relative insensitivity to the number of uncovered neighbors of a node, there is a corresponding increase in the number of edges (refer Figure 6) between these CDS nodes.

With respect to the magnitude of the difference in the number of edges among the CDS nodes, we observe that the Edge Size for a MinV-CDS is 12.4 (low network density) to 46.0 (high network density) times larger than that of the Edge Size for a MaxD-CDS. In the case of a MaxD-CDS, for fixed node mobility, as we increase the node density from low to high, there is only at most a 7% increase in the Edge Size. On the other hand, for the MinV-CDS, at fixed node mobility, as we increase the node density from low to high, the Edge Size increases as large as by 400%. This can be attributed to the huge increase (as large as by 190%) in the MinV-CDS Node Size, with increase in network density. The increase in the number of edges and nodes significantly contribute to the increase in the MinV-CDS lifetime (refer Section 4.4) as the network density is increased. For a given node density, as we increase the node mobility from low to high, the Edge Size for a MaxD-CDS does not change appreciably, whereas the Edge Size for a MinV-CDS also changes by at most 40%.

4.4 CDS Lifetime

In the case of MinV-CDS, the relatively larger CDS Node Size and Edge Size significantly contribute to the lifetime of the CDS (refer Figure 7). As the constituent nodes of the MinV-CDS are chosen based on the minimum velocity metric, the edges between the CDS nodes are bound to exist for a relatively longer time and the connectivity of the nodes that are part of the MinV-CDS is likely to be maintained for a longer time. On the other hand, the MaxD-CDS algorithm chooses nodes that are far away from each other (but still maintain an edge between them) as part of the CDS. The edges between such nodes are likely to fail sooner, leading to loss of connectivity between the nodes that are part of the MaxD-CDS. We thus observe a tradeoff between the CDS Node Size and the CDS Lifetime. If we meticulously choose slow-moving nodes to be part of the CDS, the lifetime of the CDS could be significantly improved, at the expense of the Node Size. On the other hand, if we aim to select a CDS with the minimum number of nodes required to cover all the nodes in the network, the lifetime of the CDS would be significantly lower.

With respect to the magnitude, the lifetime per MinV-CDS is 6 (low network density) to 25 (high network density) times more than that of the MaxD-CDS. The relatively high stability of MinV-CDS at high network density can be attributed to the inclusion of a significantly larger number of slow-moving nodes and their associated edges as part of the CDS. The relatively poor stability of MaxD-CDS at high network density can be attributed to the need to cover a larger number of nodes in the network without any significant increase in the number of nodes that are part of the CDS. For both MaxD-CDS and MinV-CDS, for a fixed network density, as we increase node mobility from low (v_{max} = 5 m/s) to moderate (v_{max} = 25 m/s), and from low (v_{max} = 5 m/s) to high (v_{max} = 50 m/s), the lifetime per CDS decreases by a factor of 3.3 and 5.3 respectively.

4.5 Hop Count per Path

The average hop count per path (refer Figure 8) between an *s-d* pair through the nodes that are part of the MaxD-CDS is 1.06 (at low network density) to 1.15 (at moderate and high network density) more than that incurred with MinV-CDS. The relatively lower hop count per *s-d* path, in the case of a MinV-CDS, can be attributed to the larger CDS Node Size and the presence of a larger number of edges connecting the CDS nodes. Hence, the MinV-CDS can have several *s-d* paths between any two nodes *s* and *d* in the network and we choose the minimum hop *s-d* path among them while computing the average hop count per path. On the other hand, with fewer edges in the MaxD-CDS, the paths between any two nodes through the nodes of the MaxD-CDS will have a relatively larger hop count.

The consequences of having larger hop count per path with a fewer number of nodes per MaxD-CDS are a larger end-to-end delay per data packet and unfairness of node usage. Nodes that are path of the MaxD-CDS could be relatively heavily used compared to the nodes that are not part of the MaxD-CDS. This could lead to premature failure of critical nodes, mainly nodes lying in the center of the network, resulting in reduction in network connectivity, especially in low-density networks. With MinV-CDS, as multiple nodes are part of the CDS, the packet forwarding load can be distributed across several nodes and this could enhance the fairness of node usage and help to incur a relatively lower end-to-end delay per data packet.

5. CONCLUSIONS AND FUTURE WORK

Ours is the first work to formulate an algorithm to determine stable connected dominating sets for mobile ad hoc networks, exclusively based on node velocity. Through extensive simulations, we demonstrate that the proposed algorithm, MinV-CDS, can determine connected dominating sets that have a significantly longer lifetime compared to that of the maximum density-based MaxD-CDS algorithm. The MinV-CDS also has a relatively larger number of constituent nodes and edges and this helps to reduce the hop count per path as well as the end-to-end delay and improves the fairness of node usage. We thus observe a tradeoff between the CDS Node Size and the CDS Lifetime. If we meticulously choose slow-moving nodes to be part of the CDS, the lifetime of the CDS could be significantly improved, at the expense of the Node Size. On the other hand, if we aim to choose a CDS with the minimum number of nodes required to

cover all the nodes in the network, the lifetime of the CDS would be significantly lower.

In terms of magnitude, the lifetime per MinV-CDS is 6 (low network density) to 25 (high network density) times more than that of MaxD-CDS. The Node Size for a MinV-CDS is 3.3 (low network density) to 5.8 (high network density) times larger than that of the Node Size for a MaxD-CDS. The Edge Size for a MinV-CDS is 12.4 (low network density) to 46.0 (high network density) times larger than that of the Edge Size for a MaxD-CDS. The average hop count per path between an *s-d* pair through the nodes that are part of the MaxD-CDS is 1.06 (at low network density) to 1.15 (at moderate and high network density) more than that incurred with MinV-CDS.

As future work, we will study the performance of MinV-CDS along with that of the theoretically optimal *OptCDSTrans* algorithm and compare the lifetimes of the minimum velocity-based connected dominating sets and the stable mobile connected dominating sets. Future work would also involve developing a distributed implementation of the MinV-CDS algorithm and explore its use as a virtual backbone for unicast, multicast and broadcast communication in MANETs.

6. REFERENCES

[1] Sinha, P., Sivakumar, R., and Bhargavan, V. 2001. Enhancing Ad hoc Routing with Dynamic Virtual Infrastructures. In Proceedings of the 20th IEEE INFOCOM Conference, 3, 1763-1772.

[2] Wang, F., Min, M., Li, Y., and Du, D. 2005. On the Construction of Stable Virtual Backbones in Mobile Ad hoc Networks. In Proceedings of the IEEE International Performance Computing and Communications Conference.

[3] Sakai, K., Sun, M.-T., and Ku, W.-S. 2008. Maintaining CDS in Mobile Ad hoc Networks. Lecture Notes in Computer Science. 5258 (Oct. 2008), 141-153.

[4] Sheu, P.-R., Tsai, H.-Y., Lee, Y.-P., and Cheng, J. Y. 2009. On Calculating Stable Connected Dominating Sets Based on Link Stability for Mobile Ad hoc Networks. Tamkang Journal of Science and Engineering. 12, 4, 417-428.

[5] Bao, L., and Garcia-Luna-Aceves, J. J. 2010. Stable Energy-aware Topology Management in Ad hoc Networks. Ad hoc Networks. 8, 3 (May 2010), 313-327.

[6] Kuhn, F., Moscibroda, T., and Wattenhofer, R. 2004. Unit Disk Graph Approximation. In Proceedings of the ACM DIALM-POMC Joint Workshop on the Foundations of Mobile Computing, 17-23.

[7] Alzoubi, K. M., Wan, P.-J., and Frieder, O. Distributed Heuristics for Connected Dominating Set in Wireless Ad hoc Networks. 2002. IEEE / KICS Journal on Communication Networks. 4, 1, 22-29.

[8] Butenko, S., Cheng, X., Du, D.-Z., and Paradlos, P. M. 2002. On the Construction of Virtual Backbone for Ad hoc Wireless Networks. Cooperative Control: Models, Applications and Algorithms, 43-54.

[9] Butenko, S., Cheng, X., Oliviera, C., and Paradlos, P. M. 2004. A New Heuristic for the Minimum Connected Dominating Set Problem on Ad hoc Wireless Networks. Recent Developments in Co-operative Control and Optimization, 61-73.

[10] Meghanathan, N. 2006. An Algorithm to Determine the Sequence of Stable Connected Dominating Sets in Mobile Ad hoc Networks. In Proceedings of the 2nd Advanced International Conference on Telecommunications.

[11] Meghanathan, N., and Farago, A. 2008. On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad hoc Networks. Ad hoc Networks. 6, 5 (Jul. 2008), 744-769.

[12] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. 2001. Introduction to Algorithms. 2nd Edition, MIT Press.

[13] Meghanathan, N., and Sugumar, M. 2010. A Beaconless Minimum Interference Based Routing Protocol to Minimize End-to-End Delay per Packet for Mobile Ad hoc Networks. International Journal of Interdisciplinary Telecommunications and Networking. 2, 1 (Mar. 2010), 12-26.

[14] Meghanathan, N., and Odunsi, A. 2010. Investigating the Scalability of the Fish-eye State Routing Protocol for Ad hoc Networks. Journal of Theoretical and Applied Information Technology. 12, 1 (Feb. 2010), 60-70.

[15] Meghanathan, N. 2009. Multicast Extensions to the Location Prediction Based Routing Protocol for Mobile Ad hoc Networks. Lecture Notes of Computer Science, 5682, 190-199.

[16] Bettstetter, C., Hartenstein, H., and Perez-Costa, X. 2004. Stochastic Properties of the Random-Way Point Mobility Model. Wireless Networks. 10, 5 (Sep. 2004), 555-567.