Predicted Link Expiration Time Based Connected Dominating Sets for Mobile Ad hoc Networks

Pervis Fly
Undergraduate Student, Dept. of Computer Science
Jackson State University
Jackson, MS 39217, USA

Natarajan Meghanathan
Assistant Professor, Dept. of Computer Science
Jackson State University
Jackson, MS 39217, USA

Abstract— We propose an algorithm to determine stable connected dominating sets (CDS), based on the predicted link expiration time (LET), for mobile ad hoc networks (MANETs). The proposed LET-based CDS algorithm is the first such algorithm that constructs a CDS based on edge weights represented by predicted link expiration time, rather the traditional approach of using node weights like the well-known maximum density-based CDS (MaxD-CDS) algorithm. The construction of the LET-CDS starts with the inclusion of the edge having the largest predicted link expiration time, into the CDS. Once an edge is included as the CDS edge list, the two constituent nodes of the edge becomes part of the CDS node list. The neighbors and the edges incident to either one or both the end nodes of the CDS edge are also said to be covered. The covered edges are considered in the increasing order of their predicted link expiration time, for inclusion in the CDS. If an edge has a higher predicted expiration time and is the next candidate edge to be considered for inclusion into the CDS, it is added to the CDS edge list if either one or both of the end nodes of the edge has at least one neighbor node that is yet to be covered. This procedure is repeated until all the nodes in the network are covered. Simulation results illustrate that the LET-CDS has a longer lifetime compared to the MaxD-CDS, especially in networks of moderate and high density. The LET-CDS also has a larger number of nodes and edges compared to the 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.

Keywords-Link Expiration Time; Stability; Connected Dominating Sets; Mobile Ad hoc Networks; Simulations

I. 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. Broadcast communication in MANETs has been traditionally accomplished through flooding in which each wireless node receives a copy of the broadcast message from all of its neighbors and is also responsible for forwarding the message exactly once to all of its neighbors. Recent studies (e.g., [1][2][3][4][5]) demonstrate the use of connected dominating set (CDS)-based virtual backbones to propagate the broadcast messages so that they 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 broadcast communication, the smaller the size of the CDS, the smaller is the number of unnecessary retransmissions. If the broadcast messages 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. The MaxD-CDS algorithm [10] studied in this paper is one such density-based heuristic earlier proposed by the co-author of this paper. Throughout the paper, the terms ‘link’ and ‘edge’, ‘node’ and ‘vertex’, ‘message’ and ‘packet’, ‘path’ and ‘route’ have been used interchangeably. They mean the same.

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, especially with increase in network density and node mobility. The CDS itself has to be frequently rediscovered and this adds considerable overhead to resource-constrained network. Our contribution in this paper is a predicted link expiration time (LET)-based CDS construction algorithm that gives preference to include links (and their associated end nodes) that could exist for a longer time in the CDS rather than nodes that have high neighborhood density. To the best of our knowledge, ours is the first such approach to construct a CDS based on the link weights rather than node weights (e.g. the MaxD-CDS algorithm). The proposed LET-CDS algorithm starts with the inclusion of an edge having the largest predicted expiration time, into the CDS Edge List. Once an edge is added to the CDS, the two constituent end nodes of the edges are part of the CDS Node List and all their neighbors and the incident edges are said to be covered. The covered edges are considered in the increasing order of their predicted LET, for inclusion in the CDS. If an edge has a larger predicted LET and is the next candidate edge to be considered for inclusion in the CDS, it is added to the CDS Edge List if either of its end nodes can cover at least one of their neighbor nodes 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 LET-CDS algorithm is O(|*VE*| + |*E* log E|) where |*V*| and |*E*| are the number of nodes and edges in the underlying network graph, which could be a snapshot of the network at a particular time instant. A CDS is used as long as it exists.

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, especially with increase in network density and/or node mobility. The CDS itself has to be frequently rediscovered and this adds considerable overhead to the resource-constrained network. Our contribution in this paper is a predicted link expiration time (LET)-based CDS construction algorithm that gives preference to include links (and their associated end nodes) that could exist for a longer time in the CDS rather than nodes that have high neighborhood density. To the best of our knowledge, ours is the first such approach to construct a CDS based on the link weights rather than node weights (e.g. the MaxD-CDS algorithm). The proposed LET-CDS algorithm starts with the inclusion of an edge having the largest predicted expiration time, into the CDS Edge List. Once an edge is added to the CDS, the two constituent end nodes of the edges are part of the CDS Node List and all their neighbors and the incident edges are said to be covered. The covered edges are considered in the increasing order of their predicted LET, for inclusion in the CDS. If an edge has a larger predicted LET and is the next candidate edge to be considered for inclusion in the CDS, it is added to the CDS Edge List if either of its end nodes can cover at least one of their neighbor nodes 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 LET-CDS algorithm is O(|*E* log *E*|) where |*E*| are the number of nodes and edges in the underlying 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 compare the performance of LET-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 LET-CDS has a relatively longer lifetime than MaxD-CDS with increase in network density and/or node mobility. The tradeoff is an increase in the number of nodes and number of edges that are part of the LET-CDS vis-à-vis MaxD-CDS. However, this helps the LET-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 LET-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 LET-CDS and MaxD-CDS. Section 4 presents the simulation environment and describes the simulation results comparing the performance of LET-CDS with that of MaxD-CDS. Section 5 concludes the paper and discusses future work.

II. 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]. In [12], the co-author had proposed a minimum velocity-based stable CDS (MinV-CDS) algorithm according to which slow moving nodes are preferred for inclusion into the CDS. The minimum velocity-based connected dominating sets existed for a significantly longer lifetime compared to the maximum density-based connected dominating sets; the tradeoff being a larger number of nodes and edges.

III. Algorithms to Determine LET-CDS and MaxD-CDS

A. *Prediction of the Link Expiration Time (LET)*

Given the motion parameters of two neighboring nodes, the duration of time the two nodes will remain neighbors can be predicted as follows: Let two nodes *i* and *j* be within the transmission range of each other. Let (x_i, y_i) and (x_j, y_j) be the co-ordinates of the mobile hosts *i* and *j* respectively. Let v_i, v_j be the velocities and Θ_i, Θ_j, where ($0 \le \Theta_i, \Theta_j < 2\pi$) indicate the direction of motion of nodes *i* and *j* respectively. The amount of time the two nodes *i* and *j* will stay connected, $D_{i\text{-}j}$, can be predicted using the following equation:

$$D_{i-j} = \frac{-(ab+cd)+\sqrt{(a^2+c^2)r^2-(ad-bc)^2}}{a^2+c^2} \quad \ldots\ldots\ldots\ldots\text{ (1)}$$

where: $a = v_i \cos\Theta_i - v_j \cos\Theta_j$; $b = x_i - x_j$; $c = v_i \sin\Theta_i - v_j \sin\Theta_j$; $d = y_i - y_j$

B. *Data Structures*

We maintain the following four principal data structures:

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

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

(iii) *CDS-Edge-List* – includes all the edges that are part of the CDS: i.e., the edges that exist between any two nodes in the *CDS-Node-List*.

(iv) *Covered-Edge-List* – includes edges that are not in the *CDS-Edge-List*; but, one of its two end vertices is a node in the *CDS-Node-List*.

(v) *Priority Queue* – includes edges that are in the *Covered-Edge-List* and are probable candidates for addition to the *CDS-Edge-List*. This list is sorted in the decreasing order of the predicted LET of the edges. A dequeue operation returns the edge with the largest predicted LET.

C. *Algorithm to Determine the Predicted Link Expiration Time (LET)-based CDS*

The LET-CDS (pseudo code in Figure 1) is primarily constructed as follows: The *Start Edge* is the first edge to be added to the *CDS-Edge-List*. As a result of this, all the edges that are adjacent to the *Start Edge* are added to the *Covered-Edge-List* and to an appropriate entry in the *Priority-Queue*; both the end nodes of the *Start Edge* are added to the *CDS-Node-List* and their neighbors are added to the *Covered-Node-List*. If the size of the *Covered-Node-List* is less than the number of nodes in the network (i.e., not all nodes in the network are yet covered) and the *Priority-Queue* is not empty, we dequeue the *Priority-Queue* to extract an edge *(uMaxE, vMaxE)* that has the largest predicted LET and is not yet in the *CDS-Edge-List*. If there is at least one node that has an edge with *uMaxE* or *vMaxE* and is not yet part of the *Covered-Node-List*, then the edge *(uMaxE, vMaxE)* is removed from the *Covered-Edge-List* and added to the *CDS-Edge-List*; the nodes that are newly covered with the inclusion of *(uMaxE, vMaxE)* to the *CDS-Edge-List* are added to the *Covered-Node-List* and their associated edges that are adjacent to *(uMaxE, vMaxE)* are added to the *Covered-Edge-List*. If there is no node that could be newly covered through *(uMaxE, vMaxE)*, then the edge is not added to the *CDS-Edge-List*. The above procedure is repeated until the size of the *Covered-Node-List* is less than the number of nodes in the network or the *Priority-Queue* becomes empty. If the size of the *Covered-Node-List* is equal to the number of nodes in the network, then all the nodes in the network are covered. If the *Priority-Queue* becomes empty and the *Covered-Node-List* does not have at least one node in the network, then the underlying network is considered to be disconnected. During a dequeue operation, if two or more edges have the same maximum predicted LET, we choose the edge that can bring in more nodes to the *Covered-Node-List*. If the tie cannot be still broken, we randomly choose to dequeue one of these candidate edges.

```
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:
CDS-Node-List, CDS-Edge-List, Covered-Node-List, Covered-
Edge-List, Priority-Queue, maxLETEdge
Neighbors(s) – List of neighbors of node s in graph G
LET(u–v)  – the predicted link expiration time (in seconds) of
            an edge u – v
startEdge – the first edge (maximum weight edge) to be added
            to CDS-Edge-List
Sorted-Edge-List = List of edges in E, sorted in the decreasing
                   order of LET
Output: CDS-Node-List // contains the list of nodes part of the
                         predicted LET-based CDS
Initialization:
CDS-Node-List = Φ; CDS-Edge-List = Φ; Covered-Node-List
= Φ; Covered-Edge-List = Φ; Priority-Queue = Φ

Begin Construction of LET-CDS
   startEdge = Edge with max. weight in Sorted-Edge-List
   Priority-Queue = Priority-Queue U {startEdge}
   Covered-Edge-List = Covered-Edge-List U {startEdge}
  while (|Covered-Node-List|<|V| && Priority-Queue ≠ Φ) do
      Extract the maximum weight edge (uMaxE, vMaxE)
                                          ∈ Priority-Queue
      boolean additionNeeded-uMaxE = false
      boolean additionNeeded-vMaxE = false
      for every edge (a, uMaxE) ∈ E or (uMaxE, b) ∈ E do
         if (a ∉ Covered-Node-List) or (b ∉ Covered-Node-
                                                  List) then
            Appropriately add node a or b to Covered-Node-List
            additionNeeded-uMaxE = true
            Appropriately add (a, uMaxE) or (uMaxE, b) to
                    Priority-Queue
            Appropriately add (a, uMaxE) or (uMaxE, b) to
                    Covered-Edge-List
         end if
```

```
        end for
        for every edge (a, vMaxE) ∈ E or (vMaxE, b) ∈ E do
          if (a ∉ Covered-Node-List) or (b ∉ Covered-Node-List) then
            Appropriately add node a or b to Covered-Node-List
            additionNeeded-vMaxE = true
            Appropriately add (a, vMaxE) or (vMaxE, b) to Priority-Queue
            Appropriately add (a, uMaxE) or (uMaxE, b) to Covered-Edge-List
          end if
        end for
      if (additionNeeded-uMaxE || additionNeeded-vMaxE) then
      CDS-Edge-List = CDS-Edge-List U {(uMaxE, vMaxE)}
      Covered-Edge-List=Covered-Edge-List–{(uMaxE, vMaxE)}
        if (uMaxE ∉ CDS-Node-List) then
          CDS-Node-List = CDS-Node-List U {uMaxE}
          Covered-Node-List = Covered-Node-List U {uMaxE}
        end if
        if (vMaxE ∉ CDS-Node-List) then
          CDS-Node-List = CDS-Node-List U {vMaxE}
          Covered-Node-List = Covered-Node-List U {vMaxE}
        end if
      end if
    end While

return CDS-Node-List
```

Figure 1. Pseudo Code for the Algorithm to Construct the LET-based CDS

D. *Time Complexity of the LET-CDS Algorithm*

If we use a binary heap for maintaining the Priority-Queue of |*E*| edges, each dequeue and enqueue operation can be completed in O(|log*E*|) time. There could be O(|*E*|) of such operations/iterations of the outer while loop in Figure 1. Cumulative of all the |*E*| iterations, there would be O(|*E*|) edges explored for addition to the *Covered-Edge-List*. Hence, the overall time complexity of the LET-CDS algorithm is O(|*E*| + |*E* log*E*|) = O(|*E* log*E*|).

E. *Algorithm to Determine the Maximum Density-based Connected Dominating Set (MaxD-CDS)*

The MaxD-CDS algorithm is based on node weights – prefers to choose nodes with the maximum number of uncovered neighbors until all nodes in the network are covered. The condition to check for network connectivity is the same as that of the LET-CDS. If the size of the Covered-Node-List is less than the number of nodes in the network and the Priority-Queue that has the list of nodes with the maximum number of uncovered neighbors becomes empty, then the network is pronounced to be disconnected for that time instant and we attempt to determine a CDS for the network snapshot at the subsequent time instant. Ties to choose the next node for inclusion to the MaxD-CDS are broken arbitrarily.

F. *Algorithm to Check the Existence of a CDS at any Time Instant*

The algorithm to check the existence of a CDS at a particular time instant t works as follows: Given the *CDS-Node-List* and *CDS-Edge-List* at time *t*, 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 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 return *false* – necessitating the instantiation of the appropriate CDS construction algorithm. If every non-CDS node has at least one CDS node as neighbor, the algorithms return true – the current CDS covers the entire network and there is no need to determine a new CDS.

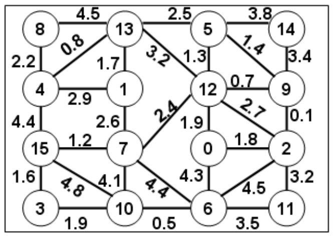
Figure 2.1. Initial Network

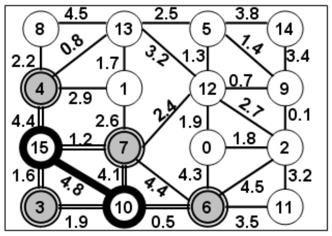
Figure 2.2. Iteration # 1

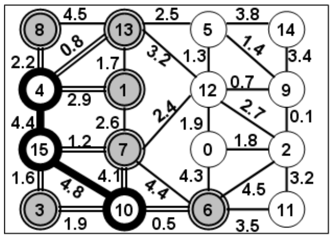
Figure 2.3. Iteration # 2

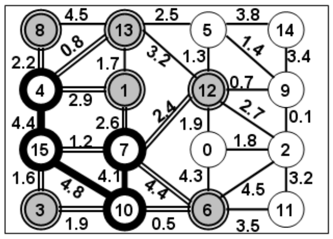
Figure 2.4. Iteration # 3

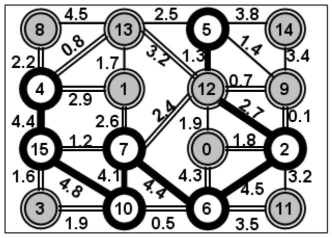
Figure 2.5. Final Iteration (# 6)

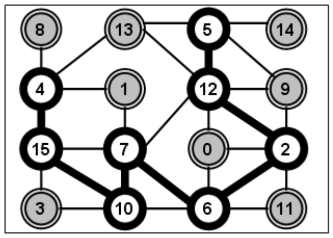
Figure 2.6. Final LET-CDS [8 CDS Nodes; 7 CDS Edges]

Figure 2. Example to Illustrate the Construction of LET-CDS

G. *Example to Illustrate the Construction of LET-CDS and MaxD-CDS*

Figures 2 and 3 illustrate examples to demonstrate the working of the LET-CDS and MaxD-CDS algorithms respectively. In these figures, each circle represents a node. In Figure 2, the integer inside the circle represents the node ID and the real number besides an edge represents the predicted

LET, the weight of the edge. In Figure 3, the integer outside the circle represents the number of uncovered neighbors of a node. In both these figures, the CDS nodes are represented with a thick black-colored circle with no shade inside the circle; the covered nodes are represented with double-bordered circles and gray-shaded inside; the uncovered nodes are represented with plain white circles. The CDS edges (edges between any two CDS nodes) are dark and bold; the covered edges (edges between a CDS node and a covered node) are double-lined; and the rest of the edges are plain. As illustrated in the examples, the LET-CDS incurs more nodes and edges that are part of the connected dominating set compared to the MaxD-CDS.

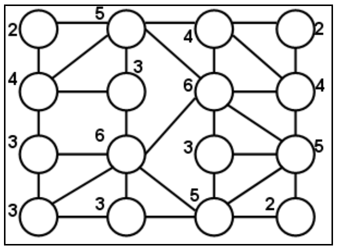
Figure 3.1. Initial Network

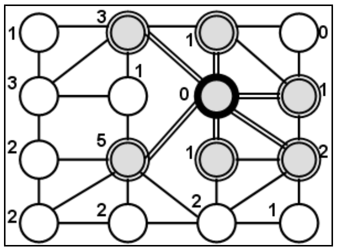
Figure 3.2. Iteration # 1

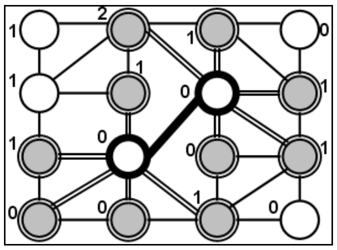
Figure 3.3. Iteration # 2

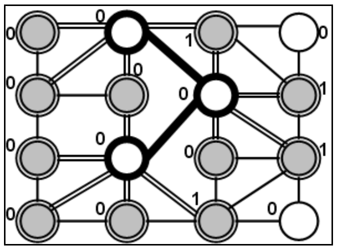
Figure 3.4. Iteration # 3

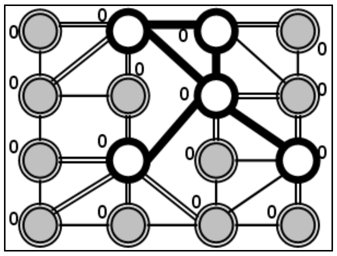
Figure 3.5. Final Iteration (# 5)

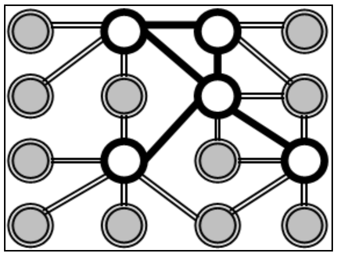
Figure 3.6. Final MaxD-CDS [5 CDS Nodes; 5 CDS Edges]

Figure 3. Example to Illustrate the Construction of MaxD-CDS

IV. Simulations

The simulations have been conducted in a discrete-event simulator developed by the authors in Java. 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 [13], 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.

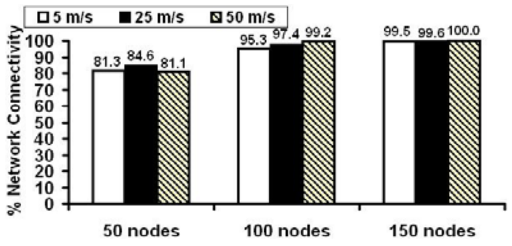

Figure 4. Average Percentage Network Connectivity

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.

A. *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 LET-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 LET-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.

B. *CDS Node Size*

The LET-CDS, based on the predicted link lifetime, 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 LET-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*.

If an edge has a larger predicted LET and is the next candidate node to be considered for inclusion (when the already covered edges are considered in the decreasing order of their predicted lifetime) in the *CDS-Edge-List*, the larger LET edge is added to the *CDS-Edge-List* if it lead to at least one uncovered node in the network to be covered. As a result, the number of nodes in the *CDS-Node-List* is relatively high for the CDS based on the predicted LET.

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 LET-CDS is 1.4 (low network density) to 2.0 (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 LET-CDS, for fixed node mobility, as we increase the node density from low to high, the Node Size can increase as large as by 43%. This can be attributed to the relative insensitivity of the LET-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.

C. *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 LET-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 LET-CDS is 2.5 (low network density) to 4.0 (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 LET-CDS, at fixed node mobility, as we increase the node density from low to high, the Edge Size increases as large as by 70%. This can be attributed to the increase in the LET-CDS Node Size, with increase in network density. The increase in the number of edges and nodes significantly contribute to the increase in the LET-CDS lifetime (refer Section 4.4) as the network density is increased.

D. *CDS Lifetime*

In the case of LET-CDS, the relatively larger CDS Node Size and Edge Size significantly contribute to a larger lifetime of the CDS (refer Figure 7). As the constituent nodes of the LET-CDS are chosen based on the larger predicted link lifetime 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 LET-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 stable edges 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 LET-CDS is 5% (low network density) to 86% (high network density) times more than that of the MaxD-CDS. The relatively high stability of LET-CDS at high network density can be attributed to the inclusion of a significantly larger number of stable CDS edges. 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.

E. *Hop Count per Source-Destination Path*

The average hop count per path (refer Figure 8) between a source-destination (*s-d*) pair through the nodes that are part of the MaxD-CDS is almost the same as that of LET-CDS (even sometimes lower) at low network density; but, could be at most 16% more than that incurred at high network density. The relatively lower hop count per *s-d* path, in the case of a LET-CDS at moderate and high network density, can be attributed to the larger CDS Node Size and the presence of a larger number of edges connecting the CDS nodes. Hence, the LET-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.

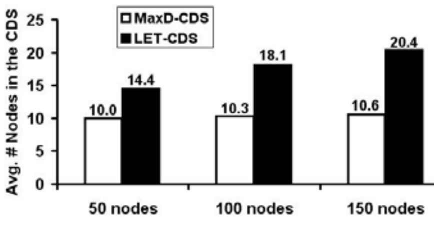
Figure 5.1. v_{max} = 5 m/s

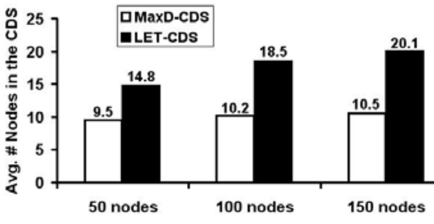
Figure 5.2. v_{max} = 25 m/s

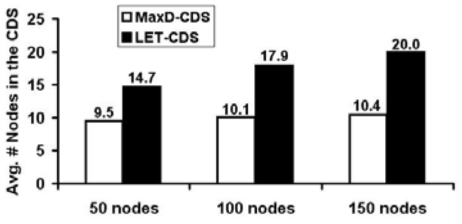
Figure 5.3. v_{max} = 50 m/s

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

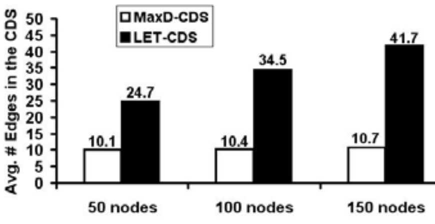
Figure 6.1. v_{max} = 5 m/s

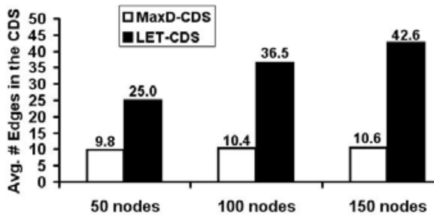
Figure 6.2. v_{max} = 25 m/s

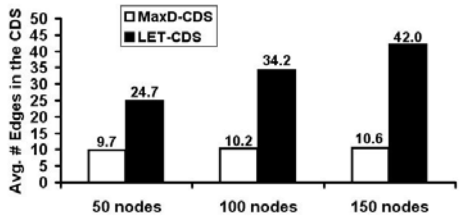
Figure 6.3. v_{max} = 50 m/s

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

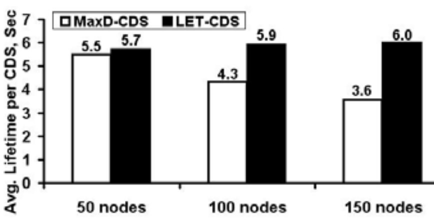
Figure 7.1. v_{max} = 5 m/s

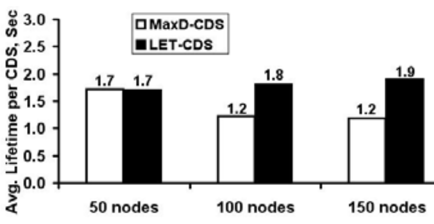
Figure 7.2. v_{max} = 25 m/s

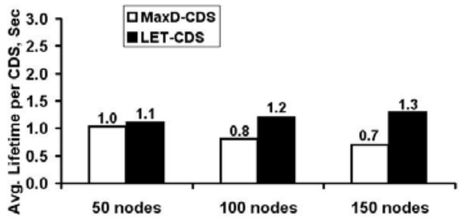
Figure 7.3. v_{max} = 50 m/s

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

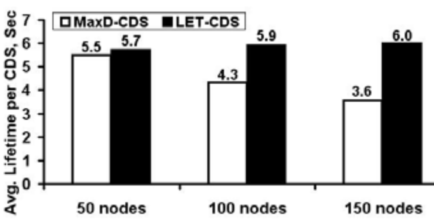
Figure 8.1. v_{max} = 5 m/s

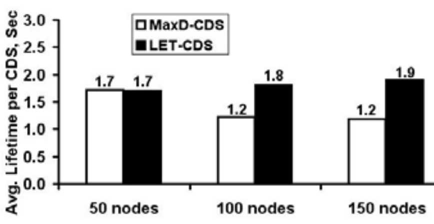
Figure 8.2. v_{max} = 25 m/s

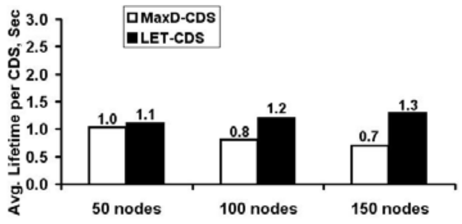
Figure 8.3. v_{max} = 50 m/s

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

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 LET-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.

V. 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 predicted link expiration time represented as edge weights. Through extensive simulations, we demonstrate that the proposed algorithm, LET-CDS, can determine connected dominating sets that could have a longer lifetime compared to that of the maximum density-based MaxD-CDS algorithm, especially with increase in network density and/or node mobility. The LET-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 stable edges 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.

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

ACKNOWLEDGMENT

This research is funded by the U.S. National Science Foundation through grant (CNS-0851646) entitled: "REU Site: Undergraduate Research Program in Wireless Ad hoc Networks and Sensor Networks," hosted by the Department of Computer Science at Jackson State University (JSU), MS, USA. The authors also acknowledge Dr. Loretta Moore, Dr. Xuejun Liang, Ms. Ilin Dasari and Mrs. Brenda Johnson (all at JSU) for their services to this program conducted during Summer 2010.

REFERENCES

[1] P. Sinha, R. Sivakumar and V. Bhargavan, "Enhancing Ad hoc Routing with Dynamic Virtual Infrastructures," *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM), vol. 3, pp. 1763 – 1772, 2001.

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

[3] K. Sakai, M.-T. Sun and W.-S. Ku, "Maintaining CDS in Mobile Ad hoc Networks," Wireless Algorithms Systems and Applications, *Lecture Notes in Computer Science*, vol. 5258, pp. 141 – 153, October 2008.

[4] P.-R. Sheu, H.-Y. Tsai, Y.-P. Lee and J. Y. Cheng, "On Calculating Stable Connected Dominating Sets Based on Link Stability for Mobile Ad hoc Networks," *Tamkang Journal of Science and Engineering*, vol. 12, no. 4, pp. 417 – 428, 2009.

[5] L. Bao and J. J. Garcia-Luna-Aceves, "Stable Energy-aware Topology Management in Ad hoc Networks," *Ad hoc Networks*, vol. 8, no. 3, pp. 313 – 327, May 2010.

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

[7] K. M. Alzoubi, P.-J Wan and O. Frieder, "Distributed Heuristics for Connected Dominating Set in Wireless Ad Hoc Networks," *IEEE / KICS Journal on Communication Networks*, Vol. 4, No. 1, pp. 22 – 29, 2002.

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

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

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

[11] N. Meghanathan and A. Farago, "On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad hoc Networks," *Ad hoc Networks*, vol. 6, no. 5, pp. 744 – 769, July 2008.

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

[13] C. Bettstetter, H. Hartenstein and X. Perez-Costa, "Stochastic Properties of the Random-Way Point Mobility Model," *Wireless Networks*, pp. 555 – 567, Vol. 10, No. 5, September 2004.

AUTHORS PROFILE

Pervis Fly is an undergraduate student (Senior) majoring in Computer Science at Jackson State University. The paper is based on the research work he conducted for the Research Experiences for Undergraduates (REU) program, sponsored by the U. S. National Science Founation at Jackson State University. During 2008-09, Pervis Fly also pariticipated in the NSF LSMAMP program for undergraduate research and has worked on an algorithm to determine ancestor-descendant relationships using binary coding. His research interests are in the areas of Algorithms, Wireless Networks and Bioinformatics.

Natarajan Meghanathan is an Assistant Professor of Computer Science at Jackson State University. He received his Ph.D. in Computer Science from The University of Texas at Dallas in May 2005 and has been working at Jackson State University since Fall 2005. He has published more than 70 papers in the areas of wireless ad hoc networks and sensor networks and has recently received grants from the National Science Foundation and Army Research Lab in these two areas. Besides these two areas, he conducts research in Network and Software Security, Computational Biology, Graph Theory and Algorithms. He also serves in the editorial board for several international journals and also in the program committee for several international conferences.