A Node Stability Index-Based Connected Dominating Set Algorithm for Mobile Ad Hoc Networks

Natarajan Meghanathan

Jackson State University
Jackson, MS 39217, USA
nmeghanathan@jsums.edu

Abstract. We propose a Node Stability Index (NSI)-based algorithm to determine stable connected dominating sets (CDS) for Mobile Ad hoc Networks (MANETs). The NSI of a node is defined as the sum of the predicted Link Expiration Times (LETs) of the links with its neighbor nodes. The NSI-CDS algorithm prefers to include (to the CDS) covered nodes that have the largest NSI value, computed based on the sum of the LETs of the uncovered neighbors. The NSI-CDS has been observed to have significantly longer lifetime than the maximum density-based CDS (MaxD-CDS) and the ID-based CDS (ID-CDS). The tradeoff is a modest increase in the CDS Node Size which however contributes significantly to the robustness of the CDS as well as to a lower hop count per path, especially in high-density networks.

Keywords: Stability, Connected Dominating Sets (CDS), Link Expiration Time, Mobile Ad hoc Networks, Maximum Density CDS, ID-CDS.

1 Introduction

A Mobile Ad hoc Network (MANET) is a resource-constrained dynamically changing network of arbitrarily moving wireless nodes that operate under limited battery charge, transmission range and bandwidth. By virtue of all these resource constraints, MANET routes are often multi-hop in nature and change with time depending on node mobility and availability. MANET routing protocols are preferred to be on-demand in nature to optimize resource usage [1][2]. The MANET on-demand routing protocols typically employ a global broadcast request-reply cycle, called flooding, to discover the paths (for unicasting) as well as trees and meshes (for multicasting) [3]. With flooding, a source node initiates the broadcast of the Route Request (RREQ) packets and these packets are forwarded exactly once by every other node to their neighbor nodes. However, with flooding, the network incurs lot of control overhead in requiring each node to broadcast (even though it is done only once) the RREQ message to the neighborhood. The redundancy of retransmissions is such that every node in the network gets a copy of the broadcast message from each of its neighbors.

Recent studies in the literature (e.g. [4][5][6]) have demonstrated that a connected dominating set (CDS) of the underlying network can be used as a backbone to broadcast a message from one node to all the other nodes in the network. A CDS of a

N. Meghanathan et al. (Eds.): CCSIT 2012, Part I, LNICST 84, pp. 31–40, 2012.

network graph is defined as the subset of the nodes in the network such that every node in the network is either in the CDS or is a neighbor of a node in the CDS. The message to be broadcast (such as a RREQ message) is forwarded only by the nodes that are part of the CDS and the non-CDS nodes (i.e., nodes that are not in the CDS) merely receive the message from one or more neighboring CDS nodes that cover their non-CDS neighbors. The efficiency of broadcasting using a CDS lies in minimizing the number of redundant retransmissions and this depends on the number of nodes that are part of the CDS (referred to as CDS Node Size). The CDS with the minimum number of nodes is referred to as a Minimum Connected Dominating Set (MCDS). Determining the MCDS for a network graph is an NP-complete problem [7]. Several heuristics have been proposed to closely approximate the optimal solution in polynomial time. The Maximum Density-based CDS (MaxD-CDS) algorithm [8] studied in this paper is one such heuristic to minimize the CDS Node Size and is based on the strategy of preferring to include nodes that have the maximum number of uncovered neighbors as part of the CDS.

In this paper, we show that aiming for the minimum number of nodes for the CDS in MANETs may not be a good strategy from a stability point of view. For a CDS to exist at any time instant, two conditions must hold: (i) The nodes that are part of the CDS must stay connected – i.e. reachable from one another directly or through one or more intermediate CDS nodes and (ii) Each non-CDS node should have at least one CDS node as a neighbor node. In the case of a MCDS, like the MaxD-CDS studied in this paper, the CDS nodes are far away from each other as we have to cover the non-CDS nodes spanning over the entire network with minimum number of CDS nodes. Thus, there are fewer links among these MCDS nodes and these links are also vulnerable to break at any time as the physical distance between the two constituent end MCDS nodes of a link is likely to be quite close to the transmission range of the nodes. With mobility, two CDS nodes that share such a vulnerable link between them could move away from the transmission range of each other at any time. Similarly, the probability of a non-CDS node not having any of its neighbor nodes to be a MCDS node is also high as there are only few nodes that are part of the MCDS. Thus, a MCDS has to be frequently reconfigured due to failure in maintaining the above two conditions for its existence. Hence, even though broadcasting through a MCDS may sound a promising idea to minimize the number of redundant retransmissions, the cost of frequently determining such a MCDS may overweigh the benefit obtained by actually getting to use the MCDS.

It is essential to ensure that a CDS is stable enough to avoid the overhead of frequent reconfigurations. This forms the objective of our work in this paper. To determine a stable CDS, we explore the idea of using the predicted link expiration time (LET) [9] of the Flow-Oriented Routing Protocol (FORP) [10] that has been observed (in [11][12]) to yield routes that are even twice the lifetime of the routes determined by the minimum-hop Dynamic Source Routing (DSR) protocol [13]. We introduce a term called the Node Stability Index (NSI) that is defined as the sum of the LETs of the links connected to the node. From a CDS-point of view, the NSI of a node is defined as the sum of the LETs of the links with its uncovered neighbors (i.e., nodes that are not yet covered by a CDS node). We propose an algorithm based on this notion of NSI, hereafter referred to as the NSI-CDS, and it is based on the idea of preferring to include nodes that have the largest NSI as part of the CDS. The algorithm

starts with including the node with the largest NSI to the CDS and adding all of its neighbor nodes to the covered node list. The NSI-values of each node in the network is updated as the sum of the LETs of the links with its uncovered neighbor nodes. During subsequent iterations of the algorithm, we pick a covered node that has the largest NSI value (of course, only if the NSI value is greater than zero), add it to the CDS and include all of its uncovered neighbor nodes to the list of covered nodes. The above procedure is repeated until all the nodes are covered.

The rest of the paper is organized as follows: Section 2 explains the design of the NSI-CDS algorithm. Section 3 presents the simulation environment, the simulation results featuring the NSI-CDS, MaxD-CDS and the ID-CDS and interprets them with respect to metrics such as CDS Lifetime, CDS Node Size, CDS Edge Size and Hop Count per Path. Section 4 concludes the paper.

2 Node Stability Index Connected Dominating Set Algorithm

The objective of the proposed Node Stability Index (NSI)-based CDS construction algorithm is to determine a long-living CDS without any substantial increase in the number of nodes constituting the CDS. We adopt the unit-disk graph model [17] according to which there exists a link between two nodes i and j if and only if the distance between the two nodes is less than or equal to the fixed transmission range. The network is homogeneous in nature and that all nodes operate with an identical and fixed transmission range, denoted as R in equation (1). The set of neighbors of a node i, $Neighbors(i)$, consists of those nodes that are within in the transmission range of node j. The predicted link expiration time (LET) of a link $i - j$ between two nodes i and j, currently at (X_i, Y_i) and (X_j, Y_j), and moving with velocities v_i and v_j in directions θ_i and θ_j (with respect to the positive X-axis) is computed using the formula proposed in [9]:

$$LET(i,j) = \frac{-(ab+cd) + \sqrt{(a^2+c^2)R^2 - (ad-bc)^2}}{a^2+c^2} \tag{1}$$

where $a = v_i{*}\cos\theta_i - v_j{*}\cos\theta_j$; $b = X_i - X_j$; $c = v_i{*}\sin\theta_i - v_j{*}\sin\theta_j$; $d = Y_i - Y_j$

At any moment, every node maintains a LET-table comprising of the estimates of the LET values to each of its neighbor nodes based on the latest beacons received from the neighbor node. Nodes periodically exchange beacons in the neighborhood. The beacon message broadcast by a node includes the current location of the node, the velocity at which the node is moving and the direction of movement of the node (denoted as the angle subscribed with respect to the positive X-axis). A node obtains its location information through the Global Positioning Scheme (GPS) [14] or any other relevant location service schemes (e.g. [15]). The Node Stability Index (NSI) of a node i during the working of the NSI-CDS algorithm is defined as the sum of the LET of the links with the neighbor nodes j that are not yet covered by a CDS node. The NSI of a node i is represented formally as:

$$NSI(i) = \sum_{\substack{j \in Neighbors(i) \\ j \notin Covered-Nodes-List}} LET(i,j) \tag{2}$$

The key data structures maintained and used in the NSI-CDS algorithm are as follows: (i) *CDS-Nodes-List*: This list includes all the nodes that are part of the CDS; (ii) *Covered-Nodes-List*: This list includes all the nodes that are either part of the CDS or is at least a neighbor node of a node in the CDS and (iii) *Priority-Queue*: This list includes all the nodes that are in the *Covered-Nodes-List* (but not in the *CDS-Nodes-List*) and are considered the candidate nodes for the next node to be selected for inclusion in the *CDS-Nodes-List*.

The NSI-CDS algorithm (pseudo code in Figure 1) works as follows: For every iteration, the algorithm selects one node from the *Priority-Queue* (the node is also in the *Covered-Nodes-List*) and adds it to the *CDS-Nodes-List*. The criterion to select a covered node from the *Priority-Queue* and include it in the *CDS-Nodes-List* is to give preference for the covered node with the maximum value of the Node Stability Index (NSI). As defined before, the NSI of a node is the sum of the Link Expiration Times (LETs) of the links with the uncovered neighbors of the node (i.e., the neighbor nodes that are not yet in the *Covered-Nodes-List*). Before the first iteration, since none of the nodes are in the *Covered-Nodes-List*, the NSI of a node is simply the sum of the LETs of the links with all of its neighbor nodes. The node with the maximum of such NSI value is the first node to be added to the *Covered-Nodes-List*, *Priority-Queue* and eventually to the *CDS-Nodes-List*. All the uncovered neighbors of the newly included node to the *CDS-Nodes-List* are now included in the *Covered-Nodes-List* as well as in the *Priority-Queue*. After every iteration, the NSI values of the nodes in the network and the *Priority-Queue* are recomputed based on the updated *Covered-Nodes-List*. Nodes whose NSI value is zero are removed from the *Priority-Queue*. The above procedure is repeated until there is at least one node that is not yet in the *Covered-Nodes-List*; if the underlying network is connected, the *Priority-Queue* will remain non-empty until all nodes are added to the *Covered-Nodes-List* and the algorithm finally returns the *CDS-Nodes-List*. If the *Priority-Queue* gets empty and there is at least one node to be added to the *Covered-Nodes-List*, then it implies the underlying network is disconnected and the algorithm returns NULL.

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-Nodes-List, *Covered-Nodes-List*, *Priority-Queue*, *startNode*

Dequeue(*Priority-Queue*) – Extracts, from the queue, the node with the maximum *NSI* (> 0) – in case of a tie, a node is randomly chosen and extracted from the queue.

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

Output: *CDS-Nodes-List* // contains the nodes that are part of the NSI-based CDS
NULL // if the underlying network graph is disconnected

Initialization: *CDS-Nodes-List* = Φ; *Covered-Nodes-List* = Φ; *Priority-Queue* = Φ;
$\forall i \in V,\ NSI(i) = \sum_{j \in Neighbors(i)} LET(i, j)$

Fig. 1. Pseudo Code for the NSI-CDS Algorithm

Begin Construction of NSI-CDS

startNode = the node $u \in V$ with the largest NSI value
Priority-Queue = {*startNode*}; *Covered-Nodes-List* = {*startNode*}

while (|*Covered-Nodes-List*| < |*V*| and *Priority-Queue* ≠ Φ) **do**
node *s* = Dequeue(*Priority-Queue*)
where $s \in$ *Covered-Nodes-List* and $s \notin$ *CDS-Nodes-List*
CDS-Nodes-List = *CDS-Nodes-List* U {*s*}
$\forall$ $v \in$ *Neighbors*(*s*),
if $v \notin$ *Covered-Nodes-List* **then**
Covered-Nodes-List = *Covered-Nodes-List* U {*v*}
Priority-Queue = *Priority-Queue* U {*v*}
end if

$\forall$ $u \in V$, $NSI(u) = \{ \sum LET(u,v) \mid v \in Neighbors(u)$
AND $v \notin$ *Covered-Nodes-List*$\}$
$\forall$ $u \in$ *Priority-Queue*,
if ($NSI(u) = 0$) **then**
remove node *u* from *Priority-Queue*
end if

if (|*Covered-Nodes-List*| < |*V*| and *Priority-Queue* = Φ) **then**
return NULL // the network is disconnected and there is no CDS
end if

end while

return *CDS-Nodes-List*

End Construction of NSI-CDS

Fig. 1. (*Continued*)

There can be at most O(|*V*|) iterations and O(|*E*|) edges have to be explored spread across all of these iterations. The dequeue operation during the beginning of each iteration takes O(|*V*|) time if the *Priority-Queue* has been implemented as an array and O(log|*V*|) time if implemented as a binary heap. After the inclusion of a node to the *CDS-Nodes-List* and its neighbor nodes to the *Covered-Nodes-List*, it takes O(|*V*|+|*E*|) time to re-compute the NSI values of all the nodes by exploring their incident edges and this is the most time-consuming step (compared to the dequeue operation) in each iteration. Thus, the overall-time complexity of the NSI-CDS algorithm can be represented as O(|*V*|*(|*V*| + |*E*|)). The MaxD-CDS and the ID-CDS algorithms also incur the same run-time complexity as the number of uncovered neighbors of a node has to be updated during each of the iterations.

3 Simulations

The performance of the NSI-CDS is compared with that of the Maximum Density-based CDS (MaxD-CDS) and the ID-based CDS (ID-CDS). The MaxD-CDS algorithm prefers to include into the CDS, a covered node with the maximum number of uncovered neighbors. The ID-CDS algorithm prefers to include into the CDS, a covered node with the largest node ID. To be fair to all the nodes in the network, every time a new ID-CDS is constructed, we randomly distribute the IDs of the nodes in the network. All of the simulations (including the implementation of the three CDS algorithms) are conducted in a discrete-event simulator developed in Java.

For a fixed network area (1000m x 1000m) and transmission range per node (250m), we conduct simulations with 50 nodes and 100 nodes representing networks of low density and high density respectively. The mobility model used in our simulations is the Random Waypoint model [16] with the velocity randomly chosen from the range [0, …, v_{max}] each time a node changes its direction. The value of v_{max} is varied by conducting simulations with 5 m/s, 25 m/s and 50 m/s representing scenarios of low, moderate and high mobility respectively.

The simulation strategy for each of the CDS algorithms is as follows: We obtain a centralized view of the network topology by generating mobility trace files for a simulation time of 1000 seconds for each combination of network density (50 and 100 nodes) and node mobility (v_{max} = 5, 25 and 50 m/s). We sample the network topology for every 0.25 seconds. If a CDS does not exist for a particular time instant, we run the CDS algorithm on the network topology snapshot at that time instant and tend to use that CDS during the subsequent time instants as long as the CDS exists (we check for existence of a CDS using the strategy described in the next paragraph). The above procedure is repeated for the entire simulation time of 1000 seconds.

In the simulations for each CDS algorithm, we use a CDS as long as it exists. We consider a CDS to 'exist' for a particular time instant if it is connected (i.e. the CDS nodes are reachable from one another directly or through multi-hop paths) and covered all nodes in the network (i.e. every non-CDS node has at least one CDS node as a neighbor node). If a CDS is determined to be not connected or not covering all the nodes in the network at a particular time instant, we determine a new CDS by running the CDS construction algorithm on a network graph snapshot corresponding to that time instant. The connectivity amongst the CDS nodes at a particular time instant is determined by running the Breadth-First-Search (BFS) algorithm [7] on a CDS-induced network sub graph involving only the nodes that are part of the CDS and the set of edges that may exist between any two CDS nodes at that time instant.

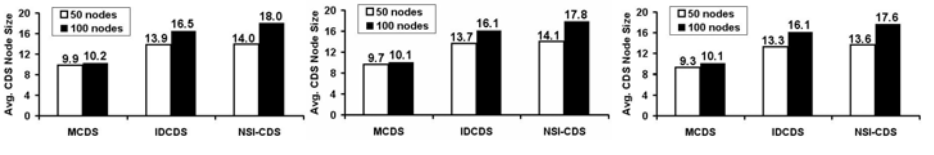
To measure the hop count per path, we run the BFS algorithm on a CDS-induced sub graph for 15 source-destination (*s-d*) pairs – the role of the source or destination could be assigned to any node in the network. The CDS-induced sub graph for a particular time instant comprises of all the nodes in the network and edges that may exist between any two CDS nodes and between a CDS node and a non-CDS node. Two non-CDS nodes have to communicate through one or more CDS nodes as intermediate nodes, even if the two non-CDS nodes are direct neighbors. However, two CDS nodes can communicate directly if they are neighbors of each other.

Each data point in Figures 2 through 5 is an average computed over 10 mobility trace files generated for every combination of network density and node mobility

values considered in the simulations. The following are the performance metrics measured in our simulations: (i) *CDS Lifetime*: We keep track of the duration of existence of each of the CDS used for the entire simulation time and compute the average value of the CDS lifetime, considering all the mobility profiles for the particular simulation condition. (ii) *CDS Node Size*: This is a time-averaged value of the number of nodes that are part of the CDS used for every time instant (i.e., the duration of the usage of the CDS is taken into consideration) over the entire simulation. (iii) *CDS Edge Size*: This is a time-averaged value of the number of edges that exist between any two CDS nodes for every time instant over the entire simulation. (iv) *Hop Count per s-d Path*: This is a time-averaged value for the number of edges (hops) in the paths determined for every *s-d* pair on CDS-induced sub graphs, considered over the entire simulation time and all the *s-d* pairs.

3.1 CDS Node Size and CDS Edge Size

The NSI-CDS includes more nodes (refer Figure 2) compared to the MaxD-CDS. The MaxD-CDS algorithm has only one objective – to cover all the nodes in the network with the minimum number of CDS nodes and hence nodes having a larger number of uncovered neighbors are preferred for inclusion to the CDS. However, such a greedy strategy is bound to have a negative effect on the CDS lifetime because it would be difficult to cover all the nodes in the network with fewer CDS nodes and also expect the CDS nodes to be connected among themselves with fewer edges that exist between these nodes. On the other hand, the NSI-CDS algorithm primarily aims to determine a stable CDS that will exist for a longer time and minimizing the CDS Node Size is only a secondary objective embedded with the primary objective of maximizing the CDS Lifetime. Thus, the number of nodes forming part of the NSI-CDS is bound to be larger than the number of nodes that are part of the MaxD-CDS. We observe that the NSI-CDS Node Size is about 45% and 75% more than the MaxD-CDS Node Size for low density and high density networks respectively. The Node Size for the ID-CDS is about the same as that of the NSI-CDS; with the NSI-CDS incurring slightly larger number of nodes (at most 3% in low-density networks and 10% in high-density networks) than the ID-CDS.

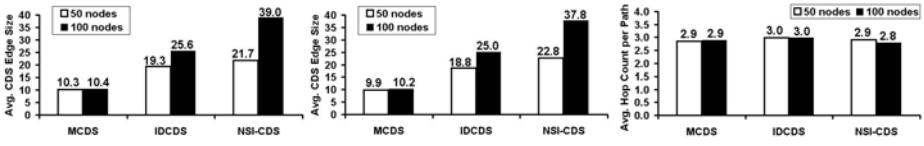


v_{max} = 5 m/s v_{max} = 25 m/s v_{max} = 50 m/s

Fig. 2. Average CDS Node Size

The larger the CDS Node Size - the larger will be the CDS Edge Size and more stable and robust will be the CDS to link failures. Since the MaxD-CDS has the lowest CDS Node Size, it also has the lowest CDS Edge Size and vice-versa for the NSI-CDS. In its pursuit to form a CDS with the least number of nodes, the MaxD-CDS algorithm chooses CDS nodes that are far away from each other such that each

CDS node individually covers as many uncovered neighbor nodes as possible. As a result, the CDS nodes are more likely to be away from each other; thus, the number of edges that are part of the CDS spanning the entire network is very low. The ID-CDS and NSI-CDS algorithms are relatively insensitive to the # of edges added to the CDS.

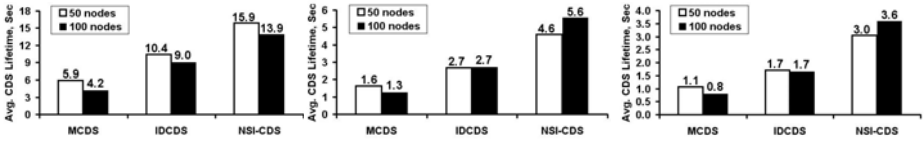


v_{max} = 5 m/s v_{max} = 25 m/s v_{max} = 50 m/s

Fig. 3. Average CDS Edge Size

3.2 CDS Lifetime

The lifetime of a NSI-CDS is significantly longer compared to that of a MaxD-CDS and an ID-CDS. For a given scenario of node mobility and network density, the sum of the lifetimes of an ID-CDS and MaxD-CDS is still less than the lifetime incurred by NSI-CDS. The stability of the NSI-CDS can also be attributed to the relatively larger CDS Edge Size that accompanies the modest increase in the CDS Node Size. Thus, even though we can say that there is a tradeoff between the CDS Lifetime vs. the CDS Node Size and Edge Size, this tradeoff is more favorable towards the NSI-CDS and it significantly gains in the Lifetime metric with a very modest increase in the Node Size. On the other hand, in pursuit of minimizing the number of nodes that are part of the CDS, the MaxD-CDS is quite unstable, especially as the node mobility and/or the network density increases. NSI-CDS is more scalable with respect to the increase in network density and node mobility. The modest increase in the Node Size (at most by 30%) at larger network density helps NSI-CDS to incur a significant gain in the Lifetime. On the other hand, in the case of MaxD-CDS, with very few additional nodes (at most 7% more nodes) to cover about 100% more nodes in the network, as we double the node density, the MaxD-CDS turns out to be quite unstable and has very low lifetime. Even the ID-CDS based approach performs significantly better than the MaxD-CDS approach in high-density networks. The lifetime per ID-CDS is about twice the lifetime per MaxD-CDS for most of the simulation conditions.

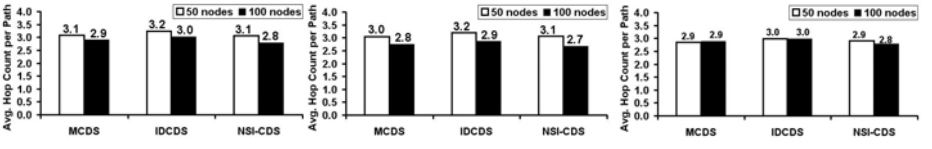


v_{max} = 5 m/s v_{max} = 25 m/s v_{max} = 50 m/s

Fig. 4. Average CDS Lifetime

3.3 Hop Count per Path

The three CDS algorithms almost incur the same hop count per path for low-density networks. However, with increase in the network density to 100 nodes, the NSI-CDS incurs a relatively slightly lower hop count (about 7-10% lower) compared to the MaxD-CDS and ID-CDS. This could be attributed to the presence of a larger number of nodes and edges as part of the NSI-CDS-induced sub graph that enables the discovery of paths that have fewer hops between the source (*s*) and destination (*d*) nodes. With fewer node and edges, the MaxD-CDS incurs more hops per *s-d* path. The ID-CDS incurs even slightly more hops per *s-d* path. The MaxD-CDS nodes could be relatively more heavily used compared to the other two CDS algorithms, because only very few nodes are part of the MaxD-CDS and all communication has to go through these CDS nodes. This could lead to premature failure of critical nodes, mainly nodes lying in the center of the network, leading to reduction in network connectivity, especially in low-density networks. With the NSI-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 as well as incur relatively lower end-to-end delay per data packet.



v_{max} = 5 m/s v_{max} = 25 m/s v_{max} = 50 m/s

Fig. 5. Average Hop Count per Source-Destination Path using the CDS Nodes

4 Conclusions

We have proposed a Node Stability Index (NSI)-based algorithm to determine stable connected dominating sets for mobile ad hoc networks and it is based on the notion of the predicted link expiration times (LETs). The NSI-CDS algorithm has the same run-time complexity as the other standard algorithms based on maximum density (MaxD-CDS) or node ID (ID-CDS). The MaxD-CDS and ID-CDS have been observed to be very unstable in the presence of node mobility. We do observe a tradeoff between the CDS Lifetime vs. the CDS Node Size. However, the tradeoff is more favorable towards the NSI-CDS. The NSI-CDS is 2.5 to 3.5 times more stable than the MaxD-CDS and 1.5 to 2.0 times stable than the ID-based CDS; this gain in the CDS Lifetime is achieved by including at most 25-45% and 3-10% more CDS nodes than MaxD-CDS and ID-CDS respectively. The relatively moderate larger number of constituent nodes and edges make the NSI-CDS more robust to node mobility and link failures as well as contribute to a lower hop count per path, especially in high-density networks.

References

1. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J.: A Performance Comparison of Multi-hop Wireless Ad hoc Network Routing Protocols. In: International Conference on Mobile Computing and Networking, pp. 85–97. ACM, Dallas (1998)
2. Johansson, P., Larson, T., Hedman, N., Mielczarek, B., DegerMark, M.: Scenario-based Performance Analysis of Routing Protocols for Mobile Ad hoc Networks. In: International Conference on Mobile Computing and Networking, pp. 195–206. ACM, Seattle (1999)
3. Siva Ram Murthy, C., Manoj, B.S.: Ad Hoc Wireless Networks: Architectures and Protocols, 1st edn. Prentice Hall (2004)
4. Sakai, K., Sun, M.-T., Ku, W.-S., Okada, H.: Maintaining CDS in Mobile Ad Hoc Networks. In: Li, Y., Huynh, D.T., Das, S.K., Du, D.-Z. (eds.) WASA 2008. LNCS, vol. 5258, pp. 141–153. Springer, Heidelberg (2008)
5. Sheu, P.-R., Tsai, H.-Y., Lee, Y.-P., Cheng, J.Y.: 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 (2009)
6. Bao, L., Garcia-Luna-Aceves, J.J.: Stable Energy-aware Topology Management in Ad hoc Networks. Ad hoc Networks 8(3), 313–327 (2010)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
8. Meghanathan, N.: An Algorithm to Determine the Sequence of Stable Connected Dominating Sets in Mobile Ad hoc Networks. In: The 2nd Advanced International Conference on Telecommunications. IARIA, Guadeloupe (2006)
9. Su, W., Lee, S.-J., Gerla, M.: Mobility Prediction and Routing in Ad hoc Wireless Networks. International Journal of Network Management 11(1), 3–30 (2001)
10. Su, W., Gerla, M.: IPv6 Flow Handoff in Ad hoc Wireless Networks using Mobility Prediction. In: Global Telecommunications Conference, vol. 1a, pp. 271–275. IEEE, Rio de Janeireo (1999)
11. Meghanathan, N.: Exploring the Stability-Energy Consumption-Delay-Network Lifetime Tradeoff of Mobile Ad hoc Network Routing Protocols. Journal of Networks 3(2), 17–28 (2008)
12. Meghanathan, N.: Path Stability based Ranking of Mobile Ad hoc Network Routing Protocols. ISAST Transactions Journal on Communications and Networking 1(1), 66–73 (2007)
13. Johnson, D.B., Maltz, D.A., Broch, J.: Ad hoc Networking. Addison-Wesley (2001)
14. Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J.: Global Positioning System: Theory and Practice, 5th edn. Springer, Heidelberg (2004)
15. Keiss, W., Fuessler, H., Widmer, J.: Hierarchical Location Service for Mobile Ad hoc Networks. ACM Mobile Computing and Communications Review 8(4), 47–58 (2004)
16. Bettstetter, C., Hartenstein, H., Perez-Costa, X.: Stochastic Properties of the Random-Way Point Mobility Model. Wireless Networks 10(5), 555–567 (2004)
17. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Unit Disk Graph Approximation. In: Workshop on the Foundations of Mobile Computing, pp. 17–23. ACM, Philadelphia (2004)