A Hierarchical Connected Dominating Set Based Clustering Algorithm for Mobile Ad hoc Networks

Deniz Cokuslu
Department of Computer Engineering
Izmir Institute of Technology
Urla, Izmir 35340, Turkey
Email: denizcokuslu@iyte.edu.tr

Kayhan Erciyes
International Computer Institute
Ege University
Bornova, Izmir 35100, Turkey
Email: kayhan.erciyes@ege.edu.tr

***Abstract*— We propose a hierarchical Connected Dominating Set (CDS) based algorithm for clustering in Mobile Ad hoc Networks (MANETs). Our algorithm is an extension of our previous Connected Dominating Set Based Clustering (CDSC) Algorithm [1]. We extended the levels of the CDS to two levels and improved functionality at each level by providing additional rules to make sure that every node belongs to a single cluster. In the first level of the algorithm, the elements of the CDS are formed, based on CDSC Algorithm heuristics with improved functionality. The second level of the algorithm is executed among the CDS elements to find the second level CDS where each element belonging to the set represents a group of CDS elements, therefore a group of clusters. We show that this approach is more scalable and simpler to implement than a single level algorithm and that it also provides more balanced two level clusters due to its distributed nature. We also show that the number of levels of the algorithm can be extended to more than two layers providing more populated clusters, therefore providing a level of cluster and group membership structure within the MANET. This hierarchical groups can be used for different application needs at each level such as multi-cast communication or security purposes in MANETs.**

I. Introduction

Routing in MANETs is a very problematic issue because of the dynamicity of the network. In dynamic networks such as MANETs, routing tables should be updated very frequently. Keeping the routing tables updated may consume a large part of the wireless traffic in the network. This traffic might sometimes be extremely dense which may possibly block the circulation of the messages between the nodes. A virtually structured network such as a connected dominating set can be considered as a good solution to make message transfers more efficient. However, even in the structured networks, a routing protocol is required in order to deliver messages to the destinations. CDS Flooding Algorithm is a flooding based routing algorithm. We first construct connected dominating set based clusters by using an efficient connected dominating set based clustering algorithm, then implement a message flooding mechanism which uses the cluster heads as the gateways of the clusters. In CDS Flooding Algorithm, flooding process takes place only between the cluster heads, therefore the algorithm significantly reduces the number of flooded messages in the network as the cluster heads consist of a small part of the entire network.

II. Background

A *Dominating Set* is a subset S of a graph $G = (V, E)$ such that every vertex in G is either in S or adjacent to a vertex in S. Dominating sets can be classified into three main categories, Independent Dominating Sets (IDS), Weakly Connected Dominating Sets (WCDS) and Connected Dominating Sets (CDS). *Independent Dominating Set* is a dominating set S of a graph G in which there are no adjacent vertices. A *Weakly Connected Dominating Set* (WCDS) is a weakly induced subgraph (S) of a graph (G) which is connected and dominating [2] [3]. Han and Jia [24] [25] proposed efficient algorithms for constructing a WCDS in MANETs. Chen and Liestman [8] and Alzoubi et al. [23] are other well known WCDS construction algorithms. A *Connected Dominating Set* (CDS) is a subset (S) of a graph (G) such that S forms a dominating set and is connected. CDS based clustering is a fundamental approach in MANETs to partition the network into a number of clusters. CDSs have many advantages in network applications such as ease of broadcasting and constructing virtual backbones [22]. Various algorithms exist for clustering in dominating sets but we are interested in CDS based clustering algorithms as they provide a backbone between clusters. Guha and Khuller [9] proposed two centralized greedy algorithms for finding suboptimal connected dominating sets. Das and Bharghavan [11] [12] provided the distributed implementations of Ghua and Khuller's algorithms [9]. Wu and Li [14], improved Das and Bhraghavan's distributed algorithm to a localized distributed algorithm. Wu and Li's algorithm works in two phases, in the first phase a node marks itself as a cluster head if any two of its neighbors are not connected to each other directly. In the second phase, a marked vertex v changes its mark to ordinary node if one of the pruning rules is met. Nanuvala [26] has extended the Wu's CDS Algorithm and added a third pruning rule. Cokuslu, Erciyes and Dagdeviren [1] added some extra heuristics to Wu and Li's algorithm [14] and provided more reliable results. They also added two more pruning rules. Heuristics shorten the runtime of the algorithm and total of four pruning rules results in a less redundant cluster heads compared to Wu and Li's algorithm. Li et al. [27] proposed an algorithm to construct CDS with bounded diameters, the algorithm first finds a maximal independent set

Proceedings of the Fifteenth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'07)
978-1-4244-1854-1/08/$25.00 ©2008 IEEE

and then selects some other nodes as clusterheads in order to build a CDS at the end. Gao et al. [28] also proposed a CDS algorithm which uses the maximal independent set as the basis of construction of the CDS.

There are various recent algorithms existing for clustering in MANETs using dominating sets [5], [6], [7], [15], [16], [17], [18], [19], [20], [21], [29].

A. The Connected Dominating Set Based Clustering Algorithm

The Connected Dominating Set Based Clustering Algorithm (CDSC) finds a minimal connected dominating set in a MANET in a distributed manner. We developed our algorithm based on Wu's CDS Algorithm [13] because it is very suitable for our purposes. It finds a connected dominating set which can be used as a backbone, it is totally distributed and it does not require a predefined routing mechanism. In the CDSC Algorithm [1], we added some extra heuristics to Wu and Li's algorithm [14] and provided more efficient results. We also added two more cluster head pruning rules. The heuristics shorten the runtime of the algorithm and a total of four pruning rules results in less redundant cluster heads compared to Wu and Li's algorithm.

III. The Two Level Connected Dominating Set Based Clustering Algorithm

We propose a distributed algorithm which finds two minimal connected dominating sets in a MANET. We developed our algorithm as an extension of our previous algorithm [1]. First, we find a CDS on MANET using our CDS algorithm and call the resulting subset of cluster heads as *First Level CDS*, then we run the same clustering algorithm on the subset of *First Level CDS*. At the end of the algorithm we get a two level connected dominating set, *First Level CDS* which is composed of *Cluster Heads* and *Second Level CDS* which is composed of *Super Cluster Heads*. The two-Level clustering provides more crowded clusters which are relatively better than our first approach in which the size of the clusters are very small compared to the number of nodes in the MANET.

A. Algorithm

We assume that the neighborhoods of the nodes remain constant in a reasonable period of time in order to complete a whole cycle in a single node. We also assume that the graph is connected, each node has a unique *node_id* and knows its adjacent neighbors. Each node has a *color* indicating whether the node is in the dominating set or not. The *color* is set to *BLACK* if the node is in the dominating set, or *WHITE* if the node is not in the dominating set. Color *GRAY* is used to indicate that the node is marked after the first phase, but it will change its color after the second phase to either *WHITE* or *BLACK*. The first level messages are *Period_TOUT* which triggers the algorithm and is sent periodically by the node itself, *Neighbor_REQ* which requests a list of distance-2 neighbors, *Neighbor_LST* which includes a list of adjacent neighbors of sending node, *Color_REQ* which requests a node's color

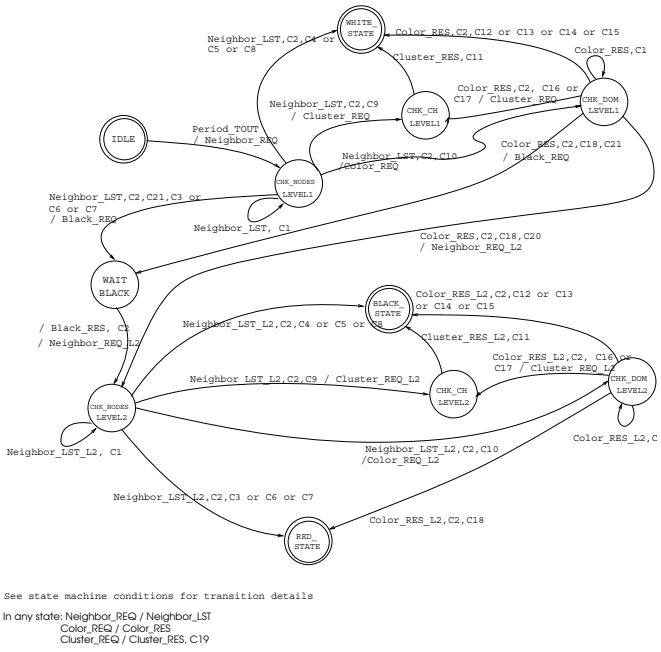

Fig. 1. Finite State Machine of the 2-Level Hierarchical Clustering Algorithm

after the first phase, *Color_RES* holds the sender's color, *Cluster_REQ* requests if a node is a *First Level Cluster Head* or not and *Cluster_RES* which informs target node that the sender is a *First Level Cluster Head* or not. The second level messages are *Black_REQ* which requests *GRAY* nodes to send their permanent color, *Black_RES* includes the sender's permanent color, *Neighbor_REQ_L2* requests a list of distance-2 *BLACK* neighbors of *BLACK* nodes, *Neighbor_LST_L2* includes a list of *BLACK* neighbors of the sender node, *Color_REQ_L2* requests a node's second level color, *Color_RES_L2* delivers the sender's second level color, *Cluster_REQ_L2* requests if a node is a *Second Level Cluster Head* or not and *Cluster_RES_L2* informs target node that the sender is a *Second Level Cluster Head* or not. Every node in the network performs the same local algorithm periodically. The finite state diagram for the algorithm can be seen in Fig. 1. During the runtime of the *Two-Level CDS Algorithm* some state machine transition conditions are needed to be defined. The definition of *CDS Algorithm Finite State Machine Transition Conditions* are described below such that:

- C1. The responses to the multi-casted message do not have been completely collected.
- C2. The responses to the multi-casted message are completely collected.
- C3. The node is isolated, its neighbor is isolated too and node's id is bigger than its neighbor's id.
- C4. The node is isolated, its neighbor is isolated too and the node's id is smaller than its neighbor's id.
- C5. The node is isolated and its neighbor is not isolated.
- C6. The node has at least one isolated neighbor.
- C7. The graph is complete and the node has the biggest

id in the graph.
- C8. The graph is complete and the node does not have the biggest id.
- C9. Node's neighbors are all connected and the graph is not complete.
- C10. The node has at least two unconnected neighbors.
- C11. Cluster Head is set to the sender's id.
- C12. CDS pruning rule 1 which is described below is true.
- C13. CDS pruning rule 2 which is described below is true.
- C14. CDS pruning rule 3 which is described below is true and the node has at least one *BLACK* neighbor.
- C15. CDS pruning rule 4 which is described below is true and the node has at least one *BLACK* neighbor.
- C16. CDS pruning rule 3 is true and the node doesn't have any *BLACK* neighbor.
- C17. CDS pruning rule 4 is true and the node doesn't have any *BLACK* neighbor.
- C18. Conditions C12 to C17 are all false.
- C19. Node's color is currently *BLACK*.
- C20. Node's neighbors completed the first level and determined their first level color.
- C21. The node still have GRAY colored neighbors in its color list.
- C22. The node does not have any neighbors.

Each node is in the *IDLE* state and colored as *UNDEFINED_COLOR* initially. When the period is timed out, the node sends a *Period_TOUT* message to itself. This message causes the node to switch its state to *CHK_NODES_LEVEL1* and send a *Neighbor_REQ* message to all of its adjacent neighbors. Then the node waits for *Neighbor_LST* messages from all of its adjacent neighbors. When all *Neighbor_LST* messages are collected, the node checks the heuristics *C3* to *C10* defined in the state machine transition conditions list to determine their next state transition. If the node is suitable for conditions C4, C5 or C8 it determines its *First Level Color* and its *Second Level Color* to *WHITE* and changes its state to *WHITE_STATE*. Such a node completes the algorithm in this step and becomes an *Ordinary Node*. If the node is suitable for the conditions C3, C6 or C7, it changes its *First Level Color* to *BLACK* and switches to state *WAIT_BLACK*. This type of node completes its *First Level Clustering* at this step as a *First Level Cluster Head*, and starts to run *Second Level Clustering*. If condition C9 is true for a node, it marks its *First Level Color* to *WHITE*, switches its state to *CHK_CH_LEVEL1* and multicasts a *Cluster_REQ* message in order to learn which neighbor became its cluster head until a *Cluster_RES* message is received. When the *Cluster_RES* message is received the node changes its state to *WHITE_STATE*, sets its *First* and its *Second Level Color* to *WHITE* and finishes the algorithm. If the node is suitable for the condition C10, it is potentially a cluster head candidate. In this case, the node switches its state to *CHK_DOM_LEVEL1*, changes its *First Level Color* to *GRAY* and multicasts a *Color_REQ* message in order to collect its neighbor's colors. When the node switches its state to *CHK_DOM_LEVEL1*, it waits for all its neighbors to send their colors. When the node v collects all of the color information, it starts to apply the *CDS pruning rules* which are described below where u and w are the neighbor nodes of the node v:

1) $\exists u \in N(v)$ which is marked $BLACK$ such that $N[v] \subseteq N[u]$;
2) $\exists u, w \in N(v)$which is marked $BLACK$ such that $N(v) \subseteq N(u) \bigcup N(w)$;
3) $\exists u \in N(v)$ which is marked $GRAY$ such that $N[v] \subseteq N[u]$ and $degree(v) < degree(u)$ OR $(degree(v) = degree(u)$ AND $id(v) < id(u))$;
4) $\exists u, w \in N(v)$ which is marked $GRAY$ OR $BLACK$ such that $N(v) \subseteq N(u) \bigcup N(w)$ AND $degree(v) < min\{degree(u), degree(w)\}$ OR $degree(v) = min\{degree(u), degree(w)\}$ AND $id(v) < min\{id(u), id(w)\}$;

If one of these pruning rules is true then the node v changes its *First Level Color* to *WHITE*. If the node is suitable for conditions C12, C13, C14 or C15, it finishes the algorithm and changes its state to *WHITE_STATE* and its colors to *WHITE*. If it is suitable for C16 or C17, it changes its state to *CHK_CH_LEVEL1* and then to *WHITE_STATE* to finish its execution. If none of the four pruning rules is true, then the node is suitable for the condition C18, it then marks itself as *BLACK*. If the node is also suitable for the condition C21, it changes its state to *WAIT_BLACK* and multicasts a *Black_REQ* message in order to wait its neighbors to determine their permanent colors. If a node's neighbors have already determined their *First Level Colors*, then node changes its state to *CHK_NODES_LEVEL2* and multicast the *Neighbor_REQ_L2* message.

Nodes which reach to the *WAIT_BLACK* or *CHK_NODES_LEVEL2* states end the *First Level Clustering* and from this point, they start to execute the *Second Level Clustering*. The algorithm which is used in the *Second Level Clustering* is the same algorithm which is used during the *First Level Clustering*. The only difference is the new set of nodes used in *Second Level Clustering* are the nodes which are *BLACK* colored after the *First Level Clustering*. We create a subset S which is composed of *First Level Cluster Heads*, and apply the *CDS* algorithm to the subset S. In the *Second Level Clustering*, nodes which are not *Level 2 Cluster Heads* end in the state *BLACK_STATE*. *Level 2 Cluster Heads* end in the state *RED_STATE*. When the *Two-Level CDS Algorithm* is finished, the nodes are in any of the *WHITE_STATE*, *BLACK_STATE* or *RED_STATE*. The cluster information for a node is held locally, each node knows only its cluster head. This makes our algorithm more flexible, thus it can be easily extended to a k-level hierarchical clustering.

At any state, a node can receive request messages to help other nodes run their algorithms. These messages are *Neighbor_REQ*, *Cluster_REQ*, *Color_REQ*, *Neighbor_REQ_L2*, *Cluster_REQ_L2*, *Color_REQ_L2* and *Black_REQ*. In such a case, the node prepares the required information requested in

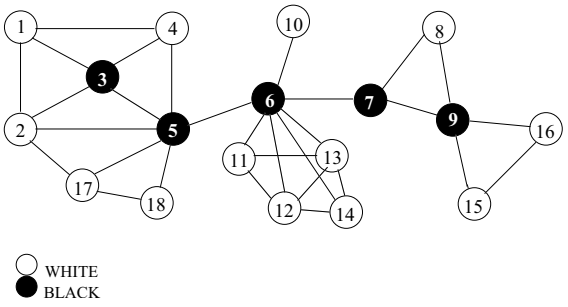

Fig. 2. First Level Output of TLCDSC Algorithm in a Sample Graph

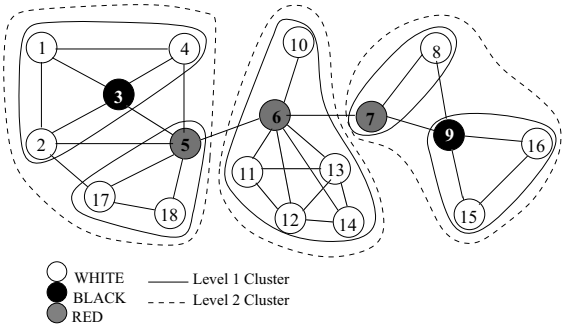

Fig. 3. Two-Level CDS Algorithm Resulting Graph

the received message and continues its current operation. No state changes are performed in this case.

B. An Example Operation

We obtained the resulting connected dominating set in Fig. 3 by using our algorithm. This section explains the algorithm step by step by in a sample graph. Execution of the algorithm is explained phase by phase, for all nodes.

Execution of First Level Clustering: At the end of the first phase of the *First Level Clustering*, nodes 6, 8, 10, 11, 14, 15, 16 and 18 determine their colors permanently. Node 6 satisfies the condition C6, thus changes its *First Level Color* to *BLACK* and finishes its *First Level Clustering* and changes its state to *WAIT_BLACK*. Nodes 8, 11, 14, 15, 16 and 18 satisfy condition C9 and change their *First Level Colors* to *WHITE*. Nodes 8, 15, 16 and 18 change their states to *CHK_CH_LEVEL1* in order to set their cluster heads. Nodes 11 and 14 set their cluster head as node 6 and finish their execution. Node 10 is an isolated node, therefore it changes its *First Level Color* to *WHITE* and sets its cluster head as node 6 and finishes its execution. Other nodes become *GRAY* colored because all of them satisfy the condition C10. In the second phase of the *First Level Clustering*, the CDS algorithm checks the conditions C12 to C18. At the end of this phase, nodes 1, 2, 4, 12, 13 and 17 determine their colors as *WHITE* because they are suitable for one of the four pruning rules. Nodes 12 and 13 select node 6 as their cluster head and finish their *First Level Clustering*. Nodes 1, 2, 4 and 17 change their states to *CHK_CH_LEVEL1* in order to set their cluster heads. Nodes 3, 5, 7 and 9 change their colors to *BLACK* as they satisfy condition the C18. At the end of the *First Level Clustering*, the resulting CDS can be seen in Fig. 2.

Execution of Second Level Clustering: The *Second Level Clustering* uses the new subset of *First Level Cluster Heads* as its domain, therefore the working set for the *Second Level Clustering* is the nodes 3, 5, 6, 7 and 9. When the *Second Level Clustering* starts its execution, nodes 3 and 9 determines their *Second Level Colors* as *WHITE* because they are suitable for the condition C5. Thus nodes 3 and 9 finishes their *Second Level Clustering* by determining their *Second Level Colors* as *WHITE* and their end states as *BLACK_STATE*. Nodes 5 and 7 are suitable for the condition C6, thus they finish their *Second Level Clustering* as *Super Cluster Heads* by determining their *Second Level Colors* as *BLACK* and their end states by *RED_STATE*. Node 6 is suitable for the condition C10, thus it changes its state to *CHK_DOM_LEVEL2*. After collecting its neighbor's second level colors, it determines its *Second Level Color* as *BLACK* and its final state as *RED_STATE*, because it is suitable for the condition C18. The overall result of the *Two-Level CDS Algorithm* can be seen in Fig. 3.

C. Analysis

Theorem 1: Time complexity of the clustering algorithm is $O(10)$.

Proof: Every node executes the distributed algorithm by the exchange of 10 messages. Since all these communication occurs concurrently, at the end of the algorithm, the members of the Two-Level CDS are determined, so the time complexity of the algorithm is $O(10)$. ■

Theorem 2: Message complexity of the clustering algorithm is O(n^2) where n is the number of nodes in the graph.

Proof: For every mark operation of a node, 10 messages are required (*Neighbor_REQ, Neighbor_LST, Color_REQ, Color_RES, Black_REQ, Black_RES, Neighbor_REQ_L2, Neighbor_LST_L2, Color_REQ_L2, Color_RES_L2*). Assuming every node has n-1 adjacent neighbors, total number of messages sent is $10(n-1)$. Since there are n nodes, total number of messages in the system is $n(10(n-1))$ Therefore messaging complexity of our algorithm has an upperbound of $O(n^2)$. ■

IV. Results

We implemented TLCDSC Algorithm using C++ on top of the network simulator *ns2*. We generated random scenarios for static and dynamic graphs.

During the experiments, we used three parameters which are the number of the nodes, mobility of the nodes and density of the network. We determined 4 "number of nodes scenarios" which have 20, 30, 40 and 50 nodes. We used the degree of the graph as the density parameter. As the surface area decreases the density of the graph increases which means that the nodes will have greater degrees. We set the surface area such that the degree of our graph will be between 4 and 10. For the mobility parameter, we generated three "mobility scenarios" namely static, low speed and high speed. In the static scenario tests,

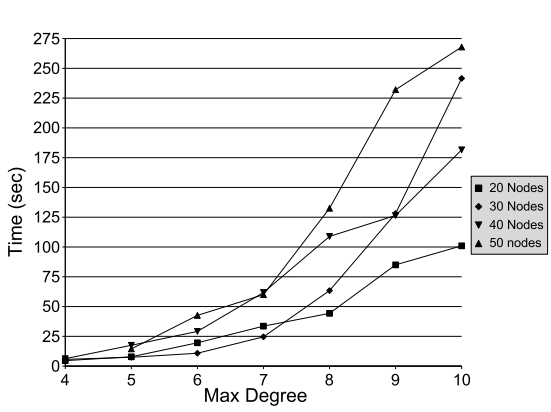

Fig. 4. Runtime Test in a Static Network

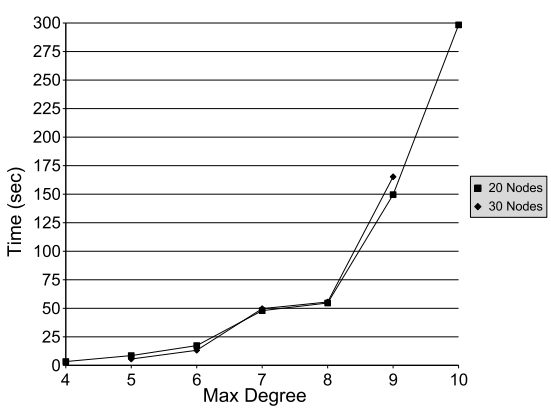

Fig. 6. Runtime Test in a High Speed Dynamic Network

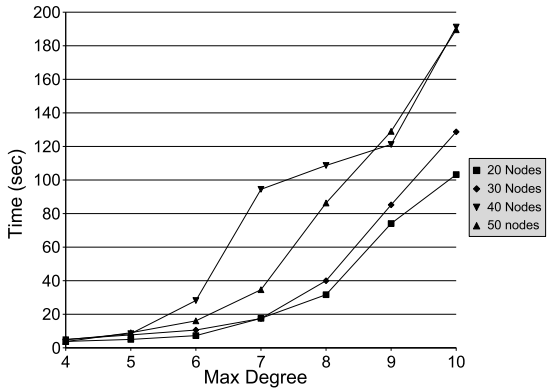

Fig. 5. Runtime Test in a Low Speed Dynamic Network

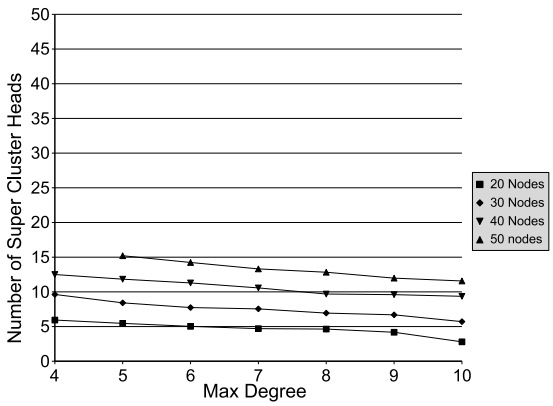

Fig. 7. Number of Super Cluster Heads in a Static Network

nodes remain constant. In the low and high mobility scenarios, respective node speeds are limited from 1 m/s to 5 m/s and from 5 m/s to 10 m/s. The speed of the nodes is determined randomly by the simulation environment within the specified velocity limits. In the dynamic graph experiments, we take into account only the experiments in which nodes are moving but the neighborhoods of the nodes do not change.

The parameters which are described above generate 84 different test cases with the specified values. During the tests, we collected an average of 60 test results for each of the 84 different test cases. Total of 5000 samples were collected during the TLCDSC Algorithm tests.

Fig. 4 shows the runtime of the algorithm. We see that runtime of the algorithm is below 20 seconds for the densities below 6. We experimented for the unlikely cases of the number of neighbors becoming larger than this value just to see the response of the algorithm. In this case, we obtain very high execution times, however, the execution time is linear. We also observe that the runtime of the algorithm is nearly the same for the nodes 20 to 50 for densities smaller than 5. This is because the algorithm runs distributed in each node and is independent from the size of the graph. In Fig. 5 and Fig. 6, we can see that in three mobility scenarios, run times are similar to each other as long as the neighborhoods remain constant. The only parameter that affects the runtime is the density of the graph which determines the number of messages exchanged between the neighbor nodes. For higher degrees, the message conflicts increase dramatically, this results in a sudden increase in the runtime of the algorithm. The message conflicts also result in anomalies in the test results which make the observations less meaningful.

Fig. 7 displays the number of super clusterheads formed using TLCDSC Algorithm. We would expect to have less clusterheads as density increases. We can see the decrease in the clusterhead counts in the graph as the degree value increases in the figure as we expected. We can see almost the same amount of decrease in the three mobility scenarios in Fig. 8 and Fig. 9.

The sizes of the formed clusters for varying parameters are recorded in Fig. 10. Typically, as the density increases, the number of clusterheads decreases. Therefore we expect to have more populated clusters as the degree increases. We can see this increase and also similar cluster sizes in the experiment results. In the different mobility scenarios, size of the clusters remain between the same range which are limited between 4 and 7. This result shows that TLCDSC

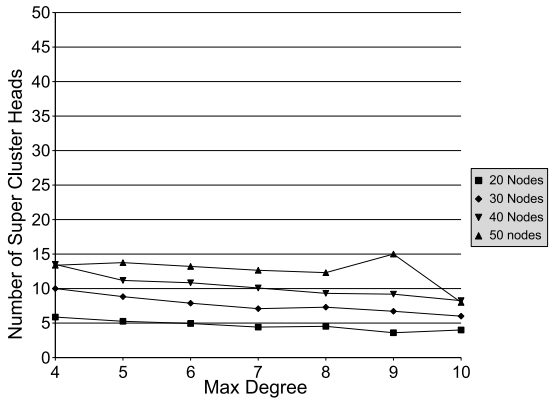

Fig. 8. Number of Super Cluster Heads in a Low Speed Dynamic Network

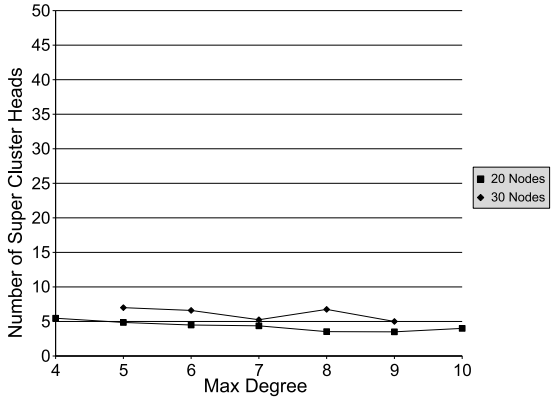

Fig. 9. Number of Super Cluster Heads in a High Speed Dynamic Network

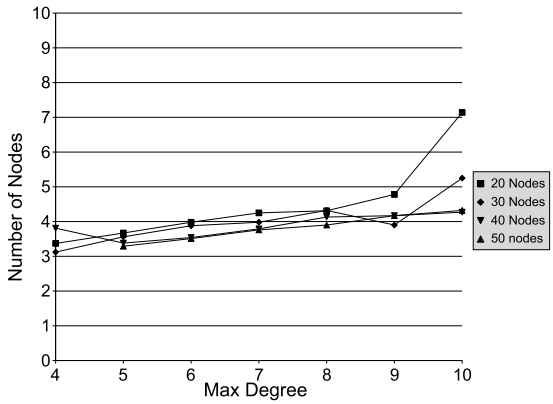

Fig. 10. Size of the Super Clusters in a Static Network

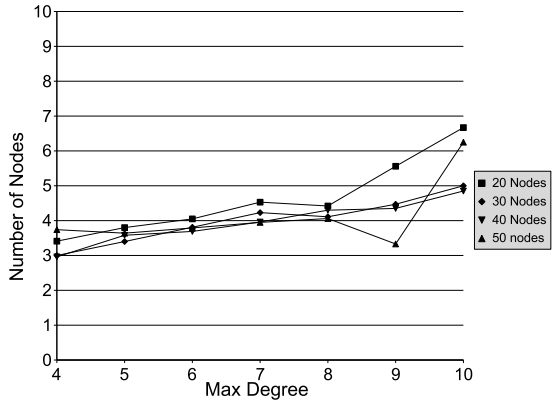

Fig. 11. Size of the Super Clusters in a Low Speed Dynamic Network

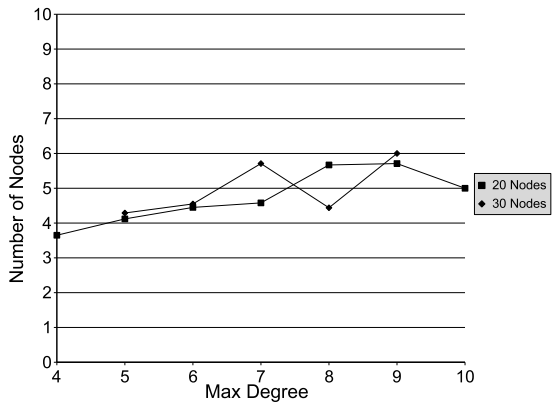

Fig. 12. Size of the Super Clusters in a High Speed Dynamic Network

Algorithm is independent from the size of the MANET in terms of the cluster qualities. The results are also similar for different mobility scenarios as can be seen in Fig. 11 and Fig. 12, which means that the algorithm is independent from the mobility too.

These results show us that the size of the resulting two level clusters and the runtime of the TLCDSC Algorithm are independent from the mobility and the size of the MANET. The TLCDSC Algorithm builds in more crowded clusters than CDSC Algorithm [1], in this terms the algorithm satisfies the main objective. We can say that the algorithm can be preferable in environments in which the density value does not exceed the maximum degree of 6 as shown in the graphs.

V. Conclusions

We described, analyzed and showed the implementation details of a two-level clustering algorithm for MANETs. Theoretically and experimentally, the proposed TLCDSC Algorithm has similar complexities for each phase which may be interpreted as being scalable. We may thus extend TLCDSC Algorithm further to, say n levels, resulting in multiple n complexity of a single level. Although the proposed algorithm runtime test results are similar to those in the CDSC Algorithm

[1], we observed that the resulting clusters are more crowded compared to CDSC Algorithm, which was our primary goal during the development of the TLCDSC Algorithm. Therefore TLCDSC Algorithm can be preferable if more crowded clusters are needed.

The local information kept at a node is minimal consisting of its neighbors and its cluster head at the lowest level which provides a high level of autonomous operation that can be used efficiently in highly distributed but coordinated MANET applications such as wide range rescue operations. One important aspect of the TLCDSC Algorithm or its extended derivatives to further levels is the co-existence of clusters at inter levels. This co-existence may be used effectively to provide simultaneous service to different application needs at each level. For example, for the *key exchange* problem in *Public Key Cryptography* in MANETs, clusters and therefore group communication at level 2 can be used where the lowest and least populated clusters can provide the basic communication backbone via their cluster heads.

One difficulty which is encountered during the implementation of TLCDSC Algorithm is the seemingly slow execution times in the *ns2* simulator. According to the investigations of the simulation results, we realized that this is not the result of more than usual number of pruning rules and heuristics but rather due to collisions of the messages at *MAC* level. We are planning to provide *MAC* level support for TLCDSC Algorithm in the near future. We are also planning to modify TLCDSC Algorithm with energy considerations of nodes, to be able to use this algorithm in *wireless sensor networks* for communication backbone formation purposes.

REFERENCES

[1] D. Cokuslu, K. Erciyes, and O. Dagdeviren, "A Dominating Set Based Clustering Algorithm for Mobile Ad hoc Networks", in *Proc. ICCS2006*, LNCS 3991, 2006, pp. 571-578.

[2] Y. P. Chen and A. L. Liestman, "Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks", in *Proc. 3rd ACM Int. Symp. Mobile Ad Hoc Net. and Comp.*, 2002, pp. 165-172.

[3] Y. P. Chen, A. L. Liestman, and J. Liu, "Clustering Algorithms for Ad Hoc Wireless Networks", *Nova Science Publisher*, 2004.

[4] D. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm", *IEEE Trans. on [legacy, pre - 1988]*, vol. 29, Issue 11, pp. 1694-1701, 1981.

[5] M. Gerla and J. T. C. Tsai, "Multicluster, mobile, multimedia radio network Wireless Networks", *ACM/Baltzer Journal of Wireless Networks.*, vol. 1, no. 3, pp. 255-265, 1995.

[6] G. Chen, F. G. Nocetti, J. S. Gonzalez and I. Stojmenovic, "Connectivity based k-hop clustering in wireless networks", in *Proc. of the 35th Annual Hawaii International Conference*, 2002, pp. 2450-2459.

[7] T. Ohta, S. Inoue and Y. Kakuda, "An Adaptive Multihop Clustering Scheme for Highly Mobile Ad Hoc Networks", in *Proc. 6th ISADS03*, 2003.

[8] Y. P. Chen and A. L. Liestman, "A Zonal Algorithm for Clustering Ad Hoc Networks", *International Journal of Foundations of Computer Science*, pp. 305-322, 2003.

[9] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets", in *Proc. LLC*, Springer-Verlag, 1998.

[10] I. Stojmenovic, M. Seddigh and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks", *IEEE Transactions on Parallel and Distributed Systems*, vol.13, pp. 14-25, 2002.

[11] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets", in *Proc. ICC 97 'Towards the Knowledge Millennium'*, IEEE International Conference on vol. 1, 1997, pp. 376-380.

[12] B. Das, R. Sivakumar and V. Bhargavan, "Routing in Ad Hoc Networks Using a Spine", in *Proc. Sixth IEEE Int. Conf. Computers Comm. and Networks*, 1997, pp. 1-20.

[13] J. Wu and H. Li "A Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks", *Springer Science and Business Media B.V., Formerly Kluwer Academic Publishers B.V.*, 2001.

[14] J. Wu and H. Li, "On Calculating Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks", in *Proc. Third Int. Workshop Discrete Algorithms and Methods for Mobile Computing and Comm.*, 1999, pp. 7-14.

[15] F. Dai and J. Wu, "An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks", *IEEE Transactions On Parallel and Distributed Systems*, vol. 15, no. 10, 2004.

[16] J. Wu, "Extended Dominating-Set-Based Routing in Ad Hoc Wireless Networks with Unidirectional Links", *IEEE Trans. Parallel and Distributed Systems*, vol. 9, no. 3, pp. 189-200, 2002.

[17] X. Yan, Y. Sun and Y. Wang, "A Heuristic Algorithm for Minimum Connected Dominating Set with Maximal Weight in Ad Hoc Networks", in *Proc. GCC2003, Grid and Cooperative Computing, Second International Workshop*, 2003, pp.719-722.

[18] P. J. Wan, K. M. Alzoubi and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks", *Springer Science and Business Media B.V., Formerly Kluwer Academic Publishers B.V.*, vol. 9, no. 2, pp. 141-149, 2002.

[19] H. Liu, Y. Pan and C. Jiannong, "An Improved Distributed Algorithm for Connected Dominating Sets in Wireless Ad Hoc Networks", in *Proc. of ISPA 2004, Parallel and Distributed Processing and Applications, Second International Symposium*, 2004, p. 340.

[20] J. Wu and F. Dai, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks", *IEEE Trans. Parallel and Distributed Systems*, vol. 15, no. 10, 2004.

[21] T. N. Nguyen and D. T. Huynh, "Connected D-Hop Dominating Sets in Mobile Ad Hoc Networks", in *Proc. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2006, pp. 1-8.

[22] I. Stojmenovic, M. Seddigh and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 14-25, 2002.

[23] K. M. Alzoubi, P. J. Wan and O. Frieder, "Maximal Independent Set, Weakly Connected Dominating Set, and Induced Spanners for Mobile Ad Hoc Networks", *International Journal of Foundations of Computer Science*, vol. 14, no. 2, pp. 287303, 2003.

[24] B. Han W. Jia, "Efficient Construction of Weakly-Connected Dominating Set for Clustering Wireless Ad Hoc Networks", in *Proc. IEEE Globecom*, 2006.

[25] B. Han, and W. Jia, "Clustering Wireless Ad Hoc Networks with Weakly-Connected Dominating Set", *Journal of Parallel and Distributed Computing*, vol. 67, no.6, pp.727-737, 2007.

[26] N. Nanuvala, "An Enhanced Algorithm to Find Dominating Set Nodes in Ad Hoc Wireless Networks", Master of Science Thesis in the College of Arts and Science, Georgia State University, 2006.

[27] Y. Li, D. Kim, F. Zou and D-Z. Du, "Constructing Connected Dominating Sets with Bounded Diameters in Wireless Networks", in *Proc. International Conference on Wireless Algorithms, Systems and Applications (WASA 2007)*, 2007, pp. 89-94.

[28] B. Gao, Y. Yang and D. Ma, "A new distributed approximation algorithm for constructing minimum connected dominating set in wireless ad hoc networks", *International Journal of Communication Systems (IJCS)*, vol. 18, pp. 743762, 2005.

[29] W. Wu, H. Du, X. Jia, Y. Li and S. Huang, "Minimum connected dominating sets and maximal independent sets in unit disk graphs", *Theoretical Computer Science*, vol. 352, no. 1, pp. 1-7, 2006.