Maintaining CDS in Mobile Ad Hoc Networks

Kazuya Sakai[1], Min-Te Sun[2], Wei-Shinn Ku[1], and Hiromi Okada[3]

[1] Auburn University, Auburn, AL 36849-5347, USA
{sakaika,weishinn}@auburn.edu
[2] National Central University, Chung-Li, Tao-Yuan 320, Taiwan
msun@csie.ncu.edu.tw
[3] Kansai University, 3-3-35 Yamate-cho Suita, Osaka 564-8680, Japan
okada@jnet.densi.kansai-u.ac.jp

Abstract. The connected dominating set (CDS) has been generally used for routing and broadcasting in mobile ad hoc networks (MANETs). To reduce the cost of routing table maintenance, it is preferred that the size of CDS to be as small as possible. A number of protocols have been proposed to construct CDS with competitive size, however only few are capable of maintaining CDS under topology changes. In this research, we propose a novel extended mobility handling algorithm which will not only shorten the recovery time of CDS mobility handling but also keep a competitive size of CDS. Our simulation results validate that the algorithm successfully achieves its design goals. In addition, we will introduce an analytical model for the convergence time and the number of messages required by the CDS construction.

1 Introduction

The connected dominating set (CDS) has been used as the virtual backbone for routing [9] and broadcast [1] in MANETs. It is defined as a subset of nodes in a network such that each node in the network is either in the set or a direct neighbor of some node in the set, and the induced graph of the nodes in the set is connected. To reduce the cost of routing table maintenance in the virtual backbone, it is preferred that the size of the corresponding CDS to be as small as possible. Although computing the minimal CDS is known to be NP-hard [3], a number of protocols [2, 5, 7, 8, 10, 11] have been proposed to construct CDS with competitive size. Among these protocols, only few [2, 7, 11] are capable of maintaining CDS under topology changes. For the rest of the protocols, when topology changes cause the CDS to be invalid, they will have to construct CDS from scratch, which in general is a time-consuming process.

In this paper, we augment the CDS construction protocols in [11] and [7] by our Extended Mobility Handling (EMH) algorithm. The algorithm consists of two parts. The first part, namely Height-Reduction (HR), focuses on shortening the recovery time of CDS mobility handling; the second part, namely Initiator-Reduction (IR), keeps the competitive size of CDS while doing mobility handling. The simulation results validate that the algorithm successfully achieves its design goals. In addition, we introduce an analytical model for the convergence time and the number of messages required by the CDS construction. We have validated that the results obtained from the analytical model match extremely well with the results from the simulation.

The rest of this paper is organized as follows. The existing distributed CDS protocols are reviewed in Section 2. The EMH algorithm is introduced in Section 3. The simulation results are presented and analyzed in Section 4. The analytical model for the convergence time and the number of messages is demonstrated and validated in Section 5. The conclusion and the future work are provided in Section 6.

2 Literature Reviews

While computing the minimum CDS is known to be NP-hard [3], a number of distributed CDS protocols [2, 5, 7, 8, 10, 11] have been proposed to construct a CDS of small size. Both Wan's protocol [8] and Li's protocol [5] obtain CDS by expending the maximal independent set. On the other hand, Wu's protocol [10] obtains CDS by eliminating unnecessary nodes through a number of rules. Although these three protocols are successful in constructing a small size of CDS based on localized information, they lack the mechanism of mobility handling. For MANETs where nodes are roaming freely all the time, it is imperative for the CDS protocol to be adaptive to the changes of topology.

Recently, several practical distributed CDS construction protocols capable of maintaining CDS [2, 7, 11] under the setting of MANETs have been proposed. Dai's protocol [2] extends the pruning rules of Wu's [10] for mobility handling. The problem of Dai's protocol is that it requires the information of two-hop neighbors. When the topology changes, it takes two beacon periods for a node to initiate the adaptation process. In addition, similar to Wu's protocol, Dai's protocol tends to introduce too much communication overhead.

The Single-Initiator (SI) protocol [11] obtains CDS by forming the dominator tree rooted from a single initiator. SI results in small size of CDS and introduces less communication overhead. In addition, SI is able to detect CDS failures caused by nodal mobility by means of transmitting "heartbeat" signals from the initiator and recovery CDS without reconstructing CDS from scratch in most of the failures. However, SI suffers from a single point of failure, i.e., it will still have to reconstruct the whole CDS from scratch should the single initiator fail.

To tackle this issue, a Multi-Initiator (MI) protocol is proposed in [7]. In MI, several initiators are elected. Each initiator generates a dominator tree in exactly the same manner as SI does in [11] and a few nodes (called bridge nodes hence after) are then included to connect the disjoint trees. In MI, the failure of an initiator will only affect the associated dominator tree. The other part of the CDS will remain intact. As each dominator tree is smaller than that of SI, maintaining CDS is much more efficient.

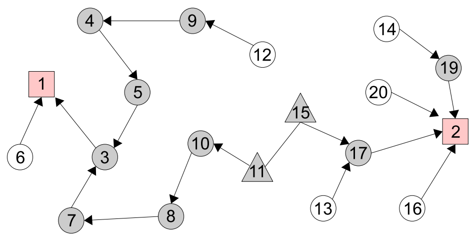


Fig. 1. Original Topology I

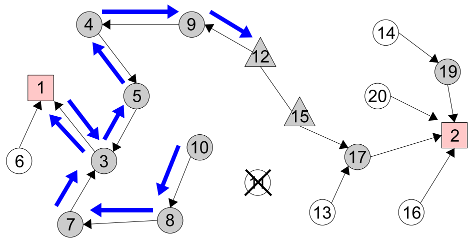


Fig. 2. Mobility Handling of MI on Topology I

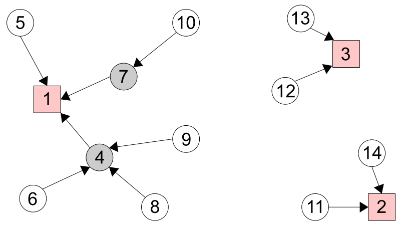


Fig. 3. Original Topology II

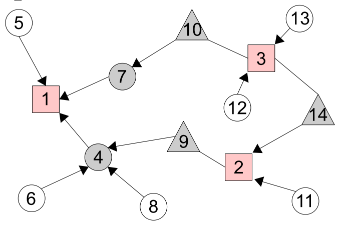


Fig. 4. Mobility Handling of MI on Topology II

3 Extended Mobility Handling for SI and MI Protocols

3.1 Motivation

In general, MI and SI are better than Wu's and Wan's protocols, as they incur less communication overhead, result in CDS of competitive size, and adapt to topology changes. SI successfully maintains a CDS under changes of the network topology that include the following four cases: 1) the initiator leaves its dominator tree; 2) a dominator leaves its dominator tree; 3) a new node joins a dominator tree after the tree is constructed. 4) a redundant dominator switches to dominatee status without disconnecting the dominator tree. In addition to the above four cases, MI also deals with the case of 5) a bridge node, a node which connects two trees, leaves its dominator tree.

However, both SI and MI suffer from two issues. First, the CDS recovery time may increase after a period of time. It is because when new nodes join the network, the height of the dominator tree may increase. when a bridge node moves away from the network, in the worst case, the initiator of one of the disconnected trees needs to assign a new bridge node by exchanging control messages between the initiator and nodes in the boundary area. The time to complete this recovery process is in proportion to the height of the tree. Note that during the recovery process, the dominating set will not be connected. Hence, to ensure the CDS to be available for a longer period of time, it is important that the height of the dominator tree to be small.

For instance, in Figure 1, a snapshot of a MANET and its connected dominating set are illustrated. In the figure, a square represents a initiator, a shaded circle is a dominator, a circle is a dominatee, and a triangle is a bridge node.

The CDS is formed by the initiators, the dominators, and the bridge nodes. An arrow between two nodes indicates their dominator/dominatee relationship (e.g., node 3 is the dominator of node 5) and a solid line between two bridge nodes means they are neighbors to each other. Assume that the bridge node 11 runs out of power, according to MI, its dominator, node 10, will first look for an alternative bridge to connect to the neighboring tree. When node 10 fails to find such a node, it sends a control message to its initiator, node 1. Then, node 1 designates a new bridge node, node 12, to connect to the neighboring tree. The control message will traverse 9 hops before a new bridge node can connect to the neighboring tree. This message traversal is illustrated in Figure 2.

Another issue of SI and MI mobility handling is the possibility of excessive initiators. When a connected network component breaks into multiple pieces due to mobility, each piece will find at least one initiator. When these pieces are merged, all these initiators will become part of the CDS. In addition, since each initiator generates a dominator tree, more initiators imply more trees, thus more bridge nodes will be needed to connect these trees. This further increases the size of CDS.

For example, in Figure 3, the topology is separated into three pieces, and each piece has its own initiator. After the change of topology, as shown in Figure 4, initiator 2 and 3 move to the proximity of the dominator tree rooted at initiator 1. To connect all these dominator trees, MI protocol will include node 9, 10, and 14 into CDS.

3.2 Extended Mobility Handling

To address the aforementioned issues, we propose the Extended Mobility Handling (EMH) algorithm on top of SI and MI. EMH incorporates two procedures, namely Height-Reduction (HR) and Initiator-Reduction (IR). The following subsections elaborate each of these procedures.

Height-Reduction To control the height of a dominator tree, a node in the tree needs to know its depth, i.e., the minimum number of hops between its initiator and itself. Recall that in both SI and MI to learn the *status* of the neighboring node, the *status* of a node is encoded in the beacon. By adding an extra *depth* field in the beacon, the *depth* of a node can be obtained without introducing extra messages. When a node receives a beacon from its dominator, it learns and updates its own *depth*, then encodes its *depth* plus one to the *depth* field of its beacon before transmitting it. If a node finds a dominator neighbor which belongs to the same dominator tree and the *depth* of the dominator is smaller than its current dominator, it changes its dominator and updates its *depth* to reduce the height of the tree. The new dominator is able to know that the node becomes its child in the next beacon period. A dominator node which child changes its dominator may no longer have children. If this situation occurs, the dominator changes it *status* to dominatee. The pseudo code of the Height-Reduction procedure is given in Figure 5.

```
/* Node $i$ executes the following: */
1.   Node $i$ receiving a beacon from $j$ /* change dominator*/
2.        if ($status(j) = dominator \wedge depth(j) < depth(dominator(i))$) then
3.             $dominator(i) \leftarrow j$
4.             $depth(i) \leftarrow depth(j) + 1$
5.   if ($status(i) = dominator$) /* eliminate unnecessary dominator */
6.        if ($\forall j \in N(i), dominator(j) \neq i$) then
7.             $status(i) \leftarrow dominatee$
```

Fig. 5. Height-Reduction

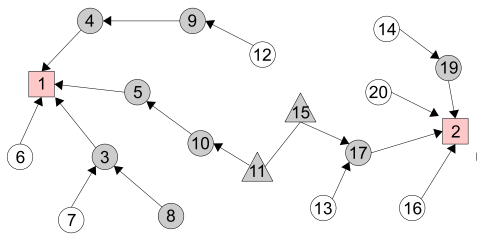


Fig. 6. Topology with Height-Reduction

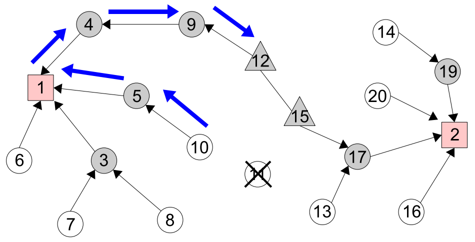


Fig. 7. Mobility Handling of MI after Height-Reduction

The following demonstrates how Height-Reduction works for the topology in Figure 1. Assume dominator node 3 is also a direct neighbor of node 8. After receiving beacons from node 3, node 8 knows $depth(3) = 1$ and $depth(7) = 2$. As node 3 is a dominator and closer to the initiator, node 8 changes its dominator to node 3 and updates its *depth*. Then, since dominator node 7 no longer has children, it changes its *status* to dominatee. Similarly, assume dominator node 5 is a direct neighbor of node 10. After node 10 finds that node 5 has a smaller depth, according to Height-Reduction it will switch its dominator to node 5 and update its *depth*. After a few beacon periods, the original dominator tree rooted at node 1 in Figure 1 will be optimized as shown in Figure 6. Now again assume that the bridge node 11 in Figure 6 runs out of power. As illustrated in Figure 7, the loss of the bridge node is handled in the same manner as in Figure 2. As the height of the tree in Figure 7 is smaller than that of Figure 2, the mobility handling for bridge node 11 can be done more efficiently. While MI needs to send 9 messages to recover the CDS, MI with Height-Reduction needs only 5.

Initiator-Reduction Since each initiator generates a tree, reducing the number of initiators is equivalent to reducing the number of trees. To reduce the number of trees, the small dominator trees can be merged to the large neighboring trees. In essence, if a tree is small, most likely its initiator will also be a boundary node, i.e., the initiator has some neighbor belonging to a different tree. If an initiator finds such a neighbor and its *id* is larger than the initiator *id* of the neighbor, it changes its *status* to dominator, its initiator *id*, and updates its *depth*. The

```
/* Node $i$ executes the following: */
1.  if (node i is initiator) /* merge small dominator tree */
2.      if ($\exists j \in N(i) \wedge initiator(j) \neq i \wedge id(i) > initiator(j)$) then
3.          $dominator(i) \leftarrow j$
4.          $initiator(i) \leftarrow initiator(j)$
5.          $depth(i) \leftarrow dept(j) + 1$
4.          if ($status(j) = dominatee$) then
5.              $status(j) \leftarrow dominator$
6.  Node $i$ receiving a beacon from $j$ /* change initiator id*/
7.      if ($dominator(i) = j \wedge initiator(j) \neq initiator(i)$) then
8.          $initiator(i) \leftarrow initiator(j)$
9.          $depth(i) \leftarrow depth(j) + 1$
```

Fig. 8. Initiator-Reduction

reason to merge the tree with larger initiator *id* to the one with smaller initiator *id* is to avoid the situation that both initiators of neighboring trees try to merge to the other tree simultaneously. After a dominator tree becomes a part of another tree, the children of the merged initiator need to change their *status*. On receiving beacon from its dominator, if a node detects its dominator's initiator *id* is different from its initiator *id*, it updates its initiator *id* and *depth* accordingly. Notice that after Initiator-Reduction procedure is complete, an initiator will not have any neighbor belonging to a different tree. This guarantees that the tree of each initiator including the bridge nodes will be at least two hops in height. The pseudo code of the Initiator-Reduction procedure is provided in Figure 8.

Figure 9 shows how Initiator-Reduction works for the topology in Figure 3. Recall that in Figure 4, initiator 2 and 3 move to the boundary area. The *status* of node 2 is still in initiator, and it has two neighbors 9 and 12 that associate with other initiators. According to the pseudo code, node 2 will switch its *status* to dominator, change its initiator *id*, and update its *depth*. Afterwards, node 9 finds that node 2 become its child, so it will switch its *status* from bridge to dominator. When Node 11 and 14 find that their dominator changed its initiator *id*, they will update their initiator *id* and *depth*. As can be seen, the size of CDS by the MI protocol's mobility handling in Figure 4 is 8, while by MI with Initiator-Reduction in Figure 9 is reduced to 7.

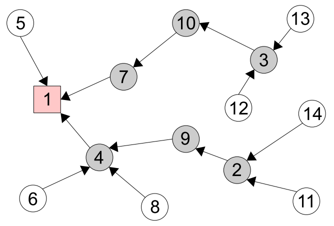


Fig. 9. Topology with Initiator-Reduction

4 Simulation Result

To evaluate the performance of our Extended Mobility Handling (EMH) algorithm, we implemented Dai's [2], SI [11], and MI [7] protocols along with SI with EMH and MI with EMH in C++. In this section, the simulation results of different CDS protocols are reported and analyzed.

4.1 Simulation Configuration

The network topology is randomly generated by placing nodes on a $1000m$ by $1000m$ square field according to uniform distribution. The simulation region is wrapped around vertically and horizontally to eliminate the edge effect. If the generated network is partitioned, it is discarded and a new network topology is generated to ensure the connectivity of the whole network at the beginning of the simulation. The transmission range of a node is set to be $150m$. The number of nodes in the simulation area is set to be 150, which corresponds to 10 neighbors per node, and some nodes are assumed to be mobile. The percentage of the mobile nodes ranges from 20% to 80% with speed up to $5m/s$. For a given simulation configuration, 1000 different network topologies are generated.

The Weighed Way Point (WWP) [4] is adopted as our mobility model. In WWP, the weight of selecting next destination and pause time for a node depends on both current location and time. The value of weights is based on empirical data carried out on University of Southern California's campus [6]. For Dai', SI and MI, the corrupted CDS is recovered according to their mobility handling procedures when the topology changes. The configurations of the heartbeat period in SI, MI, SI with EMH, and MI with EMH are set to be the same as in [7]. Each simulation lasts 1000 rounds of beacon periods. In the simulation, if the network topology is partitioned into disjoint connected components, CDS protocols maintains separate CDS within each component.

To assess the performance of different CDS protocols, four metrics are used, including the percentage of time CDS is alive, the average size of CDS, the number of extra messages, and the average amount of traffic. For MI and MI with EMH, the messages in the tree connection phase and query/response messages in the mobility handling are counted as extra messages. The total amount of time CDS is valid divided by the total simulation period is defined as the percentage of time CDS is alive. For SI, all the information exchanged between nodes are done by beacons. For Dai's protocol, beacons are considered as extra messages since the size of the beacon increases in proportion to the network density and is too large when compared with the standard beacon frame. Each protocol changes the beacon frame format to include additional information. For SI, node *id*, *status*, *color*, and dominator *id* are included in the beacon. MI enlarges the beacon of SI to include the initiator *id* and the minimal *id* of one-hop neighbors. SI with EMH and MI with EMH extend the beacon of SI and MI to add *depth*. The beacon of Dai's protocol includes node *id*, *status*, *marker*, and the list of *id*s for one-hop neighbors.

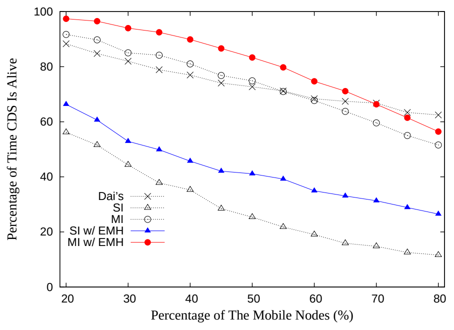


Fig. 10. Percentage of Time CDS is Alive

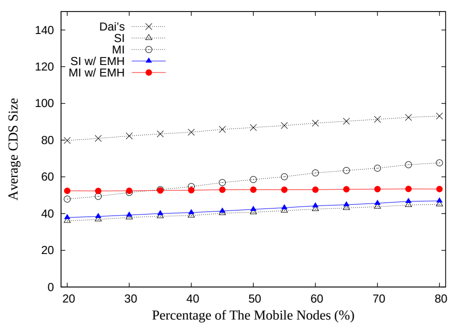


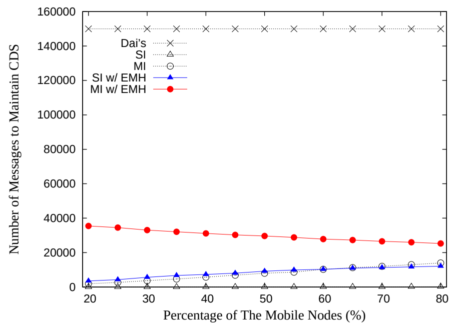
Fig. 11. Average CDS Size

4.2 Simulation Results

Figure 10 illustrates the percentage of time CDS is alive with respect to the percentage of the mobile nodes. As can be seen in Figure 10, MI with EMH almost always has the highest percentage of CDS alive time than other CDS protocols. Although smaller CDS is generally more vulnerable to topology changes, MI with EMH shows excellent mobility adaptation compared with the other protocols. Only when the percentage of mobile nodes is greater than 70% the percentage of CDS alive time of the MI-EMH becomes slightly lower than that of Dai's due to MI with EMH's smaller CDS size. As pointed out in [8], the time complexity of mobility recovery at each node of Dai's is as high as $O(\Delta^2)$, where Δ is the average number of neighbors. Thus, Dai's protocol takes more time to recover than MI with EMH. SI with EMH also improves the percentage of CDS alive time by at least 10% compared with SI. This clearly demonstrates that EMH is capable of prolonging the time CDS is available to MANETs.

Figure 11 shows the average CDS size with respect to the percentage of the mobile nodes. As illustrated in Figure 11, SI and SI with EMH consistently produces the smallest CDS and Dai's protocol consistently produces the largest CDS. While the size of CDS generated by MI increases slowly in accordance with the percentage of mobile nodes, that of MI with EMH is stable. This is because, by merging trees MI with EMH keeps the number of initiators as a constant, and consequently reduce the CDS size. In general, it is desirable that the size of the CDS is as small as possible for MANETs. From Figure 10 and Figure 11, MI with EMH can quickly adapt to topology changes while keeping CDS size competitive.

Figure 12 presents the number of extra messages to maintain CDS with respect to the percentage of the mobile nodes. As illustrated in Figure 12, SI does not introduce any extra message. MI with EMH introduces less than 25% of extra messages of Dai's. In addition, the numbers of extra messages of MI and SI with EMH are roughly the same and are at most 10% of that of Dai's. Compared with MI, MI with EMH introduces more messages. The same can be said to SI and SI with EMH. Unlike the other protocols, the number of extra message created by MI with EMH decreases as the percentage of the mobile nodes increases. This is because IR can better reduce the number of initiators when more initiator



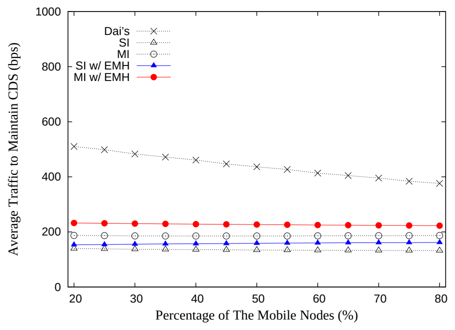


Fig. 12. Number of Messages to Maintain CDS

Fig. 13. Average Traffic to Maintain CDS

moves to the boundary area. By introducing few extra control messages, MI with EMH and SI with EMH become highly adaptive to topology changes, as shown in Figure 10.

Figure 13 shows the average traffic required at each node to maintain CDS with respect to the percentage of the mobile nodes. As can be seen in Figure 13, the average traffic of Dai's protocol is at least twice as much as that of any other protocol. The protocol incorporates EMH has slightly higher traffic than the one without EMH, but the difference is not significant. This is primarily because in EMH the beacon is 4 bit larger to include the depth value. Figure 12 and Figure 13 suggest that the traffic is mostly dominated by the extra bits in the beacon frame.

5 Analytical Model for Convergence Time and Number of Messages

In this section, an analytical model to analyze the convergence time and number of messages that SI and MI require during CDS construction is presented and validated. Note that CDS construction phases of SI and SI with EMH are the same, so are MI and MI with EMH.

5.1 Analytical Model

In the tree construction phase of MI and SI, the defer timer T_d is set to be T_{max}/n_{uc}, where T_{max} is the maximal defer time period and n_{uc} is the number of uncovered neighbors. Since SI is simpler than MI, before we discuss the convergence time of MI, let us first consider the convergence time of SI.

The convergence time of SI should be the product of the number of hops from the initiator to the edge of the tree and the average defer time. To formulate this, let us denote the width and height of the simulation region as l_x and l_y, respectively. The average distance between two nodes, $\overline{d}$, is calculated by Equation 1.

$$\overline{d} = \int_0^r \frac{2\pi x \cdot x}{\pi r^2} dx = \frac{2}{3} r \qquad (1)$$

The number of hops, denoted by h, will be the distance from the initiator to the edge of the tree divided by the average distance between two nodes. Assuming $l_x = l_y$, and $S = l_x \cdot l_y$, then the average value of h can be computed by Equation 2.

$$h = \frac{\sqrt{l_x^2 + l_y^2}}{2\overline{d}} = \frac{3\sqrt{2S}}{4r} \qquad (2)$$

Next, the average defer time of a node is T_{max} divided by the average number of uncovered neighbors n_{uc}. Let C_A and C_B be the coverage area of two neighboring nodes A and B, and d be the distance between them, the additional area that B forwards a message from node A is $|C_B \setminus C_A| = \pi r^2 - |INTC(r, d)|$, where $INTC(r, d)$ is the intersection of two circle with radius r and their centers separated by d.

$$INTC(r, d) = 4 \int_{d/2}^{r} \sqrt{r^2 - x^2} dx \qquad (3)$$

Thus, the average additional coverage area is

$$\int_0^r \frac{2\pi x(\pi r^2 - INTC(r, x))}{\pi r^2} dx \approx 0.41\pi r^2 \qquad (4)$$

Therefore, the average number of uncovered nodes is 0.41Δ, where Δ is the average number of neighbors in transmission range. In addition, a node at the edge of the tree will wait T_{max} number of beacon intervals before it changes its status into dominatee. Finally, the convergence time of SI is approximately

$$(h - 1) \cdot \frac{T_{max}}{0.41\Delta} + T_{max} \qquad (5)$$

The duration of the tree construction phase in MI can be calculated in the similar fashion. In the case of multi-initiator, the number of hops from a initiator to the edge of its tree, h_{mi}, is

$$h_{mi} = \frac{r/\overline{d}}{n_{init}\pi r^2/S} = \frac{3}{2} \cdot \frac{S}{n_{init}\pi r^2} \qquad (6)$$

For the tree connection phase in MI, a border node without any uncovered neighbor will wait T_{max} number of beacon intervals before it sends a message to its initiator. Hence, the time required for the tree connection phase is bounded by $2h_{mi} + 1$. The convergence time of MI is the total time spent on the tree construction phase and the tree connection phase, and can be computed as

$$(h_{mi} - 1) \cdot \frac{T_{max}}{0.41\Delta} + 2h_{mi} + T_{max} + 1 \qquad (7)$$

MI does not introduce extra messages except in the tree connection phase. In the tree connection phase, all nodes except initiators forward messages from their children as many times as the number of neighboring trees to their initiator. Thus, the number of message required for MI to construct CDS is

$$0.41\Delta \cdot \frac{n_{init} - 1}{n_{init}} \cdot n \qquad (8)$$

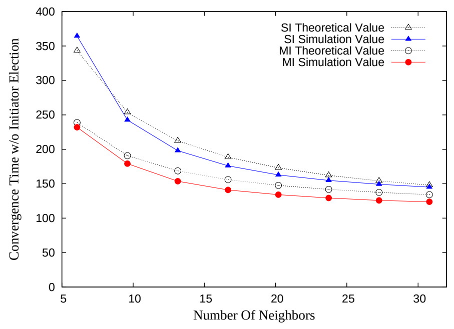


Fig. 14. The Convergence Time w/o Initiator Election

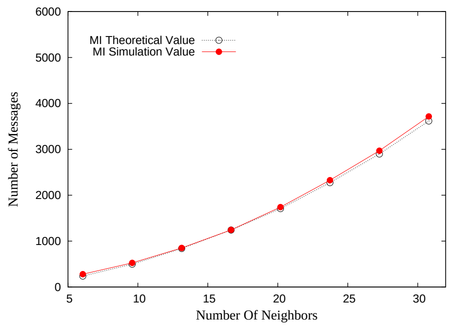


Fig. 15. The Number of Messages

5.2 Comparison of Theoretical and Simulation Results

In this subsection, the analytical model are validated by comparing with our simulation results. As the analytical model are the model for CDS construction, we conducted the simulation in static networks. The average network density changes as we change the total number of nodes. The total number of nodes placed in the field ranges from 100 to 450, which corresponds to the network density ranging from approximately 5 to 30 neighbors per node. The other configuration settings are the same as in Section 4.1.

Figure 14 demonstrates the convergence time SI and MI without the initiator election phase with respect to the average number of neighbors. Since the duration of the initiator election phase is fixed for SI and MI (it is 20 beacon periods for SI and 2 beacon periods for MI), our convergence time analysis omits the initiator election phase. The convergence time decreases in proportion to the number of neighbors because the differ timer in the tree construction phase decreases in proportion to the number of uncovered nodes. As can be seen in Figure 14, the theoretical model and simulation results are very close to each other.

Figure 15 shows the number of messages with respect to the average number of neighbors. In the simulation, the control messages in the tree connection phase are traced. Note that since SI forms only one dominator tree, there will be no control messages for tree connection. As can be seen in Figure 15, analytical model again provides a very accurate estimation.

6 Conclusion and Future Work

The CDS protocols proposed in the past either lack the ability to handle nodal mobility, result in large size of CDS, or incur large overhead. In this paper, an Extended Mobility Handling (EMH) algorithm is proposed for single and multi-initiator CDS protocols to optimize their mobility handling mechanism. By adding a few bits of overhead in the beacon frame, EMH can help SI and MI to quickly recover the corrupted CDS caused by nodal mobility. The simulation results show that EMH helps SI and MI to improve the percentage of time CDS is alive and maintain CDS of competitive size under the setting of MANETs. An analytical model is introduced to estimate the time of convergence and the number of messages for SI and MI protocol families. The estimations match extremely well with the simulation results, which validates our analytical model.

With EMH, the remaining shortcoming of SI and MI is the convergence time required in the tree construction phase. In the future, we would like to investigate means to reduce the convergence time for SI and MI. For instance, a different formula can used to replace the current one in SI and MI to achieve shorter differ timer and still allow nodes with more uncovered neighbors to be included in the dominating set earlier.

References

1. J. Cartigny, D. Simplot, and I. Stojmenovic. Localized Minimum-Energy Broadcasting in Ad-hoc Networks. In *Proceedings IEEE Conference on Computer Communications (INFOCOM)*, pages 2210–2217, Mar. 2003.
2. F. Dai and J. Wu. An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks. *IEEE Transaction on Parallel Distributed Systems*, 15(10):908–920, 2004.
3. D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997.
4. W.-J. Hsu, K. Merchant, H.-W. S., C.-H. Hsu, and A. Helmy. Weighted Waypoint Mobility Model and its Impact on Ad Hoc Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1):59–63, 2005.
5. Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, and D.-Z. Du. On greedy construction of connected dominating sets in wireless networks. *Wireless Communicationss and Mobile Compututing*, 5(8):927–932, 2005.
6. Mobilab. Community-Wide Library of Mobility and Wireless Networks Measurements. http://nile.usc.edu/MobiLib/.
7. K. Sakai, F. Shen, K. M. Kim, M.-T. Sun, and H. Okada. Multi-Initiator Connected Dominating Set for Mobile Ad Hoc Networks. In *Proceedings of International Conference on Communications (ICC)*, May 2008.
8. P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 141–149, Apr. 2002.
9. J. Wu. Extended Dominating-Set-Based Routing in Ad Hoc Wireless Networks with Unidirectional Links. *IEEE Transaction on Parallel Distributed Systems*, 13(9):866–881, 2002.
10. J. Wu and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 7–14, Aug. 1999.
11. D. Zhou, M.-T. Sun, and T. Lai. A Timer-based Protocol for Connected Dominating Set Construction in IEEE 802.11 Wireless Networks. In *Proceedings of International Symposium on Applications and the Internet (SAINT)*, pages 2–8, Jan. 2005.