Fast Connected Dominating Set Construction in Mobile Ad Hoc Networks

Kazuya Sakai
Department of CSSE
Auburn University
Auburn, AL 36849, USA

Min-Te Sun
Department of CSIE
National Central University
Jhongli City, Taoyuan Country 320, Tiwan

Wei-Shinn Ku
Department of CSSE
Auburn University
Auburn, AL 36849, USA

***Abstract*—The connected dominating set (CDS) has been commonly used for routing and broadcast in mobile ad hoc networks (MANETs). Considering the applications of MANETs, it is generally preferred that the CDS protocol not only creates CDS of small size, incurs less communication and computational overheads, adapts to the nodal mobility, but also generates CDS as quickly as possible. Unfortunately, none of the existing CDS protocols possesses all of these desirable properties. In this paper, we propose an algorithm to rapidly grow a CDS tree from an initiator. By incorporating this algorithm with the Multi-Initiator CDS protocol, we have created the CDS protocol for MANETs which enjoys all the aforementioned desirable properties. The simulation results validate that our proposed protocol achieves its design goals. In addition, an analytical model is provided that can accurately estimate the convergence time required by our proposed CDS protocol.**

I. Introduction

Mobile ad hoc networks (MANETs) are well-suited for applications where a fixed infrastructure is not readily available, such as communications in the battlefield and rescue missions in case of a catastrophe. The connected dominating set (CDS) is extensively used as a virtual backbone for MANET protocols to provide different communication primitives, such as routing [1] and broadcast [2]. A CDS is defined as a subset of nodes in a network such that each node in the network is either in the set or a neighbor of some node in the set, and the induced graph of the nodes in the set is connected.

To reduce the communication and storage overhead, the size of the CDS should be minimized. However, it is known that the problem of finding the minimum CDS is NP-hard [3]. The problem becomes even more challenging when only localized information is available at each node, which is usually the case in MANETs. This forces the existing MANET CDS protocols [4]–[9] to emphasize on constructing a small CDS distributively with localized information. When several factors, including the size of the CDS, the communication overhead incurred in the CDS construction, and the mechanism for mobility handling, are taken into consideration, studies [8], [9] have shown that the tree-based CDS protocols outperform the others. The key component in the tree-based CDS protocols is dominator tree construction. It utilizes a defer timer at each node to distributively assign a higher priority to nodes with more neighbors to CDS and effectively reduce the size of CDS with only one-hop localized information. However, the use of defer timers also creates two potential issues. First, defer timers prolong the CDS construction and increase the convergence time of the protocol. Second, if a different defer timer function is used to reduce the convergence time, the protocol will suffer from poor scalability.

In this paper, we propose a novel tree construction protocol, namely Fast Convergence Dominator Tree Construction (FC-DTC) protocol, for the tree construction phase of the tree-based CDS protocols. Unlike the original tree construction protocol, the FC-DTC protocol does not rely on defer timers. It not only significantly reduces the convergence time for CDS construction but also keeps all the advantages of the tree-based CDS protocols, such as small CDS, low overhead, and mobility handling mechanism. The simulation results have confirmed that the tree-based CDS protocols incorporating the FC-DTC protocol converge quickly and perform well for all of the aforementioned factors. In addition, an analytical model is provided to analyze the convergence time of the tree-based CDS protocols incorporating the FC-DTC protocol and the results match well with the findings of our simulation.

The rest of this paper is organized as follows. The existing distributed CDS protocols are reviewed in Section II. The FC-DTC protocol is presented in Section III. The simulation results are reported and analyzed in Section IV. The analytical model for the convergence time is demonstrated and validated in Section V. The conclusion and the future work are provided in Section VI.

II. Literature Review

Constructing the minimum CDS is known to be NP-hard [3]. While there has been research on how to approximate the minimum CDS under the assumption that the complete network topology is known [3], such an assumption is not practical for MANETs. The more practical approaches, such as [4]–[9], construct a small CDS based on localized information. Depending on the nature of the approach, these localized CDS protocols can be classified into substraction-based and addition-based.

- **Substraction-based CDS construction** - The substraction-based CDS protocol begins with the set of all nodes in the network, then systematically removes nodes to obtain the CDS. The best known CDS protocols in this category include Wu [4] and Dai's [5] protocols. Both of these two CDS protocols consist of two stages. In the first stage, each node collects two-hop neighboring information by exchanging messages with its one-hop neighbors. If a node finds that there is a direct link between any pair of its one-hop neighbors, it removes itself from the consideration of the CDS. In

the second stage, additional heuristic rules are applied to further reduce the size of the CDS. Since Wu and Dai's protocols use two-hop neighbor information, a node needs at least two beacon periods to obtain the latest topology information before it can react to any topology change caused by nodal mobility.

- **Addition-based CDS construction -** The addition-based CDS protocol starts from a subset of nodes (commonly disconnected), then includes additional nodes to form the CDS. Depending on the type of the initial subset, the addition-based CDS protocols can be further divided into MIS-based and tree-based.
  - The MIS-based CDS protocol [6], [7] obtains the CDS by expending the maximal independent set (hence the name MIS). The MIS-based protocol assumes the model of the unit-disk graph. It is also a two-stage protocol. In the first stage, the maximal independent set of the given network topology is constructed distributively by recursively selecting the nodes with the most neighbors locally. The nodes in the MIS become the skeleton of the CDS. Although nodes in the MIS are not connected, the distance between any pair of its complementary subsets is known to be exactly two hops away. Hence, in the second stage, a localized search is used to include additional nodes to connect the nodes in the MIS and form the CDS. The difference between the protocols in [6] and [7] lies in how nodes in MIS are connected in the second stage (i.e., top-down or bottom-up fashion).
  - The tree-based protocol, such as Single-Initiator (SI) version in [9] and Multi-Initiator (MI) version in [8], starts from a subset of nodes *called initiators* and grows a CDS tree from each of the initiators. The tree-based protocol can be divided into three phases. In phase one, a number of initiators is elected from the network. In phase two, each initiator utilizes the timer to grow a tree so that the nodes with more neighbors can be added to the tree. In phase three, additional bridge nodes are added to connect neighboring trees. Notice that in [9], there is only one initiator, so there is no need for the tree connection phase.

While all these aforementioned protocols create CDS with localized information, it has been shown in [8] that the addition-based protocols generally produce smaller CDS than the substraction-based ones and the tree-based protocols incur less communication overhead. In addition, the only CDS protocols capable of maintaining CDS under the change of network topology are Dai's [5] and the tree-based [8], [9] protocols. The algorithms in [10] further help to optimize the mobility handling for the tree-based CDS protocols. All the other protocols [4], [6], [7] will have to construct the CDS from scratch when the CDS is corrupted due to nodal mobility. Based on these considerations, the tree-based CDS protocols seem to be a better candidate for MANETs. However, the issue with the tree-based CDS protocols is that they use timers to generate trees. When the network size is large, the tree-based CDS protocols will need more time to construct the CDS. Hence, we are interested in finding a mean to quickly construct the CDS and at the same time enjoy the advantages of the tree-based CDS protocols.

III. Fast CDS Construction Protocol

A. Motivation

As described in Section II, tree-based CDS protocols are suitable for MANETs as they tend to result in smaller CDS, introduce less communication overhead, and adapt to nodal mobility. However, they still suffer from slow convergence due to the use of defer timers in their tree construction phase. In the rest of this paper, a tree generated from an initiator in SI and MI is referred as a dominator tree and the protocol used in the tree construction phase in SI and MI as Timer-Based Dominator Tree Construction (TB-DTC). In TB-DTC, when a node finds that one of its neighbors joins the CDS, it sets a defer timer inversely proportional to the number of uncovered neighbors and join the CDS only if it still has at least one uncovered neighbor when its timer expires. Specifically, the defer timer is calculated using Equation 1.

$$T_d = \begin{cases} \frac{T_{max}}{(n_{uc})^{\beta}} & if\ n_{uc} >= 1 \\ T_{max} & if\ n_{uc} < 1 \end{cases} \quad (1)$$

where n_{uc} is the number of uncovered neighbors. For the convenience of our discussion, we assume $\beta = 1$ in this paper. Since nodes with more uncovered neighbors clearly have a shorter timer, they have a higher probability that at least one of its neighbors will still be uncovered when its timer expires, hence a higher probability to be included in the CDS. The convergence time of dominator tree construction is bounded by $h \cdot T_{max}$, where h is the number of hops from an initiator to the edge of its dominator tree. When the network contains some long chains of nodes, it will take a long time for the protocol to converge. For instance, Figure 1 shows the worst scenario of TB-DTC. In Figure 1, the shaded square represents an initiator, a shaded circle represents a dominator, an unshaded circle represents a dominatee, and a dotted circle represents a covered or uncovered node. An arrow represents the relationship between the dominator and its dominatee (e.g., node 1 is the dominator of node 2), and a dotted line represents a link between two nodes. Suppose that in the initiator election phase, node 1 is elected as the initiator. Then, node 1 switches its status to dominator and covers node 2. On receiving beacons from its neighbors, node 2 learns that its only uncovered neighbor is node 3. Assuming $T_{max} = 100$ beacon periods, node 2 sets its timer, T_d, to 100 beacon periods. This process is repeated until all nodes turn to either dominator or dominatee as illustrated from the top to the bottom in Figure 1. The convergence time will be 300 beacon periods for this small chain with only 4 nodes. When the network density is low, this phenomenon will happen

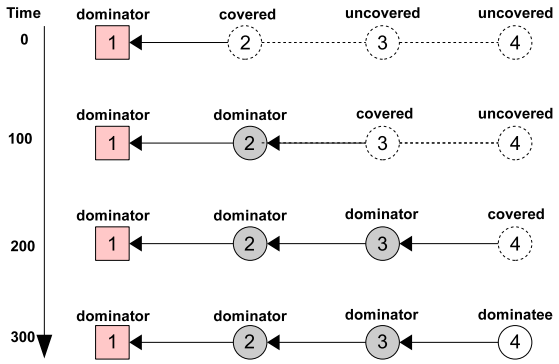

Fig. 1. An example of tree connection

TABLE I
DEFINITION OF NOTATIONS

Symbol	Definition
$N(i)$	The set of one-hop neighbors of node i
$N[i]$	$N(i) \cup \{i\}$
$init(i)$	The initiator of node i
$dominator(i)$	The dominator of node i
$n_{uc}(i)$	The number of uncovered neighbors of node i
$flag(i)$	Set to TRUE if node i is a candidate dominator in the next beacon period

frequently, which will result in long CDS convergence time for TB-DTC.

Another problem of TB-DTC is its poor scalability. In Equation 1, T_d has an upper bound of T_{max}. This is because in case of extremely high network density TB-DTC can not distinguish which node has more uncovered neighbors if each node has more than T_{max} number of neighbors.

The intuitive solution to slow convergence times is to use a different defer timer function, such as a log timer. If the log timer is upper bounded by a smaller value, the tree construction phase can be completed more quickly. However, this change will make the TB-DTC protocol even less scalable as more nodes will be mapped to the same defer timer value. Hence, it is preferable that the fast tree construction protocol does not rely on defer timers.

In this paper, we propose a novel dominator tree construction protocol, namely Fast Convergence Dominator Tree Construction (FC-DTC) protocol. By using FC-DTC instead of TB-DTC, the convergence time of SI [9] and MI [8] CDS protocols can be significantly improved. In the next subsection, we elaborate on the FC-DTC protocol.

B. Fast Dominator Tree Construction

The notations used in our FC-DTC protocol are given in Table I. Just like TB-DTC, in FC-DTC each node encodes its $initiator$ id, $status$, and $dominator$ id into the beacon frame [8]. In addition, each node also encodes the number of its uncovered nodes into its beacon. A node can be in either $dominator$, $dominatee$, $covered$, or $uncovered$ status. At the beginning, all nodes are $uncovered$. Similar to what is defined in the IEEE 802.11 standard [11], each node periodically broadcasts its beacon. Through the exchange of beacons, a node learns the information of its one-hop neighbors, such as their status and n_{uc}.

After the initiator election phase, each initiator immediately switches its status to $dominator$ and sets its initiator to itself. When an uncovered node receives the first beacon from a dominator neighbor, it sets its dominator to this neighbor, its initiator to the neighbor's initiator, and then switches its status to $covered$. By listening to beacons, each node knows how many uncovered neighbors it has. A boolean variable $flag$ is used at each node to track if a covered node should be switched to $dominator$ status. If a node has the highest value of n_{uc} among its one-hop neighbors, is in $covered$ status, and belongs to the same dominator tree for at least two beacon periods, it sets its $flag$ to be 1 and waits for one beacon period. In the next beacon period, if it still has the highest value of n_{uc} and it has at least one uncovered node, it switches to $dominator$ status. Otherwise, it sets its $flag$ back to 0 and stays in $covered$ status. In other words, if a node has the highest value of n_{uc} for two beacon periods, it changes its status to $dominator$. If two nodes have the same value of n_{uc}, the tie can be broken by their id. During the process, if a node does not have any uncovered neighbor, it switches to $dominatee$ status. Eventually, all nodes will switch their status to either $dominator$ or $dominatee$. The pseudo code of this procedure is given in Figure 2.

FC-DTC constructs a dominator tree much quicker than TB-DTC. For instance, in Figure 1, after node 1 is elected as the initiator, in one beacon period node 2 will know it has the largest n_{uc} from the beacons. It then waits one additional beacon period and switches to $dominator$ status. This process is repeated until all nodes are either $dominator$ or $dominatee$, which will take only 7 beacon periods.

FC-DTC achieves the following two design goals. First, no matter how many neighbors each node has, FC-DTC guarantees that the node with the most uncovered nodes among one-hop neighbors becomes a $dominator$ earlier. This will help reduce the size of the CDS. Second, although FC-DTC extends the beacon frame by 8 bits, it still uses only one-hop localized information and does not introduce any extra control messages. In short, FC-DTC reduces the convergence time of CDS construction while at the same time keeping the advantages of TB-DTC.

C. Mobility Handling

Since the initiator election and the tree connection phases are not changed, the SI [9] and MI [8] protocols that incorporate our FC-DTC protocol can deal with nodal mobility in the similar manner as the original SI and MI protocols. In general, the corruption of CDS is caused by one or more of the following five different nodal mobility cases.

- The initiator leaves its dominator tree.
- A redundant dominator switches to $dominatee$ status without disconnecting from the dominator tree.
- A dominator leaves its dominator tree.
- A bridge node leaves its dominator tree.

```
Node i executes following */
    Initialization:
        init(i) ← −1
        status(i) ← uncovered
        doinator(i) ← −1
        n_uc(i) ← −1
        flag(i) ← 0
    Node i detects itself as initiator:
        init(i) ← i
        status(i) ← dominator
    On receiving beacon from dominator node j:
        if (initiator(i) = −1) then
            init(i) ← init(j)
            dominator(i) ← j
            status(i) ← covered
    Node i in covered status ∧flag(i) = 0:
        if (i = candidate(i, init(i), N(i))) then
            flag(i) ← 1
            wait one beacon period
    Node i in covered status ∧flag(i) = 1:
        if (i = candidate(i, init(i), N(i))) then
            if (∃j ∈ N(i) ∧ status(j) = uncovered) then
                status(i) ← dominator
            else
                status(i) ← dominatee
        if (i ≠ candidate(i, init(i), N(i))) then
            flag(i) ← 0
/* return id with the highest value of n_nc among N[i] */
    Procedure candidate(i, init(i), N(i)):
        MaxID ← i
        MaxValue ← n_uc(i)
        for all j in N(i)
            if (init(j) = init(i) ∧ status(j) = covered
                ∧MaxValue < n_uc(j)) then
                MaxID ← j, MaxValue ← n_uc(j)
            if (status(j) = covered ∧ init(j) = init(i)
                ∧max = n_uc(j) ∧ id(j) < id(i)) then
                MaxID ← j, MaxValue ← n_uc(j)
        return MaxID
```

Fig. 2. Fast convergence dominator tree construction pseudo code

- A new node joins a dominating tree after the tree is constructed.

It is clear that the case 1, 2, and 4 can be handled in exactly the same manner as the original SI and MI protocols. As discussed in [9], case 3 can be handled by the color scheme because the color scheme does not rely on timers. Therefore, the only case that needs to be addressed is when a new node joins a dominator tree after the tree is constructed.

When a new node, say node j, joins a dominator tree, all its neighbors are in either *dominator* or *dominatee* status. If node j can find at least one dominator neighbor, it can be covered by one of its dominator neighbors. If node j cannot find a dominator neighbor, all the dominatee neighbors of node j's can switch to *covered* status. Then, the dominator tree construction phase begins again for node j and its neighbors. The pseudo code for handling this mobility case is given in Figure 3.

```
Node i executes following */
    Node i in dominatee status
        if (∃j ∈ N(i) ∧ status(j) = uncovered) then
            status(i) ← covered
            flag(i) ← 0
```

Fig. 3. Additional pseudo code for mobility handling

IV. Simulation Results

To evaluate the performance of SI with FC-DTC and MI with FC-DTC, we implement our protocol in C++ along with the other CDS protocols, including Dai's [5], Wan's [6], the original SI [9], and the original MI [8] protocols. Notice that the original SI and MI protocols use TB-DTC in the dominator tree construction phase. In this section, the simulation results of different CDS protocols are reported and analyzed.

A. Simulation Configurations

The network topology was randomly generated by placing nodes on a $1000m$ by $1000m$ square field according to the uniform distribution. The simulation region was wrapped around vertically and horizontally to eliminate the edge effect. If the generated network was partitioned, it was discarded and a new network topology was generated to ensure the connectivity of the whole network. The average network density changes as we change the total number of nodes. The total number of nodes placed in the field ranged from 100 to 450, which corresponds to the network density ranging from approximately 5 to 30 neighbors per nodes. The transmission range of a node is set to be $150m$. For a given simulation configuration, 1000 different network topologies were generated.

To assess the performance of different protocols, four metrics were used, including the size of CDS, the convergence time, the number of extra messages, and the average traffic. The convergence time is defined as the number of beacon intervals for a CDS protocol to complete. For SI with TB-DTC and SI with FC-DTC, the initiator election phase was assumed to be completed in 20 beacon periods. According to [8], 5 beacon periods is enough to elect a unique initiator. However, in the reality, there may be collisions between beacon transmissions and the nodal mobility. It is desirable to set duration time to be much larger value than the network density. For MI with TB-DTC and MI with FC-DTC, the initiator election was assumed to complete in three beacon periods. For MI with TB-DTC and MI with FC-DTC, the messages in the tree connection phase were counted as extra messages. For SI with TB-DTC and SI with FC-DTC and SI protocol, there are no extra messages as all the information exchanges between nodes were done by beacons. For Dai's protocol, beacons were considered as

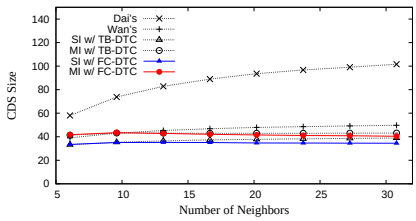

Fig. 4(a) The CDS size

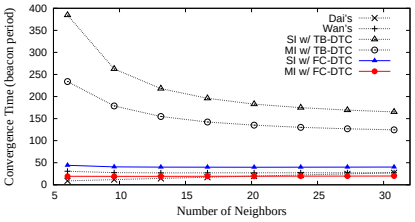

Fig. 4(b) The convergence time

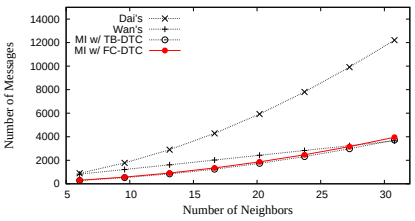

Fig. 4(c) The number of messages

Fig. 4. The simulation results of different CDS protocols

extra messages since the size of their beacon increases in proportion to the network density and was too large when compared with the standard beacon frame. For Wan's protocol, the messages exchanged in both stages are counted as extra messages. Each protocol changed the beacon frame format to include additional information. For SI with TB-DTC, node id, $status$, $color$, and dominator id were included in the beacon. MI with TB-DTC further enlarged the beacon of SI protocol to include the initiator id and the minimal id of one-hop neighbors. Both SI with FC-DTC and MI with FC-DTC enlarge the beacon by adding n_{uc} (8 bits per beacon). The beacon of Dai's protocol included node id, $status$, $marker$, and the list of ids for one-hop neighbors. For Wan's protocol, dominator id and $color$ were added to the beacon. The extra bit in the beacon and the size of extra messages for each protocol were counted toward the traffic (in bps) for CDS construction.

B. Simulation Results

Figure 4(a) shows the CDS size of different protocols with respect to the network density for different protocols. It is clear that SI with FC-DTC consistently generates the smallest CDS and Dai's consistently generates the largest CDS among all the protocols. In addition, except Dai's protocol, the size of CDS generated by other protocols is not sensitive to the density of the network. In the case of high network density, SI with FC-DTC and MI with FC-DTC result in a slightly smaller size for CDS than the original SI and MI protocols. This is because FC-DTC guarantees that nodes with more uncovered neighbors will be switched to $dominator$ first. This suggests that FC-DTC is more scalable than TB-DTC. While it has been proven that the size of Wan's CDS is suboptimal [6], both FC-DTC and MI with FC-DTC are able to generate CDS with a smaller size.

Figure 4(b) presents the convergence time with respect to the network density for different protocols. As shown in Figure 4(b), SI with FC-DTC and MI with FC-DTC significantly reduce the convergence time compared with the original SI and MI protocols. Even at the high network density, SI with FC-DTC and MI with FC-DTC reduce the convergence time by more than 80% compared with their respective TB-DTC counterpart. While the convergence time of the original SI and MI protocols are affected by the network density, that of SI with FC-DTC and MI with FC-DTC is stable. Notice that MI with FC-DTC creates CDS faster than even Dai's protocol. Compared with Wan's protocol, the convergence time of SI with FC-DTC and MI with FC-DTC is almost the same.

Figure 4(c) demonstrates the number of extra messages with respect to the network density for different protocols. As illustrated in Figure 4(c), MI with FC-DTC introduces only 30% of the extra messages that Dai's does. Compared with Wan's, MI with FC-DTC reduces the number of extra messages up to 60% when the network density ranges from 5 to 15 neighbors per node. Note that both SI with TB-DTC and SI with FC-DTC does not introduce any extra message since they do not have a tree connection phase.

Figure 5(a) shows the average traffic incurred by the CDS construction with respect to the network density for different protocols. Obviously, Dai's protocol introduces the largest traffic, and the traffic increases in proportion to the network density. This is because of the inclusion of the neighboring list in the beacon frame. For the other five protocols, the average traffic is almost the same. Note that SI with FC-DTC and MI with FC-DTC enlarges the beacon frame of the original SI and MI protocols by only 8 bit. It is interesting that even though the number of messages for MI with FC-DTC, MI with TB-DTC, and Wan's protocols increase in proportion to the number of neighbors as shown in Figure 4(c), the traffic remains relatively constant to the network density. This implies that the traffic is mostly dominated by the extra bits in the beacon frame.

V. Analytical Model for The Convergence Time

In this section, an analytical model to analyze the convergence time that SI with FC-DTC and MI with FC-DTC require during CDS construction is presented and validated.

In the initiator election phase, it takes at least n beacon periods to elect a n-hop local minimum. As described in [10], the convergence time in the tree connection phase for the original MI protocol is $2h_n + 1$, where h_n is the number of hops from an initiator to the edge of the tree when the n-hop local minimum is used as the initiator. (The calculation of h_n has been discussed in [10] and thus is omitted here.) For the original SI protocol, n is set to be 20, so $h_{20} = \frac{3\sqrt{2S}}{4r}$, where S is the simulation region and r is the transmission range. For the original MI protocol, n is set to be 2, so $h_2 = \frac{3}{2} \cdot \frac{S}{n_{init2}\pi r^2}$. Here, n_{init2} is the number of local minimum. When $n = 2$, n_{init2} is approximately $\frac{9}{25} \cdot \frac{S}{\pi r^2}$.

The one-hop neighbors of an initiator requires one beacon period to become dominator or dominatee, as they have enough information to initiate the protocol during the initiator election phase. Afterwards, it takes two beacon periods to advance one hop from the initiator. Thus, the convergence

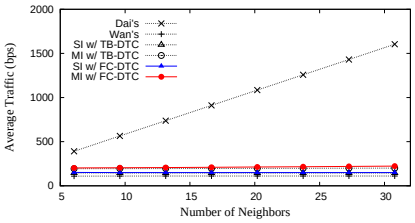

Fig. 5(a) The average traffic (bps)

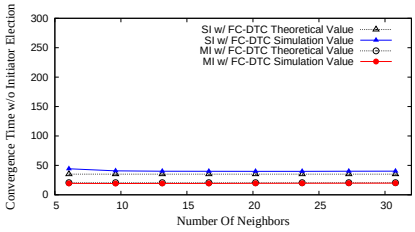

Fig. 5(b) The convergence time

Fig. 5. The simulation results of different CDS protocols

time in the dominating tree construction of FC-DTC can be calculated by $2 \cdot (h_n - 1) + 1$.

For SI with FC-DTC, the convergence time is obtained by Equation 2.

$$n + 2(h_n - 1) + 1 = n + 2h_n - 1 \tag{2}$$

For MI with FC-DTC, the convergence time is estimated by Equation 3.

$$n + 2(h_n - 1) + 1 + 2h_n + 1 = n + 4h_n \tag{3}$$

Figure 5(b) depicts the values calculated from our analytical model and the results obtained from the simulation. As can be seen in Figure 5(b), our analytical model provides a very accurate estimation in terms of the convergence time for SI with FC-DTC and MI with FC-DTC.

Remark: Note that so far we have assumed a node can collect the updated information from all its neighbors within one beacon period. In reality, since the transmission of beacons is generally done without coordination, there may be collisions between beacon transmissions. To accommodate this issue, each node can wait for a fixed number of beacon periods, say T_{wait}, after it finds it has the highest value of n_{uc}. In this case, the convergence time of the dominator tree construction for SI with FC-DTC and MI with FC-DTC will be $(T_{wait} + 1)h_n$ and $(T_{wait} + 1)(h_n - 1) + 1$, respectively. This increases the convergence time slightly, but the result will still be much faster than SI with TB-DTC and MI with TB-DTC. For instance, when $T_{wait} = 3$, the convergence time is less than 50 beacon periods for SI with FC-DTC and less than 33 beacon periods for MI with FC-DTC. Even with the highest network density (30 neighbors per node), the convergence time is at least 165 beacon periods for SI with TB-DTC and at least 124 for MI with TB-DTC.

VI. Conclusion and Future Work

It is preferred that CDS protocols result in small size of CDS, incur less communication overhead, complete quickly, and adapt to the changes of network topology. The previous works [8], [9] significantly improved the performance of the CDS protocol except for the convergence time. In this paper, we proposed the Fast Convergence Dominator Tree Construction (FC-DTC) protocol. FC-DTC significantly reduce the convergence time and the issue of scalability in the previous timer-based tree construction protocol. The simulation results show that SI with FC-DTC and MI with FC-DTC perform well or even better in terms of CDS size, the convergence time, and communication overhead. In addition, an analytical model is built to estimate the time of convergence of both SI with FC-DTC and MI with FC-DTC protocols.

In the tree construction phase, our CDS protocol introduces many control messages. While the number of extra messages is fewer than other protocols, it still increases in proportion to the network density. Since the extra messages in the tree-based CDS protocols come from the tree connection phase, in the future we would like to investigate the message optimal tree connection protocol. For instance, if disjoint trees can be connected by using local broadcasts rather than relying on initiators, the number of control messages will be significantly reduced.

References

[1] J. Wu, "Extended Dominating-Set-Based Routing in Ad Hoc Wireless Networks with Unidirectional Links," *IEEE Transaction on Parallel Distributed Systems*, vol. 13, no. 9, pp. 866–881, 2002.

[2] J. Cartigny, D. Simplot, and I. Stojmenovic, "Localized Minimum-Energy Broadcasting in Ad-hoc Networks," in *Proceedings IEEE Conference on Computer Communications (INFOCOM)*, Mar. 2003, pp. 2210–2217.

[3] D. S. Hochbaum, Ed., *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997.

[4] 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)*, Aug. 1999, pp. 7–14.

[5] 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*, vol. 15, no. 10, pp. 908–920, 2004.

[6] 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)*, Apr. 2002, pp. 141–149.

[7] 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*, vol. 5, no. 8, pp. 927–932, 2005.

[8] 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.

[9] 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)*, Jan. 2005, pp. 2–8.

[10] K. Sakai, M.-T. Sun, W.-S. Ku, and H. Okada, "Maintaining CDS in Mobile Ad Hoc Networks," in *Proceedings of International Conference on Wireless Algorithms, Systems and Applications (WASA)*, Oct. 2008.

[11] "IEEE 802.11b Standard," http://standards.ieee.org/getieee802/download/802.11b-1999.pdf.