Timer-Based CDS Construction in Wireless Ad Hoc Networks

Kazuya Sakai, *Student Member*, *IEEE*, Scott C.-H. Huang, *Member*, *IEEE*, Wei-Shinn Ku, *Member*, *IEEE*, Min-Te Sun, *Member*, *IEEE*, and Xiuzhen Cheng, *Member*, *IEEE*

Abstract—The connected dominating set (CDS) has been extensively used for routing and broadcast in wireless ad hoc networks. While existing CDS protocols are successful in constructing CDS of small size, they either require localized information beyond immediate neighbors, lack the mechanism to properly handle nodal mobility, or involve lengthy recovery procedure when CDS becomes corrupted. In this paper, we introduce the timer-based CDS protocols, which first elect a number of initiators distributively and then utilize timers to construct a CDS from initiators with the minimum localized information. We demonstrate that our CDS protocols are capable of maintaining CDS in the presence of changes of network topology. Depending on the number of initiators, there are two versions of our timer-based CDS protocols. The Single-Initiator (SI) generates the smallest CDS among protocols with mobility handling capability. Built on top of SI, the Multi-Initiator (MI) version removes the single point of failure at single-initiator and possesses most advantages of SI. We evaluate our protocols by both the ns-2 simulation and an analytical model. Compared with the other known CDS protocols, the simulation results demonstrate that both SI and MI produce and maintain CDS of very competitive size. The analytical model shows the expected convergence time and the number of messages required by SI and MI in the construction of CDS, which match closely to our simulation results. This helps to establish the validity of our simulation.

Index Terms—Connected dominating set, virtual backbone, ad hoc networks, distributed algorithms.

1 Introduction

Wireless ad hoc networks are well-suited for communications in sensor networks as well as the battlefield and rescue missions where a fixed infrastructure is not readily available [1]. The effectiveness of many communication primitives for wireless ad hoc networks, such as routing [2], multicast/broadcast [3], and service discovery [4], rely heavily on the availability of a virtual backbone. A virtual backbone of a wireless network is typically the connected dominating set (CDS) of the graph representation of the network. 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 neighbor of some node in the set, and the induced graph of the nodes in the set is connected.

In general, the smaller the CDS is, the less communication and storage overhead the protocols making use of CDS will incur. Hence, it is desired that the size of the CDS for wireless ad hoc networks to be as small as possible. On the other hand, it is known that the problem of finding the minimum CDS is NP-hard [5]. The problem becomes even more intriguing when no node has the view of the complete network topology, which is generally the case in most wireless ad hoc networks. As a result, the existing CDS protocols for wireless ad hoc networks [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] emphasize on constructing a small CDS distributively with localized information. While these protocols are successful in creating a small CDS, they either lack the mechanism to maintain the CDS transparently when network topology changes from time to time. Notice that the topology change can be the result of node mobility, the power outage of some nodes in the network, the deployment of additional nodes to the network, or the combination of the aforementioned cases. The reconstruction of CDS from scratch can keep the CDS from being available for service for an extended period of time.

In this paper, we first introduce a timer-based Connected Dominating Set protocol, namely Single-Initiator (SI). In the first phase, SI distributively elects an unique initiator. In the second phase, SI grows a dominator tree from the initiator to form the CDS by using timers. While SI has advantage over the CDS protocols in [6], [7], [9], [10], [12], [16], [17], [18], [19], [20], [21] in terms of mobility handling, it suffers from the issue of a single point of failure (i.e., the CDS has to be reconstructed when the single-initiator leaves the network). To resolve this issue, we introduce the second CDS protocol, namely Multi-Initiator (MI). Unlike SI, MI uses *a set of initiators* to grow a number of dominator trees and then connect the trees to form the CDS. Since MI uses SI to construct each dominator tree, it inherits the advantages of SI (e.g., mobility handling) and at the same time avoid the single point of failure issue. The simulation results of both static and mobile wireless ad hoc network scenarios validate that our timer-based CDS protocols construct and

- *K. Sakai and W.-S. Ku are with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849. E-mail: {sakaika, weishinn}@auburn.edu.*
- *S.C.-H. Huang is with the Department of Electrical Engineering, National Tsing Hua University, Taiwan. E-mail: chhuang@ee.nthu.edu.tw.*
- *M.-T. Sun is with the Department of Computer Science and Information Engineering, National Central University, Taiwan. E-mail: msun@csie.ncu.edu.tw.*
- *X. Cheng is with the Department of Computer Science, The George Washington University, 801 22nd St. NW, Suite 704, Washington D.C. 20052. E-mail: cheng@gwu.edu.*

Manuscript received 8 Feb. 2009; revised 21 May 2010; accepted 1 Oct. 2010; published online 17 Dec. 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-02-0045. Digital Object Identifier no. 10.1109/TMC.2010.244.

1536-1233/11/$26.00 © 2011 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

maintain CDS of competitive size with low overhead. An analytical model of the convergence time and the number of messages required by the SI and MI CDS protocols is presented and the results obtained from the analytical model match well with the results from the simulation.

The rest of this paper is organized as follows: The existing CDS protocols for wireless ad hoc networks are reviewed in Section 2. The two timer-based CDS protocols are described in detail in Section 3. The simulation results are shown in Section 4. An analytical model for the convergence time and the number of messages is presented for validation in Section 5. The conclusion and the future direction of our work are provided in Section 6.

2 Literature Review

Constructing the minimum CDS is known to be NP-hard [5]. While there has been research on how to approximate the minimum CDS under the assumption that the complete network topology is known [22], [23], [24], such an assumption is not practical for wireless ad hoc networks. The more practical approaches, such as [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], construct a CDS based on localized information. Among them, the protocols in [6], [7], [8] are to construct fault-tolerant k-connected m-dominating set where every node is either in the set or has m neighbors in the set and the removal of $k-1$ nodes will not disconnect the induced graph of the set. The protocols in [9], [10] are to construct energy-efficient CDS where the remaining energy at each node is taken into consideration in the CDS construction process. The protocols in [11] construct a directional CDS in an arbitrary directed graph, where each node has different transmission range and equips a directional antenna. These types of protocols construct CDS with special properties and thus cannot be compared fairly with ours. The most related approaches to our work are [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], which are to construct a small CDS based on localized information. Depending on the nature of the approach, these CDS protocols in [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] 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's [12], Dai's [13], and Stojmenovic's [14], [15] protocols. These CDS protocols consist of two stages. In the first stage, each node collects 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. Wu's protocol [12] uses Rules 1 and 2, where a node is removed from the CDS, if all its neighbors are covered by one or two number of its direct neighbors. The node id is used to avoid simultaneous removal. Dai's [13] generalizes this as Rule k, in which coverage is defined by an arbitrary number of connected neighbors. Note that Dai's protocol is reduced to Wu's protocol when k is 1 or 2. Stojmenovic's protocol in [14] improves Wu and Dai's protocols by two observations. First, the number of neighbors (i.e., degree) is used to determine the priority of a node to be included in CDS instead of id. Second, nodes in CDS do not exchange their information. As a result, Stojmenovic's protocol incurs less control overhead. Enhanced Dominating Set introduced in [15] further reduces the size of CDS by applying two additional rules, the extended coverage condition and the key reversal techniques. The analysis and implementations presented in [25], [26] show that protocols with one-hop information results in a large CDS and using more than two hops incurs heavy communication overhead. Therefore, Wu, Dai, and Stojmenovic's protocol with two-hop information are practical. Since these 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. The most famous protocols in this category are Wan [16], [17], [18] and Li's [19] protocols. In general, these two protocols obtain the CDS by expending the Maximal Independent Set (MIS). Their protocols also have two stages. 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 among [16], [17], [18], [19] lies in how nodes in MIS are connected in the second stage (e.g., top-down or bottom-up fashion). In addition to MIS-based CDS protocols, the cluster-based CDS protocol [20], [21] can be considered as a different flavor of the addition-based CDS construction protocols. In this protocol, nodes with the minimum id among neighbors are selected as cluster heads. Between clusters, additional nodes are included to connect cluster heads. The cluster heads and the nodes that connect the cluster heads form a CDS. While the cluster-based CDS protocol is fully distributed, it does not provide a mechanism to deal with nodal mobility.

While most of these aforementioned protocols create CDS with localized information, they incur expensive communication overhead due to the need of two-hop localized information at each node. In addition, the CDS protocols capable of maintaining CDS under the change of network topology are Dai's [13] and Stojmenovic's [14], [15] protocols. All the other protocols [12], [16], [17], [18], [19],

[20] have to construct the CDS from scratch when the CDS becomes corrupted due to nodal mobility. In [27], it has been pointed out that the addition-based protocols generally produce smaller CDS than the substraction-based ones. Therefore, we are in a position to propose new addition-based CDS protocols that are lightweight, robust, and capable of maintaining CDS in a mobile environment.

3 Timer-Based CDS Protocol

3.1 The Basic Design

The idea of our proposed protocols is based on a simple greedy strategy: To obtain a smaller dominating set, a node with more neighbors should be included in the set. To reduce the communication overhead, we propose the uses of defer timers in our distributed CDS protocols. By appropriately setting the defer timer at each node, nodes with more neighbors will have higher probability to be included in the CDS. Generally speaking, our proposed Timer-Based CDS protocols consist of three phases: 1) initiator election; 2) tree construction; and 3) tree connection, as illustrated in Algorithm 1. In the initiator election phase, a number of initiators are elected distributively. This is achieved by letting the local minimum among α-hop neighbors to be initiators. In the tree construction phase, starting from each initiator, nodes utilize the defer timer to generate disjoint dominator trees. The defer timer is set inversely proportion to the number of uncovered neighbors in order to give higher priority to nodes with more uncovered nodes. In the tree connection phase, additional nodes are identified to connect the disjoint dominator trees. Each initiator collects the information of its neighboring trees from leaf nodes in its tree, and then send a control message to a border node to connect trees. The initiators, the dominator trees, as well as the nodes that connect the trees altogether form the CDS. Depending on how many initiators are elected, two different flavors of the timer-based CDS protocols, namely SI and MI, are presented in this paper.

Algorithm 1. Timer-Based CDS Protocol Skeleton

1: **Initiator Election**
2: /* Initiators are localized elected */
3: **Tree Construction**
4: /* Each initiator grows a dominator tree independently */
5: **Tree Connection**
6: /* Additional nodes are included to connect disjoint trees */

We assume each node to have a unique id value, such as its MAC address. In the protocols, a node is always in one of the four possible states, i.e., *uncovered*, *covered*, *dominator*, and *dominatee*. Similar to what is defined in IEEE 802.11 wireless LAN specification [28], each node periodically broadcasts a beacon to its neighbors every beacon period. We assume a node's beacon contains a number of fixed-length fields, including a type, the node's id, a *color* value, the node's current state, the node's initiator id, and the node's dominator id. The $announce$ in the subsequent sections is beacons with distinct type values. Before the protocol starts, each node sets its initiator id to $INIT_0$, its state to *uncovered*, and the color value to 0. The additional notations used in the subsequent discussion are summarized in Table 1.

TABLE 1
Definition of Notations

Symbol	Definition
$id(i)$	the id of node i
$initiator(i)$	the initiator of node i
$state(i)$	the current state of node i
$dominator(i)$	the dominator of node i
$color(i)$	the current color value of node i
$ITimer$	the countdown timer for initiator election
$DTimer$	the countdown timer that determines if a node is a dominator
$CTimer$	the countdown timer that keeps track of a node's dominator
$INIT_0$	a number larger than the id of any node
$Init_{Max}$	the maximal value for $ITimer$
T_d	the defer timer used for $DTimer$ in the tree construction phase
t_d	the value used for $CTimer$ to keep tracks of the presence of a node's dominator
T_{max}	the maximal value for $DTimer$
n_{uc}	the number of uncovered neighbors
β	the coefficient used in the defer timer
δ	the threshold for color value between neighbors
$H(i,j)$	the number of hops between node i and j
n	the number of nodes in the network
n_{init}	the average number of initiators in the network
S	the size of the field the network is deployed
$Diam$	the diameter of the network
r	the transmission range of a node
Δ	the average number of neighbors
$\overline{d}$	the average distance between two neighbors
α	the range in number of hops to elect a local minimum

3.2 The SI Timer-Based CDS Protocol

If the network size (i.e., network diameter) is known in advance, a unique initiator can be elected distributively in the initiator election phase within a predefined amount of time. Since only one initiator tree will be generated from the unique initiator at the end of the tree construction phase, the tree connection phase is not required. The subsequent sections elaborate the initiator election and tree construction phases of the SI CDS protocol.

3.2.1 SI Initiator Election Phase

In the initiator election phase, the id of each node is used as the metric to determine the initiator. When the $ITimer$ expires, if a node finds that its initiator id is $INIT_0$, it first sets its initiator to its own id. The reason why an initiator id is initialized by $INIT_0$ is to deal with asynchronous environment where nodes act autonomously (see Lemma 1). When a node receives a beacon, it compares its initiator id with the initiator id in the beacon. If a node finds that its initiator id is larger than the beacon's, it sets its initiator id to the initiator id in the beacon. A node can witch its $status$, if it does not receive beacon from nodes with smaller id than it during the period of twice of $Init_{Max}$. This duration is long enough, so that, at the end of the election phase, the smallest id among all nodes will be propagated to all nodes in the network.

Note that if a new node with the smaller id than any nodes in the network joins during the initiator election phase, it takes trice of $Init_{Max}$ (detail is shown in Lemma 1).

Algorithm 2 describes the SI initiator election phase. Note that in the pseudocode, the initiator will send out $announce$ beacon in every beacon period. The primary job of $announce$ beacon includes detecting initiator failure and a disruption of a tree. For detecting initiator failure, $announce$ beacons refresh the $ITimer$ of other nodes which expires after twice of the $Init_{Max}$ beacon periods. Because $announce$ beacons are originated from the initiator, if the initiator leaves the network or there is a partition in the network, the $ITimer$ of some node will expire. In this case, the node will wait twice of the $Init_{Max}$ beacon periods to make sure the $ITimer$ of every node in the network expires and restart the initiator election process. For detecting a disruption of a tree, each initiator keeps increasing $color$ value by one at every beacon period and each node in its tree updates color value accordingly. The disconnection caused by nodal movement interrupts the propagation of $announce$ message, which will eventually lead to large color differences. When a node receives a beacon signal from its neighbors indicating a large color difference, it concludes that there is a disconnection with its dominators.

Algorithm 2. The SI initiator election

/* node i in the network executes the following: */
INITIALIZATION:
  $initiator(i) \leftarrow INIT_0$
  $state(i) \leftarrow uncovered$
  $color(i) \leftarrow 0$
  $DTimer(i) \leftarrow -1$
  send out $announce$
  $ITimer(i) \leftarrow Init_{Max}$, start $ITimer(i)$

WHEN NODE i's $ITimer(i)$ EXPIRES:
if $initiator(i) = INIT_0$ **then**
  /* initial announcement */
  $initiator(i) \leftarrow i$
  send out $announce$
  $ITimer(i) \leftarrow 2 \times Init_{Max}$, start $ITimer(i)$
else if $initiator(i) = i$ **then**
  /* node i is an initiator */
  $color(i) \leftarrow color(i) + 1$
  send out $announce$
  $ITimer(i) \leftarrow Init_{Max}$, start $ITimer(i)$
else if $initiator(i) \neq i$ **then**
  /* reelect initiator */
  $state(i) \leftarrow uncovered$
  $initiator(i) \leftarrow INIT_0$
  $color(i) \leftarrow 0$
  $ITimer(i) \leftarrow 2 \times Init_{Max}$, start $ITimer(i)$
end if

ON RECEIVING $announce$ **FROM DOMINATOR** j**:**
if $initiator(i) > initiator(j)$ **then**
  $initiator(i) \leftarrow initiator(j)$
else if $initiator(i) = initiator(j)$ && $color(i) \neq color(j)$ **then**
  $color(i) \leftarrow color(j)$
  send out $announce$
  $ITimer(i) \leftarrow 2 \times Init_{Max}$, start $ITimer(i)$
end if

3.2.2 SI Tree Construction Phase

When $ITimer$ expires, a node moves to the tree construction phase. When a node finds that it is the initiator, it immediately switches its state to $dominator$ and its initiator to itself. When an uncovered node receives a beacon from a dominator neighbor for the first time, it sets its dominator to that neighbor, its initiator to that neighbor's initiator, switches its state to $covered$, and sets a defer timer T_d in inversely proportion to the number of its uncovered neighbors using (1).

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

As long as $\beta > 0$, (1) allows nodes with more uncovered neighbors be given a smaller timer value, hence their defer timer will expire earlier. When a node's defer timer expires, if the node still has uncovered neighbors, it switches its state to $dominator$ to cover them. Otherwise, it switches its state to $dominatee$. The number of uncovered nodes, denoted as n_{uc}, can be learned from the beacon of one-hop neighbors. Note that the scale of T_d is much larger than the beacon periods. Therefore, even if each node sends out its beacon asynchronously, our differ timer scheme guarantees that a node with more uncovered neighbors has shorter timer in most of cases.

At the end of the tree construction phase, all nodes will be in either $dominator$ or $dominatee$ state. The set of dominators forms a tree, which is referred to as the dominator tree. If there is only one initiator and the network is connected, the dominator tree will be the CDS for the entire network. We formalize the SI tree construction phase in Algorithm 3.

Algorithm 3. The SI tree construction

/* initiator i executes the following */
NODE i IS ELECTED AS THE INITIATOR:
  $state(i) \leftarrow dominator$
  $color(i) \leftarrow color(i) + 1$ each time before node i sends out beacon

/* node i executes the following when it receives a beacon */
ON RECEIVING A BEACON FROM NODE j:
if $(state(j) = dominator$ **then**
  if $(color(i) < color(j))$ **then**
    $color(i) \leftarrow color(j)$
  end if
  if $state(i) = uncovered$ **then**
    $state(i) \leftarrow covered$
    $dominator(i) = j$
    $CTimer(i) \leftarrow t_d$, start $CTimer(i)$
  end if
  if $((state(i) = covered) \| (state(i) = dominatee))$ **then**
    if $(dominator(i) = j)$ **then**
      $CTimer(i) \leftarrow t_d$, start $CTimer(i)$
    end if

```
21:       if ∃ i's neighbor k, |color(j) − color(k)| > δ then
22:          /* node i has a neighbor with large color
             difference */
23:          state(i) ← dominator
24:          dominator(i) ← j
25:       end if
26:    end if
27: end if
28:
29: /* node i executes before it sends out a beacon */
30: WHEN NODE i IS IN covered OR dominatee STATE:
31: if node i has no uncovered neighbor then
32:    state(i) ← dominatee
33: else if node i has at least one uncovered neighbor then
34:    compute T_d
35:    if DTimer(i) > T_d then
36:       DTimer(i) ← T_d, start DTimer(i)
37:    end if
38: end if
39:
40: WHEN NODE i IS IN dominator STATE:
41: if (node i has no covered neighbor) & & (node i has at
    least one dominator neighbor) then
42:    state(i) ← dominatee
43:    dominator(i) ← one of node i's dominator
44:    DTimer(i) ← −1, stop Dtimer(i)
45: end if
46:
47: /* node i executes when a countdown timer expires */
48: WHEN DTimer EXPIRES
49: if node i has uncovered neighbors then
50:    state(i) ← dominator
51: end if
52:
53: WHEN CTimer EXPIRES
54:    state(i) ← uncovered
```

In Algorithm 3, if a covered node i does not receive a beacon from i's dominator for t_d beacon periods, it implies that i's dominator has left and node i can then switch its state to *uncovered*. The assignment of t_d depends on the message error rate. For the 802.11 network, a small value such as 4 is sufficient [29]. $CTimer$ is used to track the dominator. A node updates $CTimer$, if it receives a beacon from its dominator and the dominator is still in *dominator* state. If $CTimer$ expires, it implies that its dominator leaves from the network.

3.2.3 Correctness of the SI Protocol

In this section, we prove that the SI CDS protocol generates a CDS for a given network topology as long as the topology remains connected and stable for a period of time (the time required to construct a CDS) and the value of $Init_{Max}$ is larger than the network diameter.

Lemma 1. *A unique initiator is elected within $3Init_{Max}$ in the asynchronous environment where nodes act autonomously, as long as $Init_{Max}$ is larger than the network diameter $Diam$ and the topology is stable for a period of time.*

Proof. Assume the network consists of several nodes. Among them, node i has the smallest id. At time t_1, some nodes in the network start the initiator election phase. Let node j joins the network at time t_2 where $t_1 < t_2$. In the case of $id(i) \leq id(j)$, each of i's neighbors will set its initiator to i after receiving a beacon from i. Their initiator value will stay as i because $id(i)$ is the smallest among all nodes in the network. In the next round of beacon exchanges, i's neighbors will propagate their initiator information further to reach nodes two hops away from i, which again will allow more nodes to set i as their initiator. This process is repeated until all nodes in the network set their initiator as i. At the end, node i is elected as the initiator as long as $id(i) < id(j)$ for any additional new node j in $Init_{Max} + \max_{\forall k} H(i,k) \leq Init_{Max} + Diam \leq 2Init_{Max}$. Thus, the above claim is true. In the case of $id(i) > id(j)$, depending on the value of t_2, we further break it into the following two subcases.

If $t_2 \geq t_1 + H(i,j)$, when node j receives *announce* from node i at the time $t_1 + Init_{Max} + H(i,j)$, $initiator(j)$ is still $INIT_0$ because $ITimer(j)$ has not yet expired. According to the protocol, $id(i) < INIT_0$ so node j set its initiator id to be i's. In other words, all nodes will set their initiator id to i's within $Init_{Max} + \max_{\forall k} H(i,k) < 2Init_{Max}$. Thus, the above claim is true.

If $t_2 < t_1 + H(i,j)$, the $ITimer$ of node j will expire before it receives *announce* from node i. In this case, node j will send out its own *announce* message to its neighbors. The nodes that received node i's *announce* in the past override their initiator id to be j's as $id(i) > id(j)$. Node j will be elected as the initiator in $t_2 - t_1 + Init_{max} + \max_{\forall k} H(i,k) < 3Init_{Max}$. Therefore, the above claim is true. □

Lemma 2. *A new initiator will be selected by the SI initiator election phase within bounded beacon periods if the original initiator leaves the network assuming the network remains connected and stable.*

Proof. From the pseudocode of the SI initiator election phase, we can see that the initiator will send out *announce* messages every $Init_{Max}$ beacon periods. Other nodes only forward *announce* messages if its initiator id is bigger than the initiator's in the message or the color value is different. If the initiator is active, its color value will change every $Init_{Max}$ beacon periods, the refresh *announce* will keep the $ITimer$ of other nodes from expiring.

When the original initiator leaves the network, the $ITimer$ of all nodes in the network will expire since no more refresh *announce* message is coming. After $ITimer$ expires, each node sets all the information back to the initial value and the initiator election phase will start over again. According to Lemma 1, a unique initiator will once again be selected. □

In the following theorem, we prove that our SI protocol successfully obtains the connected dominating set for the given network topology:

Theorem 3. *If the network topology remains connected and stable for a period of time, the collection of all dominator nodes resulted from the SI protocol forms a connected dominating set for the entire network.*

Proof. We claim that if the network topology is connected and unchanged, eventually all the dominator nodes will form a dominating set and the induced graph is connected.

We prove by induction on number of nodes n.

Induction base. When $n = 1$, the only node is the initiator. The initiator enters dominator after the initiator election phase, which is a connected dominating set. The above claim is true.

Induction hypothesis. Assume when $n = k - 1$, the CDS generated by SI protocol covers all $k - 1$ nodes. Let us denote the graph with $k - 1$ nodes as $G(k - 1)$.

Induction step. Now one additional node (node k) joins the network. Let us denote the resulting network as $G(k)$. If one of node k's neighbors is a dominator, the CDS covering $G(k - 1)$ will cover $G(k)$ and is still a CDS.

If none of k's neighbors is a dominator, all neighbors of node k must be covered by some dominators in the CDS of $G(k - 1)$. According to the SI protocol, all node k's neighbors will go through the *covered* state and set up the defer timer appropriately. When the first one of them has its defer timer expires, that neighbor will enters the *dominator* state to cover node k. After that, there will be no more uncovered node and therefore we have a dominating set. Since the new dominator node enters the *dominator* state from the *covered* state, it must have a dominator neighbor. (A node enters covered state only after it receives a beacon from a dominator neighbor.) In other words, the nodes in the dominating set, after including the additional dominator node, are still connected. The CDS covering $G(k - 1)$ plus the new dominator node will be the CDS covering $G(k)$. □

If a node cannot correctly obtain n_{uc} due to the collisions, the size of CDS increases as nodes with fewer uncovered nodes set shorter timer. However, this never affects the correctness of SI protocol. Thus, SI protocol successfully obtains a CDS, as long as each node has bidirectional links between its neighbors.

3.2.4 SI Mobility Handling

There are many ways the network topology can change due to nodal mobility. However, regardless how the topology changes, there are only two different basic types: A node leaves the network and a new node joins the network. Considering the role of the leaving node when CDS is present in the network, we can further categorize them into the following four primitive cases:

1. A dominatee node leaves the network.
2. A dominator node leaves the network.
3. The initiator leaves the network.
4. A new node joins the network after the construction of the CDS.

In this section, we demonstrate that the SI protocol can adapt to nodal mobility by showing that the SI protocol successfully maintains the CDS under changes of network topology, which are composed of these primitive cases.

For Case 1, the departure of a dominatee will have no impact to other nodes and the original CDS should remain functional.

Case 2 is handled by the color scheme. From the SI tree construction pseudocode, we can see that a node changes its color only after receiving a larger color value from a dominator neighbor. Since the initiator keeps increasing its color value, if the CDS is intact, the color difference of neighbor nodes should be very small. If the CDS is broken into disconnected segments, the segment that contains the initiator will keep increasing its color value while other segments will have the color value unchanged. When a node sees two neighbors with large color difference, it concludes that it is a border node between the disconnected segments and enters the *dominator* state. This process will continue until a CDS is reconstructed. The threshold value of color difference (i.e., δ) is decided by the moving speed of nodes and the diameter of the network. It should be large enough to distinguish node movement from network disconnection.

To handle Case 3, all nodes in the network first have to detect that the initiator leaves the network, and then all nodes agree with a new initiator. We have already showed that SI can successfully elect a new initiator if the original initiator leaves the networks in Lemma 2.

Case 4 is exactly the situation in Theorem 4. When a new node, say node i, joins the network, it is in *uncovered* state. If node i has a dominator node, it is already covered. Otherwise, the neighbors of node i will switch their *state* to *covered*, and start $DTimer$. Hence, the node with the shortest timer enters *dominator* state and covers the new node i. Only the one-hop neighbors of a new node are involved in this process. No matter how many nodes simultaneously join the network, SI can handle Case 2 in a similar manner. Assume K number of nodes, say node i_1, $i_2, \ldots$, and $i_k (k \leq K)$, join the network at the same time. If node i_k has a dominator neighbor, it is covered. The neighbors of node $i_k (1 \leq k \leq K)$ switch their *state* to *covered* and start their $DTimer$. According to Algorithm 3, when $DTimer$ expires, a node becomes *dominator* if it has at least one uncovered node. Therefore, all new nodes $i_k (1 \leq k \leq K)$ will be covered and finally become *dominatee*.

Note that any change of network topology caused by nodal mobility after the CDS is constructed can be considered as the combination of a sequence of these four primitive cases. In addition to these cases, our SI can also optimize the CDS in case the following situation occurs:

- A redundant dominator node switches to the *dominatee* state while still keeping the dominating set connected.

This situation can be handled by the following way. From the SI tree construction pseudocode, we can see that if a dominator does not cover any neighbor and has at least one *dominator* neighbor, it will switch its state to *dominatee* and set its dominator field as the id of the dominator neighbor. Note that each node indicates its *dominator id* in the beacon frame, and thus a dominator node knows which dominatee neighbors it is covering. By doing so, the total number of *dominator* nodes is reduced and we have properly maintained the CDS. This situation could happen after a series of node movements.

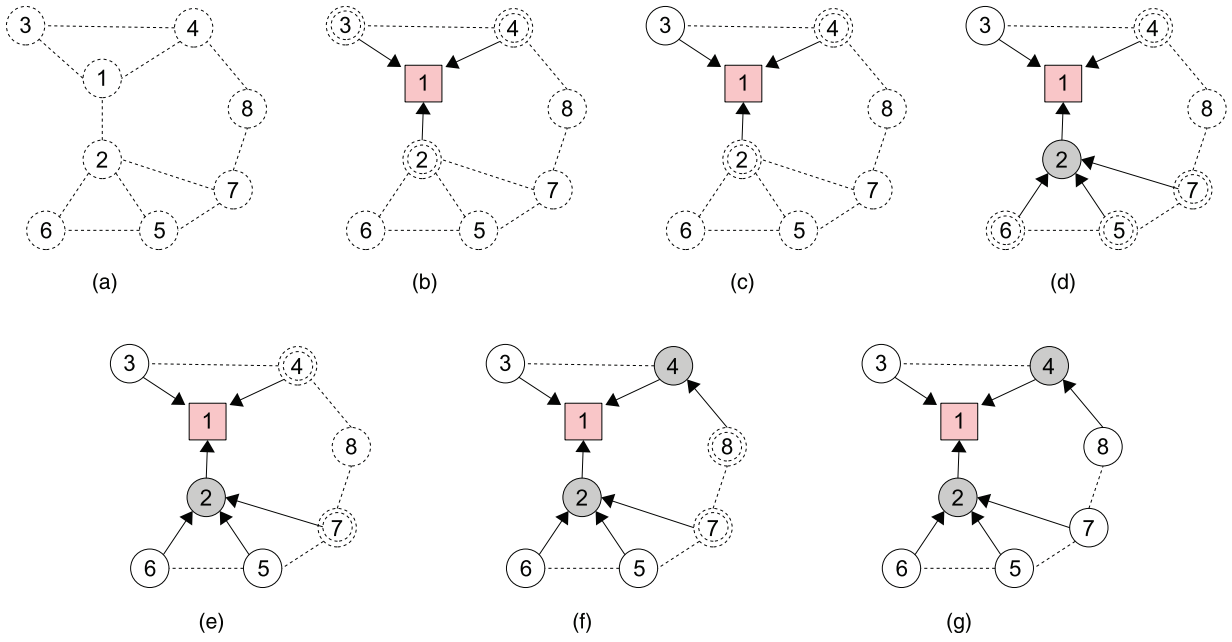

Fig. 1. An example of SI protocol.

3.2.5 Example of SI Protocol

Fig. 1a shows the snapshot of a network topology. In Fig. 1a, a dotted circle represents a node in *uncovered* state and a dotted line represents a link between two nodes. Fig. 1b shows the result of the initiator election phase. According to SI protocol, node 1 is elected as the initiator, which is denoted by a shaded square. The neighbors of node 1 that are switched to the *covered* state are denoted by a double dotted circle. In the next beacon period, node 3 finds that it has no uncovered neighbor and then switches to the *dominatee* state, which is denoted by a circle in Fig. 1c. At the same time, nodes 2 and 4 set their $DTimer$ and start counting down the timer. Since node 2 has more uncovered neighbors, it has a shorter timer and its timer will expire before node 4's. As shown in Fig. 1d, node 2 will switch to the *dominator* state, which is denoted by a gray circle. After that, nodes 5 and 6 do not have uncovered neighbors and will then switch to the *dominatee* state. On the other hand, node 7 will start its own $DTimer$, as shown in Fig. 1e. While nodes 4 and 7 have the same number of uncovered neighbors, the timer of node 4 expires sooner as it started its $DTimer$ earlier. When the timer of node 4 expires, node 4 will switch to the *dominator* state to cover node 8, as shown in Fig. 1f. Afterwards, nodes 7 and 8 find that they do not have uncovered neighbors and switch to the *dominatee* state, which is shown in Fig. 1g. At the end, the collection of the initiator and dominators form a CDS for the whole network.

3.3 The MI Timer-Based CDS Protocol

As discussed in Section 3.2, the SI protocol is able to maintain CDS in the presence of changes of network topology. However, SI requires the knowledge of the diameter of the network to properly set up the $Init_{Max}$. Although the network diameter may be estimated by the size of the region the network is deployed and the transmission range of a node, the estimation is not accurate. In addition, while SI is able to recover the CDS locally for most CDS breakdowns, the whole CDS will still have to be reconstructed from scratch should the unique initiator fail. A natural approach to resolve these issues is to elect multiple initiators. Each initiator generates a tree in exactly the same manner SI protocol does. The tree construction phase completes when all nodes are covered by dominators. This will produce several small disjoint trees and the union of these trees will form a dominating set. To obtain a CDS, we can simply include a few more nodes to connect these dominator trees. This is the basic idea of our Multi-Initiator CDS protocol. If the CDS is constructed this way, the failure of an initiator will only affects the corresponding dominator tree. The other part of the CDS will remain intact. In the following sections, we explain how the initiator election phase and tree connection phase of MI are accomplished effectively and distributively in detail. Note that the tree construction phase of MI is omitted because it is basically the same as that of SI described in Section 3.2.2.

3.3.1 MI Initiator Election Phase

The most intuitive idea to elect multiple initiators effectively and distributively is to let the local minima be the initiators. By listening to beacons, a node obtains the ids of its one-hop neighbors without introducing additional messages. If a node finds that its id is the smallest among its one-hop neighbors, it sets itself as an initiator.

The problem with this approach is that it may produce too many initiators. Given the average number of neighbors to be Δ, each node has the opportunity to be an initiator with the probability $\frac{1}{\Delta+1}$. Let n be the number of nodes uniformly distributed in a region of size S, and r be the transmission radius of each node, the average number of initiators n_{init} can be obtained by (2).

$$n_{init} = n \cdot \left(\frac{1}{\Delta + 1}\right) \approx \frac{S}{\pi r^2}. \quad (2)$$

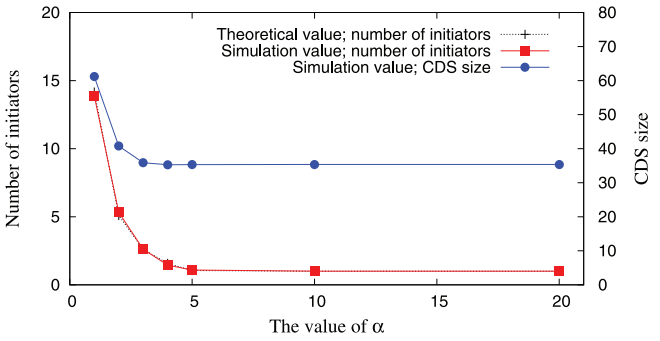

Fig. 2. The number of initiators and CDS size.

For instance, a network of 150 nodes with $150\,\mathrm{m}$ transmission radius deployed in $1{,}000 \times 1{,}000\,\mathrm{m}^2$ area will have an average of 14 initiators. Too many initiators will consequently lead to a large CDS due to two reasons. First, all initiators will eventually be included as part of the CDS. Second, after each initiator generates a dominator tree, additional nodes will be added to connect these trees. More initiators means more trees, and thus requires more nodes to connect them.

To reduce the number of initiators, the local minimum among α-hop neighbors can be elected as the initiators. As an example, to elect the local minimum among two-hop neighbors, each node can further encode the minimal id among its one-hop neighbors in its beacon. A node becomes an initiator only when its id equals to the minimal id of all its one-hop neighbors. With few rounds of beacon exchanges, the local minimum among two-hop neighbors will become the initiator. In this approach, no extra message is needed and the time to elect initiators is much shorter than the time required by SI protocol in the initiator election phase. Using the same notations, the average distance between two neighbors, $\overline{d}$, is obtained by (3).

$$\overline{d} = \int_0^r \frac{2\pi x \cdot x}{\pi r^2} dx = \frac{2}{3} r. \tag{3}$$

The average number of two-hop neighbors is $\frac{n\pi(r+\overline{d})^2}{S}$. Therefore, the average number of initiators n_{init} can be obtained by (4).

$$n_{init} = n \cdot \left(\frac{1}{\frac{n\pi(r+\overline{d})^2}{S} + 1} \right) \approx \frac{9}{25} \cdot \frac{S}{\pi r^2}. \tag{4}$$

For instance, the network of 150 nodes with $150\,\mathrm{m}$ transmission radius deployed in $1{,}000 \times 1{,}000\,\mathrm{m}^2$ area now has an average number of five initiators. This significantly reduces the possibility of a large CDS.

The multihop local minimum idea can be extended to α-hop. Since the average number of α-hop neighbors is $\frac{n\pi(r+(\alpha-1)\cdot\overline{d})^2}{S}$, the expected number of local minimum can be estimated by (5). Note that when α is equal or larger than the network diameter, only one initiator will be elected in the network and the protocol is reduced to SI.

$$n_{init} = max\left\{ n \cdot \left(\frac{1}{\frac{n\pi(r+(\alpha-1)\cdot\overline{d})^2}{S} + 1} \right), 1 \right\}. \tag{5}$$

Fig. 2 shows the number of initiators and CDS size with respect to the value of α in the network of 150 nodes with $150\,\mathrm{m}$ transmission radius deployed in $1{,}000 \times 1{,}000\,\mathrm{m}^2$ area. As illustrated in Fig. 2, the number of initiators as well as the size of CDS decrease as the value of α increases. In the case $\alpha = 2$, the size of CDS is competitive and each dominator tree is small and easy to maintain. Therefore, in the rest of the paper, the α value of the MI protocol is set to 2.

3.3.2 MI Tree Connection Phase

In this phase, additional nodes are included to connect neighboring dominator trees. If a node has a neighbor belonging to a different initiator, it is referred to as a border node. To connect the dominator trees distributively, the most intuitive idea is to have all the border nodes turn into dominators. Although this approach does not introduce extra messages, it will create a very large CDS as there are possibly many border nodes between each pair of neighboring dominator trees. To limit the size of CDS, it is better that the root of a tree (i.e., initiator) determines what border nodes are utilized to connect the neighboring trees.

Since an initiator does not know what neighboring trees it has with only the localized information, extra messages have to be introduced so that the initiator can collect such information from its border nodes. After the defer timer expires, if a node finds that it is a border node, it sends a message to its initiator which includes the initiator's id of the neighboring tree. Fig. 3a depicts a snapshot of the network after the tree construction phase completes. For example, border node 3 belonging to the tree rooted at the initiator 1 finds that its neighbor 8 belongs to the tree rooted at the initiator 2 from node 8's beacon. Node 3 then sends a message to node 5 containing the id of the initiator 2.

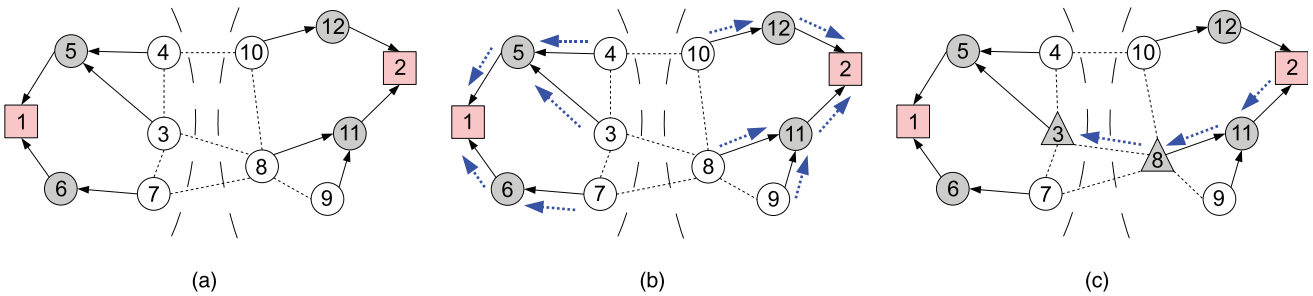

Fig. 3. An example of MI protocol.

Similarly, nodes 4, 7, 10, and 8 send a message to their initiator containing the initiator id of their neighboring tree.

At the first glance, this process seems to introduce many messages. However, when a dominator receives messages from the neighbors it covers about the neighboring trees, it only forwards one copy of the messages if the initiator id contained in the messages are the same. For instance, in Fig. 3b, when node 5 receives messages from nodes 3 and 4 about the same neighboring dominator tree, it only forwards one copy of the message to its dominator node 1. By doing so, the number of messages can be controlled to $O(n \cdot m)$ where n is the number of nodes in the network and m is the number of neighboring dominator trees.

When an initiator learns about its neighboring trees, it can then instruct only border nodes on a particular path to each of its neighboring trees to switch the state to dominator and connect to its neighboring trees. The border nodes that are used to connect the trees are referred to as the bridge nodes. Any metric can be used to elect bridge nodes among border nodes. In this paper, the node with the smallest id is elected as bridge nodes. For example, in Fig. 3c, initiator 2 elects node 8 as a bridge because it has the smallest id among border nodes which connects the neighboring tree 1. Our CDS consists of the dominator nodes in the dominator trees and the bridge nodes that connect the trees.

If each initiator tries to connect to each of its neighboring dominator trees, it is likely that there will be two paths between each pair of neighboring dominating trees. In the worst case, at most four border nodes (two for each path) will become dominators. While having two paths between neighboring dominator trees may improve the degree of fault tolerance and system throughput, it will create a larger CDS. To limit the size of CDS, an initiator makes a connection to a neighboring dominator tree only when the id of the initiator of the neighboring tree is smaller than its own. This roughly reduces the number of the bridge nodes by half.

3.3.3 Correctness of MI Protocol

Since MI is built on top of SI, most of the theorems in Section 3.2.3 directly apply to MI. The only additional issue is whether or not the tree connection phase can successfully connect the dominator trees.

Theorem 4. *If the network topology remains connected and stable for a period of time, the collection of all dominator nodes resulted from the MI protocol forms a connected dominating set for the entire network.*

Proof. After the initiator election and tree construction phases, a set of the disjoint dominator trees will be created. The nodes in these dominator trees together form a dominating set. The neighboring dominator trees will be at most three hops away. It is because if two neighboring dominator trees are more than three hops away, one of the nodes will not be covered by any dominator. For example, in Fig. 4, the neighboring trees are distanced by four hops. This will leave node 5 to be $uncovered$. However, this situation should not happen because according to the tree construction protocol, when the $DTimer$ of nodes 4 and 6 expire, they will find node 5 as an uncovered neighbor and therefore one of them will switch to dominator to cover node 5. In other words, there will be at most two adjacent covered nodes and each of them has a different initiator. These two border nodes will report to their initiators and, according to MI, the initiator with larger id will then initiate the connection to the neighboring tree, providing that the network topology remains connected and stable for a period of time. □

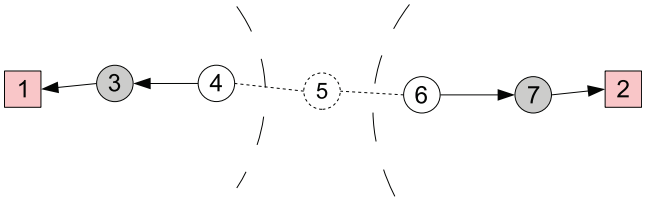

Fig. 4. An impossible case at the end of MI tree construction phase.

3.3.4 MI Mobility Handling

In MI, each dominator tree is maintained in the same manner as SI. This allows the MI protocol to take advantage of the mobility handling capability of SI inside each of the dominator trees. The root of a dominator tree periodically broadcasts the $announce$ message so that any topology changes due to mobility can be captured and handled in a timely fashion. Therefore, MI can naturally handle the four different mobility cases mentioned in Section 3.2.4. However, since MI will create multiple dominator trees, there are additional mobility cases that involve more than one tree. Notice that most of these new cases can be considered as the combinations of the aforementioned four cases. For instance, if a dominator moves from one tree to another, it can be seen as Case 2 for the first tree plus Case 4 for the second tree. Consequently, most of these new cases can be handled and resolved properly and locally with no change to the protocol. The only new case that needs to be addressed is when a bridge node leaves its dominator tree.

Given a bridge node x, assume that it belongs to a dominator tree with root a, and x is used by a to connect to a neighboring dominator tree with the root b. Clearly, both a and b are initiators. If x leaves the tree, the dominator that covers x, say node p, will find out after a couple of beacon periods. In this case, p will first try to fix the problem locally by querying its neighbors if any of them has a neighbor belonging to the initiator b. If p receives a positive response from some of its border neighbors, it instructs one of them to switch to dominator state, i.e., act as the new bridge between two trees. If p does not hear back from its neighbors for a period of time, it sends a message to the initiator a, which will send a treewide query down to all its border nodes looking for a possible connection to the neighboring dominator tree rooted at b. The responses sent back from the border nodes are handled similarly to the messages in the tree connection phase. Afterwards, if a receives some responses from the border nodes with neighbors belonging to initiator b, it instructs the border nodes on one of the paths to turn to dominators (i.e., become bridge nodes) to connect two trees. If a does not receive any response, that implies that the dominator tree rooted at b is no longer a neighboring dominator tree for a. In this case, nothing needs to be done by node a.

TABLE 2
Simulation Parameters

Parameters	Value
Simulation area	$1000m \times 1000m$
Number of nodes	100 to 450
Transmission range	$150m$
1 beacon period	1 second
T_{max}	100 beacon periods
β	1
$Init_{Max}$	20 beacon periods
α	2
Mobility model	WWP
Percentage of mobile nodes	20% to 80%
Node speed	up to $5m/s$
Number of simulations	1000

TABLE 3
Parameters for ns-2

Parameters in ns-2	Value
Propagation	Two-Ray Ground
MAC protocol	IEEE 802.11
Data rate	2 Mbps
Traffic	CBR
Number of flows	5 concurrent flows
Number of packet	5 packets/flow
Inter-arrival time	0.25 second
Packet size	128-byte
Percentage of mobile nodes	50% or 100%
Number of simulations	100

It may seem odd that the dominating set will remain connected if nothing is done after the departure of a bridge node makes two neighboring dominator trees no longer neighbors to each other. If we regard each dominator tree as a big cluster, what the tree connection phase is doing is to create an edge between each neighboring clusters. If two clusters (trees) are no longer neighbors to each other, as long as the whole network remains connected, they should be connected via the other clusters (trees). If the departure of a bridge node partitions the network into components, it will be impossible to create a CDS for the network. However, the MI CDS protocol can still maintain a CDS inside each of the connected components and recover back to a single CDS when the components reconnect to each other. This feature is especially important for mobile ad hoc networks.

3.3.5 Differences between MI and Cluster-Based CDS Protocol

When $\alpha = 1$, it may seem that MI behaves similar to the cluster-based CDS protocol in [20], [21]. Indeed, when $\alpha = 1$, according to MI the nodes with minimum id among neighbors will be elected as initiators, so we could consider initiators as cluster heads and trees as clusters. However, there are still several major differences between these two protocols. First, MI is timer-based. The use of timers allows MI to easily accommodate nodal mobility. Second, in MI, each node belongs to only one initiator. Last, unlike the protocol in [20], there exist two bridge nodes to connect a pair of neighboring trees/clusters.

4 Simulation Results

To evaluate the performance of SI and MI, we implement our protocol in C++ and ns-2 [30] along with the other CDS protocols, including Dai's with two-hop neighboring information [13], Stojmenovic's with two-hop neighboring information [14], and Wan's [18]. In this section, the simulation results of different CDS protocols are reported and analyzed.

4.1 Simulation Configurations

The network topology is randomly generated by placing nodes in a $1{,}000\,\mathrm{m}$ by $1{,}000\,\mathrm{m}$ square field according to uniform distribution. If the generated network is partitioned, it is discarded and a new network topology is generated to ensure the connectivity of the whole network. The transmission range of a node is set to be $150\,\mathrm{m}$. Two different network scenarios, static networks and mobile networks, are considered and simulated to evaluate the performance of CDS. For a given simulation configuration, 1,000 different network topologies are generated. The parameters used for simulations and ns-2 configuration are summarized in Tables 2 and 3, respectively.

Static networks. In this scenario, the average node 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 node density ranging from approximately 5 to 30 neighbors per nodes. For SI and MI, the T_{max} and β in the tree construction phase are set to be 100 beacon periods and 1. According to [29], the value of $Init_{Max}$ for SI is set to be 20 to accommodate beacon collisions. On the other hand, α for MI is set to be 2, i.e., two-hop local minimum will be elected as initiators.

Mobile networks. In this scenario, the average node density is set to be 10 neighbors per node, and some nodes are assumed to be mobile. The percentage of the mobile nodes ranges from 20 to 80 percent with speed up to $5\,\mathrm{m/s}$. The Weighed Way Point (WWP) [31] 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 the University of Southern California's campus [32]. For Dai's, Stojmenovic's, SI and MI, the corrupted CDS is recovered according to their mobility handling procedures when the topology changes. Each simulation lasts 1,000 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. In the limited mobility environment, 50 percent nodes are mobile with $5\,\mathrm{m/s}$, and in the high-mobility environment, all nodes are mobile. In AODV with MI, only nodes in a CDS forward a route request (RREQ) packet. The propagation model used in ns-2 routing simulations is the two-ray ground model. IEEE 802.11 is used as the MAC layer protocol, and the data rate is 2 Mbps. For each network realization, five pairs of source and destination are randomly selected. A new pair of source and destination will be selected if they are not connected. Each source node generates constant bit rate (CBR) traffic flows to its destination simultaneously. Each CBR flow sends five consecutive packets of 128 bytes. The interarrival time of packets is set to be 0.25 seconds.

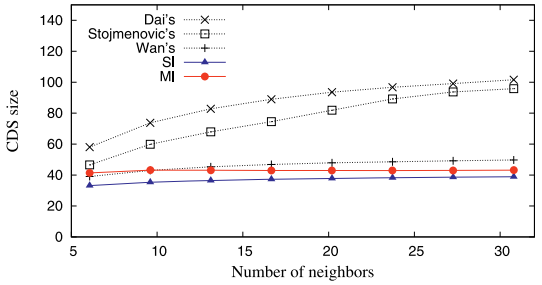

Fig. 5. CDS size.

To assess the performance of different CDS protocols, five metrics are used, including the size of CDS, the number of extra messages, the average traffic, the convergence time, and the percentage of time CDS is alive. To assess the performance of routing with CDS protocols, three metrics are used, including delivery rate, end-to-end delay, and the number of RREQ packets. For MI, the messages in the tree connection phase and query/response messages in the mobility handling are counted as extra messages. For SI, all the information exchanged between nodes are done by beacons. For Dai's and Stojmenovic's protocols, beacons are considered as extra messages since the size of the beacon increases in proportion to the node 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*, *state*, *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. The beacon of Dai's and Stojmenovic's protocols include node *id*, *state*, *marker*, and the list of *id*s for one-hop neighbors. For Wan's protocol, dominator *id* and *color* are added to the beacon. The extra bit in the beacon and the size of extra messages for each protocol are counted toward the traffic (in bps) for CDS construction. The period of time for CDS protocols to complete is defined as the convergence time in the number of beacon intervals. For SI, the initiator election is assumed to be completed in 20 rounds of beacon intervals. For MI, the initiator election is done in two rounds of beacon intervals. For the mobile network scenario, the total amount of time CDS is valid divided by the total simulation period is defined as the percentage of time CDS is alive.

4.2 Simulation Results for Static Network Scenario

In this section, the simulation results of different CDS protocols in static network scenario are presented.

Fig. 5 shows CDS size of different protocols with respect to the node density. It is clear that SI consistently generates the smallest CDS and Dai's consistently generates the largest CDS among all the protocols. Stojmenovic's protocol creates a smaller CDS than that of Dai's, but compared with SI and MI it generates a larger CDS. While it is proven that the size of Wan's CDS is suboptimal [16], the size of the CDS of MI is smaller than Wan's and is very close (within few nodes) to SI. In addition, the CDS size in SI and MI remains constant when the node density increases. This suggests that SI and MI are scalable.

Fig. 6 demonstrates the number of extra messages with respect to the node density. As illustrated in Fig. 6, MI

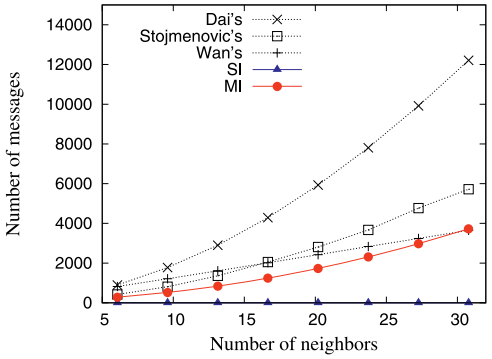

Fig. 6. Number of messages.

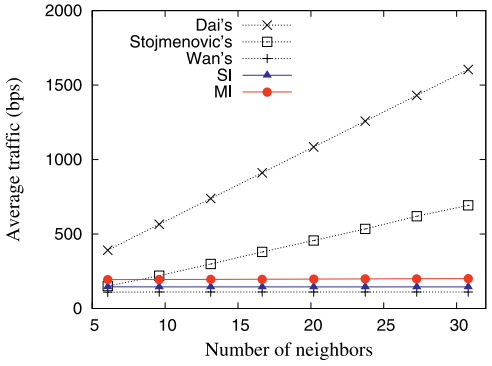

Fig. 7. Average traffic (kbps).

introduces only 30 percent of the number of extra messages introduced by Dai's. Compared with Wan's, MI reduces the number of extra messages up to 60 percent when the node density ranges from 5 to 15 neighbors per node. While Stojmenovic's reduces the number of messages from Dai's implementation, it still introduces more messages than MI does. As discussed in Section 3.2, SI does not introduce any extra message.

Fig. 7 shows the average traffic required for CDS construction with respect to the node density. Obviously, Dai's protocol produces the largest traffic, and the traffic required by Dai's and Stojmenovic's protocols increase in proportion to the node density. This is because of the inclusion of the neighboring list in the beacon frame. For the other three protocols, the average traffic is almost the same. It is interesting that for Wan's and MI protocols, even the number of messages increases in proportion to the number of neighbors as shown in Fig. 6, the traffic remains relatively constant to the node density. This implies that the traffic is mostly dominated by the extra bits in the beacon.

Fig. 8 presents the convergence time with respect to the node density. As shown in Fig. 8, the convergence time of SI and MI decreases quickly as the node density increases. This indicates that the convergence time of SI and MI is dominated by the time of tree construction, in which a node in dense networks is likely to have more uncovered neighbors and will have a smaller defer timer. MI takes longer time to form CDS than SI. This is due to the additional time for the tree connection phase.

4.3 Simulation Results for Mobile Network Scenario

In this section, the simulation results of different CDS protocols in the mobile network scenario are presented.

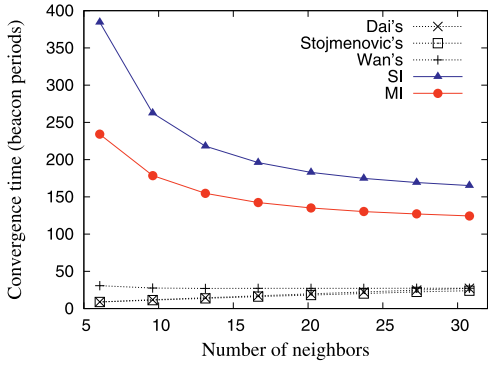

Fig. 8. Convergence time.

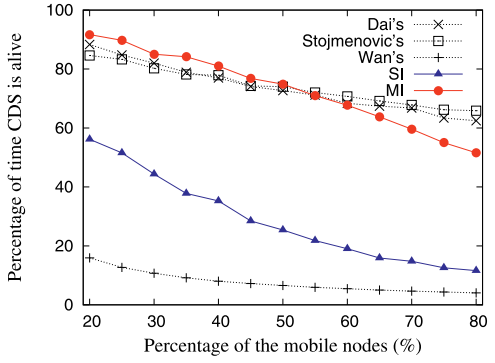

Fig. 9. Percentage of time CDS is alive.

Fig. 9 illustrates the percentage of time that CDS is alive with respect to the percentage of mobile nodes. As can be seen in Fig. 9, MI has the highest percentage of CDS alive time than other CDS protocols except when the percentage of mobile nodes is more than 60 percent. Although smaller CDS is generally more vulnerable to topology changes, MI shows excellent mobility adaptation compared with the other protocols. Only when the percentage of mobile nodes is greater than 65 percent does MI's *percertage of time CDS is alive* become slightly lower than that of Dai's and Stojmenovic's due to MIs smaller CDS size. As pointed out in [16], the time complexity of mobility recovery at each node of Dai's is as high as $O(\Delta^2)$. While Stojmenovic's protocol requires $O(\Delta)$ to maintain a CDS, it is vulnerable to nodal mobility as it further reduces the size of CDS than Dai's protocol. In addition, Dai's and Stojmenovic's protocols require two-hop information, in which if one of neighbors changes its state, a node has to wait two beacon periods to initiate their protocol. Thus, mobility handling procedures used in Dai's and Stojmenovic's protocols are not as efficient as that of MI. SI has smaller percentage of CDS alive time than MI, Dai's, and Stojmenovic's. This is because of the possible failure of the unique initiator, which results in the reconstruction of the whole CDS from scratch.

Fig. 10 shows the average CDS size with respect to the percentage of the mobile nodes. As illustrated in Fig. 10, SI consistently produces the smallest CDS and Dai's protocol consistently produces the largest CDS. The size of CDS by Stojmenovic's protocol is larger than MI. The average CDS size of MI is 5 to 35 percent higher than Wan's. However, as can be seen in Fig. 9, Wan's protocol is vulnerable to the nodal mobility.

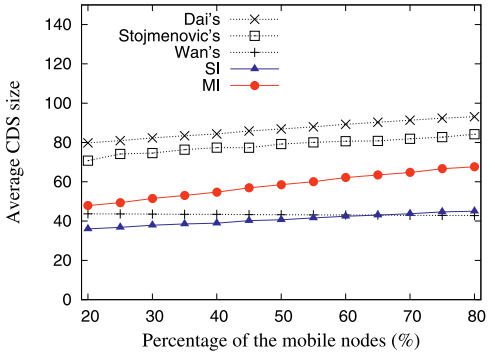

Fig. 10. Average CDS size.

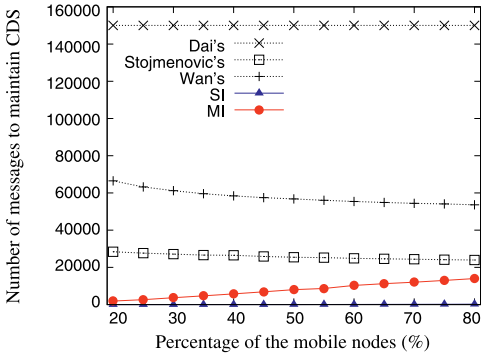

Fig. 11. Number of Messages to Maintain CDS.

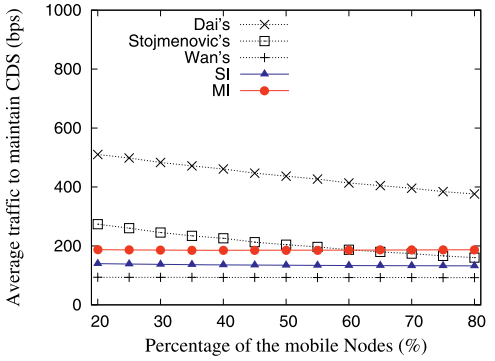

Fig. 12. Average traffic to maintain CDS.

Fig. 11 presents the number of extra messages to maintain CDS with respect to the percentage of the mobile nodes. As illustrated in Fig. 11, MI requires a low number of extra messages to maintain CDS. The number of messages is only 60 percent of that of Stojmenovic's, 20 percent of that of Wan's, and 8 percent of Dai's. This demonstrates that MI can handle topology changes efficiently with only a small number of messages.

Fig. 12 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 Fig. 12, the average traffic of MI and SI are at least 50 percent lower than that of Dai's. Compared with Wan's and SI, MI has slightly higher traffic, but the difference is not significant. This is because the beacon in MI is slightly larger than that of Wan's and SI. As we have pointed out in Section 4.2, the traffic is mostly dominated by the extra bits in the beacon frame rather than the extra messages. This again is validated by Figs. 11 and 12.

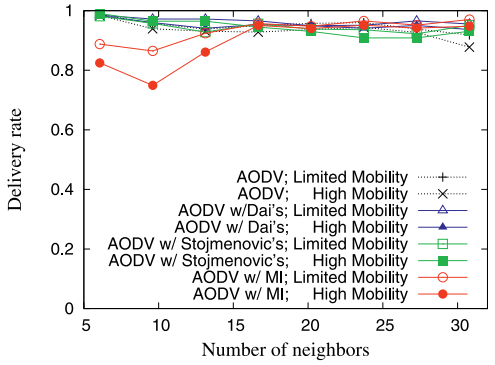

Fig. 13. Delivery rate.

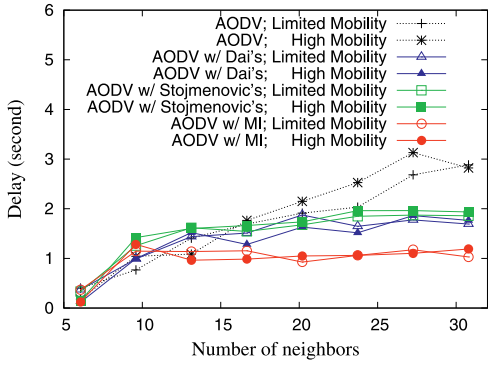

Fig. 14. End-to-end delay.

Fig. 13 depicts the packet delivery rate with respect to the node density. In the case of sparse networks, AODV with MI results in lower delivery rate than the original AODV, AODV with Dai's, and AODV with Stojmenovic's. This is because in such low node density the network is often not connected, which leads to the failure of route discovery. However, when the average number of neighbors is more than 15, AODV with MI results in higher delivery rate than AODV, and has the similar delivery rate with AODV with Dai's and AODV with Stojmenovic's.

Fig. 14 presents the end-to-end delay with respect to the node density. As can be seen in Fig. 14, the delay of AODV increases in proportion to the number of neighbors, while that of AODV with MI remains stable. Compared with AODV with Dai's and AODV with Stojmenovic's, the delay of AODV with MI is only 50 percent. It implies that in networks with high-density AODV, and AODV with Dai's and AODV with Stojmenovic's incur more collisions and take longer time to discover a route.

Fig. 15 shows the number of RREQ packets with respect to the node density. It is clearly shown that AODV with MI introduces lower number of RREQ packets. This is because that only nodes in the CDS forward RREQ packets. On the other hand, AODV with Dai's and AODV with Stojmenovic's incur more number of RREQ packets than AODV with MI due to lower disconnections of a CDS. Considering the results presented in Figs. 13, 14, and 15, we are confident that CDS constructed by our proposed protocols improve routing performance and reduce routing overhead in mobile ad hoc networks.

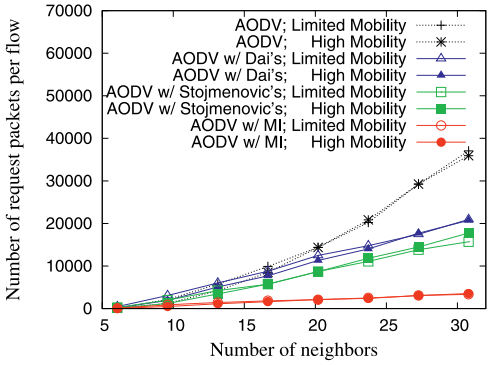

Fig. 15. Number of RREQ packets.

5 Analytical Model

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. Our analytical models estimate the average performance when nodes are randomly placed in a network in the uniform distribution.

5.1 Analytical Model

The initiator election phase of SI and MI completes in $2Init_{Max}$ and α beacon periods, respectively. We first consider the convergence time of SI.

The convergence time of SI in the tree construction phase is 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 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 (6).

$$h = \frac{\sqrt{l_x^2 + l_y^2}}{2\overline{d}} = \frac{3\sqrt{2S}}{4r}. \tag{6}$$

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 circles of radius r with their centers separated by d.

$$INTC(r,d) = 4\int_{d/2}^{r} \sqrt{r^2 - x^2}dx. \tag{7}$$

Thus, the average additional coverage area is

$$\int_0^r \frac{2\pi x(\pi r^2 - INTC(r,x))}{\pi r^2}dx \approx 0.41\pi r^2. \tag{8}$$

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

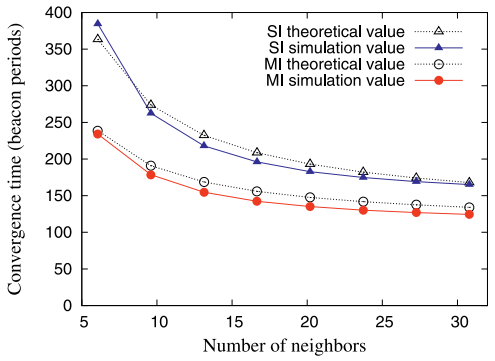

Fig. 16. The convergence time.

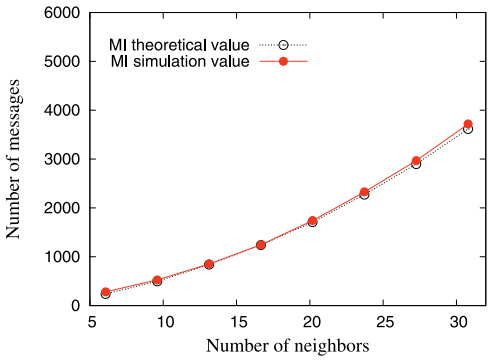

Fig. 17. The number of messages.

$$2Init_{Max} + (h-1)\cdot\frac{T_{max}}{0.41\Delta} + T_{max}. \quad (9)$$

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 an initiator to the edge of its tree, h_{mi}, is

$$h_{mi} = \frac{r/\bar{d}}{n_{init}\pi r^2/S} = \frac{3}{2}\cdot\frac{S}{n_{init}\pi r^2}. \quad (10)$$

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. The time required for the tree connection phase is bounded by $2h_{mi}+1$. Hence, 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

$$\alpha + (h_{mi}-1)\cdot\frac{T_{max}}{0.41\Delta} + T_{max} + 2h_{mi} + 1. \quad (11)$$

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

$$0.41\Delta\cdot\frac{n_{init}-1}{n_{init}}\cdot n. \quad (12)$$

5.2 Theoretical and Simulation Result Comparisons

In this section, the analytical model are validated by comparing with our simulation results.

Fig. 16 demonstrates the convergence time of SI and MI with respect to the average number of neighbors. The convergence time decreases in proportion to the number of neighbors because the defer timer in the tree construction phase decreases in proportion to the number of uncovered nodes. As can be seen in Fig. 16, the theoretical model and simulation results are very close to each other.

Fig. 17 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 Fig. 17, analytical model again provides a very accurate estimation.

6 Conclusion and Future Work

While the existing CDS protocols are successful in constructing CDS of small size, they miss a number of key features that are important in wireless ad hoc networks. In this paper, we introduce two timer-based CDS protocols, namely SI and MI, that not only create CDS of competitive size with low overheads but also address the shortcomings of the existing protocols. SI utilizes timers to distributively construct and maintain CDS in the presence of changes of network topology. Built on top of SI, MI requires minimum localized information to construct and maintain CDS efficiently. The performance of the SI and MI protocols are verified by both ns-2 simulations under static/mobile network settings and an analytical model. Both performance assessments provide very close results, which establishes the validity of our simulations.

Since both protocols use timers, which inevitably prolongs the convergence time required for CDS construction. In the future, we would like to investigate means to reduce the required convergence time.

References

[1] J. Blum, M. Ding, A. Thaeler, and X. Cheng, "Connected Dominating Set in Sensor Networks and MANETs," *Handbook of Combinatorial Optimization*, pp. 329-369, Kluwer Academic Publishers, 2005.

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

[3] J. Cartigny, D. Simplot, and I. Stojmenovic, "Localized Minimum-Energy Broadcasting in Ad-Hoc Networks," *Proc. IEEE INFOCOM*, pp. 2210-2217, Mar. 2003.

[4] A. Helmy, S. Garg, P. Pamu, and N. Nahata, "CARD: A Contact-Based Architecture for Resource Discovery in Ad Hoc Networks," *Mobile Networks and Applications*, vol. 10, no. 1, pp. 99-113, 2004.

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[6] F. Dai and J. Wu, "On Constructing k-Connected k-Dominating Set in Wireless Networks," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS)*, 2005.

[7] Y. Wu and Y. Li, "Construction Algorithms for k-Connected m-Dominating Sets in Wireless Sensor Networks," *Proc. Ninth ACM MobiHoc*, May 2008.

[8] Y. Wu, F. Wang, M.T. Thai, and Y. Li, "Constructing Algorithms for k-Connected m-Dominating Sets in Wireless Sensor Networks," *Proc. IEEE Military Comm. Conf. (MILCOM)*, pp. 29-31, Oct. 2007.

[9] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *Proc. ACM MobiCom*, July 2001.

[10] J. Wu, F. Dai, M. Gao, and I. Stojmenovic, "On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks," *J. Comm. and Networks,* vol. 4, no. 1, pp. 59-70, 2002.
[11] S. Yang, J. Wu, and F. Dai, "Efficient Directional Network Backbone Construction in Mobile Ad Hoc Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 12, pp. 1601-1613, Dec. 2008.
[12] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proc. Third Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm. (DIALM),* pp. 7-14, Aug. 1999.
[13] F. Dai and J. Wu, "An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 15, no. 10, pp. 908-920, Oct. 2004.
[14] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 1, pp. 14-25, Jan. 2002.
[15] D. Simplot-Ryl, I. Stojmenovic, and J. Wu, "Energy-Efficient Backbone Construction, Broadcasting, and Area Coverage in Sensor Networks," *Handbook of Sensor Networks: Algorithms and Architectures,* pp. 343-379, Wiley, 2005.
[16] P.J. Wang, K.M. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Proc. IEEE INFOCOM,* pp. 141-149, Apr. 2002.
[17] K.M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks," *Proc. ACM MobiHoc,* pp. 157-164, June 2002.
[18] P.-J. Wan, L. Wang, and F. Yao, "Two-Phased Approximation Algorithms for Minimum CDS in Wireless Ad Hoc Networks," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 337-344, June 2008.
[19] 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 Comm. and Mobile Computing,* vol. 5, no. 8, pp. 927-932, 2005.
[20] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.,* vol. 15, no. 7, pp. 1265-1275, Sept. 1997.
[21] F.G. Nocetti, J.S. Gonzalez, and I. Stojmenovic, "Connectivity Based k-Hop Clustering in Wireless Networks," *Telecomm. Systems,* vol. 22, pp. 1-4, 2003.
[22] *Approximation Algorithms for NP-Hard Problems,* D.S. Hochbaum, ed., PWS Publishing Co., 1997.
[23] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica,* vol. 20, no. 4, pp. 374-387, 1998.
[24] B.S. Baker, "Approximation Algorithms for NP-Complete Problems on Planar Graphs," *J. ACM,* vol. 41, no. 1, pp. 153-180, 1994.
[25] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, "Localized Protocols for Ad Hoc Clustering and Backbone Formation: A Performance Comparison," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 4, pp. 292-306, Apr. 2006.
[26] S. Basagni, M. Mastrogiovanni, and C. Petrioli, "A Performance Comparison of Protocols for Clustering and Backbone Formation in Large Scale Ad Hoc Networks," *Proc. IEEE Conf. Mobile Ad-Hoc and Sensor Systems,* pp. 70-79, Oct. 2004.
[27] K. Sakai, F. Shen, K.M. Kim, M.-T. Sun, and H. Okada, "Multi-Initiator Connected Dominating Set Construction for Mobile Ad Hoc Networks," *Proc. IEEE Int'l Conf. Comm. (ICC),* May 2008.
[28] "IEEE 802.11b Standard," http://standards.ieee.org/getieee802/download/802.11b-1999.pdf, 2011.
[29] D. Zhou, M.-T. Sun, and T. Lai, "A Timer-Based Protocol for Connected Dominating Set Construction in IEEE 802.11 Multihop Mobile Ad Hoc Networks," *Proc. Int'l Symp. Applications and the Internet (SAINT),* pp. 2-8, Jan. 2005.
[30] "The Network Simulator (ns-2)," http://www.isi.edu/nsnam/ns/, 2011.
[31] 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 Comm. Rev.,* vol. 9, no. 1, pp. 59-63, 2005.
[32] "Mobilab: Community-Wide Library of Mobility and Wireless Networks Measurements," http://nile.usc.edu/MobiLib/, 2009.

Kazuya Sakai received the BS and MS degrees in electronics engineering from Kansai University, Osaka, Japan, in 2004 and 2007, respectively, and the MS degree in computer science from Auburn University, Alabama, in 2010. Since 2010, he has been a graduate student in the Department of Computer Science and Engineering at The Ohio State University. His research interests are in the area of wireless networks, mobile computing, and security. He is a student member of the IEEE.

Scott C.-H. Huang received the BS degree in mathematics from the National Taiwan University in 1998 and the PhD degree in computer science from the University of Minnesota in 2004. He joined the faculty of City University of Hong Kong as a research fellow and is currently a lecturer there. His research area includes ad hoc and sensor networks, network security, and combinatorial optimization. He is a member of the IEEE.

Wei-Shinn Ku received the PhD degree in computer science from the University of Southern California (USC) in 2007 and the MS degrees in computer science and electrical engineering from the USC in 2003 and 2006, respectively. He is a graduate of the National Taiwan Normal University. Now, he is an assistant professor with the Department of Computer Science and Software Engineering at Auburn University. His research interests include spatial and temporal data management, mobile data management, geographic information systems, and security and privacy. He is a member of the IEEE.

Min-Te Sun received the BS degree in mathematics from the National Taiwan University in 1991, the MS degree in computer science from Indiana University in 1995, and the PhD degree in computer and information science from The Ohio State University in 2002. Since 2008, he has been with the Department of Computer Science and Information Engineering at the National Central University, Taiwan. His research interests include distributed algorithm design and wireless network protocol engineering. He is a member of the IEEE.

Xiuzhen Cheng received the MS and PhD degrees in computer science from the University of Minnesota-Twin Cities, in 2000 and 2002, respectively. She is an associate professor at the Department of Computer Science, The George Washington University. Her current research interests include cyberphysical systems, wireless and mobile computing, sensor networking, wireless and mobile security, and algorithm design and analysis. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She worked as a program director for the US National Science Foundation (NSF) from April to October in 2006, and from April 2008 to May 2010. She received the NSF CAREER Award in 2004. She is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.