# Multi-Initiator Connected Dominating Set Construction for Mobile Ad Hoc Networks

*Kazuya Sakai*[2], *Fangyang Shen*[1], *Kyoung Min Kim*[1], *Min-Te Sun*[1], and *Hiromi Okada*[2]
[1]Dept. of CSSE, Auburn University, Auburn, Alabama 36849-5347
[2]Dept. of Elec. Eng., Kansai University, Yamate-cho, Suita, Osaka, Japan 564-8680

***Abstract*— The connected dominating set (CDS) has been extensively used for routing and broadcast in mobile ad hoc networks. While existing CDS protocols are successful in constructing CDS of competitive size with localized information, they either lack the mechanism to properly handle nodal mobility or require lengthy period of time to recover when CDS becomes corrupted. In this paper, a novel protocol, namely Multi-Initiator Connected Dominating Set protocol (MI-CDS), is proposed that constructs and maintains CDS of competitive size efficiently without introducing much communication overhead. The simulation results demonstrate that MI-CDS permits CDS to be available for the highest percentage of time in the mobile network scenario compared with the other CDS protocols.**

## I. Introduction

Mobile ad hoc networks (MANETs) are well-suited for communications in the battlefield and rescue missions where a fixed infrastructure is not readily available. Many MANET protocols that provide different communication primitives, such as routing [1] and broadcast [2], rely on the availability of a virtual backbone. A virtual backbone of a MANET is the connected dominating set (CDS) of the graph representation of the MANET. 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 it will incur. When it comes to constructing CDS for a MANET, it is desirable that the size of the CDS is as small as possible. On the other hand, it is known that the problem of finding the minimum CDS is NP-hard [3]. The problem becomes even more challenging when no node has the view of the complete network topology. As a result, the existing MANET CDS protocols [4]–[6] emphasize constructing a small CDS distributively with localized information. While they are successful in creating a small CDS, they either lack the mechanism for mobility handling or incur significant overhead when maintaining the CDS. For a MANET where nodes are roaming freely all the time, it is likely that the CDS will not be available for service for a significant amount of time due to either frequent CDS reconstruction or lengthy CDS maintenance should these protocols be adopted.

In this paper, we propose the Multi-Initiator Connected Dominating Set protocol (MI-CDS). MI-CDS first elects a small number of initiators among nodes in the network, and then grows a dominator tree from each of the initiators distributively, and finally connects these trees via a small number of bridge nodes to form a CDS. Unlike the CDS protocols in [4], [5], MI-CDS is able to handle mobility without the need of reconstructing the CDS from scratch when the network topology changes. Compared with the CDS protocol in [6], MI-CDS does not suffer from the problem of the single point of failure and thus can maintain the CDS more efficiently. The simulation results of both static and mobile network scenarios validate that MI-CDS constructs and maintains CDS of competitive size with low overhead. In case of the mobile network scenario, MI-CDS permits CDS to be available for the highest percentage of time compared with the other CDS protocols.

The rest of this paper is organized as follows. The existing MANET CDS protocols are reviewed in Section II. The MI-CDS protocol is presented in detail along with how it handles nodal mobility is described in Section III. The simulation results are shown in Section IV. The conclusion and the future direction of our work are provided in Section V.

## II. Literature Review

Constructing the minimum CDS is known to be NP-hard. While there are some works that discuss how to approximate the minimum CDS under the assumption that the complete network topology is known [3], such an assumption is not practical when it comes to mobile ad hoc networks. The more practical approach, such as [4]–[6], is to construct a small CDS based on localized information. Depending on how much localized information is used during CDS construction, these protocols can be classified as two-hop based and one-hop based.

**Two-hop localized CDS construction -** The most noticeable CDS protocols in this category include Wu's [4] and Wan's [5]. Wu's CDS protocol consists of two stages. In the first stage, each node collects two-hop neighboring information via 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 CDS. In the second stage, additional heuristic rules are applied to further reduce the size of the CDS.

Wan's CDS protocol [5] 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 maximal independent set become the skeleton of the CDS. Although nodes in the maximal independent set 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 in CDS to connect the nodes in the maximal independent set.

While both Wu and Wan's CDS protocols are fully distributed and localized, the first stage of Wu's as well as the second stage of Wan's introduce a large number of messages. Additionally, none of them provides the mechanism to deal with mobility. In other words, even a slight change of network topology may cause the whole CDS to be reconstructed from scratch.

**One-hop localized CDS construction -** In [6], Zhou et al proposed a CDS construction protocol based on one-hop localized information. Zhou's protocol is again a two-stage process. In the first stage, a distributed leader election is used to select an unique initiator. In the second stage, a CDS tree is generated from the initiator by making use of the defer timer. A node sets its defer timer inversely proportional to the number of uncovered neighbors. When a node's timer expires, if all its neighbors are covered by some dominators, the node is switched to the dominatee status. Otherwise, the node joins the CDS to cover its uncovered neighbors.

In general, one-hop localized CDS protocol is better than two-hop ones in terms of the message overhead. Additionally, Zhou's protocol makes use of periodical heartbeat signals from the initiator to detect mobility events that cause the CDS to be corrupted and provides mechanisms to fix the corrupted CDS locally. However, when the initiator fails or moves away from the network, the CDS will have to be fully reconstructed. Another issue of Zhou's protocol is the duration of the first stage. If it is set to be too long, the CDS construction will take too much time to complete. If it is set to be too short, it may end up having more than one initiator, which will result in a disconnected dominating set.

## III. Multi-Initiator Connected Dominating Set Construction

### A. Protocol Skeleton

As stated in Section II, one-hop localized CDS construction is preferred as it incurs less message overhead. However, Zhou's protocol in [6] suffers from the single point of failure i.e., the whole CDS will have to be reconstructed should the single initiator fail. To tackle this issue, a natural approach is to elect multiple initiators. Each initiator generates a tree in exactly the same manner the single initiator does in [6]. This will produce several small disjoint trees and the union of these trees will form a dominating set. In the remainder of this paper, we refer to the tree generated from an initiator as the dominator tree. To obtain a CDS, we can simply include a few more nodes to connect these dominator trees. If the CDS is constructed this way, the failure of an initiator will only affects the associated dominator tree. The other part of the CDS will remain intact. Table I illustrates the skeleton of our Multi-Initiator Connected Dominating Set (MI-CDS) protocol.

TABLE I
Multi-Initiator Connected Dominating Set Protocol

1. **Initiator Election**
   /* Initiators are localized elected without introducing extra messages */
2. **Tree Construction**
   /* Each initiator grows a dominator tree independently */
3. **Tree Connection**
   /* Additional nodes are included to connect disjoint dominator trees */

In the following subsections, we explain how each of the three phases of our MI-CDS can be accomplished effectively and distributively in detail.

*1) Initiator Election Phase:* The most intuitive idea to elect multiple initiators effectively and distributively is to let the local minimum be the initiators. We assume that each node in the network carries a unique identifier (referred as $id$ hence after), such as its MAC address. Similar to what has been defined in IEEE802.11b specification [7], a node periodically broadcasts the beacon with its $id$. By listening to beacons, a node obtains the $id$s 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 that $n$ nodes uniformly distributed in a region of size $S$, the transmission radius of each node to be $r$, and the average number of neighbors to be $\Delta$, the average number of initiators $n_{init}$ can be obtained by Equation 1.

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

For instance, a network of 50 nodes with $150m$ transmission radius deployed in $1000m \times 1000m$ 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 two-hop neighbors is elected as the initiators. To achieve this, each node encodes the minimal $id$ among its one-hop neighbors in the beacon. A node becomes an initiator only when its $id$ equals to the minimal $id$ of the one-hop neighbors encoded in the beacon from all of its one-hop neighbors. With few rounds of beacon exchanges, the local minimum among its 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 Zhou's CDS protocol in the first stage. Using the same notation, the average distance between two neighbors $\overline{d}$ and the average number of initiators $n_{init}$ can be obtained by Equation 2 and 3, respectively.

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

$$n_{init2} = n \cdot (\frac{1}{\frac{n\pi(r+\overline{d})^2}{S} + 1}) \approx \frac{9}{25} \cdot \frac{S}{\pi r^2} \quad (3)$$

For instance, the network of 50 nodes with $150m$ transmission radius deployed in $1000m \times 1000m$ area now has an average number of 5 initiators. This significantly reduces the possibility of a large CDS.

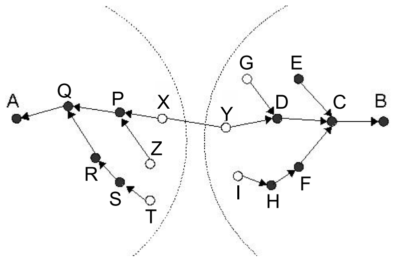


Fig. 1. Example of Tree Connection

*2) Tree Construction Phase:* In this phase, each initiator generates a dominator tree independently and distributively. The process is similar to Zhou's CDS protocol in [6]. For dominator tree generation, a node further includes its status, a dominator's $id$ if it is covered by a dominator, and an initiator's $id$ in the beacon. The status of a node can be either *uncovered*, *covered*, *dominator*, or *dominatee*. At the beginning, every node is in uncovered status. After an initiator is elected, it 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 that neighbor, its initiator to that neighbor's initiator, switches its status to covered, and sets a timer inversely proportional to the number of uncovered neighbors. When the timer expires, if the node still has uncovered neighbors, it switches its status to dominator to cover these neighbors. Otherwise, it switches its status to dominatee.

At the end of the process, each node is either a dominator or a dominatee. The dominators belonging to an initiator form a tree rooted at the initiator, which is referred as the dominator tree. If two dominator trees are close to each other i.e., separated by either one or two covered neighbors, they are referred to as neighboring dominator trees.

*3) 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 to use to connect to 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 the initiator's $id$ of the neighboring tree. For instance, in Figure 1 a border node $X$ belonging to initiator $A$ finds that one of its neighbors $Y$ belongs to initiator $B$ from $B$'s beacon, $X$ then sends a message to $A$ containing the $id$ of the initiator $B$. Similarly, $Y$ will send a message to $B$ containing the $id$ of the initiator $A$.

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 Figure 1, when node $P$ receives messages from $X$ and $Z$ about the same neighboring dominator tree, it only forwards one copy of the message to its dominator $Q$. Similarly, $Q$ only forwards one copy of the message it receives from $P$ and $R$ to its dominator $A$. By doing this, the number of messages can be controlled to $O(n)$, where $n$ is the number of nodes in the network.

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 status 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. 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.

### *B. Mobility Handling*

In MI-CDS, each dominator tree is maintained in the same way as Zhou's protocol [6]. This allows our protocol to take advantage of the mobility handling capability of Zhou's protocol inside each of the dominator trees. The root of a dominator tree periodically broadcasts a heartbeat signal consisting of the initiator's $id$ and a sequence number to nodes under the tree so that any topology changes due to mobility can be captured and handled in a timely fashion. As described in [6], this mechanism is able to handle the following four different mobility cases:

1) The initiator leaves its dominator tree.
2) A dominator leaves its dominator tree.
3) A new node joins a dominator tree after the tree is constructed.
4) A redundant dominator switches to dominatee status without disconnecting the dominator tree.

MI-CDS will create multiple dominator trees, so 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 above four cases. For instance, if a dominator moves from one tree to another, it can be seen as the second case for the first tree plus the third case 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 status 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 tree-wide 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 $A$.

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, MI-CDS 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.

## IV. Simulation Results

To evaluate the performance of MI-CDS, we implement our protocol in C++ along with the other CDS protocols, including Wu [4], Wan [5], and Zhou's [6] protocols. In this section, the simulation results of different CDS protocols are reported and analyzed.

### A. Simulation Configurations

The network topology is randomly generated by placing nodes on a $1000m$ by $1000m$ square field according to uniform distribution. The simulation region is wrapped around vertically and horizontally to eliminate the edge effect. If the generated network is partitioned, it is discarded and a new network topology is generated to ensure the connectivity of the whole network. The transmission range of a node is set to be $150m$. For a given simulation configurations, 100 different network topologies are generated. Two different network scenarios are considered and simulated as follows.

**Static Networks -** In this scenario, the average network density changes as we change the total number of nodes. The total number of nodes placed in the field ranges from 100 to 450, which corresponds to the network density ranging from approximately 5 to 30 neighbors per nodes. For Zhou's and MI-CDS, the configurations in the tree construction phase is set to be the same as in [6].

**Mobile Networks -** In this scenario, the average network 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 50% with speed up to $5m/s$. The Weighed Way Point (WWP) [8] 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 in University of Southern California's campus [9]. Since Wu and Wan's protocols do not have mobility handling mechanisms, the whole CDS is reconstructed if the CDS is found corrupted. For Zhou's and MI-CDS, the corrupted CDS is recovered according to their mobility handling procedures when the topology changes. The configurations of the heartbeat period is set to be the same as in [6]. Each simulation lasts 3600 rounds of beacon period. In the simulation, if the network topology is partitioned into disjoint connected components, MI-CDS maintains separate CDS within each component.

To assess the performance of different 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. For MI-CDS, the messages in the tree connection phase and query/response messages in the mobility handling are counted as the extra messages. For Zhou's protocol, all the information exchanges between nodes are done by beacons. For Wu's protocol, beacons are considered as extra messages since the size of its beacon increases in proportion to the network density and is 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 changes the beacon frame format to include additional information. For Zhou's, node $id$, $status$, $color$, and dominator $id$ are included in the beacon. MI-CDS enlarges the beacon of Zhou's to include the initiator $id$ and the minimal $id$ of one-hop neighbors. The beacon of Wu's protocol includes node $id$, $status$, $marker$, 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 kbps) 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 Zhou's protocol, the initiator election is assumed to be completed in 20 rounds of beacon intervals. For MI-CDS, the initiator election is done in three rounds of beacon intervals. For the mobile 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.

### B. Simulation Results for Static Network Scenarios

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

Figure 2 shows CDS size of different protocols with respect to the network density. It is clear that Zhou's consistently

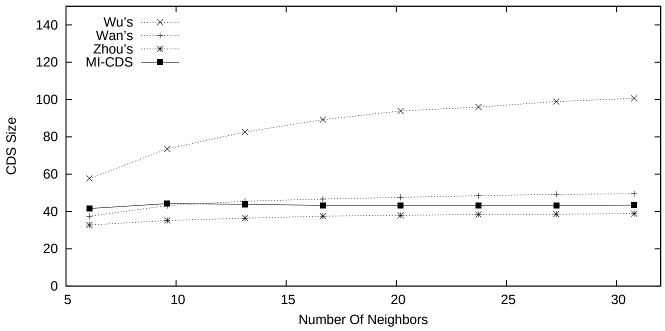


Fig. 2. CDS Size

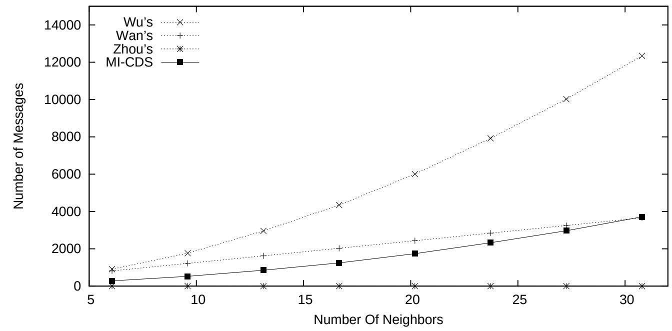


Fig. 3. Number of Messages

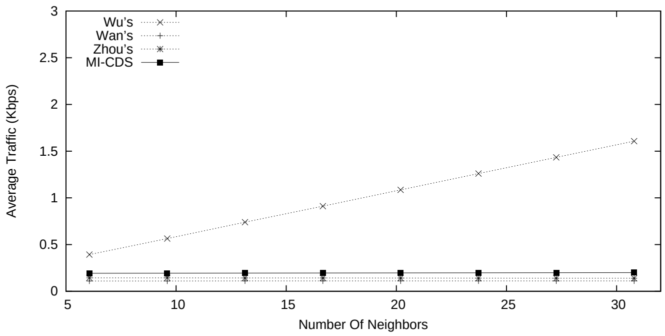


Fig. 4. Average Traffic (kbps)

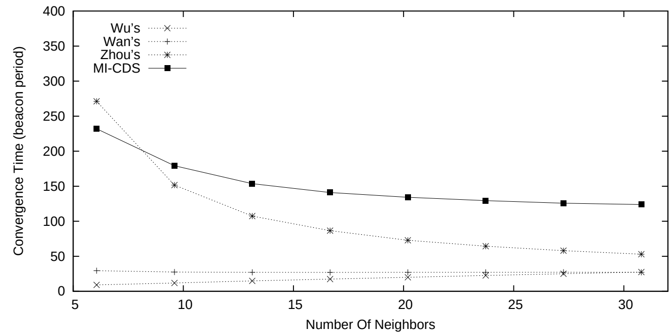


**Fig. 5. Convergence Time**

generates the smallest CDS and Wu's consistently generates the largest CDS among all the protocols. While it is proven that the size of Wan's CDS is suboptimal [5], the size of CDS of MI-CDS is smaller than Wan's and is very close (within few nodes) to Zhou's. In addition, the CDS size in MI-CDS remains constant when the network density increases. This suggests that MI-CDS is scalable.

Figure 3 demonstrates the number of extra messages with respect to the network density. As illustrated in Figure 3, MI-CDS introduces only 30% the number of extra messages introduced by Wu's. Compared with Wan's, MI-CDS reduces the number of extra messages up to 60% when the network density ranges from 5 to 15 neighbors per node. As discussed in [6], Zhou's protocol does not introduce any extra message.

Figure 4 shows the average traffic introduced for CDS construction with respect to the network density. Obviously, Wu'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 three protocols, the average traffic is almost the same. It is interesting that for Wan's and MI-CDS even the number of messages increases in proportion to the number of neighbors as shown in Figure 3, the traffic remains relatively constant to the network density. This implies the traffic is mostly dominated by the extra bits in the beacon.

Figure 5 presents the convergence time with respect to the network density. As shown in Figure 5, the convergence time of MI-CDS and Zhou's decreases quickly as the network density increases. This indicates that the convergence time of MI-CDS and Zhou's is dominated by the time spent for the tree construction, in which a node in dense networks is likely to have more uncovered neighbors and will have a smaller defer timer. MI-CDS takes longer time to form CDS than Zhou's protocol. This is due to the additional time spent in the tree connection phase.

### *C. Simulation Results for Mobile Scenarios*

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

Figure 6 depicts the percentage of time CDS is alive with respect to the percentage of the mobile nodes. As can be seen in Figure 6, MI-CDS always has the highest percentage of CDS alive time than other CDS protocols. The more mobile nodes the network has, the better performance MI-CDS achieves in terms of the CDS alive time percentage compared with the other protocols. For instance, in the case of 50% of mobile nodes, CDS built by MI-CDS is alive eight times longer than Wu and Wan's protocols, and more than twice longer than Zhou's. Although smaller CDS is generally more vulnerable to topology changes, MI-CDS shows excellent mobility adaptation compared with the other CDS protocols.

Figure 7 shows the average CDS size with respect to the percentage of the mobile nodes. As illustrated in Figure 7, Zhou's consistently produces the smallest CDS and Wu's consistently produces the largest CDS. The average CDS size of MI-CDS is only few nodes higher than Wan's.

Figure 8 presents the number of extra messages to maintain CDS with respect to the percentage of the mobile nodes. As illustrated in Figure 8, MI-CDS requires a low number of extra messages to maintain CDS. The number of messages is only 20% of that of Wan's and 8% of Wu's. This demonstrates that MI-CDS can handle topology changes efficiently with only a

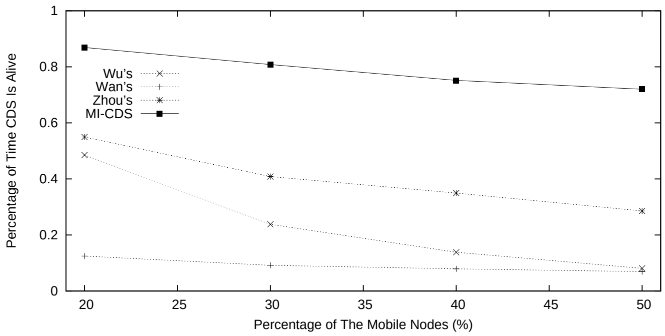


Fig. 6. Percentage of Time CDS is Alive

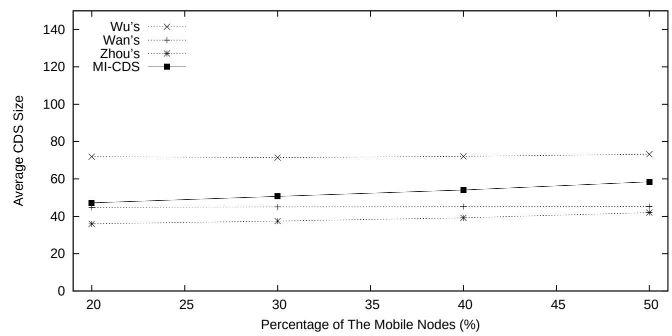


Fig. 7. Average CDS Size

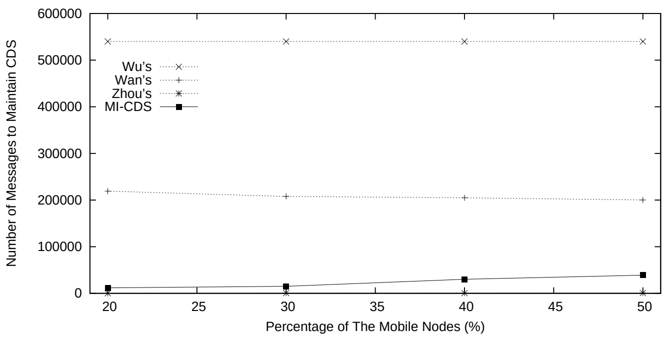


Fig. 8. Number of Messages to Maintain CDS

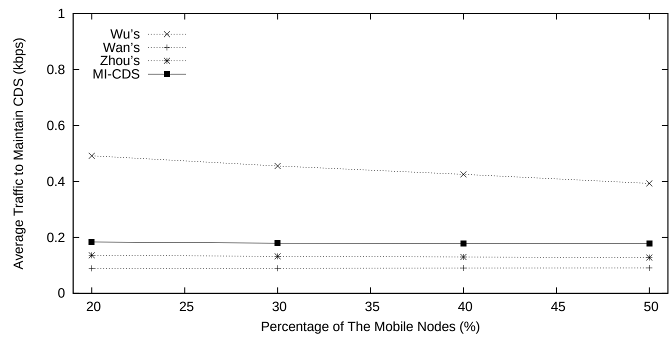


Fig. 9. Average Traffic to Maintain CDS

small number of messages.

Figure 9 shows the average traffic required at each node to maintain CDS with respect to the percentage of the mobile nodes. As can be seen in Figure 9, the average traffic of MI-CDS is at least 50% lower than that of Wu's. Compared with Wan's and Zhou's, MI-CDS has slightly higher traffic, but the difference is not significant. This is because the beacon in MI-CDS is slightly larger than that of Wan's and Zhou's. As we have pointed out in Subsection IV-B, the traffic is mostly dominated by the extra bits in the beacon frame rather than the extra messages. This again is validated by Figure 8 and Figure 9.

## V. Conclusion and Future Work

The CDS protocols proposed in the past either lack the ability to handle nodal mobility, or recover slowly when the constructed CDS becomes corrupted. In this paper, the Multi-Initiator Connected Dominating Set protocol (MI-CDS) is proposed that can construct and maintain CDS efficiently. Instead of relying on the single initiator, MI-CDS elects a small number of initiators to generate CDS. This allows MI-CDS to fix the corrupted CDS quickly caused by different types of nodal mobility. The simulation results demonstrate that MI-CDS consistently generates CDS of competitive size with very low communication overhead. In case of mobile networks, the simulation results confirm that MI-CDS allows CDS to be available for the highest percentage of the time compared with the other CDS protocols. In the future, we would like to consider the virtual backbone construction for multi-hop wireless mesh networks, where each node may be equipped with multiple radio interfaces. The definition of the connected dominating set may need to be extended so that it can be applied to multi-radio wireless mesh networks. In addition, the performance metrics of a virtual backbone for multi-radio wireless mesh networks should include throughput, which makes the CDS protocol performance more difficult to assess since the throughput is also dependent on the routing protocol.

## References

[1] J. Wu, "Extended Dominating-Set-Based Routing in Ad Hoc Wireless Networks with Undirectional Links," *IEEE Transactions on Parallel and Distributed Computing*, 22, 1-4, pp. 327-340, Sep. 2002.

[2] J. Cartigny, D. Simplot, and I. Stojemenovic, "Localized Minimum-energy Broadcasting in Ad Hoc Networks," *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pp. 2210-2217, Mar. 2003.

[3] D. S. Hochbaum, *Approximation Algorithms for NP-Hard Problems*, PWA Publishing Company, Boston, MA, 1995.

[4] J. Wu, H. L. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proc. of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 7-14, Aug. 1999.

[5] P. J. Wang, K. M. Alzoubi, O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pp. 141-149, Apr. 2004.

[6] D. Zhou, M.-T. Sun and T.H. Lai, "A Timer-based Protocol for Connected Dominating Set Construction in IEEE 802.11 Wireless Networks," *Proc. of International Symposium on Applications and the Internet (SAINT)*, pp. 2-8, Jan. 2005.

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

[8] W. Hsu, K. Merchant, H. Shu, C. Hsu, A. Helmy, "Weighted Waypoint Mobility Model and its Impact on Ad Hoc Networks," *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, pp. 59-63, Jan. 2005.

[9] Mobilab: Community-wide Library of Mobility and Wireless Networks Measurements http://nile.usc.edu/MobiLib/