Distributed Heuristics for Connected Dominating Sets in Wireless Ad Hoc Networks

Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder

Abstract: **A connected dominating set (CDS) for a graph $G(V,E)$ is a subset V' of V, such that each node in $V-V'$ is adjacent to some node in V', and V' induces a connected subgraph. CDSs have been proposed as a virtual backbone for routing in wireless ad hoc networks. However, it is NP-hard to find a minimum connected dominating set (MCDS). An approximation algorithm for MCDS in general graphs has been proposed in the literature with performance guarantee of $3+\ln\Delta$ where Δ is the maximal nodal degree [1]. This algorithm has been implemented in distributed manner in wireless networks [2]–[4]. This distributed implementation suffers from high time and message complexity, and the performance ratio remains $3+\ln\Delta$. Another distributed algorithm has been developed in [5], with performance ratio of $\Theta(n)$. Both algorithms require two-hop neighborhood knowledge and a message length of $\Omega(\Delta)$. On the other hand, wireless ad hoc networks have a unique geometric nature, which can be modeled as a unit-disk graph (UDG), and thus admits heuristics with better performance guarantee. In this paper we propose two destributed heuristics with *constant* performance ratios. The time and message complexity for any of these algorithms is $O(n)$, and $O(n\log n)$, respectively. Both of these algorithms require only single-hop neighborhood knowledge, and a message length of $O(1)$.**

Index Terms: **Ad hoc networks, connected dominating set, independent set, leader election, spanning tree.**

I. INTRODUCTION

Wireless ad hoc networks can be flexibly and quickly deployed for many applications such as automated battlefield operations, search and rescue, and disaster relief. Unlike wired networks or cellular networks, no physical backbone infrastructure is installed in wireless ad hoc networks. A communication session is achieved either through a single-hop radio transmission if the communication parties are close enough, or through relaying by intermediate nodes otherwise. In this paper, we assume that all nodes in a wireless ad hoc network are distributed in a two-dimensional plane and have an equal maximum transmission range of one unit. Each node has a unique ID. Scheduling of transmission is the responsibility of the MAC layer. The topology of such wireless ad hoc network can be modeled as a *unit-disk graph* (UDG) [6], a geometric graph in which there is an edge between two nodes if and only if their distance is at most one (see Fig. 1).

Although a wireless ad hoc network has no *physical* backbone infrastructure, a *virtual* backbone can be formed by nodes in a connected dominating set (CDS) of the corresponding UDG [2]–[4]. In general, a *dominating set* (DS) of a graph $G=(V,E)$ is a subset $V'\subset V$ such that each node in $V-V'$ is adjacent to some node in V', and a *connected dominating set* (CDS) is a dominating set that also induces a connected subgraph. A (connected) dominating set of a wireless ad hoc network is a (connected) dominating set of the corresponding UDG. A virtual backbone, also referred to as a *spine*, plays a very important role in routing, where the number of nodes responsible for routing can be reduced to the number of nodes in the CDS. The virtual backbone also plays an important role for broadcasting and connectivity management in wireless ad hoc networks [2]. Broadcasting responsibility can be reduced to the nodes in the CDS instead of all the nodes in the graph. To reduce the communication overhead, to increase the convergence speed, and to simplify the connectivity management, it is desirable to find a minimum connected dominating set (MCDS) of a given set of nodes.

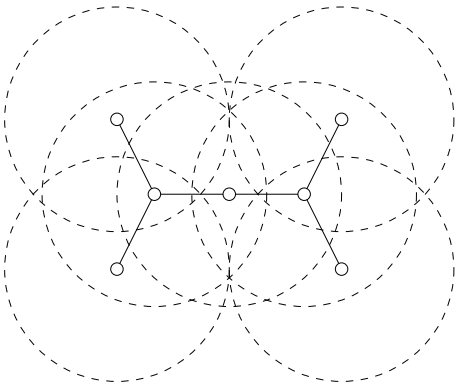

Fig. 1. Modeling the topology of wireless ad hoc networks by unit-disk graphs.

The MCDS in general graphs has been studied in [1]. An approximation preserving reduction from the set-cover problem [7] to MCDS was given in [1], which implied that for any fixed $0<\epsilon<1$, no polynomial-time algorithm can find a connected dominating set in general graphs within $(1-\epsilon)H(\Delta)$ times the MCDS unless $NP\subset DTIME\left[n^{O(\log\log n)}\right]$ [8], where Δ is the maximum degree and H is the harmonic function. Two greedy heuristics with performance guarantee of $2H(\Delta)+2$ and $ln\Delta+3$ respectively were also given in [1]. To find an MCDS in a UDG is still NP-hard [6]. A 10-approximation *centralized* algorithm for MCDS in UDG was first proposed in [9].

In this paper we concentrate on the construction of an CDS in UDG. The construction of the CDS should be distributed and simple. Since the networking nodes in wireless ad hoc networks are very limited in resources, a virtual backbone should not only be "thinner," but should also be constructed with low communication and computation costs. In addition, the communication

Manuscript received October 1, 2001.

The authors are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, e-mail: {alzoubi, wan, ophir }@cs.iit.edu.

and computation costs should be scalable as the wireless ad hoc networks are typically deployed with large network size.

To achieve a performance ratio within a small constant factor, we take advantage of the property of the maximal independent set (MIS), where the approximation ratio of any heuristic for constructing an MIS in a UDG is at most 5. An MIS S is an independent DS, i.e., all pairwise nodes in S are nonadjacent. Thus the construction of an MIS is a construction of a DS with approximation ratio of 5. In this paper we propose a distributed heuristic for constructing the MIS. Before starting the construction process each node is assigned a unique rank. We use two approaches for rank assignment. In the first (ID-Based) approach the rank of each node is its own ID. The time and message complexity for this approach are both linear. In the second (Level-Based) approach, an arbitrary spanning tree (ST) T is constructed before the rank assignment, then the rank of each node is the ordered pair (level, ID), where level is the number of hops to the root in T. The time and message complexity of the Level-Based approach is $O(n)$ and $O(n \log n)$ respectively.

In this paper we propose two distributed heuristics for constructing the CDS. The first heuristic uses the ID-Based approach for rank assignment. This approximation algorithm has a constant factor of 12. The second heuristic uses the Level-Based approach for rank assignment, and has a constant factor of 8. The second algorithm is a distributed implementation of the centralized approximation algorithm in [9], which has a performance ratio of 10. In our analysis, we show a tighter performance ratio of 8, instead of 10. This algorithm uses the breadth first spanning (BFS) tree as a building block for the construction of an CDS. The distributed construction of any BFS tree has a time and message complexity of $O(n^2)$. In our implementation, we reduce this complexity overhead by replacing the BFS tree with an arbitrary spanning tree. The message complexity of this implementation is reduced to $O(n \log n)$, and the time complexity is reduced to $O(n)$, while still maintaining a ratio of 8. This complexity is dominated by the leader election procedure.

The remainder of this paper is organized as follows. In Section II, we review related work . In Section III, we discuss the properties of the maximal independent set (MIS), and provide two approaches of the MIS construction. In Section IV we propose a distributed construction for the CDS with a performance factor of 12. In Section V we give a distributed implementation of the MCDS heuristic in [9], then we prove a tighter performance ratio of 8. Finally, we conclude this paper in Section VI.

II. LITERATURE REVIEW

Distributed approximation algorithms for MCDS in wireless ad hoc networks were first developed in a series of papers [2]–[4]. These algorithms provided distributed implementations of the greedy heuristics given in [1]. The CDS is referred to in these papers as a spine, and functions as a virtual backbone. The primary task of the spine is the route computation and maintenance. Nodes in the spine maintain up to date information about their domains (neighbors of the CDS nodes), and are used to exchange such information between each other to store global information of the network. The advantage of this strategy is the storage of global information in fewer nodes than the number of nodes in the network, which reduces the access overhead for this information, and reduces the update overhead. According to this strategy, it is highly desirable to reduce the size of the CDS in order to minimize the access and update overhead. The spine is not necessarily used to route packets in the network, even though it can be used to provide a temporary backup routes for fault tolerance.

We notice that these algorithms lack mechanisms to bridge two consecutive stages. Specifically, individual nodes have no way to tell when the next stage should begin. Furthermore, these approximation algorithms suffer from high performance ratio, and high implementation complexities, $O(n^2)$ time and messages. The construction of the MCDS requires 2-hop neighborhood knowledge, which means larger message size, frequent updates, slower convergence speed, and more memory.

In [5], a distributed algorithm was proposed for the construction of an approximation MCDS. This algorithm runs in two phases. In the first phase, each node first broadcasts to its neighbors the *entire* set of IDs of its neighboring nodes, and after receiving this adjacency information from all neighbors it declares itself as dominator if and only if it has two nonadjacent neighbors. These dominators form the initial CDS U. In the second phase, a node u in U is considered as *locally redundant* if it has either a neighbor in U with larger ID which dominates all other neighbors of u, or two adjacent neighbors with larger IDs which together dominate all other neighbors of u. The algorithm then removes all locally redundant nodes from U.

As indicated in [5], the theoretical performance of this algorithm in terms of the number of nodes in the output CDS remains unspecified. In this paper, we provide an instance showing a performance factor of $\Theta(n)$. Consider the instance when an even number of nodes n are evenly distributed over the two horizontal sides of a unit-square. Each horizontal side has exactly m nodes, and each node has exactly m neighbors, one in the opposite horizontal side and the rest on the same horizontal side. Any MCDS consists of a pair of nodes lying in a vertical segment. However, the CDS output by the algorithm in [5] consists of all nodes. Indeed, for each node u, the unique neighbor lying in the opposite horizontal side is not adjacent to all other neighbors of u. Thus, the initial CDS U constructed by the first phase consists of all nodes. In addition, no single neighbor of a node u can dominate all other neighbors of u. Furthermore, if a pair of neighbors of u are adjacent, they must lie in the same horizontal side as u, and therefore neither of them is adjacent to the unique neighbor of u lying in the opposite horizontal side. Thus, the second phase can't reduce the size of the initial CDS. Consequently, the output CDS still consists of all nodes, and the performance ratio for this algorithm is $\Theta(n)$, exactly $n/2$.

It is claimed in [5] that the total message complexity is $O(n\Delta)$ and the time complexity at each node is $O(\Delta^2)$. A more accurate message complexity is $\Theta(m)$ where m is the number of edges in the UDG, as each edge contributes two messages in the first phase. However, the $O(\Delta^2)$ time complexity is not correct. In fact, in order to decide whether it is locally redundant in the second phase, a node u in the initial CDS may have to examine as many as $O(\Delta^2)$ pairs of neighbors, and for each pair of neighbors, as much as $O(\Delta)$ time may be taken to find out whether such pair of neighbors together dominates all other

neighbors of u. Therefore, the time complexity at each node may be as high as $O\left(\Delta^3\right)$, instead of $O\left(\Delta^2\right)$. Note that m and Δ can be as many as $O\left(n^2\right)$ and $O\left(n\right)$ respectively. Thus, the message complexity and the time complexity of the distributed algorithm in [5] are $O\left(n^2\right)$ and $O\left(n^3\right)$ respectively.

In the context of clustering and broadcasting, Stojmenovic et al. [10] presented a distributed construction of the CDS. The CDS consists of two types of nodes: The cluster-heads and the border-nodes. The cluster-heads form an MIS. Several algorithms for MIS were described in [10], which can be generalized to the following framework:

- Each node has a unique *rank* parameter such as the ID only [11], [12], an ordered pair of degree and ID [13], an order pair of degree and location [10]. The ranks of all nodes give rise to a total ordering of all nodes.
- Initially, each node which has the lowest rank among all neighbors broadcasts a message declaring itself as a cluster-head. Note that such node does exist.
- Whenever a node receives a message for the *first* time from a cluster-head, it broadcasts a message giving up the opportunity as a cluster-head.
- Whenever a node has received the giving-up messages from all of its neighbors with lower ranks, if there is any, it broadcasts a message declaring itself as a cluster-head.

After a node learns the status of all neighbors, it joins the cluster centered at the neighboring cluster-head with the lowest rank by broadcasting the rank of such cluster head. The border-nodes are those which are adjacent to some node from a different cluster.

The implementation cost of these algorithms given in [10] depends on the choice of the rank. If the rank is ID only, which remains unchanged throughout the process, both the time complexity and the message complexity of this algorithm are $\Theta\left(n\right)$. If the rank involves the degree, which would change dynamically throughout the process, a significant amount of time and messages have to be devoted to rank updating and synchronization. The algorithms in [10] didn't provide these implementation details. But we believe that $O\left(n^2\right)$ messages and time may be required for rank updating and synchronization. Regardless of the choice of the rank, all algorithms in [10] have $\Theta\left(n\right)$ approximation factor. Such inefficiency stems from the non-selective inclusion of all border-nodes.

A centralized heuristic with constant performance ratio of 10 was developed for a UDG in [9], but centralized heuristics are not practical for wireless ad hoc networks. Also the performance ratio can be shown to have a tighter bound of 8 instead of the given ratio of 10. A key component of this heuristic is the BFS tree. The distributed implementation of this component is a bottleneck in the message complexity, as it may use $O\left(n^2\right)$ messages.

III. DISTRIBUTED CONSTRUCTION FOR MIS

The minimum dominating set (MDS) in a UDG admits a polynomial-time approximation scheme (PTAS) [14]. In other words, for any fixed $\epsilon > 0$, there exists a polynomial-time (in the size of the nodes and ϵ) algorithm which computes a DS of size at most $1 + \epsilon$ times the minimum. The PTAS for MDS in a UDG is based on a sophisticated use of the shifting strategy [15] that was previously employed, among other results, for obtaining PTASs for various optimization problems in planar graphs [16]. However, this PTAS is not suitable for distributed implementation in wireless ad hoc networks, due to its implementation complexity.

An alternative approach is to construct an MIS. In general, an *independent set* (IS) of a graph $G = (V, E)$ is a subset of pairwise non-adjacent nodes in V, and a *maximal independent set* (MIS) is an independent set such that any other node is adjacent to some node in the MIS. Obviously, any MIS is also a DS, and conversely, any independent DS must be an MIS. An MIS should intuitively have a small size as the nodes in an independent set are "sparsely" distributed with certain distance between any pair of nodes. Indeed, the size of any MIS in a UDG is at most five times of the size of the MDS, as each node is adjacent to at most five independent nodes [9].

In a general graph, an MIS can be constructed in the following simple way: Initially all nodes are unmarked (white). While there is some unmarked nodes, select an arbitrary unmarked node v, mark it black and mark all its neighbors gray. When all nodes are marked, all black nodes form an MIS. In a wireless ad hoc network each node has a unique *rank* parameter used in the construction process. In this paper we consider two approaches for rank assignment. In the first (ID-Based) approach, the rank of each node is simply its ID. A node with the lowest ID among all its neighbors has the lowest rank. In the second (Level-Based) approach, the rank of a node is an ordered pair of the node's level and ID. To define the rank in the Level-Based approach, we first apply the distributed leader election algorithm in [17], $O\left(n\right)$ time complexity and $O\left(n \log n\right)$ message complexity, to construct a rooted spanning tree T rooted at a node v. After such construction is completed, each node identifies its tree level with respect to T (i.e., its graph distance in T from the root) as follows: The root first announces its level 0. Each other node, upon receiving the level announcement message from its parent in T, obtains its own level by increasing the level of its parent by one, and then announces this level. Each node also records the levels of its neighbors in the UDG.

When a leaf node has determined its level, it transmits a LEVEL-COMPLETE message to its parent. Each internal node will wait till it receives this LEVEL-COMPLETE message from each of its children and then forward it up the tree toward the root. When the root receives the LEVEL-COMPLETE message from all its children, each node knows the levels and IDs of its own and its neighbors. The rank of each node is then given by the ordered pair of level and ID of a node. The ranks of all nodes are sorted in the lexicographic order. Thus the root, which is at level 0, has the lowest rank.

The following principles can be used for distributed construction of the MIS:

- Initially each node has the status candidate.
- Any node which has the lowest rank among all neighbors marks itself black and declares itself as a dominator by broadcasting a DOMINATOR message.
- Whenever a node receives a DOMINATOR message for the *first* time, it marks itself gray and declares itself as a

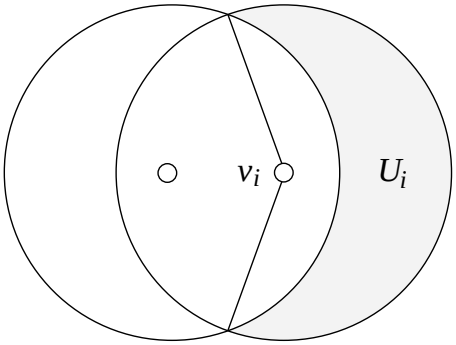

Fig. 2. U_i lie in a sector of at most 240 degree within the coverage range of node v_i.

dominatee by broadcasting a DOMINATEE message.

- Whenever a node has received the DOMINATEE messages from all of its neighbors with lower ranks, if there is any, it marks itself black and declares itself as a dominator by broadcasting a DOMINATOR message.

If the Level-Based approach is used for rank assignment, and a reporting of the MIS completion is necessary, a reporting process can be performed as follow: When a leaf node is marked, it transmits an MIS-COMPLETE message to its parent. Each internal node will wait till it receives this MIS-COMPLETE message from each of its children and then forward it up the tree toward the root.

Obviously, the time complexity for either approach is $O(n)$. The message complexity is $O(n)$ for the ID-Based approach, and $O(n \log n)$ for the Level-Based approach. Next, we bound the number of black nodes in terms of the size of an MCDS, denoted by *opt*. Intuitively, the nodes in an independent set are "sparsely" distributed with certain distance between any pair of nodes. Indeed, it is well-known that in a UDG each node is adjacent to at most five independent nodes. This immediately implies that the size of any independent set is at most $5 \cdot opt$. Next, we show a stronger bound on the size of any independent set.

Lemma 1: The size of any maximal independent set in a UDG $G = (V, E)$ is at most $4 \cdot opt + 1$, where *opt* is the size of MCDS.

Proof: Let U be any maximal independent set of V. Let OPT be any MCDS, and choose an arbitrary spanning tree T of OPT. Pick an arbitrary node in OPT as the root of T. Let $v_1, v_2, \cdots, v_{opt}$ be an arbitrary preorder traversal of T. Let U_1 be the set of nodes in U that are adjacent to v_1. For any $2 \leq i \leq opt$, let U_i be the set of nodes in U that are adjacent to v_i but none of $v_1, v_2, \cdots, v_{i-1}$. Then $U_1, U_2, \cdots, U_{opt}$ form a partition of U. From the above discussion, $|U_1| \leq 5$. For any $2 \leq i \leq opt$, at least one node in $v_1, v_2, \cdots, v_{i-1}$ is adjacent to v_i. Thus U_i lie in a sector of at most 240 degree within the coverage range of node v_i (see Fig. 2). This implies that $|U_i| \leq 4$. Therefore,

$$|U| = \sum_{i=1}^{opt} |U_i| \leq 5 + 4(opt - 1) = 4 \cdot opt + 1.$$

This completes the proof. □

By definition, any pair of nodes in an MIS are separated by at least two hops. However, a subset of nodes in an MIS U may be three hops away from its complementary subset in U. This case may appear when an ID-Based approach is used for rank assignment. In the next theorem, we show that the MIS constructed by using the Level-Based approach for rank assignment guarantees that the distance between any pair of complementary subsets is *exactly* two hops.

Theorem 1: Let U be the set of MIS nodes constructed by using the Level-Based approach for rank assignment. The distance between any pair of complementary subsets of U is exactly two hops.

Proof: Let $U = \{u_i : 1 \leq i \leq k\}$ where u_i is the i^{th} node which is marked black. For any $1 \leq j \leq k$, let H_j be the graph over $\{u_i : 1 \leq i \leq j\}$ in which a pair of nodes is connected by an edge if and only if their graph distance in G is two. We prove by induction on j that H_j is connected. Since H_1 consists of a single vertex, it is connected trivially. Assume that H_{j-1} is connected for some $j \geq 2$. When the node u_j is marked black, its parent in T must be already marked gray. Thus, there is some node u_i with $1 \leq i < j$ which is adjacent to u_j's parent in T. So (u_i, u_j) is an edge in H_j. As H_{j-1} is connected, so must be H_j. Therefore, H_j is connected for any $1 \leq j \leq k$. The connectedness of H_k then implies that the bipartite separation of U is exactly two. □

IV. ID-BASED APPROACH FOR CDS

A. Overview of the Algorithm

The construction of the CDS in this section uses the MIS generated by the ID-Based approach for rank assignment. A variation of this algorithm was proposed in [18], but the Level-Based approach was used for rank assignment. Thus, the performance ratio was 8 instead of 12. In this section the distributed algorithm for CDS consists of three procedures: Leader Election, MIS Construction, and Dominating Tree Construction. The Leader Election procedure elects a node e.g., with the smallest ID, as the leader. The distributed algorithm in [17] for leader election can be adopted. This algorithm has a message complexity of $O(n \log n)$. When the leader is found, it broadcasts its identity to all the nodes in the network.

When a node receives the leader's identity, it starts the MIS construction procedure, which is described in the previous section, and using the ID-Based approach for rank assignment. Each node will either be colored with black (as a dominator) or gray (as a dominatee). The leader will also select a black node as the root of the dominating tree as follows: If the leader is marked black, it selects itself as the root of the tree, otherwise it selects one of the black neighbors to be the root.

The Dominating Tree Construction procedure is initiated by the root to construct a tree containing all black nodes in addition to some gray nodes. All nodes in this dominating tree form an CDS. The root joins the dominating tree first, then it sends an invitation message to all black nodes within 3-hop distance to join the tree. When a node receives an invitation message, and all its neighbors have been marked either black or gray, it responds to

the message. When each black node joins the dominating tree, it will also send an invitation to all black nodes within 3-hop distance to join the tree. This invitation will be relayed through the gray nodes within 2-hops distance. Each black node will join the tree when it receives the invitation for the first time together with the gray nodes which relays the invitation to itself. This process should be repeated until all black nodes are in the tree.

B. Implementation Detail

Two types of messages will be used by all nodes: INVITE and JOIN. An INVITE message is initiated by a black node upon joining the dominating tree and relayed by gray nodes to solicit other black nodes to join the dominating tree. It is a broadcast message which consists of two fields: $senderID$ which represents the ID of the sender, and hop which represents the number of hops by which this message has been relayed. Since an invitation is targeted for black nodes within 3-hop distance, the field hop can only have three different values: $0, 1$ or 2. A JOIN message is initiated by a black node upon receiving the first invitation and sent in the reverse direction along the path in which the first invitation came along. It is a unicast message which consists of the two fields: $senderID$ which represents the ID of the sender, and $receiverID$ which represents the ID of the receiver.

A gray node will not relay each INVITE or JOIN message it receives. Instead, for INVITE messages it only transmits at most one message on behalf of all black neighbors and at most one message on behalf of all 2-hop distance black nodes; for JOIN messages it transmits at most once. To achieve this, each gray node maintains two local variables: $inviter$ and $counter$. The variable $counter$ can have three different values: $0, 1$ or 2. It is initialized to 0. If it has transmitted an INVITE message initiated from a black neighbor, it will be set to 1. If the node has transmitted an INVITE message initiated from a black neighbor two hops away, it will be set to 2. The variable $inviter$ is initialized to null, and will hold the ID of the sender of the *first* received INVITE message with $hop = 0$ or 1 through the whole construction process.

The dominating tree is initially empty. The root will be the first one to join the tree. When a black node joins the dominating tree, it will first send an INVITE message with $hop = 0$. When a gray node receives an INVITE message, it will ignore the message if it is COMPLETE, or if either in the received INVITE message $hop = 2$ or its local variable $counter = 1$. Otherwise, if its local variable $counter = 0$, it sets the local variables $counter = hop + 1$ and $inviter = senderID$, modifies the INVITE message by resetting the $senderID$ field in the message with its own ID and incrementing hop by one, and then transmits the INVITE message. If its local variable $counter = 2$, and the variable $hop = 0$ in the INVITE message, it sets the local variable $counter = hop + 1$, modifies the INVITE message by resetting the $senderID$ field in the message with its own ID and incrementing hop by one, and then transmits the INVITE message. However, if its local variable $counter = 2$, and the variable $hop = 1$ in the INVITE message, the message is ignored.

When a black node not in the dominating tree receives an INVITE message for the first time, it puts the sender of the received INVITE message as its parent, then sends back a JOIN message in which the field $senderID$ is set to its own ID and the field $receiverID$ is set to the value of $senderID$ in the received INVITE message, and finally sends out a new INVITE message. When a node, gray or black, receives a JOIN message *addressed to itself*, it puts the sender of the JOIN message as its child. In addition, when a gray node receives a JOIN message addressed to itself for the *first* time, it also puts the node whose ID is stored in its local variable $inviter$ as its parent, and then sends a JOIN message in which the field $senderID$ is set to its own ID and the field $receiverID$ is set to the local variable $inviter$.

The construction of the dominating tree is completed when all black nodes have joined the dominating tree. A reporting process if necessary, can be performed by constructing a spanning tree rooted at the leader to notify the leader of the completion. A gray node reports a COMPLETE message to its parent in the spanning tree if it has received a COMPLETE message from each child in the spanning tree. A black node reports a COMPLETE message to its parent in the spanning tree if it has received a COMPLETE message from each child in the spanning tree and itself has joined the dominating tree.

C. Analyses of the Algorithm

The next theorem proves the correctness of the algorithm, analyzes its performance ratio and the message/time complexity.

Theorem 2: At the end of the third phase, all nodes in the dominating tree form an CDS with size at most $12opt + 3$. In addition, the algorithm has $O(n \log n)$ message complexity and $O(n)$ time complexity.

Proof: First we claim that all black nodes will eventually join the dominating tree. Suppose to the contrary that some black nodes are outside the dominating tree. Let v be a closest black node to the dominating tree. Then the distance between v and the dominating tree is at most three hops. Therefore, v will receive an INVITE message initiated from some black node in the dominating tree, and would then join the dominating tree, which is a contradiction. Second we notice that when each black node other than the root joins dominating tree, it brings together at most gray nodes to join the dominating tree at the same time. As the number of black nodes is at most $4 \cdot opt + 1$ from Lemma 1, the number of nodes in the dominating tree is at most

$$3 \cdot (1 + (4opt)) = 12opt + 3.$$

The procedure Leader Election has $O(n \log n)$ message complexity and $O(n)$ time complexity. The procedure MIS Construction has $O(n)$ message complexity and $O(n)$ time complexity. In the procedure Dominating Tree Construction, each black nodes sends exactly one INVITE message and one JOIN message; each gray node sends at most two INVITE messages, and at most one JOIN message. If the report process is implemented, an additional COMPLETE message is sent by each node. Thus the third procedure has $O(n)$ message complexity and $O(n)$ time complexity. Therefore, the algorithm has $O(n \log n)$ message complexity and $O(n)$ time complexity, which is dominated by the Leader Election procedure. □

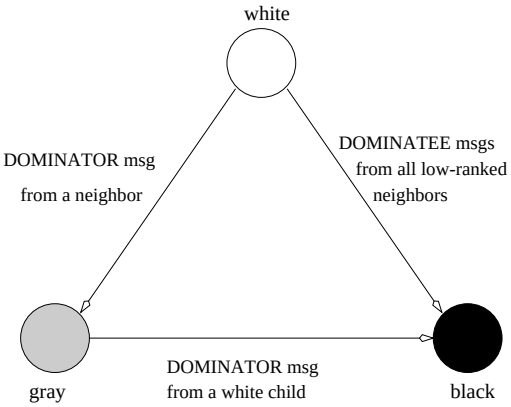

Fig. 3. State transition diagram.

V. LEVEL-BASED APPROACH FOR CDS

The construction of the CDS in this section uses the MIS generated by the Level-Based approach for rank assignment. This algorithm can be divided into three phases: The Leader Election Phase, the Level Calculation Phase, and the Color Marking Phase. The distributed algorithm in [17] for leader election can be adopted. This algorithm has $O(n \log n)$ message complexity, and $O(n)$ time complexity. After the construction of the spanning tree is completed, and when the root receives LEVEL-COMPLETE messages from all its children, each node knows the level numbers of all its neighbors. The pair (level, ID) of a node defines the *rank* of this node. The ranks of all nodes are sorted in the lexicographic order. Thus the leader, which is at level 0, has the lowest rank. In the Color Marking Phase all nodes are initially unmarked (white), and will eventually get marked either black or gray. Two types of messages are used by the nodes during this phase, the DOMINATOR message and the DOMINATEE message. The DOMINATOR message is sent by a node after it marks itself black, and the DOMINATEE message is sent by a node after it marks itself gray. Both messages contain the sender's ID. The algorithm can be described as a color markup process. Initially, the root marks itself black, and then broadcasts to its neighbors a DOMINATOR message. All other nodes act according to the following principles.

- Whenever a white node receives a DOMINATOR message from a *white* neighbor for the *first* time , it marks itself gray and broadcasts the DOMINATEE message.
- When a white node has received a DOMINATEE message from *each* of its neighbors of lower rank, it marks itself black and broadcasts the DOMINATOR message.
- When a gray node receives a DOMINATOR message for the *first* time from one of its children in T, which has never sent a DOMINATEE message, it *remarks* itself black and broadcasts the DOMINATOR message.

Fig. 3 shows the state transition diagram of this phase. Eventually each node will be either black (a dominator) or gray (dominatee). A reporting process, if necessary, can be performed to notify the root of the completion. When a leaf node has determined its status, it transmits a report COMPLETE message to its parent. Each internal node will wait till it receives this report COMPLETE message from each of its children and then forward it up the tree to the root. When the root receives the report COMPLETE message from all its children then the tree is completed.

Theorem 3: The size of any CDS in a UDG $G = (V, E)$ generated by the above algorithm is at most $8 \cdot opt + 1$.

Proof: The black nodes can be classified into two types: Those which are marked black from white, and those which are first marked gray from white and then remarked black from gray. Let k be the number of levels of the BFS tree. For each level $0 \leq \ell \leq k-1$, let S_ℓ denote the black nodes of the first type at level ℓ, and P_ℓ denote the black nodes of the second type at level ℓ. Then S_0 consists only of the leader, and $S_1 = P_0 = P_{k-1} = \emptyset$. In addition, for each $1 \leq \ell \leq k-2$, each node in P_ℓ is the parent of some node in $S_{\ell+1}$,and thus $|P_\ell| \leq |S_{\ell+1}|$. Therefore,

$$\begin{aligned}\left|\bigcup_{\ell=0}^{k-1} P_\ell\right| &= \sum_{\ell=0}^{k-1} |P_\ell| = \sum_{\ell=1}^{k-2} |P_\ell| \\ &\leq \sum_{\ell=1}^{k-2} |S_{\ell+1}| = \sum_{\ell=0}^{k-1} |S_\ell| - 1 = \left|\bigcup_{\ell=0}^{k-1} S_\ell\right| - 1.\end{aligned}$$

On the other hand, all nodes in $\bigcup_{\ell=0}^{k-1} S_\ell$ are independent, and thus from Lemma 1,

$$\left|\bigcup_{\ell=0}^{k-1} S_\ell\right| \leq 4opt + 1.$$

This implies that the total number of black nodes is at most

$$\left|\bigcup_{\ell=0}^{k-1} S_\ell\right| + \left|\bigcup_{\ell=0}^{k-1} P_\ell\right| \leq 2\,(4opt + 1) - 1 = 8opt + 1.$$

Therefore, the approximation ratio of the above algorithm is at most 8. □

Fig. 4 illustrates the algorithm for color marking phase. In the graph, the IDs of the nodes are labelled beside the nodes, and node 0 is the leader elected in the first (leader election) phase. The solid lines represent the edges in the ST tree, and the dashed lines represent all other edges in the UDG. The ordering of the nodes by rank is given by 0, 4, 12, 2, 5, 8, 10, 3, 6, 9, 11, 1, 7. A possible execution scenario is shown in Fig. 4(a)–(f), which is explained below.

1. Node 0 marks itself black and sends out a DOMINATOR message (see Fig. 4(a)).
2. Upon receiving the DOMINATOR message from node 0, nodes 4 and 12 mark themselves gray, and then send out the DOMINATEE messages (see Fig. 4(b)).
3. Upon receiving the DOMINATEE message from node 4, node 2 marks itself black and send out a DOMINATOR message, as all its low-ranked neighbors (node 4 only) have been marked gray; and node 8 has to wait for node 5, since node 5 has a lower rank. Similarly, upon receiving the DOMINATEE message from node 12, node 5 marks it black and sends out a DOMINATOR message; and node 10 has to wait for node 5 (see Fig. 4(c)).
4. Upon receiving the DOMINATOR message from node 2, node 3 marks itself gray and send out a DOMINATEE

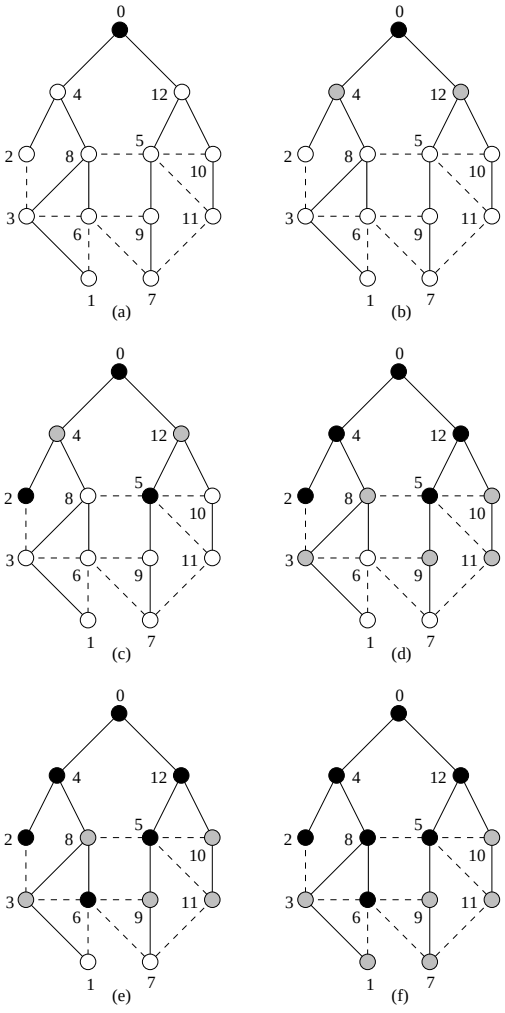

Fig. 4. An example of the algorithm for color marking.

message; and node 4 remarks itself black and sends out a DOMINATOR message. Upon receiving the DOMINATOR message from node 5, nodes 8, 9, 10 and 11 mark themselves gray and send out DOMINATEE messages; node 12 remarks itself black and sends out a DOMINATOR message (see Fig. 4(d)).

5. Upon receiving the DOMINATEE messages from nodes 3 and 8, node 6 marks itself black and sends out a DOMINATOR message, as all its low-ranked neighbors (nodes 3, 8) have been marked gray (see Fig. 4(e)).
6. Upon receiving the DOMINATOR message from node 6, nodes 1 and 7 mark themselves gray and send out DOMINATEE messages; node 8 remarks itself black and sends out a DOMINATOR message (see Fig. 4(f)).

A. Correctness and Performance Analysis

The next theorem proves the correctness of the algorithm, analyzes its performance ratio and its time and message complexity.

Theorem 4: At the end of the third phase, all black nodes form an CDS with size at most $8opt + 1$. In addition, the message complexity of the algorithm is $O(n \log n)$, and the time complexity is $O(n)$.

Proof: Obviously, all black nodes form a DS, as all nodes are either marked gray or black and each gray node is adjacent to at least one black node. To show that all black nodes are connected, it is sufficient to prove that between any black node and the root, there is a "black" path, i.e., a path consisting of only black nodes. We prove it by contradiction. Assume to the contrary and let u_1 be such a black node that is marked black at the *earliest* time. Then u_1 must be of the first type, i.e., u_1 marks itself black from white. Let u_2 be parent of u_1. Then by the time u_1 marks itself black, u_2 is already marked gray. Let u_3 be the black node whose DOMINATOR message causes u_2 to mark itself gray from white. Then u_3 is marked black earlier than u_1. From the selection of u_1, there is a black path from u_3 to the root. On the other hand, u_2 will eventually mark itself black, upon receiving the DOMINATOR message either from u_1 or some other child which has never sent a DOMINATEE message previously. By concatenating the path $u_1u_2u_3$ and the black path from u_3 to the root, we obtain a black path from u_1 to the root, which is a contradiction. To prove the performance ratio follow the proof of theorem 3. Now we count the total number of messages. The message complexity of the first phase is $O(n \log n)$. The message complexity of the second phase is $O(n)$. The message complexity of the third phase is also $O(n)$, as each gray node or black node of the first type sends exactly one message and each black node of the second type sends two messages. Thus the total message complexity of the algorithm is $O(n \log n)$. The time complexity for the first phase is $O(n)$ [17]. It is obvious also that the time complexity for the second and the third phase is $O(n)$. □

Table 1. Performance comparison.

	[2]–[4]	[5]	[10]	This paper
Approx. factor	$O(\log n)$	$O(n)$	$O(n)$	$8-12$
Msg. complexity	$O(n^2)$	$O(n^2)$	$O(n)$–$O(n^2)$	$O(n \log n)$
Time complexity	$O(n^2)$	$O(\Delta^3)$	$O(n)$–$O(n^2)$	$O(n)$
Ngh. knowledge	two-hop	two-hop	single-hop	single-hop

VI. CONCLUSION

In this paper, we investigated three known distributed approximation algorithms for MCDS. And then we presented our own algorithms. In the ID-Based algorithm with approximation factor of 12, each node only maintains knowledge about its own ID and the IDs of all its neighbors. In the Level-Based algorithm with approximation factor of 8, each node maintains knowledge about its own ID and level, and the IDs and levels of all its neighbors. The performance comparison of these algorithms is listed in Table 1. From this table, we can conclude that our algorithms outperform the existing algorithms.

Finally, we appreciate the valuable comments from the reviewers.

REFERENCES

[1] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, 20(4), pp. 374–387, Apr. 1998.

[2] V. Bharghavan and B. Das, "Routing in ad hoc networks using minimum connected dominating sets," in *Proc. Int. Conf. Commun.'97*, Montreal, Canada. June 1997.
[3] B. Das, R. Sivakumar, and V. Bhargavan, "Routing in ad-hoc networks using a spine," in *Proc. Int. Conf. Comput. and Commun. Networks*, Las Vegas, NV., Sept. 1997.
[4] R. Sivakumar, B. Das, and V. Bharghavan, "An improved spine-based infrastructure for routing in ad hoc networks," in *Proc. IEEE Symp. Comput. and Commun.*, Athens, Greece, June 1998.
[5] J. Wu and H.L. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proc. The 3rd ACM Int. Workshop Disc. Algor. and Methods for Mobile Computing and Commun.*, 1999, pp. 7–14.
[6] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Disc. Math.*, 86, pp. 165–177, 1990.
[7] U. Feige, "A threshold of $\ln n$ for approximating set-cover," in *Proc. 28th ACM Symp. Theory Comput.* , 1996, pp. 314–318.
[8] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems," *J. the ACM*, 41(5), pp. 960–981, 1994.
[9] M. V. Marathe *et al.*, "Simple heuristics for unit disk graphs," *Networks*, vol. 25, pp. 59–68, 1995.
[10] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," in *Proc. IEEE Hawaii Int. Conf. System Sciences*, Jan. 2001.
[11] M. Gerla and J. Tsai, "Multicluster, mobile, multimedia radio network," *ACM-Baltzer J. Wireless Networks*, vol.1, no.3, pp.255–265, 1995.
[12] C.R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE J. Select. Areas Commun.*, vol. 15, no. 7, pp. 1265–1275, Sept. 1997.
[13] G. Chen and I. Stojmenovic, "Clustering and routing in wireless ad hoc networks," *Technical Report TR-99-05*, Computer Science, SITE, University of Ottawa, June 1999.
[14] H. B. Hunt III *et al.*, "NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs," *J. Algorithms*, 26(2), pp. 238–274, 1998.
[15] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *J. the ACM*, 32(1), pp. 130–136, 1985.
[16] B. S. Baker, "Approximation algorithms for NP-complete problems on planar graphs," *J. the ACM*, 41(1), pp. 153–180, 1994.
[17] I. Cidon and O. Mokryn, "Propagation and leader election in multihop broadcast environment," in *Proc. 12th Int. Symp. Distr. Computing*, Greece, Sept. 1998, pp.104–119.
[18] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "New distributed algorithm for connected dominating set in wireless ad hoc networks," *HICSS35*, Hawaii, Jan. 2002.

Khaled M. Alzoubi is a Ph.D. candidate in Computer Science at Illinois Institute of Technology since 1998. He received his MS in Computer Science from Northeastern Illinois University in 1991, and BS in Applied Mathematics and Computer Science from University of Illinois in 1987. His research interests include Distributed Algorithms and wireless networks.

Peng-Jun Wan has been an Assistant Professor in Computer Science at Illinois Institute of Technology since 1997. He received his PhD in Computer Science from University of Minnesota in 1997, MS in Operations Research and Control Theory from Chinese Academy of Science in 1993, and BS in Applied Mathematics from Qsinghua University in 1990. His research interests include optical networks and wireless networks

Ophir Frieder is the IITRI Professor of Computer Science at the Illinois Institute of Technology. His research interests span the general area of distributed information systems. He is a Member of ACM and a Fellow of the IEEE.