Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks

Peng-Jun Wan* Khaled M. Alzoubi* Ophir Frieder*

Abstract—**Connected dominating set (CDS) has been proposed as virtual backbone or spine of wireless ad hoc networks. Three distributed approximation algorithms have been proposed in the literature for minimum CDS. In this paper, we first reinvestigate their performances. None of these algorithms have constant approximation factors. Thus these algorithms can not guarantee to generate a CDS of small size. Their message complexities can be as high as $O(n^2)$, and their time complexities may also be as large as $O(n^2)$ and $O(n^3)$. We then present our own distributed algorithm that outperforms the existing algorithms. This algorithm has an approximation factor of at most 8, $O(n)$ time complexity and $O(n \log n)$ message complexity. By establishing the $\Omega(n \log n)$ lower bound on the message complexity of any distributed algorithm for nontrivial CDS, our algorithm is thus message-optimal.**

I. Introduction

Wireless ad hoc networks can be flexibly and quickly deployed for many applications such as automated battlefield, 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. The topology of such wireless ad hoc network can be modeled as a *unit-disk graph* [6], a geometric graph in which there is an edge between two nodes if and only if their distance is at most one (see Figure 1).

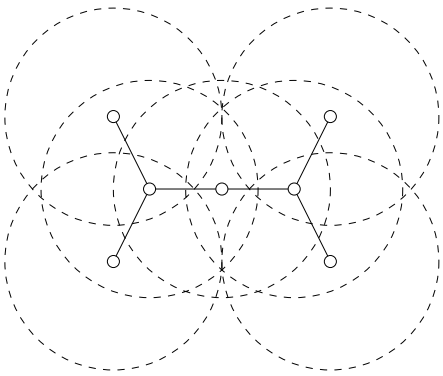

Fig. 1. Model the topology of wireless ad hoc networks by unit-disk graphs.

Although a wireless ad hoc network has no *physical* backbone infrastructure, a *virtual* backbone can be formed by nodes in a connected dominating set of the corresponding unit-disk graph [1][7][10]. Such virtual backbone, also referred to as *spine*, plays a very important role in routing, broadcasting and connectivity management in wireless ad hoc networks [1]. 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 which also induces a connected subgraph. A (connected) dominating set of a wireless ad hoc network is a (connected) dominating set of the corresponding unit-disk graph. To simplify the connectivity management, it is desirable to find a minimum connected dominating set (MCDS) of a given set of nodes. However, finding an MCDS in unit-disk graphs is NP-hard [6], and thus only distributed approximation algorithms in polynomial time are practical for wireless ad hoc networks. In this paper, we further show that any distributed algorithm for *nontrivial* CDS requires at least $O(n \log n)$ messages, where n is the number of nodes and the message length has the same order of the number of bits representing the node IDs.

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 and computation costs should be scalable as the wireless ad hoc networks are typically deployed with large network size. In this paper, we first reinvestigate the performance of the three known distributed approximation algorithms for MCDS, proposed by Das et al in [1][7][10], by Wu and Li in [12] and by Stojmenovic et al in [11], respectively. After that we propose a new distributed algorithm. We show that our algorithm outperforms significantly the existing algorithms. A remark is that this paper focuses on the generation of a CDS. The maintenance of CDS is not discussed in this paper.

II. Lower Bound on Message Complexity

In this section, we establish the $\Omega(n \log n)$ lower bound on the message complexity for distributed algorithms for leader election, spanning tree and nontrivial CDS in wireless ad hoc networks. The reduction is made from the following well-known bound on the message complexity of distributed leader election in asynchronous ring networks with point-to-point transmission.

Theorem 1: [2] In asynchronous rings with point-to-point transmission, any distributed algorithm for leader election in sends at least $\Omega(n \log n)$ messages.

*Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. Email: {wan, alzoubi, ophir }@cs.iit.edu.

Theorem 2: In asynchronous wireless ad hoc networks whose unit-disk graph is a ring, any distributed algorithm for leader election sends at least $\Omega(n \log n)$ messages.

Proof: Let $\mathcal{A}$ be any distributed algorithm for leader election in wireless ad hoc networks whose unit-disk graph is a ring. Let $\mathcal{A}^*$ be the algorithm by replacing each wireless transmission by two point-to-point transmissions. Then $\mathcal{A}^*$ is a distributed algorithm for leader election in asynchronous rings with point-to-point transmission. Note that the algorithm $\mathcal{A}^*$ sends twice messages of that sent by $\mathcal{A}$. Thus from Theorem 1, $\mathcal{A}$ must also send at least $\Omega(n \log n)$ messages. ■

Theorem 3: In asynchronous wireless ad hoc networks whose unit-disk graph is a ring, any distributed algorithm for spanning tree sends at least $\Omega(n \log n)$ messages.

Proof: Let $\mathcal{A}$ be any distributed algorithm for spanning tree in wireless ad hoc networks whose unit-disk graph is a ring. Note that any spanning tree of a ring consists of all edges in the ring except one. Thus it has exactly two leaves which are also neighbors. Thus after an spanning tree is completed, the two leaves can exchange a message to select the leader between them according to some symmetry-breaking criterion, for example by their IDs. After the leader is identified, it then notifies all other nodes in linear number of message. Thus from algorithm $\mathcal{A}$, we can derive a distributed algorithm for leader election whose message complexity is $\Theta(n)$ more than the number of messages sent by $\mathcal{A}$. From Theorem 2, the message complexity of $\mathcal{A}$ is at least $\Omega(n \log n)$. ■

A distributed algorithm for leader election in wireless ad hoc networks has been proposed in [5]. This algorithm has message complexity $O(n \log n)$ and therefore is message-efficient. Its actual implementation also constructs a spanning tree rooted at the leader.

Theorem 4: In asynchronous wireless ad hoc networks whose unit-disk graph is a ring, any distributed algorithm for nontrivial CDS sends at least $\Omega(n \log n)$ messages.

Proof: Let $\mathcal{A}$ be any distributed algorithm for CDS in wireless ad hoc networks whose unit-disk graph is a ring. Note that for any nontrivial CDS of a ring consists of all nodes except either a unique node or two neighboring nodes. So after an nontrivial CDS is completed, the leader can be elected as follows. A dominatee declares itself as the leader if both its neighbors are dominators, or one of its neighbor is a dominatee but has larger ID. The leader then notifies all other nodes in linear number of message. Thus from algorithm $\mathcal{A}$, we can derive a distributed algorithm for leader election whose message complexity is $\Theta(n)$ more than the number of messages sent by $\mathcal{A}$. From Theorem 2, the message complexity of $\mathcal{A}$ is at least $\Omega(n \log n)$. ■

III. Das et al's Algorithm

The centralized version of the distributed algorithm proposed by Das et al consists of three stages. The first stage finds an approximation to Minimum Dominating Set, which is essentially the well-studied Set Cover problem. Not surprisingly, the heuristic proposed by das et al in [1][7][10] is a translation of Chvatal's greedy algorithm [4] for Set Cover, and thus guarantees an approximation factor of $H(\Delta)$, where Δ is the maximum degree and H is the harmonic function. Let U denote the dominatg set output in this stage. The second stage constructs a spanning forest F. Each tree component in F is a union of stars centered at the nodes in U. The stars are generated by letting each dominatee node pick up an arbitrary neightbor in U. The thrid stage expands the spanning forest F to a spanning tree T . All internal nodes in T form a CDS. It is easy to show that the CDS generated in this way contains at most $3|U|$ nodes, and therefore is a $3H(\Delta)$-approximation of MCDS.

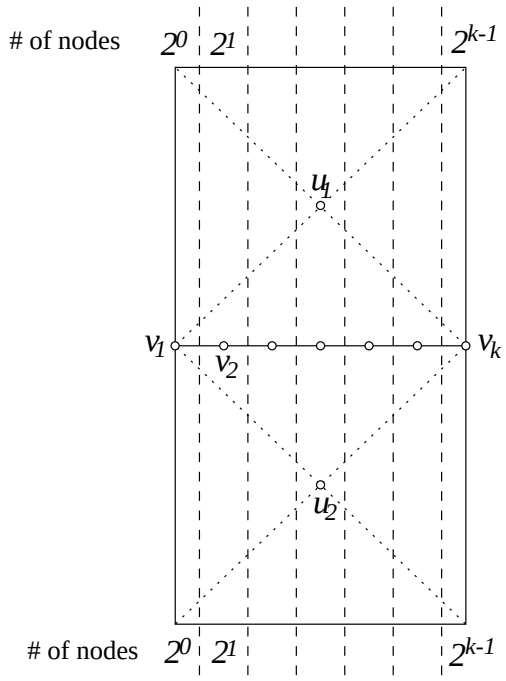

Fig. 2. Instance for which the size of the solution computed by the greedy algorithm, $\{v_1, v_2, \cdots, v_k\}$, is larger than the optimum solution, $\{u_1, u_2\}$, by a logarithm factor.

Figure 2 shows a family of instances for which the size of the solution computed by the above greedy algorithm is larger than the optimum solution by a logarithm factor. All points lie in a rectangle whose horizontal side has length one and whose vertical side has length $2\sqrt{1-\left(\frac{1}{2(k-1)}\right)^2}$. The two nodes $v_1$and v_k are the centers of the left and right vertical sides respectively. The $k-2$ nodes $v_2, v_3, \cdots, v_{k-1}$ are evenly distributed within the line segment between $v_1$and v_k from left to right. The two nodes $u_1$and u_2 are the centers of the two sub-rectangles above and below the segment between $v_1$and v_k respectively. The rest points lie in the two horizontal sides. In each horizontal side,

$2^0 = 1$ node lies to the left of (and excluding) the perpendicular bisector of v_1v_2, 2^{k-1} nodes lie to the right of (and excluding) the perpendicular bisector of $v_{k-1}v_k$, and 2^{i-1} nodes lie between (and excluding) the perpendicular bisector of $v_{i-1}v_i$ and the perpendicular bisector of v_iv_{i+1}. Thus, the total number of nodes is

$$n = k + 2 + 2\sum_{i=1}^{k} 2^{i-1} = k + 2^{k+1}.$$

Note that u_1 is adjacent to all nodes lying in the top subrectangle, u_2 is adjacent to all nodes lying in the bottom subrectangle, and they are adjacent to each other. Thus, $\{u_1, u_2\}$ forms an MCDS. On the other hand, the above greedy algorithm would add $v_k, v_{k-1}, \cdots, v_1$ sequentially to the dominating set in the first stage and output the set $\{v_1, v_2, \cdots, v_k\}$ as the CDS at the end of the second stage. This can be proven by induction as follows.

Initially, the degree of node v_i is

$$2 \cdot 2^{i-1} + (k-1) + 2 = 2^i + k + 1;$$

the degrees of the node u_1 and u_2 are both

$$\sum_{i=1}^{k} 2^{i-1} + k + 1 = 2^k + k;$$

and the degree of any other node is

$$\sum_{i=1}^{k} 2^{i-1} - 1 + 1 + 1 = 2^k.$$

So v_k is the first node to be selected. Now we assume that the nodes $v_k, v_{k-1}, \cdots, v_j$ have been added to the dominating set. For any node v_i with $i < j$, the number of its neighbors that have not been dominated yet is $2 \cdot 2^{i-1} = 2^i$; for the node u_1 or u_2, the number of its neighbors that have not been dominated yet is

$$\sum_{i=1}^{j-1} 2^{i-1} = 2^{j-1} - 1;$$

and for any other rest node, the number of its neighbors that have not been dominated yet is

$$\sum_{i=1}^{j-1} 2^{i-1} - 1 = 2^{j-1} - 2.$$

So the node v_{j-1} is then added to the dominating set. Therefore, by induction, the nodes $v_k, v_{k-1}, \cdots, v_1$ are added sequentially to the dominating set. Note that $\{v_1, v_2, \cdots, v_k\}$ is a CDS. The first stage will stop after v_1 is added, and the second stage would add no more nodes.

Since $n = k + 2^{k+1}$ and $\Delta = 2^k + k + 1$, we have $k > \log n - 2$ and $k > \log \Delta - 1$. Therefore, the instance shown in Figure 2 implies the lower bounds $\frac{\log n}{2} - 1$ and $\frac{\log \Delta}{2} - \frac{1}{2}$ on the approximation factor of the greed algorithm.

The distributed implementation of the above greedy algorithm proposed in [1][7][10] has very high time complexity and message complexity. Indeed, both time complexity and message complexity can be as high as $\Theta\left(n^2\right)$. We also notice that such distributed implementation is technically incomplete. For example, the distributed implementation consists of multiple stages, but the implementation lacks lack mechanisms to bridge two consecutive stages. Thus, individual nodes have no way to tell when the next stage should begin. While these technical incompleteness are possibly to be fixed, we will not take such effort here as the approximation factor of the greedy algorithm is intrinsically poor.

In summary, we have the following performance results of the distributed algorithm in [1][7][10].

Theorem 5: The approximation factor of the distributed algorithm proposed by Das et al in [1][7][10] is between $\frac{\log \Delta}{2} - \frac{1}{2}$ and $3H\left(\Delta\right)$. Both its message complexity and time complexity are $O\left(n^2\right)$.

IV. Wu and Li's Algorithm

While the algorithm proposed by Das et al first finds a DS and then grow this DS into a CDS, the algorithm proposed by Wu and Li in [12] takes an opposite approach. The algorithm in [12] first finds a CDS and then prune certain redundant nodes from the CDS. The initial CDS U consists of all nodes which have at least two non-adjacent neighbors. 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 dominates all other neighbors of u. The algorithm then removes all locally redundant nodes from U. This algorithm applies only to wireless ad hoc networks whose unit-disk graph is not a complete graph. As indicated in [12], the approximation factor of this algorithm remains unspecified. Obviously, the MCDS of any wireless ad hoc network whose unit-disk graph is not complete graph consists of at least two nodes. On the other hand, any CDS contains at most n nodes. Thus, the approximation factor of the above algorithm is at most $\frac{n}{2}$ where n is the number of nodes. Next, we show that the approximation factor of the above algorithm is exactly $\frac{n}{2}$. This means that the above algorithm does perform extremely poorly over certain instances.

When n is even, we consider the instance illustrated in Figure 3(a). These nodes are evenly distributed over the two horizontal sides of a unit-square. Each node has exactly m neighbors, one in the opposite horizontal side and the rest in 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 [12] 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 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. So no node is locally redundant. Consequently the output CDS still consists of all nodes.

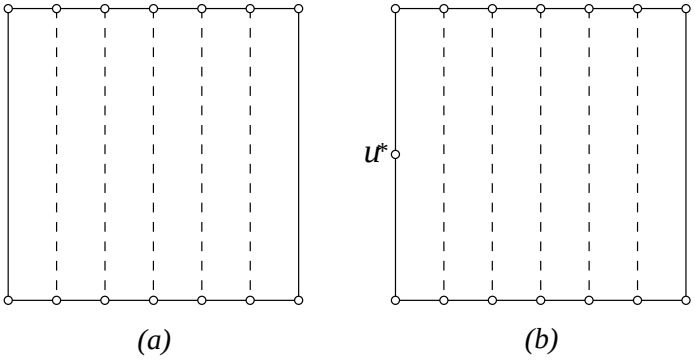

Fig. 3. Instance for which the CDS computed by Wu and Li's algorithm consists of all nodes but the MCDS consists of only two nodes.

When n is odd, we consider the instance illustrated in Figure 3(b). The node with the largest ID, denoted by u^*, is the center of the left vertical side of a unit-square, and all other $n-1$ nodes are evenly distributed over the two horizontal sides of the unit-square. The two nodes at the left two corners of the unit-square forms an MCDS. On the other hand, the CDS output by the algorithm in [12] also consists of all nodes. In fact, following the same argument as in the even case, all nodes other than u^* are in the initial CDS U. The node u^* is also in the initial CDS U as well. Since u^* is not adjacent to the two nodes at the right corners of the unit-square, all nodes other than u^* are not locally redundant. The u^* itself is also not locally redundant as it has the maximum ID. Therefore, the output CDS still consists of all nodes.

The distributed implementation of the above algorithm given in [12] runs in two phases. In the first phase, each node first broadcasts to its neighbors the *entire* set of its neighbors, 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. In the second phase, a dominator declares itself as a dominatee if it is locally redundant. Note a dominator can find whether it is locally redundant from the adjacency information of all its neighbors. It is claimed in [12] 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 unit-disk graph, as each edge contributes two messages in the first phase. The $O(\Delta^2)$ time complexity, however, 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 $O(\Delta^3)$, instead of $O(\Delta^2)$. Note that m and Δ can be as many as $O(n^2)$ and $O(n)$ respectively. Thus, the message complexity and the time complexity of the distributed algorithm in [12] are $O(n^2)$ and $O(n^3)$ respectively. The instances shown in Figure 3 do require such complexities.

In summary, we have the following performance results of the distributed algorithm in [12].

Theorem 6: The approximation factor of the distributed algorithm proposed by Wu and Li in [12] is exactly $\frac{n}{2}$. Its message complexity is $\Theta(m)$ and its time complexity is $O(\Delta^3)$.

V. Stojmenovic et al's Algorithm

In the context of clustering and broadcasting, Stojmenovic et al [11] presented a distributed construction of CDS. The CDS consists of two types of nodes: the cluster-heads and the border-nodes. The cluster-heads form a maximal independent set (MIS), i.e., a dominating set in which any pair nodes are non-adjacent. Several algorithms for MIS were described in [11], which can be generalized to the following framework:

- Each node has a unique *rank* parameter such as the ID only [8][9], an ordered pair of degree and ID [3], an order pair of degree and location [11]. 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. Then all cluster-heads and all border-nodes form a CDS.

Regardless of the choice of the rank, the algorithm in [11] have an $\Theta(n)$approximation factor. Such inefficiency stems from the non-selective inclusion of all border-nodes. In fact, if the rank is ID only, Figure 4 shows a family of instances which would imply the approximation factor to be exactly n. In these instances, the node with the largest ID is located at the center of a unit-disk and all other nodes are evenly distributed in the boundary of the unit-disk. After the cluster-heads are selected, all other nodes become border-nodes. Thus the CDS would consist of all nodes. On the other hand, the node at the center dominates all other nodes. If the rank is an ordered pair of degree and ID or an order pair of degree and location, the instances shown in Figure 3 imply that their approximation factors are at least $\frac{n}{2}$. If the rank is ID only, it is easy to construct an instance to show that the approximation factor is exactly n, the worst possible.

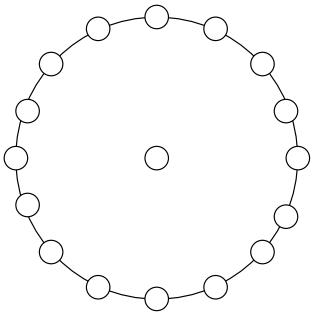

Fig. 4. Instance for which the CDS output by Stojmenovic et al's algorithm consists of all nodes but the MCDS consists of only one node.

The implementation cost of these algorithms given in [11] 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 $\Theta(n)$. This does not contradict to Theorem 4, as it may output a trivial CDS. 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 [11] didn't provide these implementation details. But at least $O(n^2)$ messages and time may be required for rank updating and synchronization.

In summary, we have the following performance results of the distributed algorithm in [11].

Theorem 7: If the rank is ID only, the distributed algorithm proposed by Stojmenovic et al in [11] has an approximation factor of n and linear message and time complexities. If the rank involves the degree, the distributed algorithm proposed by Stojmenovic et al in [11] has an approximation factor of $\frac{n}{2}$ and $O(n^2)$ message and time complexities.

VI. A Better Distributed Algorithm

A. Algorithm Description

Our distributed algorithm for CDS consists of three phases: the Leader Election Phase, the Level Calculation Phase, and the Color Marking Phase. The Leader Election Phase elects a leader v and constructs a spanning tree T rooted at the leader. The distributed algorithm in [5] for leader election can be adopted. Note that any criteria can be used to define the leadership, such as ID or the combination of degree and ID. This algorithm has $O(n)$ time complexity and $O(n \log n)$ message complexity. At the end of the first phase, each node knows its parent and its children in T.

In the Level Calculation Phase, each node identifies its level in T. It starts with the root announcing its level 0. Each 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 announce this own level. Each node also records the levels of its neighbors in the unit-disk graph. If we need to report the completion of the tree, a report process has to be performed upwards along the T. When a leaf node has determined its level, it transmits a COMPLETE message to its parent. Each internal node will wait till it receives this COMPLETE message from each of its children and then forward it up the tree toward the root. When the root receives the COMPLETE message from all its children, then it starts the third phase. Obviously, the total number of messages sent in this phase is $O(n)$. At this moment, each node knows the levels and IDs of its own and 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 contains the sender's ID. The third phase is initiated by the root which 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 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. It will then choose its parent in T as its own dominator.
- When a gray node receives a DOMINATOR message for the *first* time from some *child* in T which has never sent a DOMINATEE message, it *remarks* itself black and broadcasts the DOMINATOR message.
- If a black node has rank higher than all its neighbors and all its neighbors are all black, it *remarks* itself gray and broadcasts the DOMINATOR message.

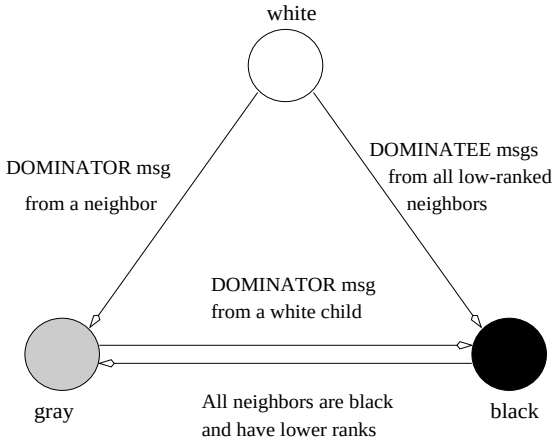

Fig. 5. The state transition diagram of the Color Marking Phase.

Figure 5 shows the state transition diagram of this phase. Eventually each node will be either black (a dominator) or gray

(dominatee). A reporting process as in the second phase, if necessary, can be performed to notify the root of the completion.

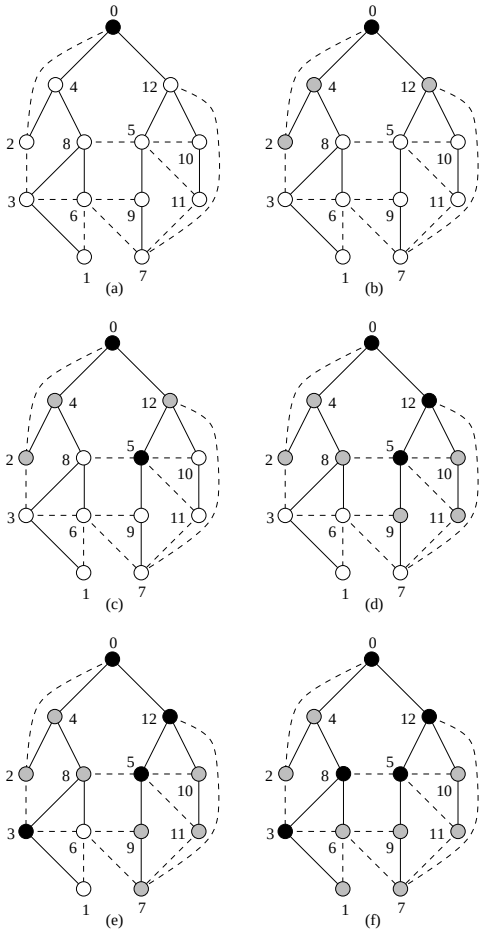

Fig. 6. An example of the algorithm for color marking.

Figure 6 illustrates the algorithm for color marking in this phase. In the graph, the IDs of the nodes are labelled beside the nodes, and node 0 is the leader elected in the first phase. The solid lines represent the edges in the spanning tree output at the first phase, and the dashed lines represents other edges in the unit-disk graph. 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 Figure 6(a)–(f), which is explained below.
1. Node 0 marks itself black and sends out a DOMINATOR message (see Figure 6(a)).
2. Upon receiving the DOMINATOR message from node 0, nodes 2, 4 and12 mark themselves gray, and then send out the DOMINATEE messages (see Figure 6(b)).
3. Upon receiving the DOMINATEE message from node 2, node 3 has to wait for node 8. Upon receiving the DOMINATEE message from node 4, node 8 has to wait for node 5. Upon receiving the DOMINATEE message from node 12, node 5 marks it black as all its low-ranked neighbors (node 12 only) have been marked gray; node 7 has to wait for nodes 6, 9, 11; and node 10 has to wait for node 5 (see Figure 6(c)).
4. 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 Figure 6(d)).
5. Upon receiving the DOMINATOR message from node 12, node 7 marks itself gray and sends out a DOMINATEE message. Upon receiving the DOMINATEE message from node 8, node 3 marks itself black as all its low-ranked neighbors (nodes 2, 8) have been marked gray; node 6 has to wait for node 3 (see Figure 6(e)).
6. Upon receiving the DOMINATOR message from node 3, nodes 1 and 6 mark themselves gray and send out DOMINATEE messages; node 8 remark itself black and send out a DOMINATOR message (see Figure 6(f)).

Note that at the end of the last step, node 4 will receive the DOMINATOR message from node 8. But it will not remark its color from gray to black as node 8 has sent a DOMINATEE message previously.

B. Correctness

We distinguish the black nodes into two types. A black node is of the first type if it is marked black from white, and is of the second type if it is first marked gray from white and then remarked black from gray.

Theorem 8: At the end of the third phase, all black nodes form a nontrivial CDS.

Proof: Since all nodes are either marked gray or black and each gray node is adjacent to at least one black node, all black nodes form a DS. In order 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. Let u_1 be the black node that is marked black at the *earliest* time among those black nodes which have no black path from the root. Then u_1 must be of the first type, i.e., u_1 marks itself black from white. Let u_2 be the 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. This contradicts to the assumption that u_1 has no black path from the root.

It is obvious that the CDS is nontrivial. ■

C. Performance Analysis

Since all three phases have $O(n)$ time complexity, the overall time complexity of our distributed algorithm is $O(n)$. 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 our algorithm is $O(n \log n)$.

Next, we bound the number of black nodes in terms of the size of an MCDS, denoted by opt. A set of nodes are said to be *independent* if they are pairwise non-adjacent. 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 unit-disk graph 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 9: The size of any independent set in a unit-disk graph $G = (V, E)$ is at most $4 \cdot opt + 1$.

Proof: Let U be any independent set of V, and let T^* be any spanning tree of an MCDS. Consider an arbitrary preorder traversal of T^* given by $v_1, v_2, \cdots, v_{opt}$. 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. As v_1 can be adjacent to at most five independent nodes, $|U_1| \leq 5$. 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 Figure 7). 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. ■

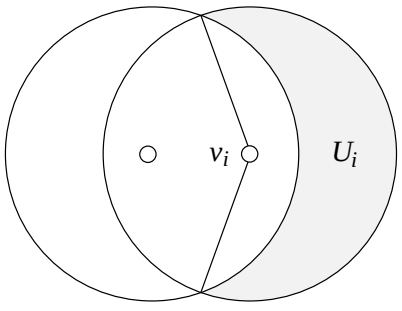

Fig. 7. U_i lie in a sector of at most 240 degree within the coverage range of node v_i.

Let k be the number of levels of the spanning tree T constructed in the first phase. 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,

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

On the other hand, all nodes in $\bigcup_{\ell=0}^{k-1} S_\ell$ are independent, and thus from Lemma 9,

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

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

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

In summary, we have the following performance results of the distributed algorithm in [12].

Theorem 10: Our distributed algorithm has an approximation factor of at most 8, $O(n)$ time complexity, and $O(n \log n)$ message complexity.

	[1][7][10]	[12]	[11]	This paper
Approx. factor	$O(\log n)$	$O(n)$	$O(n)$	≤ 8
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

TABLE I
PERFORMANCE COMPARISON.

VII. CONCLUSION

In this paper, we have established a $\Omega(n \log n)$ lower bound on message complexity of any distributed algorithm for non-trivial CDS. We then reinvestigated three known distributed approximation algorithms for MCDS. After that we presented our own algorithm. The performance comparison of these four algorithms is listed in Table I. From this table, we can conclude that our algorithm outperforms the existing algorithms.

REFERENCES

[1] V. Bharghavan and B. Das, "Routing in Ad Hoc Networks Using Minimum Connected Dominating Sets", *International Conference on Communications'97*, Montreal, Canada. June 1997.

[2] J. Burns, "A Formal Model for Message Passing Systems", *Technical Report TR-91*, Computer Sceince Department, Indiana University, May 1980.

[3] 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.

[4] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operation Research*, 4(3):233–235, 1979.

[5] I. Cidon and O. Mokryn, “Propagation and Leader Election in Multihop Broadcast Environment”, *12th International Symposium on DIStributed Computing (DISC98)*, September 1998, Greece. pp.104–119.

[6] B. N. Clark, C. J. Colbourn, and D. S. Johnson, “Unit Disk Graphs”, *Discrete Mathematics*, 86:165–177, 1990.

[7] B. Das, R. Sivakumar, and V. Bhargavan, “Routing in Ad-Hoc Networks Using a Spine”, *International Conference on Computers and Communications Networks ’97*, Las Vegas, NV. September 1997.

[8] M. Gerla, and J. Tsai, “Multicluster, mobile, multimedia radio network”, *ACM-Baltzer Journal of Wireless Networks*, Vol.1, No.3, pp.255-265(1995).

[9] C.R. Lin and M. Gerla, “Adaptive Clustering for Mobile Wireless Networks”, *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 7, Sept. 1997, pp. 1265-1275

[10] R. Sivakumar, B. Das, and V. Bharghavan, “An Improved Spine-based Infrastructure for Routing in Ad Hoc Networks”, *IEEE Symposium on Computers and Communications ’98*, Athens, Greece. June 1998.

[11] I. Stojmenovic, M. Seddigh, J. Zunic, “Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks”, *Proc. IEEE Hawaii Int. Conf. on System Sciences*, January 2001.

[12] J. Wu and H.L. Li, “On calculating connected dominating set for efficient routing in ad hoc wireless networks”, *Proceedings of the 3rd ACM international workshop on Discrete algorithms and methods for mobile computing and communications*, 1999, Pages 7–14.