New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks

Khaled M. Alzoubi Peng-Jun Wan Ophir Frieder
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
Email: {alzoubi, wan, ophir}@cs.iit.edu

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.**

Keywords: wireless ad hoc networks, distributed algorithm, 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, 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).

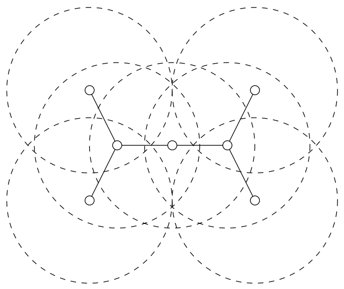

Fig. 1. Model the topology of wireless ad hoc networks by unit-disk graphs.

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][11]. 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.

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 two 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. The dominating set is initially empty. The greedy algorithm iteratively adds to U a node adjacent to the maximum number of nodes not yet dominated by U, and terminates when U become a dominating set. The second stage assigns each edge in the unit-disk graph G with a weight equal to the number of endpoints not in U, and then finds a minimum spanning tree T in the resultant weighted graph. All internal nodes of T then 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. Note that the weighted graph described here is slightly different from the one defined in [1][7][10]. The weighted graph defined in [1][7][10] excludes all edges whose endpoints are both in U. Such weighted graph may not be connected and thus does not have a spanning tree. Such bug is fixed in the weighted graph described here. Next, we show that the approximation factor of the above greedy algorithm is $\Omega(H(\Delta))$.

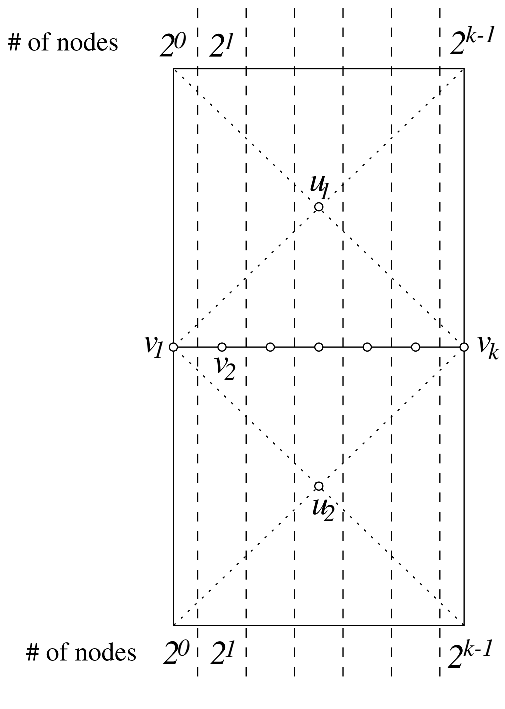

Fig. 2. Instance for which the size of the solution computed by the greedy algorithm, $\{v_1, v_2, \cdots, v_k\}$, is larger than the optimum solution, $\{u_1, u_2\}$, by a logarithm factor.

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 sub-rectangle, u_2 is adjacent to all nodes lying in the bottom sub-rectangle, 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 [10].

Theorem 5: The approximation factor of the distributed algorithm proposed by Das et al in [10][1][7][10] is between $\frac{\log \Delta}{2} - \frac{1}{2}$ and $3H(\Delta)$. 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 remove 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.

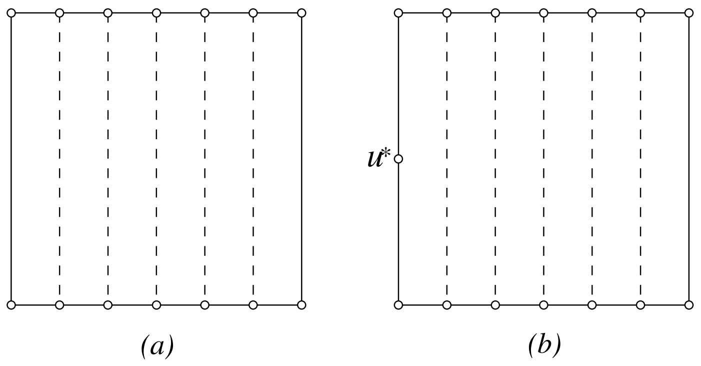

Fig. 3. Instance for which the CDS computed by Wu and Li's algorithm consists of all nodes but the MCDS consists of only two nodes.

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 $\Theta(n)$ 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.

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 trivi

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 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 [11] have an $\Theta\left(n\right)$approximation factor. In fact, 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. Such inefficiency stems from the non-selective inclusion of all border-nodes.

VI. Our Algorithm

Our distributed algorithm for CDS consists of two phases. These two phases construct an maximal independent set (MIS), and a dominating tree, respectively. They are described and analyzed in the next three subsections.

A. MIS Construction

By definition, any pair of nodes in an MIS are separated by at least at two hops. However, a subset of nodes in an MIS may be three hops away from the subset of the rest nodes in this MIS. The MIS constructed in this section guarantees that the distance between any pair of complementary subsets is *exactly* two hops. Our construction also follows the general framework described in the previous section, but with a carefully chosen rank definition.

To define the rank, we first apply the distributed leader election algorithm in [5], $O\left(n\right)$ 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 announce this 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 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. An MIS can then be constructed in the following simple way:

- Initially, each 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 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.
- When a leaf node is marked, it transmits a 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.

Let U be the set of black nodes. We show that it has the following property.

Theorem 7: 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 in 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. ■

Obviously, this phase has $O\left(n\right)$ time complexity and $O\left(n\log n\right)$ message complexity. 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 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 8: 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 4). 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. ■

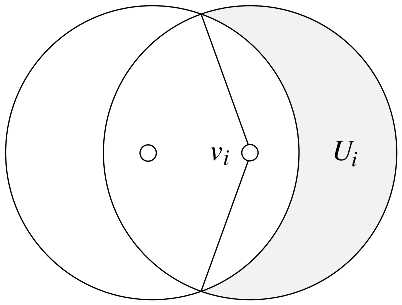

Fig. 4. U_i lie in a sector of at most 240 degree within the coverage range of node v_i.

B. Dominating Tree Construction

The second phase constructs a tree spanning all the black nodes, referred to as *dominating tree*. All nodes in this dominating tree form a CDS. The dominating tree is initially empty . The root joins the dominating tree first. When each black node joins the dominating tree, it sends an invitation to all black nodes that are two hops away and outside the current dominating tree to join the dominating tree. This invitation will be relayed through the gray nodes. Each black node will join the tree when it receives the invitation for the first time together with the gray node which relays the invitation to itself. This process should be repeated until all black nodes are in the tree. The next is the implementation detail:

- The root sends an INVITE message.
- When a gray node receives for the *first* time an INVITE message from a black neighbor, it stores the ID of this black neighbor in its local variable $inviter$, and then relays such INVITE message.
- When a black node receives for the *first* time an INVITE message from a gray neighbor, it puts this gray neighbor as its parent in the dominating tree, then sends back a JOIN message towards this gray neighbor and finally initiated an INVITE message.
- Whenever a gray node receives a JOIN message towards itself from a black neighbor, it puts this black neighbor as its child. In addition, upon the receiving of the *first* JOIN message towards itself, it sends a JOIN message towards the black neighbor whose ID is stored in the local variable $inviter$.
- Whenever a black node receives a JOIN message towards itself from a gray neighbor, it puts this gray neighbor as its child.

Theorem 7 guarantees that whenever there is any black node outside the current dominating tree, at least one black node would join the dominating tree. Thus eventually all black nodes will join the dominating tree. A reporting process described as follows, if necessary, can be performed along the spanning tree T to notify the root of the completion. A gray node reports a COMPLETE message to its parent in the spanning tree if all 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 all has received a COMPLETE message from each child in the spanning tree and itself has joined the dominating tree.

Note that each black node initiates one INVITE message and one JOIN message (except the root); each gray node relays one INVITE message and at most one JOIN message. So the construction of the dominating tree requires $O(n)$ messages and $O(n)$ time. The same is true for the optional reporting process. Therefore, the total message complexity and time complexity of our algorithm are $O(n \log n)$ and $O(n)$ respectively.

Finally, we bound the size of the dominating tree. Since each gray node appearing in the dominating tree is the parent of at least one black node, the total number of gray nodes in the dominating tree is at most one less than the number of black nodes. From Lemma 8, the total number of nodes in the dominating tree is at most

$$2\,(4opt + 1) - 1 = 8opt + 1.$$

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

Theorem 9: Our distributed algorithm has an approximation factor of at most 8, $O(n)$ time complexity, and $O(n \log n)$ message complexity.

VII. Conclusion

In this paper, we have established a $\Omega(n \log n)$ lower bound on message complexity of any distributed algorithm for nontrivial 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.

	[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.

Finally, we appreciate the valuable comments from the reviewers.

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.