WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2006; **6**:183–190
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.378

Virtual backbone construction in multihop *ad hoc* wireless networks

Xiuzhen Cheng[1]*,†, Min Ding[1], David Hongwei Du[2] and Xiaohua Jia[2]

[1]*Department of Computer Science, The George Washington University, Washington, DC, USA*
[2]*Department of Computer Science, City University of Hong Kong, Hong Kong*

Summary

Recent research points out that the flooding mechanism for topology update or route request in existing *ad hoc* routing protocols greatly degrades the network capacity. If we restrict the broadcast of control packets within a small subset of hosts in the network, the protocol overhead can be substantially reduced. This motivates our research of constructing a *virtual backbone* by computing a *connected dominating set* (*CDS*) in unit-disk graphs. In this paper, we propose two distributed algorithms to approximate a minimum CDS. These algorithms take linear time. Their performance is verified by a complete theoretical analysis. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: multihop ad hoc wireless network; connected dominating set; unit-disk graph

1. Introduction

Ad hoc wireless networks are autonomous systems consisting of (mobile) hosts (as routers) connected by wireless links, and have no fixed infrastructure and no central administration. All routing and network management functions must be performed by ordinary nodes. Due to the features of dynamic topology, multihop communication, and strict resource limitations, *ad hoc* routing becomes the most challenging problem. Almost all existing topology-based routing protocols involve the *global flooding*, which suffers from the notorious *broadcast storm problem* [8]. This causes high protocol overhead and interference to ongoing traffics. Inspired by the *physical backbone* in a wired network, many researchers proposed the concept of *virtual backbone* infrastructure [7], which organizes a hierarchical structure of ordinary nodes to achieve scalability and efficiency.

In this paper, we will study the problem of efficiently constructing a virtual backbone for *ad hoc* wireless networks. We assume that a given *ad hoc* network instance contains n hosts in a two-dimensional plane. Each host is equipped with an omnidirectional antenna that makes the transmission coverage a disk. Each transceiver also has the same communication range R. The footprint of an *ad hoc* wireless network is thus a unit-disk graph. In graph-theoretic terminology, the network topology can be represented as $G = (V, E)$ where V contains all nodes and E is the set of links. A link between nodes u and v exists if and only if their distance is at most R. Furthermore, we require the number of hosts forming the virtual backbone to be minimized, as the virtual

*Correspondence to: Xiuzhen Cheng, Department of Computer Science, The George Washington University, Washington, USA.
†E-mail: cheng@gwu.edu

Contract/grant sponsor: NSF CAREER Award; Contract/grant number: CNS-0347674.
Contract/grant sponsor: RGC CERG Grant; Contract/grant number: CityU 1149/04E.

backbone is used to mainly disseminate control packets. These observations motive us to study the *minimum connected dominating set (MCDS) problem in unit-disk graphs*, which is NP-hard [6]. In other words, we will use the induced topology of a CDS in unit-disk graphs to model a virtual backbone for *ad hoc* networks.

In this paper, we propose two algorithms for distributed CDS construction. We further analyze their performance when the underlying topology is a unit-disk graph. Our first algorithm grows a tree from a unique leader, whose election takes $O(n \log n)$ messages. This algorithm achieves a comparable performance compared to all those based on a single leader, but takes less number of messages. The second algorithm is initiated by multiple locally elected leaders. This algorithm generates a CDS with size at most $147 \cdot \text{opt} + 33$, which is better than the best published result $(192 \cdot \text{opt} + 48)$ [1] with linear message complexity, where opt is the cardinality of an MCDS. Note that in this paper, we assume each node has a unique id, and its one-hop neighborhood information is available. For a survey of existing CDS construction techniques we refer the readers to Reference [3] and the references therein.

This paper is organized as follows. We first introduce the basic concepts and preliminary results needed in our algorithm design in Section 2. Then, we propose our algorithms together with their performance analysis in Section 3 and 4. We conclude this paper by a discussion in Section 5.

2. Preliminaries

Given a graph $G = (V, E)$, two vertices are *independent* if they are not neighbors. An *independent set* (IS) S of G is a subset of V such that for $\forall\, u, v \in S$, $(u, v) \notin E$. S is *maximal* (denoted by MIS) if any vertex not in S has a neighbor in S. A *dominating set* D of G is a subset of V such that each node not in D has at least one neighbor in D. Each node in D is called a *dominator*, while a node not in D is called a *dominatee*. If the induced subgraph of D is connected, then D is a *connected dominating set* (CDS). Among all CDSs of graph G, the one with minimum cardinality is called a MCDS. Note that a maximal independent set is also a dominating set. In this paper, we focus on connected unit-disk graphs.

Based on the general geometric properties of an MIS in a unit-disk graph, Alzoubi, Wan and Frieder [1] deduce the following lemma.

Lemma 2.1. *Let S be any* MIS *of a unit-disk graph G. For $\forall u \in S$,*

1. *The number of nodes in S that are at most two hops away from u is at most 23.*
2. *The number of nodes in S that are at most three hops away from u is at most 47.*

By exploring local connectivity information, we obtain the following result for two nodes that are two hops away.

Lemma 2.2. *Let S be any* MIS *of a unit-disk graph G. For $\forall u, v \in S$, if u and v are two hops away, then the number of nodes in S that are at most three hops away from either u or v is at most 64.*

Proof. The idea to prove Lemma 2.1 in Reference [1] is applied here again, as shown in Figure 1. We consider the worst case when the distance from u to v is exactly two. All nodes in S that are at most three hops away from either u or v lie within the two annuluses centered at u and v of radius 0.5 and 3.5. The disks of radius 3.5 centered at u and v have an overlapping area A. From Figure 1, the area of A is

$$2 \times \frac{2 \times \arccos \frac{1}{3.5}}{2\pi} \times 3.5^2 \times \pi - 2 \times \frac{2 \times \sqrt{3.5^2 - 1} \times 1}{2} \approx 24.677$$

Thus the number of nodes in S that are at most three hops away from either u or v is less than

$$\frac{\pi \times 3.5^2 \times 2 - \pi \times 0.5^2 \times 2 - A}{\pi \times 0.5^2} \approx 64.58$$

■

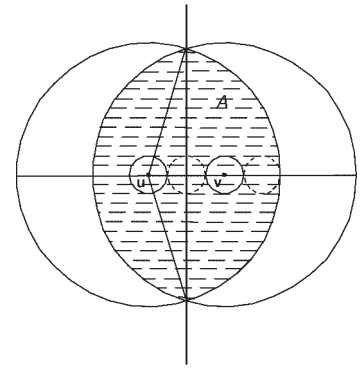

Fig. 1. Illustration of the proof of Lemma 2.2.

Wan, Alzoubi and Frieder [9] have proved the following result that relates the size of any MIS of a unit-disk graph G to that of its MCDS.

Lemma 2.3. *Let S be any* IS *and D be any* MCDS *of a unit-disk graph G. Then $|S| \leqslant 4 \cdot |D| + 1$ for $|D| > 1$.*

Now let S be any MIS. Note that for $\forall u \in S$, there exists another node $v \in S$ such that u and v is either two or three hops away, as long as graph G is connected. If any pair of nodes in S is at least three hops away, then the following lemma holds.

Lemma 2.4. *Let S be any* MIS *and D be any* MCDS *of a unit-disk graph G. If $\forall u, v \in S$ are at least three hops away from each other, then $|S| \leqslant |D|$.*

Proof. We claim that $\forall u \in D$ is either in S or adjacent to at most one node in S. Otherwise, there will exist two nodes in S that are two hops away, a contradiction. On the other hand, because D is an MCDS, $\forall u \in S$ is either in D or adjacent to at least one node in D. Therefore $|S| \leqslant |D|$. ■

3. Algorithm I: CDS Construction Based on a Single Leader

Algorithm I grows a spanning tree distributedly from the leader, with all non-leaf nodes form a CDS.

3.1. Algorithm Description

To facilitate elaboration, we use colors to represent different states during the CDS construction. Initially each node is colored white. Algorithm I contains two steps:

1. Apply the leader-election heuristic in Reference [5] to compute a leader.
2. The leader first colors itself black and broadcasts a $<$*dominator*$>$ message specifying itself as its dominator. The actions and broadcasted messages for each non-leader node are stated below.
   - Upon receiving a $<$*dominator*$>$ message from u whose dominator is v, a white or yellow node colors itself gray and broadcasts a $<$*dominatee*$>$ message, specifying u as its dominator; Node v, if it is gray, will color itself black and then broadcast a $<$*dominator*$>$ message.
   - Upon receiving a $<$*dominatee*$>$ message, a white node colors itself yellow and broadcasts an $<$*active*$>$ message.
   - The yellow node whose id is the minimum among all of its one-hop yellow neighbors colors itself black and broadcasts a $<$*dominator*$>$ message, specifying the gray neighbor with the smallest id as its dominator.

Note that Algorithm I grows a tree from the leader in a step-by-step fashion. At any time, all the inner nodes of the tree are colored black while most of the leaf nodes are colored gray. In the first step, the leader and all of its neighbors are added to the tree. In each of the following steps, a gray node v and one of its yellow neighbors u are colored black. All the white and yellow neighbors of u and v are colored gray and added to the tree as leaves. This algorithm terminates when no white and yellow node left. All the black nodes form a CDS.

3.2. An Example

Figure 2 demonstrates an example of the CDS construction process by Algorithm I. We use the same network topology as that in Reference [9] for convenience. In this graph, the number labeling each node is the node id. The network topology is denoted by edges between nodes. Node 0 is selected as the leader. In Figure 2(a) and 2(b) the leader first colors itself black and its one-hop neighbors are marked gray. Then upon receiving $<$*dominatee*$>$ messages, nodes 3, 8, 5, 10, and 7 change their colors to yellow, as shown in Figure 2(c). Now each yellow node competes with its yellow neighbors. Those with the minimum id among their yellow neighbors (nodes 3, 5, and 7, as shown in Figure 2(d)) become dominators and change their colors to black. They specify nodes 2 and 12 as their dominators. Finally, all white/yellow neighbors of dominator nodes color themselves gray, indicating their dominatee status, as shown in Figure 2(e). The algorithm terminates when no white or yellow node left. The black nodes 0, 12, 2, 5, 3, and 7 form a CDS. Comparing to the approach in Reference [9], our algorithm broadcasts less number of messages and finishes with fewer steps.

3.3. Performance Analysis

In this subsection, we study the performance of Algorithm I.

Theorem 3.1: *Algorithm I has message complexity $O(n \log n)$ and time complexity $O(n)$, where n is the total number of vertices.*

Proof. Step 1 has message complexity $O(n \log n)$ [5]. Step 2 involves the broadcasting of three different

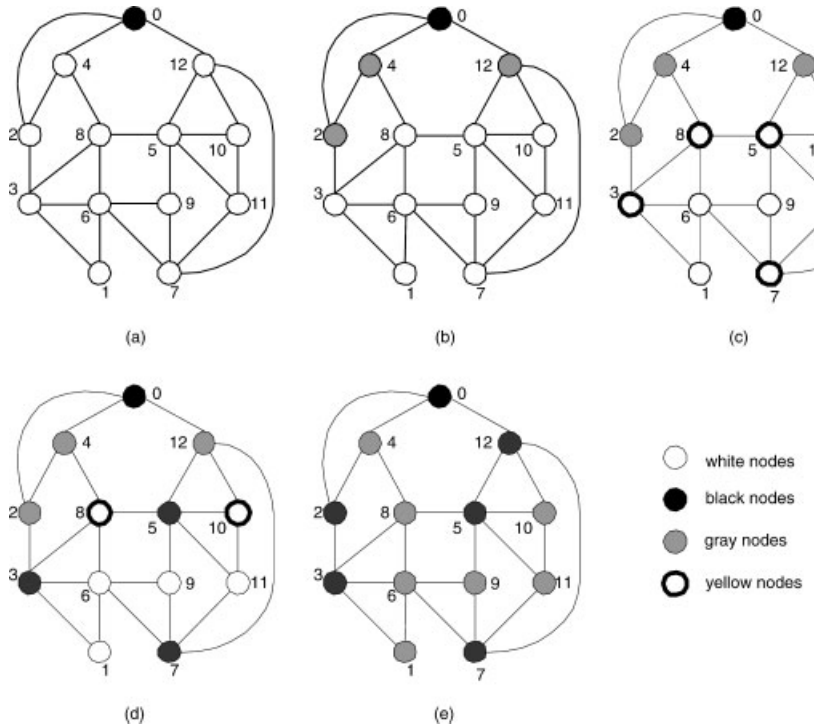

Fig. 2. An example to illustrate Algorithm I. Note that we use the same network topology as that in Reference [9] for convenience.

kinds of messages: $< dominator >$, $< dominatee >$, and $< active >$. Each node broadcasts each message at most once. Therefore the total number of messages in Algorithm I is at most $O(n \log n)$.

The time complexity for Step 1 is $O(n)$ [5]. In Step 2, there are $O(n)$ number of broadcastings. Each broadcast takes unit time, thus Step 2 also takes time $O(n)$. Therefore Algorithm I has time complexity $O(n)$. ■

Theorem 3.2. *The size of the connected dominating set generated by Algorithm* I *is at most* $8 \cdot opt + 1$, *where opt is the size of the minimum connected dominating set*.

Proof. We can partition all the dominators into two sets: A and B. Set A contains all vertices with color changing from *white or yellow to black directly* and B contains all vertices with color changing from *white or yellow to gray then to black*. The first step adds the leader to A. In the following steps, while one node is added to B, there is at least one node added to A. Thus $|A| \geq |B| + 1$.

Now we claim that A is an independent set. This is obvious since each vertex u in A is colored black from white or yellow. This means u has no black neighbors because each neighbor of a black node has gray color. From Lemma 2.3, $|A| \leqslant 4 \cdot opt + 1$. Thus $|A| + |B| \leq 8 \cdot opt + 1$. ■

Remark: Note that intuitively Algorithm I is not better than the best published result in Reference [9], which is denoted by WAF. A careful study indicates that Algorithm I broadcasts less number of messages for CDS construction. WAF relies on the tree constructed by leader election procedure. In Reference [9], the algorithm traverses the tree multiple times for computing an MIS and then connecting all nodes in the MIS. WAF assigns dominator or dominatee status based on the level and effective degree of each node in the tree, thus it takes larger number of messages to report level and effective degree information. Our algorithm grows a spanning tree from the leader. It does not rely on any degree information, thus it is more message-efficient.

4. Algorithm II: A Linear Message Complexity Algorithm

The message complexity of Algorithm I is $O(n \log n)$, which is dominated by leader election. As proved by

References [2,5,9], the lower bound on the message complexity for distributed leader election is $\Omega(n \log n)$. Thus, the best achievable message complexity for leader election based CDS construction is $O(n \log n)$. Recently, Alzoubi, Wan and Frieder [1] introduce an algorithm with linear time and message complexities. This algorithm generates a CDS with size at most $192 \cdot opt + 48$, where *opt* is the size of any MCDS. In this section, we are going to propose a better heuristic that computes a CDS with size at most $147 \cdot opt + 33$. Our algorithm also has linear time and message complexities. Compared to Algorithm I, Algorithm II exploits parallelism to a much higher degree.

4.1. Algorithm Description

Algorithm II contains three steps. Step 1 constructs a forest in which each tree is rooted at a node with minimum id among its one-hop neighbors. Step 2 collects neighboring information, which will be used in step 3 to connect neighboring trees. Initially all nodes are colored white. The actions and broadcasted messages are stated below.

1. **Constructing a forest** F. This step constructs a forest in which each tree is rooted at a node whose id is the smallest among all its one-hop neighbors. Roots are colored red; non-leaf tree nodes are colored either black or blue. All red/black/blue nodes are dominators while gray nodes are dominatees. The procedure to construct a rooted tree is similar to Algorithm I. A tree is identified by its root. Each node knows the root of the tree to which it belongs.
   - Any node u that has the smallest id among *all neighbors* in the closed neighbor set of u colors itself red and broadcasts a $<$ *root* $>$ message.
   - Upon receiving a $<$ *root* $>$ message from u, a white node colors itself gray and broadcasts a $<$ *dominatee* $>$ message, specifying u as its dominator and its root.
   - Upon receiving a $<$ *dominatee* $>$ message, a white node that has the smallest id among *all white neighbors* in its closed neighbor set colors itself black and broadcasts a $<$ *dominator* $>$ message, specifying the gray neighbor v with the smallest id as its dominator and v's root as its root.
   - Upon receiving a $<$ *dominator* $>$ message from u whose dominator is v, a white node colors itself gray and broadcasts a $<$ *dominatee* $>$ message, specifying u as its dominator and u's root as its root. Meanwhile, the gray node v colors itself blue and broadcasts a $<$ *connector* $>$ message.
   - Upon receiving a $<$ *connector* $>$ message from u, a white node colors itself gray and broadcasts a $<$ *dominatee* $>$ message, specifying u as its dominator and u's root as its root.

2. **Collecting information**. Note that step 1 terminates when there is no white node left. After the forest is constructed, each red/black node finds out its nearest red/black neighbors in neighboring trees,[‡] and compute the 'smallest paths'[§] to them.
   - Upon receiving $<$ *dominator* $>$, or $<$ *dominatee* $>$, or $<$ *root* $>$ messages from all neighbors, a gray/blue node u broadcasts a $<$ *report-one-hop* $>$ message, reporting all its one-hop red/black neighbors together with their roots.
   - Upon receiving $<$ *report-one-hop* $>$ messages from all gray/blue neighbors, a gray/blue node reports its two-hop red/black neighbors together with their roots by broadcasting message $<$ *report-two-hop* $>$.

3. **Constructing a CDS**. After step 2, each red/black node knows its red/black neighbors belonging to different trees within three-hop distance. A CDS can be constructed by the broadcasting of $<$ *relaying* $>$ messages, which contain the smallest paths to the red/black neighbors of the red/black node that initiates the message.
   - Upon receiving $<$ *report-two-hop* $>$ messages from all neighbors, a red/black node constructs and broadcasts a $<$ *relaying* $>$ message, containing the one-hop intermediate relaying nodes in the smallest paths to all the three-hop red/black

[‡]Two trees τ_1 and τ_2 are neighbors if at least one node in τ_1 is a one-hop neighbor of some node in τ_2. A neighboring tree of a node u is a tree containing a node that is at most three-hops away from u.

[§]The smallest path between two nodes u and v is the shortest path whose ordered list of node ids (including u and v) are the smallest alphabetically. For example, if the two shortest paths between nodes 1 and 10 are $1 \rightarrow 5 \rightarrow 6 \rightarrow 10$ and $1 \rightarrow 9 \rightarrow 4 \rightarrow 10$, then the smallest path is $1 \rightarrow 9 \rightarrow 4 \rightarrow 10$, whose ordered list of ids is '1 4 9 10', which is smaller than '1 5 6 10' alphabetically. The *smallest path between two neighboring trees* τ_1 *and* τ_2 is the smallest of all the smallest paths between two neighboring black/red nodes u and v, where $u \in \tau_1$ and $v \in \tau_2$. The *smallest path between a node u and a neighboring tree* τ is the smallest of all the smallest paths between u and the nearest neighboring red/black nodes in τ.

neighbors, and the unique intermediate relaying nodes in the smallest paths to all the two-hop red/black neighbors whose ids are higher.

- Upon receiving a $<$ *relaying* $>$ message from all one-hop red/black black neighbors, a gray node whose id is included in at least one $<$ *relaying* $>$ message will color itself yellow and broadcast a $<$ *connector* $>$ message.

Remarks. (i) The smallest path computed in step 2 for any pair of red/black nodes that are two or three hops away is unique, as long as node ids are unique. This guarantees the correctness of Algorithm II. A smallest path must be a shortest path first. (ii) The computation of the smallest path requires the availability of three-hop neighborhood information for each red/black node, which is possible through the broadcastings of $<$ *report-one-hop* $>$ and $<$ *report-two-hop* $>$ messages. (iii) In the third step, only red/black nodes within three-hop distance that belong to different trees are connected.

4.2. An Example

Figure 3 shows an example to illustrate how Algorithm II is applied to compute a CDS. For convenience, we use the same network topology as that in Algorithm I. In the first step, every node that has the smallest id among all its one-hop neighbors begins to construct the forest. In Figure 3(a), nodes 0, 1, 5 color themselves red and each broadcasts a $<$ *root* $>$ message. Then all of their one-hop neighbors, which include nodes 2, 4, 12, 3, 6, 8, 9, 10, 11 in Figure 3(b), change to gray and broadcast $<$ *dominatee* $>$ messages. Upon receiving a $<$ *dominatee* $>$ message, node 7 becomes a dominator, specifying node 6 as its dominator and node 1 as its root. This process is illustrated in Figure 3(c) and 3(d). Up to now, the first step terminates and there is no white node left. There are three trees constructed as shown in Figure 3(d). In the second step, red/black nodes find out their nearest red/black neighbors within three-hop distance in neighboring trees through $<$ *report-one-hop* $>$ and $<$ *report-two-hop* $>$ messages broadcasted by their gray/blue neighbors. In this example, $0 \rightarrow 2 \rightarrow 3 \rightarrow 1$ is found to be the smallest path between node 0 and node 1. Furthermore, Node 0 specifies $0 \rightarrow 12 \rightarrow 5$ as the smallest path to node 5, and $0 \rightarrow 12 \rightarrow 7$ as the smallest path to node 7. Note that path $5 \rightarrow 9 \rightarrow 7$ is selected as the smallest path between node 5 and node 7 according to our definition of the smallest path between two nodes. Similarly, path

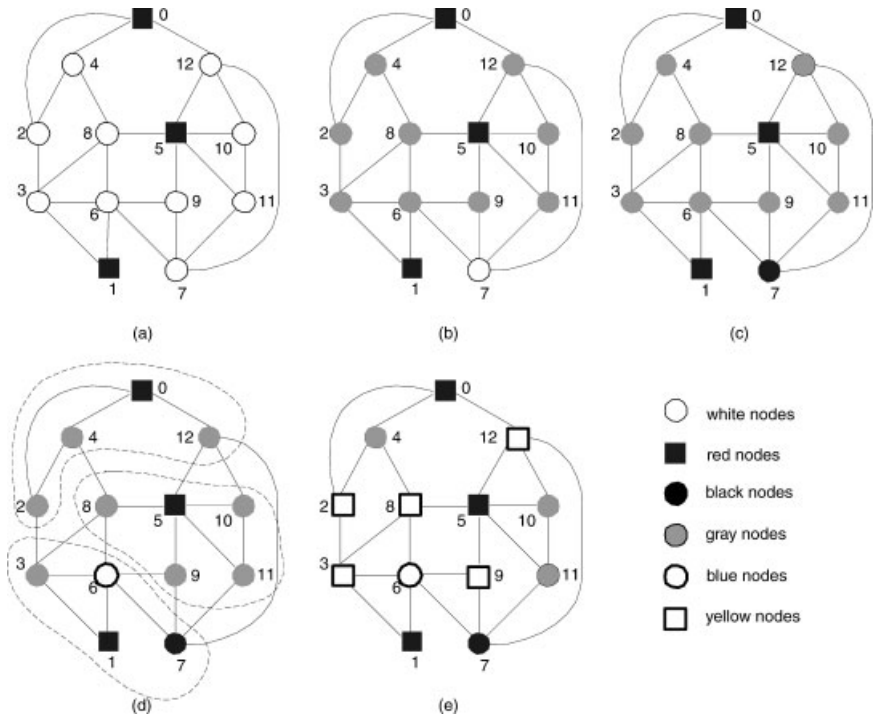

Fig. 3. A connected dominating set (CDS) construction example by Algorithm II. We use the same network topology as that in algorithm I.

$1 \rightarrow 3 \rightarrow 8 \rightarrow 5$ is the smallest path between node 1 and node 5. Figure 3(e) illustrates the third step to connect the neighboring trees. Nodes 2, 3, 12, 8, and 9 turn yellow since their ids are included in the $<relaying>$ messages from their one-hop red/black neighbors. When the algorithm terminates, nodes 0, 1, 5, 7, 6, 2, 3, 12, 8, and 9 form a CDS.

4.3. Performance Analysis

The message and time complexities, together with the performance ratio of Algorithm II are reported in this subsection. We will first prove its correctness.

Lemma 4.1. *In Algorithm II, there is no white node left after step 1 completes.*

Proof. We prove this lemma by contradiction. Assume there is at least one white node after step 1 completes. Let u be the white node with the smallest id. If u is adjacent to a red or black or blue node, then it should have been colored gray after receiving a $<root>$, or a $<dominator>$, or a $<connector>$ message. If u is adjacent to a gray node, then it should have been colored black as it has the smallest id among all of its one-hop white neighbors. Finally if u is adjacent to only white nodes, then it should have been colored red at the beginning of Algorithm II. In all cases, node u with white color can not exit. ■

Lemma 4.2. *After Algorithm II completes, all red/black nodes form an MIS.*

Proof. Nodes with the smallest id among all one-hop neighbors are colored red in step 1. Thus two red nodes are at least two hops away. A node is colored black in step 1 only if it has the smallest id among all its white neighbors, and it is not adjacent to any black or red node, as all neighboring nodes of a red or a black node are colored gray. Therefore, any pair of red/black nodes are at least two hops away. Thus, all red/black nodes form an independent set S. On the other hand, all blue nodes and yellow nodes are colored gray first, and each gray node has at least one red or one black neighbor. Based on Lemma 4.1, S is maximum. ■

Theorem 4.3. *Algorithm II finds a CDS containing all red, black, blue, and yellow nodes.*

Proof. Note that the set containing all red, black, and blue nodes are equivalent to the set containing all dominators in the forest. Step 3 of algorithm II connects neighboring trees with yellow nodes. Therefore the set containing all red, black, blue, and yellow nodes form a CDS. ■

Theorem 4.4. *Algorithm II has message complexity $O(n)$ and time complexity $O(n)$, where n is the total number of vertices.*

Proof. There are seven types of messages involved in Algorithm II and each node broadcasts each type of message at most once. Thus the message complexity is $O(n)$.

Each broadcasting takes unit time. Based on Lemma 2.1, any red/black node in the MIS constructed in step 1 has constant number of two-hop and three-hop independent red/black neighbors. Thus the smallest path computation in step 2 also takes constant time. Therefore the time complexity for algorithm II is $O(n)$. ■

Theorem 4.5. *The size of the connected dominating set generated by Algorithm II is at most* $147 \cdot opt + 33$.

Proof. Let S_1 be the set containing all red nodes; S_2 be the set containing all black nodes; S_3 be the set containing all blue connectors; and S_4 be the set containing all yellow connectors. There are two kinds of red nodes in S_1. $S_1^{(1)}$ is the set in which each red node has at least one red/black neighbor that is two hops away; $S_1^{(2)}$ is the set containing the red nodes at least three hops away from any other red/black nodes. We have $S_1 \equiv S_1^{(1)} + S_1^{(2)}$.

From Lemma 4.2 and Lemma 2.3, $|S_1| + |S_2| \leqslant 4\ opt + 1$. On the other hand, a gray node is colored blue only because a white node is colored black in step 1, thus $|S_3| \leqslant |S_2|$. Finally in step 3, based on Lemma 2.1 and Lemma 2.2, each node in $S_1^{(1)}$ and S_2 is charged for at most 32×2 yellow and blue connectors, and each node in $S_1^{(2)}$ is charged for at most 47×2 yellow connectors. Note that in this charging analysis, each yellow/blue connector is counted twice. Therefore $|S_4| \leqslant \frac{1}{2}[(|S_1^{(1)}| + |S_2|) \times 32 + |S_1^{(2)}| \times 47] \times 2 - |S_3|$. Based on Lemma 2.4, $|S_1^{(2)}| \leqslant opt$. Therefore,

$$\begin{aligned} |D| &= |S_1| + |S_2| + |S_3| + |S_4| \leqslant 4 \cdot opt + 1 \\ &\quad + (|S_1^{(1)}| + |S_2|) \times 32 + |S_1^{(2)}| \\ &\quad \times 47 \leqslant 147 \cdot opt + 33 \end{aligned}$$

■

Remarks. (i) The proof of Theorem 4.5 does not consider the following fact: step 3 only connects

neighboring red/black nodes in different trees. Thus, our performance analysis may not be tight. (ii) This algorithm constructs and connects a forest, while the one proposed in Reference [1] computes and connects an MIS.

5. Conclusion and Discussion

In this paper, we have studied the problem of effectively and efficiently constructing a virtual backbone for *ad hoc* wireless networks, which is modeled by computing a minimum connected dominating set in unit-disk graphs. Two distributed message/time efficient algorithms have been proposed. Both of them achieve good performance compared with existing ones in literature.

Note that we have proved in Reference [4] that MCDS in unit-disk graphs has a PTAS, which means that it can be approximated to any degree. However, the best algorithm with $O(n \log n)$ message complexity has performance ratio of 8 and the best algorithm with linear message complexity obtains a CDS with size at most $147 \cdot opt + 33$. Therefore, one of our future research is to design better heuristics to bridge the gap.

References

1. Alzoubi KM, Wan P-J, Frieder O. Message-optimal connected dominating sets in mobile ad hoc networks, *ACM MOBIHOC*, 2002; pp. 157–164.
2. Awerbuch B. Optimal distributed algorithm for minimum weight spanning tree, counting, leader election and related problems. *Proceedings of the 19th ACM Symposium on Theory of Computing, ACM*, 1987; pp. 230–240.
3. Blum J, Ding M, Thaeler A, Cheng X. Connected dominating sets in sensor networks and MANETs. In *Handbook of Combinatorial Optimization*, Du D-Z, Pardalos P (eds). Kluwer Academic Publishers, Netherlands, 2004; pp. 329–369.
4. Cheng X, Huang X, Li D, Wu W, Du D-Z. A polynomial-time approximation scheme for the minimum connected dominating set in ad hoc wireless networks. *Networks* 2003; **42**(4): 202–208.
5. Cidon I, Mokryn O. Propagation and leader election in multihop broadcast environment, *the 12th International Symposium on Distributed Computing (DISC), 1998; pp. 104–119.*
6. Clark BN, Colbourn CJ, Johnson DS. Unit disk graphs. *Discrete Mathematics* 1990; **86**: 165–177.
7. Das B, Bharghavan V. Routing in ad hoc networks using minimum connected dominating sets, *ICC '97*, Montreal, Canada, June 1997.
8. Ni S-Y, Tseng Y-C, Chen Y-S, Sheu J-P. The broadcast storm problem in a mobile ad hoc network. *Proceedings of MOBICOM*, Seattle, August 1999; pp. 151–162.
9. Wan P-J, Alzoubi KM, Frieder O. Distributed construction of connected dominating set in wireless ad hoc networks. *IEEE INFOCOM*, 2002; pp. 1597–1604.

Author's Biographies

Xiuzhen Cheng is an assistant professor in the Department of Computer Science at the George Washington University. She received her M.S. degree and Ph.D. in Computer Science from University of Minnesota—Twin Cities in 2000 and 2002, respectively. Her current research interests include localization and fault-tolerant information processing in sensor networks; routing in mobile *ad hoc* networks; and approximation algorithm design and analysis. She received the NSF CAREER Award in 2004.

Min Ding received her B.S. in Computer Science and Engineering from Tianjin University in 1995, and the M.S. degree in Information Science from the Institute of Scientific and Technical Information of China in 1999. During August to December 1999, she was a visiting student in College of Information Studies, University of Maryland, College Park. From 2002, she is a Ph.D. student at Computer Science Department, the George Washington University. Her research interests include localization and tracking, in-network aggregation and query processing in sensor networks.

David Hongwei Du received his B.Sc. degree in Computer Science from the Central China Normal University in 2003. He is currently a M.phil. student in the department of Computer Science, City University of Hong Kong. His research interests include computer networks, and internet and mobile computing.

Xiaohua Jia received his B.S. degree in 1984 and M.Eng. in 1987 from the University of Science and Technology of China, and obtained his D.Sc. in 1991 in Information Science from the University of Tokyo, Japan. He is currently a professor in Department of Computer Science at City University of Hong Kong, adjunct with School of Computing, Wuhan University, China. His research interests include distributed systems, computer networks, WDM optical networks, Internet technologies, and mobile computing.