Efficient Distributed Low-Cost Backbone Formation for Wireless Networks

Yu Wang, *Member*, *IEEE*, Weizhao Wang, *Student Member*, *IEEE*, and Xiang-Yang Li, *Member*, *IEEE*

Abstract—Backbone has been used extensively in various aspects (e.g., routing, route maintenance, broadcast, scheduling) for wireless ad hoc or sensor networks recently. Previous methods are mostly designed to minimize the size of the backbone. However, in many applications, it is desirable to construct a backbone with small *cost* when each wireless node has a cost of being in the backbone. In this paper, we first show that previous methods specifically designed to minimize the backbone size may produce a backbone with large cost. Then, an efficient distributed method to construct a weighted backbone with low cost is proposed. We prove that the total cost of the constructed backbone is within a small constant factor of the optimum for homogeneous networks when either the nodes' costs are smooth (i.e., the maximum ratio of costs of adjacent nodes is bounded) or the network maximum node degree is bounded. We also show that, with a small modification, the backbone is efficient for unicast: The total cost (or hop) of the least cost (or hop) path connecting any two nodes using backbone is no more than three (or four) times the least cost (or hop) path in the original communication graph. Our theoretical results are corroborated by our simulation studies. Finally, we discuss several possible ad hoc network applications of our proposed backbone formation algorithms.

Index Terms—Connected dominating set, clustering, distributed algorithm, wireless ad hoc networks.

1 Introduction

Wireless networks have drawn lots of attention in recent years due to potential applications in various areas. Many routing protocols have been proposed for wireless ad hoc networks recently. The simplest routing method is to flood the message, which not only wastes the rare resources of wireless nodes but also diminishes the throughput of the network. One way to avoid flooding is to let each node communicate with only a selected subset of its neighbors, or to use a hierarchical structure like the Internet, e.g., connected dominating set (CDS) based routing [1], [2], [3].

Efficient distributed algorithms for constructing connected dominating sets in ad hoc networks were well studied [1], [2], [3], [4], [5], [6], [7], [8]. Most of the proposed methods try to minimize the number of clusterheads, i.e., the number of nodes in the backbone. However, in many applications of ad hoc networks, minimizing the size of the backbone is not sufficient. For example, different wireless nodes may have different costs for serving as a clusterhead, due to device differences, power capacities, and information loads to be processed. Therefore, in this paper, for the succinctness of our presentation, we assume that each wireless node has a *generic cost* (or *weight*). The cost may represent the *fitness* or *priority* of each node to be a clusterhead. A lower cost means a higher priority. In practice, the cost could represent the power consumption rate of this node if a backbone with small power consumption is needed, the robustness of this node if fault-tolerant backbone is needed, or a function of its security level if a secure backbone is needed. Therefore, by defining different costs, our proposed low-cost backbone formation algorithms can be used in various practical applications. Recently, many proposed clustering algorithms [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] also considered different weights as a *priority criterion* to decide whether a node will be a clusterhead. However, the ultimate goal of the majority of protocols is still to minimize the size of the cluster (or backbone), not the total weight of the cluster (or backbone). In this paper, we study how to construct a sparse backbone efficiently for a set of weighted wireless nodes such that the total cost of the backbone is minimized and there is a cost (or hops) efficient route connecting every pair of wireless nodes via the constructed network backbone.

We propose a novel distributed method to generate a weighted backbone with a good approximation ratio while using a small communication cost. Our methods work not only for homogeneous networks, but also for heterogeneous networks. We prove that the total cost of the constructed backbone is within $\min(4\delta+1, 18\log(\Delta+1))+10$ times the optimum for homogeneous networks when all nodes have the same transmission range. Here, δ is the maximum ratio of costs of two adjacent wireless nodes and Δ is the maximum node degree in the communication graph. Notice that the advantage of our backbone is that the total cost is small compared with the optimum when either the costs of wireless nodes are smooth, i.e., two neighboring nodes' costs differ by a small constant factor, or the maximum node degree is low. The total number of messages of our method is $O(m)$ for any network composed of n wireless

- *Y. Wang is with the Department of Computer Science, University of North Carolina at Charlotte, 9201 University City Blvd., Charlotte, NC 28223. E-mail: ywang32@uncc.edu.*
- *W. Wang and X.-Y. Li are with the Department of Computer Science, Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616. E-mail: wangwei4@iit.edu, xli@cs.iit.edu.*

Manuscript received 6 Dec. 2004; revised 31 May 2005; accepted 27 June 2005; published online 25 May 2006.
Recommended for acceptance by J. Wu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0299-1204.

devices and m total pairs of nodes that can directly receive signals from each other. We also show that, with a small modification, the constructed backbone is efficient for unicast: The total cost (or hop) of the least cost (or hop) path connecting any two nodes using backbone is no more than 3 (or 4) times the least cost (or hop) path in the original communication graph. This is significant since our backbone structure is much sparser than the original communication graph, which significantly reduces the cost of routing without losing much ground on the performance of unicast.

The rest of the paper is organized as follows: In Section 2, we provide preliminaries necessary for describing our new algorithms. In Section 3, we review the related work in literature, including formation of connected dominating sets and weighted clustering methods. Then, the possible bad performances of several classical methods are shown by examples in Section 4. Section 5 presents our new weighted backbone formation algorithms and Section 6 gives the theoretical performance analysis of the proposed algorithms. Section 7 presents some experimental results. In Section 8, we discuss several possible network applications of our proposed algorithms. Finally, we conclude our paper in Section 9 by discussing dynamic maintenance of the backbone and some future research directions.

2 Preliminaries

In this section, we give some definitions and notations that will be used later in our presentation. We assume that all wireless nodes are given as a set V of n points in a two-dimensional space. Each wireless node has an omnidirectional antenna. This is attractive for a single transmission of a node can be received by all nodes within its vicinity. We always assume that the nodes are almost static in a reasonable period of time. A communication graph $G = (V, E)$ over a set V of wireless nodes has an edge uv between nodes u and v if and only if u and v can communicate directly with each other, i.e., inside the transmission region of each other. Hereafter, we always assume that G is a connected graph. Let $d_G(u)$ be the degree of node u in a graph G and Δ be the maximum node degree of all wireless nodes (i.e., $\Delta = \max_{u \in V} d_G(u)$). Each wireless node u has a cost $c(u)$ of being in the backbone. Here, the cost $c(u)$ could be the value computed based on a combination of its remaining battery power, its mobility, its node degree in the communication graph, and so on. We will discuss several possible weight functions for different applications in Section 8. In general, a smaller $c(u)$ means that the node is more suitable for being in the backbone. Let $\delta = \max_{ij \in E} c(i)/c(j)$, where ij is the edge between nodes i and j, E is the set of communication links in the wireless network G, and the maximum operation is taken on all pairs of adjacent nodes i and j in G. In other words, δ is the maximum ratio of costs of two adjacent nodes. We call δ the *cost smoothness* of the wireless networks. When δ is bounded by some small constant, we say the node costs are *smooth*.

When the transmission region of every wireless node is modeled by a unit disk centered at itself, the communication graph is often called a *unit disk graph*, denoted by $UDG(V)$, in which there is an edge between two nodes if and only if their distance is at most 1. We also call such wireless networks *homogeneous networks*.

We call all nodes within a constant k hops of a node u in the communication graph G the *k-local nodes* or *k-hop neighbors* of u, denoted by $N_k(u)$, which includes u itself. The k-local graph of a node u, denoted by $G_k(u)$, is the induced graph of G on $N_k(u)$, i.e., $G_k(u)$ is defined on $N_k(u)$, and contains all edges in G with both endpoints in $N_k(u)$.

A subset of vertices in a graph G is an *independent set* if, for any pair of vertices, there is no edge between them. It is a *maximal independent set* if no more vertices can be added to it to generate a larger independent set. It is a *maximum independent set* (MIS) if no other independent set has more vertices. The independence number, denoted as $\alpha(G)$, of a graph G is the size of the MIS of G. The *k-local independence number*, denoted by $\alpha^{[k]}(G)$, is defined as $\alpha^{[k]}(G) = \max_{u \in V} \alpha(G_k(u))$. It is well-known that, for a unit disk graph, $\alpha^{[1]}(UDG) \leq 5$ [19] and $\alpha^{[2]}(UDG) \leq 18$ [20].

A subset S of V is a *dominating set* if each node in V is either in S or is adjacent to some node in S. Nodes from S are called dominators, while nodes not in S are called dominatees. Clearly, any maximal independent set is a dominating set. A subset C of V is a *connected dominating set* (CDS) if C is a dominating set and C induces a connected subgraph. Consequently, the nodes in C can communicate with each other without using nodes in $V - C$. A dominating set with minimum cardinality is called a *minimum dominating set* (MDS). A CDS with minimum cardinality is the *minimum connected dominating set* (MCDS). In ad hoc networks, assume that each node u has a cost $c(u)$. Then, a CDS C is called a *weighted connected dominating set* (WCDS). A subset C of V is a *minimum weighted connected dominating set* (MWCDS) if C is a WCDS with minimum total cost. In this paper, we study efficient algorithms to construct a low-cost backbone which can approximate the MWCDS well.

3 Related Work

Efficient distributed algorithms for constructing connected dominating sets in ad hoc networks were well studied [1], [2], [3], [4], [5], [6], [7], [8]. The notion of cluster organization has been used for wireless networks since their early appearance. Baker et al. [5], [6] introduced a fully distributed linked cluster architecture, mainly for hierarchical routing, and demonstrated its adaptivity to network connectivity changes. The notion of the cluster has been revisited by Gerla et al. [21], [22] for multimedia communications, with the emphasis on the allocation of resources to support multimedia traffic in an ad hoc environment. Alzoubi et al. [4] proposed a method to approximate a *minimum connected dominating set* within 8 whose message complexity is $O(n \log n)$ and time complexity is $O(n)$ for wireless networks modeled by unit disk graphs. Alzoubi et al. [23] continued to propose a localized method approximating the MCDS within a constant time using a linear number of messages. Marathe et al. [24] studied several approximation results for unit disk graphs, such as methods for *maximum independent set*, *minimum vertex cover*, *minimum coloring*, and *minimum dominating set*. Existing clustering methods first choose some

nodes to act as coordinators of the clustering process, i.e., clusterheads. Then, a cluster is formed by associating the clusterhead with some (or all) of its neighbors. Previous methods differ on the criterion for the selection of the clusterhead, which is either based on the lowest (or highest) ID among all unassigned nodes [6], [22] or based on the maximum node degree [21] or based on some generic weight [10], [15]. In [8], Chen et al. also proposed a localized algorithm to build CDS for topology maintenance, where a node becomes a dominator when two of its neighbors cannot reach each other either directly or via one or two dominators. Similarly, Wu and Li [2] proposed their localized connected dominating set method using a *marking process* where a node is marked true if it has two unconnected neighbors. It is shown that the set of marked nodes forms a CDS. They then reduced the size of the CDS by applying two *dominant pruning rules*. In [7], Dai and Wu further extended their pruning rules to k-hop neighborhoods in order to achieve better results. Recently, Kuhn and Wattenhofer [25] proposed a new distributed MDS approximation algorithm based on linear programming (LP) relaxation techniques. For an arbitrary parameter k, their algorithm computes a dominating set of expected size $O(k\Delta^{2/k}\log\Delta|MDS|)$ in $O(k^2)$ rounds where each node has to send $O(k^2\Delta)$ messages of size $O(\log\Delta)$. Moreover, the authors further gave the time lower bounds for the distributed approximation of MDS in [26].

Many proposed clustering algorithms [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] also considered different weights as a *priority criterion* to decide whether a node will be a clusterhead. Notice the ultimate goal of most protocols is still to minimize the number of clusterheads (or the size of the backbone), not the total weight of clusterheads (or the backbone). For example, methods in [11], [16] considered the stability or mobility of each node as the weight. They preferred the node with high stability and low mobility to be the clusterhead. In [13], the authors also combined the stability with the degree of each node as the weight. The higher priority is given to relatively stable and high degree nodes. Methods in [12], [14] considered clustering in heterogeneous sensor networks, where each node has a different energy level. Most of them used the remaining energy or energy consumption rate as the weight. Both [18] and [17] considered two factors in the priority: available energy and speed, though they used different equations to combine them. In both [10] and [15], authors considered a combined weight metric that takes into account several system parameters like the node-degree, transmission power, mobility, and battery power of the nodes. Most of these proposed weighted clustering algorithms applied simple greedy algorithms where the nodes with highest priority (lowest cost) become clusterheads. For example, [10] selects a node with the lowest cost among its unchosen neighbors to serve as a clusterhead. These greedy heuristics work well in practice, but we will show in Section 4 that they may generate a backbone with a high cost compared with the optimum. Some of these methods [12], [14] are randomized algorithms; nodes become clusterheads randomly with a weighted election probability. All of these cluster methods do not guarantee any approximation ratio of the weighed cluster (or backbone) compared with the optimum. Notice that Basagni [9] gave an algorithm to solve the *maximal weighted independent set* in wireless networks and Basagni et al. [27] studied the performance of a greedy clustering algorithm (highest weight nodes become clusterheads) for the *maximum weighted independent set* in peer-to-peer networks, but, here, our solution for cluster is a distributed approximation algorithm for the *minimum weighted dominating set* and the *minimum weighted connected dominating set*, which are well-known NP-hard problems. Li and Wang [28] presented a centralized approximation algorithm for the weighted maximum independent set for some special graphs. Guha and Khuller [29] studied centralized algorithms for the weighted minimum connected dominating set in general graphs. By combining a weighted set cover approximation algorithm and a node-weighted Steiner tree approximation algorithm, they achieved the approximation ratio $3\ln n$. In [30], they further improved the approximation ratio to $1.35\ln n$, which is the best ratio known. In addition, any approximation algorithm with the ratio α for the unweighted (connected) dominating set problem automatically gives the ratio $\alpha\cdot\delta$ for the weighted version. In particular, the known PTAS for the dominating set in UDG [31] implies that the weighted dominating set in UDG can be approximated with the ratio $(1+\epsilon)\cdot\delta$ for arbitrary $\epsilon>0$.

4 Classical Greedy Methods Do Not Work Well

Most of the methods proposed in the literature aim to find a small dominating set for homogeneous networks. Many of them are based on classical greedy algorithms. If we insist on applying these greedy methods to approximate the minimum weighted dominating set, they may produce a backbone that is arbitrarily worse than the optimum. We will show by examples that three classical methods do not generate a dominating set whose cost is always comparable with ours in the worst case.

The first method to generate a dominating set is to generate a maximal independent set as follows [10], [19]: First, all nodes are originally marked as WHITE, which represents that the node is not assigned any role yet. A node u sends a message IamDominator to all its one-hop neighbors if it has the smallest cost (ID is often used if every node has a unit cost) among all its WHITE neighbors. Node u also marks itself Dominator. When a node v received a message IamDominator from its one-hop neighbors, node v then marks itself Dominatee. Node v then sends a message IamDominatee to all its one-hop neighbors. Clearly, the nodes marked with Dominator indeed form a dominating set. We then show by example that the produced dominating set may be arbitrarily larger than the optimum solution. Although the instance illustrated here uses UDG as the communication graph, it is not hard to extend this to general communication graphs. See Fig. 1a for an illustration. Assume that three wireless nodes u, v, and w are distributed along a line with one-unit intervals. The nodes' costs of u, v, and w are ∞, 1, and $1-\epsilon$, respectively. The dominators selected by the first method are nodes w and u

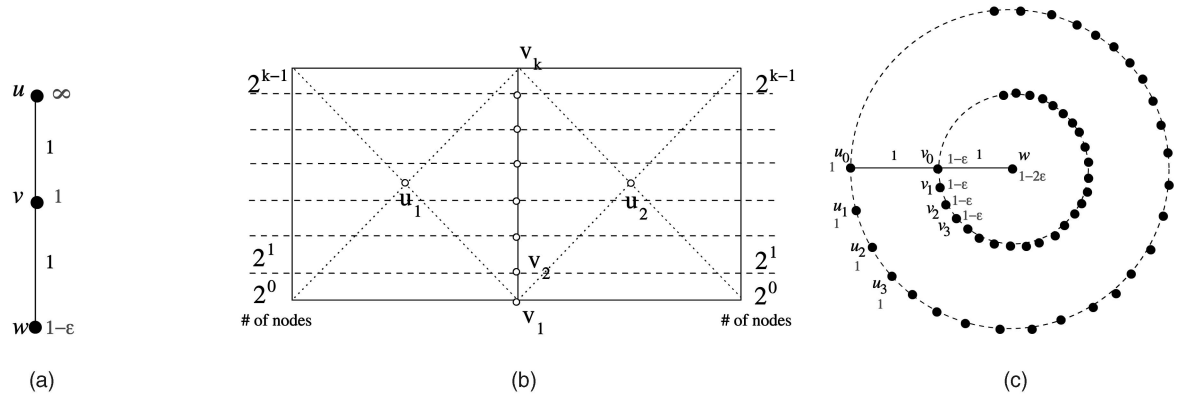


Fig. 1. Examples where the greedy methods fail to produce low-cost weighted connected dominating sets. (a) Example for greedy 1. (b) Example from [4] for greedy 2. (c) Example for greedy 3.

and the total cost of the solution is ∞. However, the optimal solution is formed by v with a total cost 1. Our method presented later does produce a dominating set of total cost $2-\epsilon$.

The second method of constructing a dominating set [3] is based on the minimum weighted set cover [32]. The method can be described in a centralized way as follows: In each round, we select an unselected node i with the minimum ratio $c(i)/d_i$, where d_i is the number of nodes not covered by previously selected dominators. It is well-known that this centralized method produces a dominating set whose total cost is no more than $\log(\Delta+1)$ times the optimum, where Δ is the maximum original degree of all nodes. In [4], Alzoubi et al. gave an example (as in Fig. 1b) with a family of instances for which the size of the solution computed by the second method is larger than the optimum solution by a logarithm factor when all nodes have the same weight. Again, though the instance illustrated here uses UDG as the communication graph, we can easily extend this to a general communication graph. For the detail of this example, see [4]. Moreover, this method is expensive to implement in a distributed way. It is expensive to find the node i with the minimum ratio $c(i)/d_i$ among all unchosen nodes. Our method described later will produce a dominating set whose size is no more than five times the optimum for unit weighted UDG. More importantly, our method is a fully distributed method.

The third method to select the dominating set is proposed by Bao and Garcia-Luna-Aceves [18]. Unlike the previous two methods, this is a fully localized method and it can be executed in two rounds using synchronous communication model. A node decides to become a dominator if either one of the following two criteria are satisfied: 1) The node has the smallest cost in its one-hop neighborhood or 2) the node has the smallest cost in the one-hop neighborhood of one of its one-hop neighbors. We show by an example that the produced dominating set may be arbitrarily larger than the optimum solution. See Fig. 1c for an illustration of an instance in UDG. Assume that $2n+1$ wireless nodes are distributed as shown in Fig. 1c. The nodes' costs of u_i, v_i, and w are 1, $1-\epsilon$, and $1-2\epsilon$, respectively. The dominators selected by the third method are nodes w and v_i $(0 \le i < n)$ and the total cost of the solution is $n(1-\epsilon)+1-2\epsilon$. However, the optimal solution formed by node w and seven nodes from u_i has a total cost $8-2\epsilon$. It is easy to show that seven unit disks centered at 7 nodes among some u_i can cover all u_i. Our method described later will produce an optimal dominating set in this special case.

5 Low-Cost Backbone Formation Algorithms

In this section, we propose a distributed algorithm that constructs a low-cost backbone (weighted connected dominating set) for a wireless ad hoc network G. We will prove that the total cost of the constructed backbone is no more than

$$\min(\alpha^{[2]}(G)\log(\Delta+1), (\alpha^{[1]}(G)-1)\delta+1)+2\alpha^{[1]}(G)$$

times the optimum solution. Notice that, for homogeneous wireless networks modeled by UDG, it implies that the constructed backbone has a cost no more than

$$\min(18\log(\Delta+1), 4\delta+1)+10$$

times the optimum.

We assume that each node knows the IDs and costs of all its one-hop neighbors, which can be achieved by requiring each node to broadcast its ID and cost to its one-hop neighbors initially. This protocol can be easily implemented using synchronous communications as was done in [5], [6]. Our method has the following two phases: The first phase (clustering phase) is to find a set of wireless nodes as the dominators[1] and the second phase is to find a set of nodes, called *connectors*, to connect these dominators to form the final backbone. Notice that these two phases could interleave in the actual construction method. We separate them just for the sake of easy presentation.

5.1 Finding Dominators

We now propose our method of constructing a dominating set whose total cost is comparable with the optimum solution. Our method first constructs a maximal independent set (MIS) using node weight as selection criterion. For each node v in MIS, we then run a local greedy set cover

1. We will interchange the terms clusterhead and dominator. A node that is not a clusterhead is also an *ordinary* node or *dominatee*.

method on the *local neighborhood* $N_2(v)$ to find some nodes ($GRDY_v$) to cover all one-hop neighbors of v. If $GRDY_v$ has a total cost smaller than v, then we use $GRDY_v$ to replace v, which further reduces the cost of MIS. Our method is illustrated in Algorithm 1. For the example illustrated by Fig. 1a, the MIS will be two nodes w and u, whose cost is large. Node u is PossibleDominator and, thus, performs the local set cover. Clearly, $N_2(u) = \{u, v, w\}$ and $N_1(u) = \{u, v\}$. The local set cover will select v to cover all nodes in $N_1(u)$ since v covers both nodes in $N_1(u)$. Note that $c(v) < c(u)$, so node u will let v be a dominator. The other PossibleDominator w will keep itself as a dominator since the local set cover gets worse solution than itself. The final dominating set is then $\{v, w\}$, which is close to optimum $\{v\}$.

Algorithm 1 Construct Low-Cost Dominating Set
1: First, assume that all nodes are originally marked WHITE.
2: A node u sends a message ItryDominator to all its one-hop neighbors if it has the smallest cost among all its WHITE neighbors. Node u also marks itself PossibleDominator.
3: When a node v received a message ItryDominator from its one-hop neighbors, node v then marks itself Dominatee. Node v then sends a message IamDominatee to all its one-hop neighbors.
4: When a node w receives a message IamDominatee from its neighbor v, node w removes node v from its list of WHITE neighbors.
5: Each node u marked with PossibleDominator collects the cost and ID of all of its two-hop neighbors $N_2(u)$.
6: Using the greedy method for minimum weighted set cover (like the second method), node u selects a subset of its two-hop neighbors to cover *all* the one-hop neighbors (including u) of node u. If the cost of the selected subset, denoted by $GRDY_u$, is smaller than the cost of node u, then node u sends a message YouAreDominator(w) to each node w in the selected subset. Otherwise, node u just marks itself Dominator.
7: When a node w receives a message YouAreDominator(w), node w marks itself Dominator.

5.2 Finding Connectors

The second step of weighted connected dominating set formation is to find some *connectors* (also called *gateways*) among all the dominatees to connect the dominators. The connectors and the dominators form a CDS (also called the backbone). Several methods [5], [6], [19], [21], [23] have been proposed in the literature to find the connectors. However, all of these methods only consider the unweighted scenario and they generally do not produce a weighted connected dominating set (WCDS) with a good approximation ratio.

Given a dominating set S, let $VirtG$ be the graph connecting all pairs of dominators u and v if there is a path in the original graph G connecting them with, at most, three hops. It is well-known that $VirtG$ is connected [4]. It is natural to form a CDS by finding connectors to connect any pair of dominators u and v if they are connected in $VirtG$. This strategy was used in several previous methods, e.g., [4], [5], [6], [19], [22].

Our connector selection method for WCDS is also based on this observation. First, we define two dominators u and v as *neighboring dominators* if they are at most three hops away, i.e., they are neighbors in $VirtG$. Let $\mathrm{LCP}(u, v, G)$ denote the least cost path $uv_1v_2 \cdots v_kv$ between nodes u and v on a weighted graph G, and $\mathcal{L}(u, v, G)$ denote the total cost of nodes on path $\mathrm{LCP}(u, v, G)$ excluding u and v, i.e.,

$$\mathcal{L}(u, v, G) = \sum_{1 \le i \le k} c(v_k).$$

For every pair of neighboring dominators u and v, our method will find the shortest path with at most three hops to connect them. The nodes on this shortest path will be assigned a role of connector. Our method uses the following data structures and messages:

1. $D_k(v)$ is the list of dominators that are k hops away from a node v.
2. $P_k(v, u)$ is the least cost path from v to u using at most k hops. (Notice u and v may be less than k hops away.)
3. OneHopDominatorList($v, D_1(v)$): Nodes $D_1(v)$ are the dominators of node v that are one hop from v.
4. TwoHopDominator($v, u, w, c(w)$): Node u is a two-hop dominator of node v and the path uwv has the least cost.

Algorithm 2 illustrates our method in detail. By combining all the dominators and the connectors selected by the algorithms, we get a WCDS (the backbone). Notice that since we run MST on $VirtG$, the constructed backbone is a sparse graph, i.e., it has only a linear number of links.

Algorithm 2 Low-cost Connector Selection
1: Every dominatee node v broadcasts to its one-hop neighbors the list of its one-hop dominators $D_1(v)$ using message OneHopDominatorList($v, D_1(v)$). When a node w receives OneHopDominatorList($v, D_1(v)$) from one-hop neighbor v, it puts the dominator $u \in D_1(v)$ to $D_2(w)$ if $u \notin D_1(w)$. Update the path $P_3(z, u)$ as uvw if it has a smaller cost.
2: When a dominatee node w received messages OneHopDominatorList from *all* its one-hop nodes, for each dominator node $u \in D_2(w)$, node w sends out message TwoHopDominator($w, u, x, c(x)$), where wxu is the least cost path $P_2(w, u)$.
3: When a dominator z receives a message TwoHopDominator($w, u, x, c(x)$) from its neighbor w, it puts u to $D_3(z)$ if $u \notin D_2(z)$ and updates the path $P_3(z, u)$ as $uwxz$ if $c(w) + c(x)$ has a lower cost.
4: Each dominator u builds a virtual edge $\widetilde{uv}$ to connect each neighboring dominator v. The length of $\widetilde{uv}$ is the cost of path $P_3(u, v)$. Notice that here the cost of end nodes u and v is not included. All virtual edges form an *edge weighted* virtual graph $VirtG$ in which all dominators are its vertices.
5: Run a distributed algorithm to build an MST on graph $VirtG$. Let $VMST$ denote $MST(VirtG)$.
6: For any virtual edge $e \in VMST$, select each of the dominatees on the path corresponding to e as a connector.

6 Performance Guarantee

In this section, we first study the performance of the proposed weighted backbone in terms of the total node cost. Then, by a small modification of the backbone formation algorithm, we can make our weighted backbone more efficient for unicast routing.

6.1 Total Cost of the Backbone

Remember, we want to build a backbone whose total node cost is as low as possible. We will show that the backbone constructed by our method is comparable to the optimum when the network is not dense or the costs of the nodes do not have a dramatic change, i.e., they are smooth. The following analysis is on homogeneous networks, but it can be extended to general heterogeneous networks without difficulty. Before describing our result, we first review an important observation of the *dominating set* on UDG, which will play an important role in our proofs later. After clustering, one dominator node can be connected to many dominatees. However, it is well-known that a dominatee node can only be connected to at most *five* independent nodes in the UDG model. In other words, the *1-local independence number* of UDG, $\alpha^{[1]}(UDG)$, is 5. Generally, it is well-known that, for each node, there are, at most, a constant number ($\alpha^{[k]}(UDG)$) of independent nodes that are, at most, k units away. The following lemma, which bounds the number of independent nodes within k units from a node v, is proved in [19] by using a simple area argument.

Lemma 1. *For every node v, the number of independent nodes inside the disk centered at v with radius k units, $\alpha^{[k]}(UDG)$, is bounded by a constant $\ell_k = (2k+1)^2$.*

The bounds on ℓ_k can be improved by a tighter analysis. In [20], Li and Wan gave a detailed proof to show that, for UDG, the number of independent nodes in a two-hop neighborhood (not including the one-hop neighbors) is, at most, 13, while the number of independent nodes in one-hop neighborhoods is, at most, 5. Therefore, there are, at most, 18 independent nodes inside the disk centered at a node v with radius 2, i.e., $\alpha^{[2]}(UDG) = 18$.

Theorem 2. *Algorithm 1 constructs a dominating set whose total cost is no more than $\min(18\log(\Delta+1), 4\delta+1)$ times the optimum for networks modeled by UDG.*

Proof. First, we prove the total cost of the maximal independent set MIS formed by all PossibleDominator nodes is no more than $4\delta+1$ times the optimum. Assume node u is a node from the optimum OPT. If u is not a PossibleDominator node, then there are, at most, five PossibleDominator nodes around u. Let $v_1^u, v_2^u, \cdots, v_5^u$ denote them. The cost of one of these five nodes is smaller than the cost of u; otherwise, node u will be selected as a PossibleDominator node. Without loss of generality, let $c(v_1^u) \le c(u)$. We also know that $c(v_i^u) \le \delta \cdot c(u)$ for $2 \le i \le 5$. Thus, $\sum_{1\le i\le 5} c(v_i^u) \le (4\delta+1)c(u)$. If we summarize the inequalities for all nodes in the optimum dominating set OPT, we get

$$\sum_{u\in OPT}\sum_{1\le i\le 5} c(v_i^u) \le (4\delta+1)\sum_{u\in OPT} c(u) = (4\delta+1)c(OPT).$$

Notice that every node in MIS will appear as v_i^u for at least one node $u \in OPT$ since OPT is a dominating set. Thus, $c(MIS) = \sum_{v\in MIS} c(v) \le \sum_{u\in OPT}\sum_{1\le i\le 5} c(v_i^u)$. It follows that $c(MIS) \le (4\delta+1)c(OPT)$.

Then, we prove the total cost of the nodes selected by the greedy method in Step 6 of Algorithm 1 is no more than $18\log(\Delta+1)$ times the optimum. Assume that node u runs the greedy algorithm and gets the subset as $GRDY_u$, and the cost of the selected subset $c(GRDY_u)$ is at most $c(u)$. It is well known that the dominating set generated by the greedy algorithm for set cover is no more than $\log f$ times the optimum if every set has at most f items. Here, we know that every dominator can cover at most Δ dominatees, thus

$$c(GRDY_u) \le \log(\Delta+1)\cdot c(LOPT_u).$$

Here, $LOPT_u$ is an optimum dominating set (using nodes from $N_2(u)$) when the set of nodes to be covered are the one-hop neighborhood of u (including u). Assume that OPT_u is the subset of the global optimum solution, denoted as OPT, for $MWCDS$, which falls in the two-hop neighborhood of u, i.e.,

$$OPT_u = OPT \bigcap N_2(u).$$

Obviously, OPT_u is a dominating set for $N_1(u)$. Thus, we have $c(LOPT_u) \le c(OPT_u)$ since $LOPT_u$ is the local optimum. Therefore,

$$c(GRDY_u) \le \log(\Delta+1)\cdot c(LOPT_u) \le \log(\Delta+1)\cdot c(OPT_u).$$

Considering all nodes in the MIS, we get

$$c(GRDY) \le \sum_{u\in MIS} c(GRDY_u) \le \log(\Delta+1)\cdot\sum_{u\in MIS} c(OPT_u).$$

Remember that, for each node v, the number of independent nodes in the two-hop neighborhood of v is bounded by 18. Therefore, each dominator is counted at most 18 times (once for each node $u \in MIS$ that selects v to $GRDY_u$). Thus, $\sum_{u\in MIS} c(OPT_u) \le 18c(OPT)$.

For each node u in MIS, we either use u as a dominator or use $GRDY_u$ as dominators, whichever has a smaller cost. Then, the total weight of the final dominating set is at most

$$\begin{aligned}\sum_{u\in MIS} &\min(c(u), c(GRDY_u)) \\ &\le \min\Big(\sum_{u\in MIS} c(u), \sum_{u\in MIS} c(GRDY_u)\Big) \\ &\le \min(4\delta+1, 18\log(\Delta+1))\cdot c(OPT).\end{aligned}$$

This finishes our proof. □

Notice that, here, the approximation ratio is

$$\min(18\log(\Delta+1), 4\delta+1).$$

So, if one of $\log(\Delta+1)$ and δ is a constant, the approximation ratio is a constant. Our analysis is also pessimistic as our simulation shows that the practical performance is much better than this theoretical bound. It is easy to generalize the above result to heterogeneous networks.

Theorem 3. *For a network modeled by a graph G, Algorithm 1 constructs a dominating set whose total cost is no more*

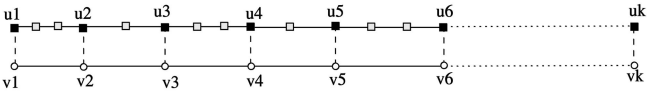


Fig. 2. $\mathcal{L}(u,v,G) \geq 2 \cdot \mathcal{L}(u,v,VirtG)$.

than $\min(\alpha^{[2]}(G)\log(\Delta+1), (\alpha^{[1]}(G)-1)\delta+1)$ *times the optimum.*

Now, we need to prove the total cost of connectors selected by Algorithm 2 is also bounded. The following lemma about the relationship between $\mathcal{L}(u,v,G)$ and $\mathcal{L}(u,v,VirtG)$ will be used in the proof:

Lemma 4. *For any pair of dominators u and v,*

$$\mathcal{L}(u,v,VirtG) \leq 2 \cdot \mathcal{L}(u,v,G).$$

Proof. Notice that the original graph is node weighted, while the virtual graph $VirtG$ is edge weighted. Here, let $c(e)$ be the weight of edge $e = \widetilde{u_i u_j}$ and $c(e) = \mathcal{L}(u_i, u_j, G)$. We assume that path $uv_1v_2\cdots v_kv$ is the least cost path connecting u and v in the original graph G, as shown in Fig. 2.

For any dominatee node p in the original communication graph, it must be dominated by at least one dominator. Thus, we can assume that node u_i is node v_i's dominator, as shown in Fig. 2. For dominators u_i and u_{i+1}, we argue that the length of $\widetilde{u_i u_{i+1}}$ is, at most, the summation of the cost of v_i and v_{i+1}. Notice that $u_iv_iv_{i+1}u_{i+1}$ is a three-hop path between u_i and u_{i+1} whose length is $c(v_i)+c(v_{i+1})$. Thus, the length of $\widetilde{u_i u_{i+1}}$ is, at most, $c(v_i)+c(v_{i+1})$. Thus, we have $c(\widetilde{u_i u_{i+1}}) \leq c(v_i)+c(v_{i+1})$ for $1 \leq i \leq k-1$. Similarly, we also have $c(\widetilde{uu_1}) \leq c(v_1)$ and $c(\widetilde{u_k v}) \leq c(v_k)$. Summing all these inequalities, we get

$$\mathcal{L}(u,v,VirtG) \leq c(\widetilde{uu_1}) + c(\widetilde{v_k v}) + \sum_{i=1}^{k-1} c(\widetilde{u_i u_{i+1}}) \leq 2\sum_{i=1}^{k} c(v_i).$$

This finishes our proof. □

In graph G, we set all dominators' cost to 0 to obtain a new graph G'. Assume T_{opt} is the tree with the minimum cost that spans all dominators selected by Algorithm 1. The following lemma shows that there exists a tree T'_{opt} whose cost equals the cost of T_{opt} and every dominatee node u in T'_{opt} has a node degree at most $\alpha^{[1]}(G)$.

Lemma 5. *There exists a tree T'_{opt} in G' spanning all dominators selected in Algorithm 1 and connectors in this tree have degree at most $\alpha^{[1]}(G)$.*

Proof. We prove this by construction. Consider any optimum cost tree T_{opt} spanning all dominators. In tree T_{opt}, assume there exist some connectors whose degrees are greater than $\alpha^{[1]}(G)$. We choose any one of them as the root. The depth of a connector is defined as the hops from this connector to the root in T_{opt}. We process all connectors u in T_{opt} whose degree is greater than $\alpha^{[1]}(G)$ in an increasing order of their depths. Notice that, as we will see later, the depth of a node does change in our construction, but it will only increase. Assume that currently we are processing a node u with more than $\alpha^{[1]}(G)$ neighbors. Clearly, there are at least two neighbors of u in tree T_{opt} that are connected, say p, q. Notice either p's or q's depth is greater than u since u only has one parent. Without loss of generality, we assume that p's depth is bigger than u's depth. We then remove edge uq and add edge pq. Then, u's degree decreases by 1 while all other connectors whose depth is less than or equal to u's remains unchanged and p's degree increases by 1. Notice this will result in a new tree spanning all dominators while keeping the cost of the tree unchanged. Update the depth of node q and all nodes of the subtree rooted at q (the depths will increase by 1). Repeat the above iteration until all nodes are processed. It is obvious that the above process will terminate. The resulting tree is T'_{opt}. □

For tree T'_{opt}, we define its weight $c(T'_{opt})$ as the sum of the cost of all connectors. We also define $c(T) = \sum_{e \in T} c(e)$ for an edge weighted tree T. Lemma 5 implies that there is an optimum tree connecting all dominators with node degree at most 5 for networks modeled by UDG.

Theorem 6. *The connectors selected by Algorithm 2 have a total cost no more than $2 \cdot \alpha^{[1]}(G)$ times the optimum for networks modeled by G.*

Proof. Let K_G be another virtual complete graph whose vertices are all dominators selected in Algorithm 1 and whose edge length equals the cost of the least cost path between two dominators on original graph G. Following, we argue the weight of MST on graph K_G is at most $\alpha^{[1]}(G)$ times the weight of tree T'_{opt}.

For spanning tree T'_{opt}, we root it at an arbitrary node and duplicate every link in T'_{opt} (the resulting structure is called DT'_{opt}). Clearly, every node in DT'_{opt} has an even degree now. Thus, we can find an Euler circuit, denoted by $EC(DT'_{opt})$, that uses every edge of DT'_{opt} exactly once, which is equivalent to saying that every edge in $T'_{opt}(G)$ is used exactly twice. Consequently, every node v_k in $V(T'_{opt})$ is used exactly $d_{T'_{opt}}(v_k)$ times. Here, $d_G(v)$ denotes the degree of a node v in a graph G. Thus, the total weight of the Euler circuit is at most $\alpha^{[1]}(G)$ times $c(T'_{opt})$, i.e.,

$$c(EC(DT'_{opt})) \leq \alpha^{[1]}(G) \cdot c(T'_{opt}).$$

Notice that, here, if a node v_k appears multiple times in $EC(DT'_{opt})$, its weight is also counted multiple times in $c(EC(DT'_{opt}))$.

If we walk along $EC(DT'_{opt})$, we visit all dominators and the length of any subpath between dominators u_i and u_j is not smaller than $\mathcal{L}(u_i,u_j,G)$. Therefore, the cost of $EC(DT'_{opt})$ is at least $c(MST(K_G))$ since $MST(K_G)$ is the minimum cost tree spanning all dominators and the edge u_iu_j in $MST(K_G)$ corresponds to the path with the least cost between u_i and u_j. In other words,

$$c(EC(DT'_{opt})) \geq c(MST(K_{UDG})).$$

Consequently, we have

$$c(MST(K_G)) \leq c(EC(DT'_{opt})) \leq \alpha^{[1]}(G) \cdot c(T'_{opt}). \quad (1)$$

Now, we prove the weight of $MST(VirtG)$ is at most two times the weight of $MST(K_G)$. For any edge $e = u_iu_j \in MST(K_G)$, from Lemma 4, we have

$$c(e) \geq \mathcal{L}(u_i, u_j, G) \geq \frac{\mathcal{L}(u_i, u_j, VirtG)}{2}.$$

For each edge $e = u_iu_j \in MST(K_G)$, we connect them in graph $VirtG$ using path $\mathrm{LCP}(u_i, u_j, VirtG)$. This constructs a connected subgraph MST' on graph $VirtG$ whose cost is not greater than twice the weight of $MST(K_G)$. Thus,

$$c(MST(VirtG)) \leq c(MST') \leq 2 \cdot c(MST(K_G)). \quad (2)$$

The theorem follows by combining (1) and (2):

$$c(MST(VirtG)) \leq 2c(MST(K_G)) \leq 2\alpha^{[1]}(G) \cdot c(T'_{opt}).$$

□

Notice that Theorem 6 also implies the following side-product result: Given a group of receivers in a node weighted network, the connectors found through VMST have a total cost no more than $2\alpha^{[1]}(G)$ times the minimum cost multicast tree. For the special case of UDG, the total cost of the connectors is no more than 10 times the optimum multicast tree. Here, we assume that the receivers have cost 0.

Combining Theorem 3 and Theorem 6, we get the following theorem, which is one of the main contributions of this paper.

Theorem 7. *For any communication graph G, our algorithm constructs a weighted connected dominating set whose total cost is no more than*

$$\min(\alpha^{[2]}(G)\log(\Delta+1), (\alpha^{[1]}(G)-1)\delta+1) + 2\alpha^{[1]}(G)$$

times the optimum.

Specifically, for homogeneous wireless networks modeled by a unit disk graph, our algorithm constructs a weighted connected dominating set whose total cost is no more than $\min(18\log(\Delta+1), 4\delta+1)+10$ times the optimum.

6.2 Unicast Performance

After we construct the backbone WCDS, if a node u wants to broadcast a message, it follows the following procedure. If node u is not a dominator, then it sends the message to one of its dominators. When the message reaches the backbone, it will be broadcast along the virtual minimal spanning tree. Previously, we prove that the total cost of WCDS is no more than a constant times the optimum, which implies that our structure is energy efficient for broadcast. Notice that in the construction of the low-cost backbone we apply MST (virtual minimal spanning tree) to reduce the total cost of the backbone, it makes the backbone very sparse, which may hurt the performance of the unicast routing since less-power-efficient paths can be used for routing. Therefore, when considering unicast routing, we can remove the MST step and use the paths in $VirtG$ as the backbone. Specifically, we can modify our backbone formation algorithms by 1) removing steps 5, 6, and 7 (collecting two-hop information and running the greedy algorithm for the set over) from Algorithm 1, 2) modifying PossibleDominator to Dominator in step 2 of Algorithm 1, and removing steps 5 and 6 (building VMST) from Algorithm 2. Notice that the changes to Algorithm 1 are not necessary, as we will see later. Let $UWCDS$ be the constructed backbone. If a node u wants to unicast a message, it follows the following procedure: If node u is not a dominator and node v is not a neighbor of u, u sends the message to one of its dominators. Then, the dominator will transfer the message to the target or a dominator of the target through the backbone. Now, we prove that the backbone is a spanner for the unicast application, i.e., every route in the constructed network topology is efficient. Remember, a route is *efficient* if its total cost (or total hop number) is no more than a constant factor of the minimum total cost (or hop number) needed to connect the source and the destination in the original communication graph. The constant is called cost (or hops) stretch factor.

Theorem 8. *For any communication graph, the cost stretch factor of UWCDS is, at most, 3.*

Proof. Consider any source node s and target node t that are not connected directly in the original communication graph G. Assume the least cost path $\mathrm{LCP}(s,t,G)$ from s to t in G is $\Pi_{G_h}(s,t) = v_1v_2...v_k$, where $v_1 = s$ and $v_k = t$, as illustrated by Fig. 2. We construct another path in UWCDS from s to t and the total cost of this path is at most three times the cost of the least cost path $\mathrm{LCP}(s,t,G)$.

For any dominatee node p in original communication graph G, we will show that there must exist one dominator q whose cost is not greater than p's cost. First, from our selection procedure of the maximal independent set, node p is not selected to MIS implies that, at some stage, there is a neighbor, say, u, with smaller cost selected to MIS, which will be PossibleDominator. Notice that this PossibleDominator node u may not appear in our final structure. However, this node is not selected only if $c(GRDY_u)$ is smaller than $c(u)$. Notice that, clearly, there is at least one node, say v, in $GRDY_u$ that dominates node p since p is a one-hop neighbor of node u and $GRDY_u$ covers all one-hop neighbors of u (including u). Clearly, all dominators in $GRDY_u$ have costs no more than $c(u)$ from $c(GRDY_u) \leq c(u)$. If node u is in final structure, we set q as u; otherwise, set q as node v. We denote node q as p's *small dominator*. Notice that q and p can be the same node.

For each node v_i in the path $\mathrm{LCP}(s,t,G)$, let u_i be its small dominator if v_i is not a dominator, else let u_i be v_i itself. Notice that there is a three-hop path $u_iv_iv_{i+1}u_{i+1}$ in the original communication graph G. Then, from Algorithm 2, we know there must exist one or two connectors connecting u_i and u_{i+1} and also the cost summation of these connectors is at most the cost summation of v_i and v_{i+1}. We define a path, denoted by $\mathrm{LCP}(s,t,UWCDS)$, to connect s and t in UWCDS as the concatenation of all paths $\mathrm{LCP}(u_i, u_{i+1}, VirtG)$, for $1 \leq i \leq k-2$, and a least cost path (with $\leq$ two hops) connecting u_{k-1} and t. Remember that the path $\mathrm{LCP}(u_i, u_{i+1}, VirtG)$ is only the

least cost path among all paths connecting u_i and u_{i+1} using at most three hops.

We then show that the path $\mathrm{LCP}(s,t,UWCDS)$ has a cost no more than three times the path $\mathrm{LCP}(s,t,G)$, where $\mathrm{LCP}(s,t,G)$ is the least cost path connecting s and t in the original communication graph G. Clearly,

$$\sum_{i=1}^{k-2}\mathcal{L}(u_i,u_{i+1},VirtG)\le c(v_1)+2\cdot\sum_{i=2}^{k-2}c(v_i)+c(v_{k-1}).$$

Notice that, in our unicast routing algorithm, when the target node t is within two hops of the dominator node u_{k-1}, node u_{k-1} will not send the data to dominator node u_k. Instead, if target t is a one-hop neighbor of node u_{k-1}, it will send data directly to node t; otherwise, node u_{k-1} will find a least cost node, say w, to connect directly to the target node t. Obviously, $c(w)\le c(v_{k-1})$ since node v_{k-1} connects u_{k-1} and target t. Thus, the total cost of the path in the constructed backbone is

$$\begin{aligned}&\sum_{i=1}^{k-2}\mathcal{L}(u_i,u_{i+1},VirtG)+\mathcal{L}(u_{k-1},t,VirtG)+\sum_{i=1}^{k-1}c(u_i)\\&\le c(v_1)+2\sum_{i=2}^{k-2}c(v_i)+c(v_{k-1})+c(v_{k-1})+\sum_{i=1}^{k-1}c(v_i)\\&<3\sum_{i=1}^{k-1}c(v_i).\end{aligned}$$

This finishes our proof. □

Similarly to the proof in [19], we can prove the following theorem:

Theorem 9. *For any communication graph (not necessarily a UDG), the hops stretch factor of UWCDS is, at most, 4.*[2]

6.3 Message and Time Complexity

Compared with data processing, a wireless node spends more energy in data communication. Here, we show that our algorithms are efficient in term of communication complexity.

Theorem 10. *Algorithm 1 uses $O(n)$ messages if the networks are modeled by UDG and the geometry information of all nodes is known.*

Proof. First, for messages ItryDominator and IamDominatee, every node sends out at this kind of message, at most, once. Thus, the total number of these two messages is $O(n)$. Second, for each PossibleDominator node, it needs to collect the costs and IDs of all of its two-hop neighbors. This step may cost lots of communications (at most, $O(m)$ messages when no geometry information is known, where m is the number of links in the original UDG). Recently, Calinescu [33] proposed a communication efficient method (using $O(n)$ messages) to collect $N_2(u)$ for every node u, when the geometry information is known for networks modeled by UDG. Third, after applying the greedy method, node u may send a message YouAreDominator to node v, but, since the number of independent nodes u in two hops of v is bounded by a constant, the total number of this kind of message is also $O(n)$. Consequently, Algorithm 1 uses $O(n)$ messages. □

2. Actually, the bound is $3+\frac{2}{k}$, where k is the number of hops of the shortest hop path in the original communication graph. The basic idea of the proof is similar with the idea used in proof of Lemma 4 and illustrated by the example in Fig. 2. Since one-hop neighbors can directly communicate with each other, for any nodes that are at least two hops away, the bound is 4.

It is easy to show that Algorithm 1 uses $O(m)$ messages for a general network or when the geometry information of all nodes is unknown. For Algorithm 2, the number of messages in the first three steps is at most $O(m)$. Obviously, we can construct the minimum spanning tree on $VirtG$ using $O(m+n\log n)$ number of messages. In practice, we may not need to construct the minimum spanning tree exactly: A localized approximation of the minimum spanning tree [34], which has a message complexity only $O(n)$, may perform well enough. In addition, if only unicast is running on the backbone, we can ignore the MST construction. Then, the message complexity is only $O(m)$.

We also study the time complexity of our algorithms. For Algorithm 1, the first four steps take, at most, $O(n)$ in time. To collect the information of two-hop neighbors, we apply the method proposed by Calinescu [33], which also takes, at most, $O(n)$ in time. Notice that the time complexity of the greedy method in [3] (based on the set covering method in [32]) is, at most, $O(m\Delta)$, where m is the number of nodes participating in the algorithm and Δ is the maximum node degree. So, the sixth step of Algorithm 1 takes at most $O(\Delta_2\Delta)$, where Δ_2 is the maximum number of two-hop neighbors. Since $\Delta_2\le n$ and $\Delta_2\le\Delta^2$, the sixth step takes at most $O(\Delta^3)$ (or $O(n\Delta)$). Therefore, the time complexity of Algorithm 1 is $O(n\Delta)$ in the worst case. For Algorithm 2, the most time-consuming step is to build an MST on $VirtG$. Obviously, we can construct the MST using, at most, $O(m+n\log n)$ time.

7 Simulation Results

In this section, we conduct simulations on random networks to evaluate the performances of our proposed weighted backbone and compared them with previously greedy algorithms. The simulation platform was developed by the authors using C++. In the simulation, we assume nodes have unlimited buffering and ignore all possible retransmissions at the MAC and PHY layers. The main purpose of these simple settings of simulations is only to evaluate the *nonnetwork* performances (geometric properties) of the different backbones formed by different algorithms, such as the total weight of the backbone and the hop (or cost) spanning ratios of the backbone.

7.1 Practical Implementation

Since the distributed construction of MST in Algorithm 2 is expensive in terms of message complexity ($O(m+n\log n)$), we implement a localized approximation of MST, *localized minimum spanning tree* (LMST) [34] to reduce the messages to $O(n)$. For a general edge weighted graph G, the k-local minimum spanning tree ($LMST_k(G)$) contains a *directed* edge $\overrightarrow{uv}$ if edge uv belongs to $MST(N_k(u))$. In our case, for the edge weighted graph $VirtG$, each dominator node u will first collect all dominator nodes that are at most k hops

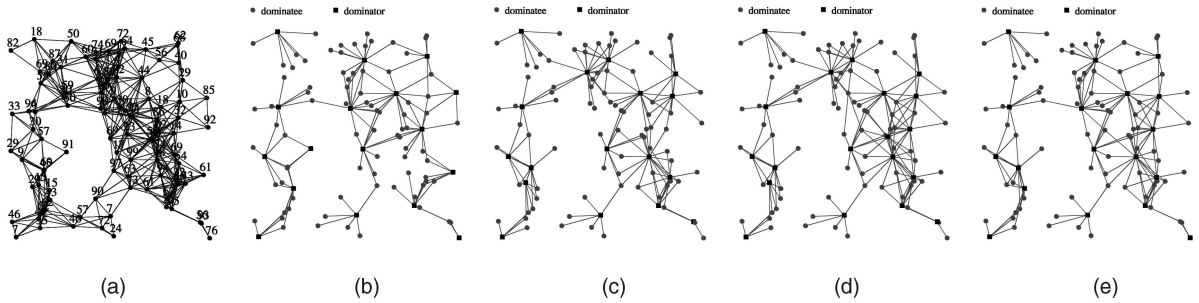


Fig. 3. Different dominating sets by different greedy methods from the same unit disk graph. (a) UDG. (b) Greedy 1. (c) Greedy 2. (d) Greedy 3. (e) Our method.

away in $VirtG$. Typically, k is 1 or 2 in our methods. Node u then constructs the minimum spanning tree $MST(N_k(u))$ and keeps all edges $uv \in MST(N_k(u))$. The union of all such selected links forms the LMST. Notice that, here, the weight of a link uv is the cost of the least cost path (with $\leq$ 3 hops) connecting u and v in G. It is easy to prove that the global minimum spanning tree $MST(G)$ is a subgraph of the local minimum spanning tree $LMST_k(G)$. Unfortunately, in the worst case, the total cost of $LMST_k(G)$ could be arbitrarily larger than the cost of $MST(G)$. However, our simulations show that it is within a small constant factor on average. The advantage of using the LMST instead of the global MST is the significant reduction in the communication cost.

7.2 Performance Comparisons

In our experiments, we randomly generated a set V of n wireless nodes with random costs drawn from $[1, 100]$ and the induced $UDG(V)$ and, then, tested the connectivity of $UDG(V)$. If it is connected, we construct different clustering algorithms on $UDG(V)$ to form dominating sets and measure the total costs of these dominating sets. Then, we apply our new method to construct the weighted backbone. We test the total cost of the final backbone and measure the average and maximum cost/hop spanning ratios. In the experimental results presented here, n wireless nodes are randomly distributed in a $500\ \mathrm{m} \times 500\ \mathrm{m}$ square, and the transmission range is set to $100\ \mathrm{m}$. We tested all algorithms by varying n from 50 to 275, where 50 vertex sets are generated for each case. The average and the maximum were computed over all these 50 vertex sets. Note, the parameter setting of our experiments here is just for demonstrations. We have tried various other settings, and the results and performances are stable. Due to space limitations, we cannot present all of them here.

7.2.1 Cost of Dominators

First, we compare our algorithm with the three previous greedy algorithms to find a dominating set (DS). Fig. 3 gives an example of the original communication graph with node costs (Fig. 3a)) and different DSs by different greedy methods (Figs. 3b, 3c, 3d, and 3e; the black squares are the dominators). We plotted the performances (average total cost of the backbone and average number of dominators) of all methods in Fig. 4. Our method produces a DS whose cost is significantly less than that produced by the MIS-based method (greedy 1) and is on a similar level to the other two methods. In addition, our method produces a DS whose size is significantly less than that produced by the method in [18] (greedy 3) and is on a similar level with other two methods. The set-cover-based method (greedy 2) is the only one that is comparable with our method for both metrics. However, it is a centralized method, while ours is a distributed method with a small communication cost.

7.2.2 Cost of Backbone

After getting the dominating set (Fig. 3e) by Algorithm 1, we apply Algorithm 2 to find the connectors. Fig. 5e shows the backbone after adding some connectors to the dominating set. Notice that we used the local minimum spanning tree to find the connectors instead of the global minimum spanning tree. (That is why the graph WCDS in Fig. 5e is not a tree.) We also apply Algorithm 2 to find the connectors for the other three greedy methods for comparison, and the results are shown in Figs. 5a, 5b, and 5c. We

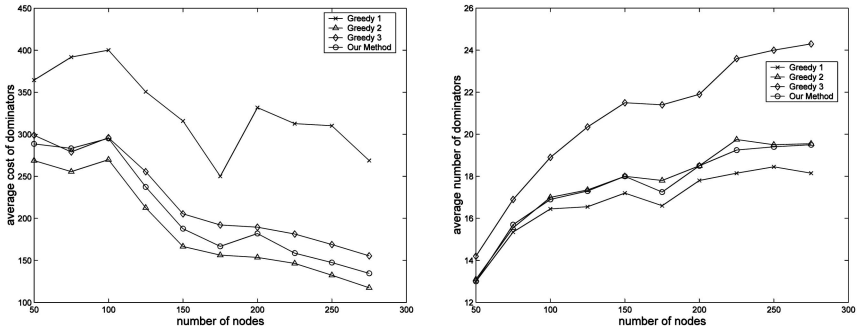


Fig. 4. Total cost and number of clusterheads of different greedy methods (number of nodes: from 50 to 275).

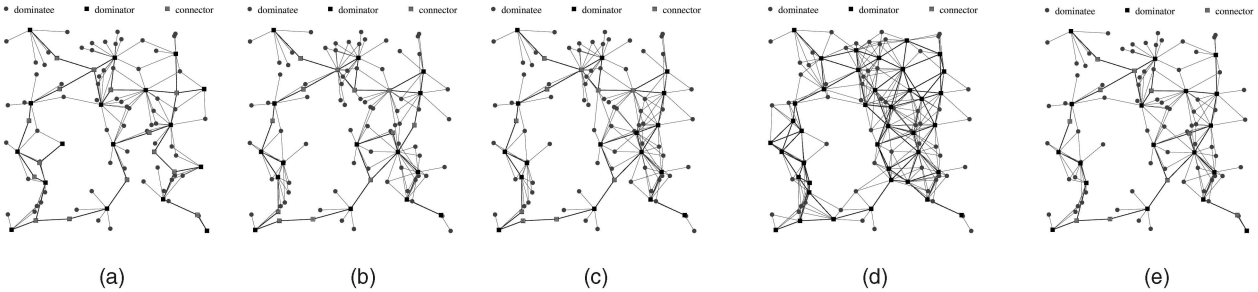


Fig. 5. Different connected dominating sets by different methods from the same unit disk graph. (a) Greedy 1. (b) Greedy 2. (c) Greedy 3. (d) Wu and Li's method. (e) Our method.

also implement a variation of another connected dominating set method by Wu and Li [2]. Their method uses a *marking process* plus two *dominant pruning rules* to build the CDS. We modified it to compare costs instead of IDs in the dominant pruning rules. The resulting CDS is shown in Fig. 5d. Obviously, their method generates more dominators than other methods; the reason is that they did not use MST (or LMST) in the formation. Also, their original method is used to minimize the size of CDS, not its total cost. Notice that more dominators in the backbone means better performances during the unicast routing. We plot the total cost of the weighted backbone in Fig. 6a. As expected, the total cost of the backbone produced by our method is less than that produced by the MIS-based method (greedy 1) and that produced by Wu and Li's method. However, the results from the other two greedy methods are slightly better than ours (though on a similar level). The main reason is that we use the same MST-based method to select the connectors for all greedy methods. One interesting observation is that, though the size of the backbone becomes stable when the network becomes denser, the average total cost of the backbone for greedy methods decreases over the increasing of the network density. This may be due to that dense network provides more candidates for backbone with potential lower costs.

7.2.3 Cost of Unicast Routing

For unicast, we can simplify Algorithm 2 by directly using VirtG as the final backbone. Spanning ratios of the final unicast backbone are plotted in Fig. 6b. Notice that the average cost and hop spanning ratios are indeed small (almost 1). The maximum cost spanning ratio is less than 3. The maximum hop spanning ratio is no more than 4. These map well to the theoretical bounds, which are 3 and 4, respectively.

8 Practical Applications in Ad Hoc Networks

As we mentioned in the introduction, the proposed distributed algorithms for MWCDS can be used in ad hoc networks to form a low-cost network backbone for unicast routing or broadcasting application. The cost which we used as the input of our algorithms could be a *generic* cost, defined by various practical applications. Here, we list some possible weights that may be used in ad hoc networks.

Energy Consumption Rate. Most backbone-based unicast routing or broadcasting protocols [1], [2], [3] deliver packets only through the backbone or restrict the flooding packets in the backbone; thus the nodes serving as clusterheads or connectors in the backbone consume more energy than ordinary nodes. If we use the energy consumption rate at each node as its weight, using the proposed low-cost backbone formation algorithm, we can achieve an energy efficient backbone, where the total energy consumption of this backbone is at most constant times the energy consumption of the optimum. Also, the unicast carried on the backbone is power efficient, compared with the least energy consumption path in the original communication graph. Another way to build energy-efficient backbone is to select nodes with the maximum remaining energy.

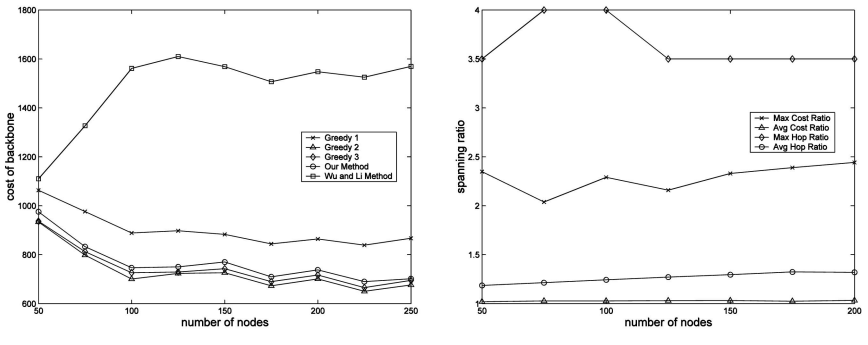


Fig. 6. Performance of backbones (number of nodes: from 50 to 200). (a) Total cost. (b) Spanning ratio.

Fault-Tolerant Rate. Fault tolerance is also an important issue in wireless networks since nodes are mobile and in a dynamic environment. If each node estimates its probability of being faulty and we treat it as the weight, we can use our proposed algorithm to build a fault-tolerant backbone for routing. The fault-tolerant rate can be evaluated by considering the mobility (stability, speed) of the node, the quality of links (link failures) around the node, the interference level at the node, or another metric. Some research along this line has been done in [11], [13], [16]. Assume that p_i is the probability that the wireless node $v_i \in V$ will have a fault in computing or communicating with its neighbors. Two possible criteria could be used to measure the fault-tolerant quality of a backbone (i.e., a CDS $S \subset V$): $\sum_{v_i \in S} p_i$ or $\Pi_{v_i \in S} p_i$. In the first case, the cost of node v_i is assigned as $c(v_i) = p_i$, while, in the latter case, the cost of v_i is assigned as $c(v_i) = \log p_i$. Then, building the most fault-tolerant backbone is equivalent to finding a CDS with the minimum total cost.

Security Level. Our proposed algorithm can also be applied in designing secure routing protocols. Since ad hoc networks lack a central authority for authentication and key distribution, security is hard to achieve. In [35], Liu et al. proposed a dynamic trust model for an ad hoc network where each node has a security level by observing its neighbor. By using the security level information obtained by their method, we can apply our low-cost method to build a backbone for routing with high security. We could assign the cost to a node using a method analog to the case of fault tolerance discussed above.

More different metrics can be considered as the weight in our method, such as traffic load, signal overhead, battery level, and coverage. As done in [10], [15], we can also use a combined weight function to integrate various metrics in consideration to form a more robust and efficient backbone for wireless ad hoc networks in general applications.

Beside forming the backbone for routing, our weighted clustering algorithm (Algorithm 1) can also be used in other applications, such as selecting the mobile agents to perform intrusion detection in ad hoc networks [37] (to achieve more robust and power efficient agent selection) or selecting the rendezvous points to collect and store data in sensor networks [36] (to achieve the energy efficiency and storage balancing).

9 Summary and Future Work

In this paper, we present a new algorithm to construct a sparse structure for network backbone in wireless ad hoc networks. A communication efficient distributed algorithm was presented for the construction of a weighted connected dominating set, whose size is guaranteed to be within a small constant factor of the minimum (when either δ or Δ is a constant). We also show that, with a small modification, the constructed backbone is efficient for both cost and hops (though losing the low cost property). This topology can be constructed locally. Our simulations confirmed that our new backbone indeed performs well in random networks.

In our algorithms, we assume that the nodes are almost-static in a reasonable period of time. However, in some ad hoc network applications, the network could be highly dynamic. Therefore, after the generation of the weighted backbone, the dynamic maintenance of the backbone is also an important issue. Two major events may cause the backbone to become obsolete: 1) *topology changes* due to node moving, node joining or leaving, or node failure and 2) *weight changes* when weights are assigned based on some observed status of nodes. Note that some of the practical weights we discussed above change frequently, such as battery level and quality of links. Thus, a dynamic update method for our backbone is needed. Usually, there are two kinds of update methods: on-demand update or periodical update. Most of the existing clustering algorithms are invoked periodically, while some algorithms (e.g. [10]) perform the updating only when it is required. Our algorithm can adapt and combine both these update methods. If no major topology change or no remarkable weight change occurs, no update will be performed until some preset timer expires. This kind of global update also ensures the load balance throughout the network. But, for some major topology change (e.g., a clusterhead dies) or tremendous change of weights (e.g., a big drop of security level), an on-demand update will be performed. Notice that, since our algorithm is a localized algorithm,[3] the update process can be performed only in a local area where the change occurs, i.e., the backbone is easy to maintain locally when the nodes move around. However, it remains an open problem how to update the topology efficiently while preserving the approximation quality.

Remember that we use the following assumptions on the wireless network model: an omnidirectional antenna and a single transmission received by all nodes within the vicinity of the transmitter. The problem studied here will become much more complicated if we relax some of these assumptions. It is also interesting to see the practical performance differences of all proposed methods, such as the methods by Baker et al. and Alzoubi et al. and our methods proposed here, in a mobile environment.

Acknowledgments

The work of Y. Wang was supported, in part, by funds provided by the University of North Carolina at Charlotte. The work of X.-Y. Li is partially supported by US National Science Foundation CCR-0311174.

References

[1] J. Wu and H. Li, "A Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks," *Telecomm. Systems J.*, vol. 3, pp. 63-84, 2001.

[2] J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," *Proc. Third Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm.*, 1999.

[3] B. Das and V. Bharghavan, "Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets," *Proc. IEEE Int'l Conf. Comm.*, 1997.

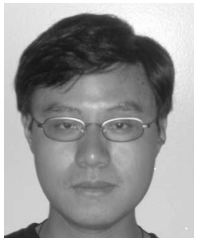
[4] K.M. Alzoubi, P.-J. Wan, and O. Frieder, "New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks," *Proc. IEEE Hawaii Int'l Conf. System Sciences*, 2002.

[5] D.J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm," *IEEE Trans. Comm.*, vol. 29, no. 11, pp. 1694-1701, Nov. 1981.

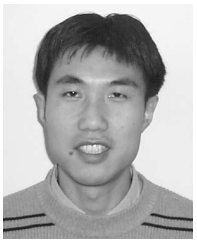
3. By using a localized minimum spanning tree (LMST) instead of MST, our distributed algorithm becomes a localized algorithm.

[6] D.J. Baker, A. Ephremides, and J.A. Flynn, "The Design and Simulation of a Mobile Radio Network with Distributed Control," *IEEE J. Selected Areas in Comm.*, vol. 2, pp. 226-237, 1984.
[7] F. Dai and J. Wu, "An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 15, no. 10, pp. 908-920, Oct. 2004.
[8] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *Wireless Network*, vol. 8, no. 5, pp. 481-494, 2002.
[9] S. Basagni, "Finding a Maximal Weighted Independent Set in Wireless Networks," *Telecomm. Systems*, vol. 18, nos. 1-3, pp. 155-168, Sept. 2001.
[10] M. Chatterjee, S.K. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks," *J. Cluster Computing*, vol. 5, no. 2, pp. 193-204, 2002.
[11] C. Bettstetter and R. Krausser, "Scenario-Based Stability Analysis of the Distributed Mobility-Adaptive Clustering (DMAC) Algorithm," *Proc. Second ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, 2001.
[12] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. 33rd Hawaii Int'l Conf. System Sciences*, 2000.
[13] U.C. Kozat, G. Kondylis, B. Ryu, and M. Marina, "Virtual Dynamic Backbone for Mobile Ad Hoc Networks," *Proc. IEEE Int'l Conf. Comm.*, 2001.
[14] G. Smaragdakis, I. Matta, and A. Bestavros, "SEP: A Stable Election Protocol for Clustered Heterogeneous Wireless Sensor Networks," *Proc. Second Int'l Workshop Sensor and Actor Network Protocols and Applications*, 2004.
[15] G. Chen, F. Nocetti, J. Gonzalez, and I. Stojmenovic, "Connectivity-Based k-Hop Clustering in Wireless Networks," *Proc. 35th Ann. Hawaii Int'l Conf. System Sciences*, 2002.
[16] M. Min, F. Wang, D.-Z. Du, and P.M. Pardalos, "A Reliable Virtual Backbone Scheme in Mobile Ad-Hoc Networks," *Proc. First IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems*, 2004.
[17] H. Liu and R. Gupta, "Selective Backbone Construction for Topology Control," *Proc. First IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems*, 2004.
[18] L. Bao and J.J. Garcia-Luna-Aceves, "Topology Management in Ad Hoc Networks," *Proc. Fourth ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, 2003.
[19] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder, "Geometric Spanners for Wireless Ad Hoc Networks," *IEEE Trans. Parallel and Distributed Processing*, vol. 14, no. 4, pp. 408-421, Apr. 2003.
[20] X.-Y. Li and P.-J. Wan, "Theoretically Good Distributed CDMA/OVSF Code Assignment for Wireless Ad Hoc Networks," *Proc. 11th Int'l Computing and Combinatorics Conf. (COCOON)*, 2005.
[21] M. Gerla and J.T.-C. Tsai, "Multicluster, Mobile, Multimedia Radio Network," *Wireless Networks*, vol. 1, no. 3, pp. 255-265, 1995.
[22] C.R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.*, vol. 15, no. 7, pp. 1265-1275, Sept. 1997.
[23] K.M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks," *Proc. Third ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, 2002.
[24] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, and D.J. Rosenkrantz, "Simple Heuristics for Unit Disk Graphs," *Networks*, vol. 25, pp. 59-68, 1995.
[25] F. Kuhn and R. Wattenhofer, "Constant-Time Distributed Dominating Set Approximation," *Proc. 22nd ACM Symp. Principles of Distributed Computing*, 2003.
[26] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "What Cannot Be Computed Locally!" *Proc. 23rd ACM Symp. Principles of Distributed Computing*, 2004.
[27] S. Basagni, I. Chlamtac, and A. Farago, "A Generalized Clustering Algorithm for Peer-to-Peer Networks," *Proc. Workshop Algorithmic Aspects of Comm.*, 1997.
[28] X.-Y. Li and Y. Wang, "Simple Heuristics and PTASs for Intersection Graphs in Wireless Ad Hoc Networks," *Proc. ACM Sixth Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm. (DIALM)*, 2002.
[29] S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, vol. 20, no. 4, pp. 374-387, 1998.
[30] S. Guha and S. Khuller, "Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets," *Information and Computation*, vol. 150, no. 1, pp. 57-74, 1999.
[31] H.B. Hunt, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns, "NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs," *J. Algorithms*, vol. 26, no. 2, pp. 238-274, 1998.
[32] V. Chvátal, "A Greedy Heuristic for the Set-Covering Problem," *Math. Operations Research*, vol. 4, no. 3, pp. 233-235, 1979.
[33] G. Călinescu, "Computing 2-Hop Neighborhoods in Ad Hoc Wireless Networks," *Proc. Int'l Conf. Ad-Hoc Networks and Wireless (AdHoc-Now)*, 2003.
[34] X.-Y. Li, Y. Wang, and W.-Z. Song, "Applications of *k*-Local MST for Topology Control and Broadcasting in Wireless Ad Hoc Networks," *IEEE Trans. Parallel and Distributed Processing*, vol. 15, no. 12, pp. 1057-1069, Dec. 2004.
[35] Z. Liu, T. Joy, and R. Thompson, "A Dynamic Trust Model for Mobile Ad Hoc Networks," *Proc. 10th IEEE Int'l Workshop Future Trends in Distributed Computing Systems*, 2004.
[36] R. Zheng, G. He, I. Gupta, and L. Sha, "Time Indexing in Sensor Networks," *Proc. First IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems*, 2004.
[37] O. Kachirski and R. Guha, "Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Networks," *Proc. IEEE Workshop Knowledge Media Networking*, 2002.

Yu Wang received the PhD degree in computer science from the Illinois Institute of Technology in 2004 and the BS and MS degrees in computer science from Tsinghua University, China, in 1998 and 2000, respectively. He is an assistant professor in the Department of Computer Science at the University of North Carolina at Charlotte. His current research interests include computer networks, wireless networks, mobile computing, algorithm design, and artificial intelligence. He is a member of the ACM, the IEEE, and the IEEE Communications Society.



Weizhao Wang received the BS and MS degrees in computer science from Shanghai Jiaotong University, China, in 1999 and 2002. He is currently a PhD candidate in the Department of Computer Science at the Illinois Institute of Technology. His current research interests include wireless networks, game theory, algorithm design, and next generation Internet. He is a student member of the IEEE and a member of ACM SIGACT.



Xiang-Yang Li received the bachelor's degrees in computer science and business management from Tsinghua University, China, in 1995 and the MS and PhD degrees in 2000 and 2001, respectively, from the Department of Computer Science at the University of Illinois at Urbana-Champaign. He has been an assistant professor of computer science at the Illinois Institute of Technology since 2000. His research interests span the wireless ad hoc networks, game theory, computational geometry, and cryptography and network security. He is a member of the ACM, the IEEE, and the IEEE Communication Society.