Connected Domination in Multihop Ad Hoc Wireless Networks

Mihaela Cardei * Xiaoyan Cheng *
Xiuzhen Cheng * Ding-Zhu Du *

Abstract

The idea of *virtual backbone routing* for ad hoc wireless networks is to operate routing protocols over a virtual backbone. One purpose of virtual backbone routing is to alleviate the serious *broadcast storm problem* suffered by many exiting on-demand routing protocols for route detection. Thus constructing a virtual backbone is very important. In our study, the virtual backbone is approximated by a *minimum connected dominating set (MCDS)* in a unit-disk graph. This is a NP-hard problem [6]. We propose a distributed approximation algorithm with performance ratio at most 8. This algorithm has time complexity $O(n)$ and message complexity $O(n \cdot \Delta)$, where n is the number of hosts and Δ is the maximum degree. To our knowledge, this is the best (time and message efficient) distributed algorithm known so far. We first find a maximal independent set. Then we use a Steinter tree to connect all vertices in the set. The performance of our algorithm is witnessed by both simulation results and theoretical analysis.

Keywords: Multihop ad hoc wireless network, connected dominating set, virtual backbone routing.

1 Introduction

Ad hoc wireless network has applications in emergency search-and-rescue operations, decision making in the battlefield, data acquisition operations in inhospitable terrain, etc. It is featured by dynamic topology (infrastructureless), multihop communication, limited resources (bandwidth, CPU, battery, etc) and limited security. These characteristics put special challenges in routing protocol design.

Existing routing protocols can be classified into two categories: *proactive* and *reactive*. Proactive routing protocols (see [12] and [9] as examples) ask each host (or many hosts) to maintain global topology information, thus a route can be provided immediately when requested. But large amount of control messages are required to keep each host updated for the newest topology changes. Reactive routing protocols (See [10] and [13] as examples) have the feature *on-demand*. Each host computes route for a specific destination only when necessary. Topology changes which do not influence active routes do not trigger any route maintenance function, thus communication overhead is lower compared to proactive routing protocol. But, if a host knows nothing about the destination, flooding must be applied to detect route.

On-demand routing protocols attract much attention due to their better scalability and lower protocol overhead. But most of them use flooding for route discovery. Flooding suffers from *broadcast storm problem* [11]. Broadcast storm problem refers to the fact that flooding may result in excessive *redundancy*, *contention*, and *collision*. This causes high protocol overhead and interference to other ongoing communication sessions. On the other hand, the *unreliability* of broadcast [14] may obstruct the detection of the shortest path, or simply can't detect any path at all, even though there exists one.

Recently an effective approach based on overlaying a virtual infrastructure (termed *core*) on an ad hoc network is proposed in [14]. Routing protocols are operated over the core. Route request packets are unicasted to core nodes and a (small) subset of non-core nodes. No broadcast is involved in core path detection. Simulation results when running DSR [10] and AODV [13] over this core indicate that the core structure is *effective* in enhancing the performance of the routing protocols. Actually prior to this work, inspired by the *physical backbone* in a wired network, many researchers proposed the concept of *virtual backbone* for unicast, multicast/broadcast in ad hoc wireless networks (see [7] and [15]). The virtual backbone is mainly used to collect topology information for route detection. It also works as a backup when route is unavailable temporarily.

In this paper, we will study the problem of efficiently constructing virtual backbone for ad hoc wireless networks. The number of hosts forming the virtual backbone must be as small as possible to decrease protocol overhead. The algorithm must be time/message efficient due to resource scarcity. We use a connected dominating set (CDS) to approximate the virtual backbone. We assume a given ad hoc network instance contains n hosts. Each host is in the ground and is mounted by an omni-directional antenna. Thus the transmission range of a host is a disk. We further assume that each transceiver has the same communication range R. Thus the footprint of an ad hoc wireless

*Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA. Email: {mihaela,xiaoyan,cheng,dzd}@cs.umn.edu.

network is a unit-disk graph. In graph-theoretic terminology, the network topology we are interested in is a graph $G = (V,E)$ where V contains all hosts and E is the set of links. A link between u and v exists if their distance is at most R. In a real world ad hoc wireless network, sometime even when v is located in u's transmission range, v is not reachable from u due to *hidden/exposed terminal problems*. But in this paper we only consider bidirectional links. From now on, we use host and node interchangeably to represent a wireless mobile.

There exist several distributed algorithms ([1][7][15]) for MCDS computation in the context of ad hoc wireless networking. The one in [1] first builds a rooted tree distributedly. Then the status (inside or outside of the CDS) is assigned for each host based on the level of the host in the tree. Das and Bharghavan in [7] provide the distributed implementation of the two centralized algorithms given by Guha and Khuller [8]. Both implementations suffer from high message complexities. The one given by Wu and Li in [15] has no performance analysis. it needs at least two-hop neighborhood information. The status of each host is assigned based on the connectivity of its neighbors. We summarize the performance comparison of these algorithms in Table 1. The parameters used for comparison include the (upper bound of the) cardinality of the generated CDS, the message and time complexities, the message length and neighborhood information.

Table 1: Performance comparison of the algorithms in [7], [15], [1] and those proposed in this paper. Here opt is the size of the given instance; Δ is the maximum degree; $|C|$ is the size of the generated connected dominating set; m is the number of edges; n is the number of hosts.

	[7]-I	[7]-II	[15]	[1]	A
Cardinality	$\leq (2ln\Delta+3)opt$	$\leq (2ln\Delta+2)opt$	N/A	$\leq 8\,opt+1$	$< 8\,opt$
Message	$O(n\|C\|+m+nlog\,n)$	$O(n\|C\|)$	$O(n\Delta)$	$O(nlog\,n)$	$O(n)$
Time	$O((n+\|C\|)\Delta)$	$O(\|C\|(\Delta+\|C\|))$	$O(\Delta^2)$	$O(n\Delta)$	$O(n\Delta)$
Msg length	$O(\Delta)$	$O(\Delta)$	$O(\Delta)$	$O(1)$	$O(1)$
Information	2-hop	2-hop	2-hop	1-hop	1-hop

Note that the last column (labeled by A) in Table 1 corresponds to our algorithm. We see our algorithm is superior over the two algorithms in [7] for all parameters. Algorithm in [15] takes less time than our algorithm but it has much higher message complexity and it uses more complicated message information. The algorithm in [1] is comparable with our algorithms in many parameters. But from simulation, we know that our algorithm computes small connected dominating set in average.

This paper is organized as follows. Section 2 provides basic concepts related to this topic. Our algorithm and its theoretic performance analysis are presented in Section 3. Simulation result is demonstrated in Section 4. Section 5 concludes the paper.

2 Preliminaries

Given graph $G = (V,E)$, two vertices are *independent* if they are not neighbors. For any vertex v, the set of *independent neighbors* of v is a subset of v's neighbors such that any two vertices in this subset are independent. An *independent set (IS)* S of G is a subset of V such that $\forall u,v \in S$, $(u,v) \notin E$. S is *maximal* if any vertex not in S has a neighbor in S (denoted by MIS).

A *dominating set (DS)* D of G is a subset of V such that any node not in D has at least one neighbor in D. 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 *minimum connected dominating set (MCDS)*. Computing an MCDS in a unit graph is NP-hard [6]. Note that the problem of finding an MCDS in a graph is equivalent to the problem of finding a spanning tree (ST) with maximum number of leaves. All non-leaf nodes in the spanning tree form the MCDS. An MIS is also a DS.

For a graph G, if $e = (u,v) \in E$ `iff` $length(e) \leq 1$, then G is called a *unit-disk graph*. We will only consider unit-disk graphs in this paper. From now on, when we say a "graph G", we mean a "unit-disk graph G". [1] proves the following lemma. This lemma relates the size of any MIS of unit-disk graph G to the size of its optimal CDS.

Lemma 2.1 *[1] The size of any MIS of G is at most* $4 \times opt + 1$, *where opt is the size of any optimal CDS of G.*

For a minimization problem $\mathcal{P}$, the *performance ratio* of an approximation algorithm A is defined to be $\rho_A = sup_{i \in I} \frac{A_i}{opt_i}$, where I is the set of instances of $\mathcal{P}$, A_i is the output from A for instance i and opt_i is the optimal solution for instance i. In other words, ρ is the supreme of $\frac{A}{opt}$ among all instances of $\mathcal{P}$.

3 An 8-approximate algorithm to compute CDS

In this section, we propose a distributed algorithm to compute CDS. This algorithm contains two phases. First, we compute a maximal independent set (MIS); then we use a Steiner tree to connect all vertices in the MIS. We will show that our algorithm has performance ratio at most 8 and is message and time efficient.

3.1 Algorithm description

Initially each host is colored *white*. A dominator is colored *black*, while a dominatee is colored *gray*. we assume that each vertex knows its distance-one neighbors and their effective degrees d^*. This information can be collected by periodic or event-driven hello messages. The *effective degree* of a vertex is the total number of white neighbors.

We also designate a host as the *leader*. This is a realistic assumption. For example, the leader can be the commander's mobile for a platoon of soldiers in a mission. If it is impossible to designate any leader, a distributed leader-election algorithm can be applied to find out a leader. This adds message and time complexity. The best leader-election algorithm (see [3]) takes time $O(n)$ and message $O(n\,log\,n)$ and these are the best-achievable results. Assume host s is the leader.

Phase 1. Host s first colors itself black and broadcasts message `DOMINATOR`. Any white host u receiving `DOMINATOR` message the first time from v colors itself gray and broadcasts message `DOMINATEE`. u selects v as its dominator. A white host receiving at least one `DOMINATEE` message becomes active. An active white host with highest (d^*, id) among all of its active white neighbors will color itself black and broadcast message `DOMINATOR`. A white host decreases its effective degree by 1 and broadcasts message `DEGREE` whenever it receives a `DOMINATEE` message. Message `DEGREE` contains the sender's current effective degree. A white vertex receiving a `DEGREE` message will update its neighborhood information accordingly. Each gray vertex will broadcast message `NUMOFBLACKNEIGHBORS` when it detects that none of its neighbors is white. Phase 1 terminates when no white vertex left.

Phase 2. When s receives message `NUMOFBLACKNEIGHBORS` from all of its gray neighbors, it starts phase 2 by broadcasting message `M`. A host is "ready" to be explored if it has no white neighbors. We will use a Steiner tree to connect all black hosts generated in Phase 1. The idea is to pick those gray vertices which connect to many black neighbors. We will modify the classical distributed depth first search spanning tree algorithm given in [2] to compute the Steiner tree.

A black vertex without any dominator is *active*. Initially no black vertex has a dominator and all hosts are *unexplored*. Message `M` contains a field *next* which specifies the next host to be explored. A gray vertex with at least 1 active black neighbors are *effective*. If `M` is built by a black vertex, its *next* field contains the *id* of the unexplored gray neighbor which connects to maximum number of active black hosts. If `M` is built by a gray vertex, its *next* field contains the *id* of any unexplored black neighbor. Any black host u receiving an `M` message the first time from a gray host v sets its dominator to v by broadcasting message `PARENT`. When a host u receives message `M` from v that specifies u to be explored next, if none of u's neighbors is white, u then colors itself black, sets its dominator to v and broadcasts its own `M` message; otherwise, u defer its operation until none of its neighbors is white. Any gray vertex receiving message `PARENT` from a black neighbor will broadcast message `NUMOFBLACKNEIGHBORS`, which contains the number of active black neighbors. A black vertex becomes *inactive* after its dominator is set. A gray vertex becomes *ineffective* if none of its black neighbors is active. A gray vertex without active black neighbor, or a black vertex without effective gray neighbor, will send message `DONE` to the host which activates its exploration or to its dominator. When s gets message `DONE` and it has no effective gray neighbors, the algorithm terminates.

Note that phase 1 sets the dominators for all gray vertices. Phase 2 may modify the dominator of some gray vertex. The main job for phase 2 is to set a dominator for each black vertex. All black vertices form a CDS.

In Phase 1, each host broadcasts each of the messages `DOMINATOR` and `DOMINATEE` at most once. The message complexity is dominated by message `DEGREE`, since it may be broadcasted Δ times by a host, where Δ is the maximum degree. Thus the message complexity of Phase 1 is $O(n \cdot \Delta)$. The time complexity of Phase 1 is $O(n)$.

In phase 2, vertices are explored one by one. The total number of vertices explored is the size of the output CDS. Thus the time complexity is at most $O(n)$. The message complexity is dominated by message `NUMOFBLACKNEIGHBORS`, which is broadcasted at most 5 times by each gray vertex because a gray vertex has at most 5 black neighbors in a unit-disk graph. Thus the message complexity is also $O(n)$.

From the above analysis, we have

Theorem 3.1 *The distributed algorithm has time complexity $O(n)$ and message complexity $O(n \cdot \Delta)$.*

Note that in phase 1 if we use (id) instead of (d^*, id) as the parameter to select a white vertex to color it black, the message complexity will be $O(n)$ because no `DEGREE` messages will be broadcasted. $O(n \cdot \Delta)$ is the best result we can achieve if effective degree is taken into consideration.

3.2 Performance Analysis

In this subsection, we study the performance of our algorithm.

Lemma 3.2 *Phase 1 computes an MIS which contains all black nodes.*

Proof. A node is colored black only from white. No two white neighbors can be colored black at the same time since they must have different (d^*, id). When a node is colored black, all of its neighbors are colored gray. Once a node is colored gray, it remains in color gray during Phase 1. ∎

From the proof of Lemma 3.2, it is clear that if (id) instead of (d^*, id) is used, we still get an MIS. Intuitively, this result will have a larger size.

Lemma 3.3 *In phase 2, at least one gray vertex which connects to maximum number of black vertices will be selected.*

Proof. Let u be a gray vertex with maximum number of black neighbors. At some step in phase 2, one of u's black neighbor v will be explored. In the following step, u will be explored. This exploration is triggered by v. ■

Lemma 3.4 *If there are c black hosts after phase 1, then at most $c-1$ gray hosts will be colored black in phase 2.*

Proof. In phase 2, the first gray vertex selected will connect to at least 2 black vertices. In the following steps, any newly selected gray vertex will connect to at least one new black vertex. ■

Lemma 3.5 *If there exists a gray vertex which connects to at least 3 black vertices, then the number of gray vertices which are colored black in phase 2 will be at most $c-2$, where c is the number of black vertices after phase 1.*

Proof. From Lemma 3.3, at least one gray vertex with maximum black neighbors will be colored black in phase 2. Denote this vertex by u. If u is colored black, then all of its black neighbors will choose u as its dominator. Thus, the selection of u causes more than 1 black hosts to be connected. ■

Theorem 3.6 *Our algorithm has performance ratio at most 8.*

Proof. From Lemma 3.2, phase 1 computes a MIS. We will consider two cases here.

If there exists a gray vertex which has at least 3 black neighbors after phase 1, from Lemma 2.1, the size of the MIS is at most $4 \cdot opt + 1$. From lemma 3.5, we know the total number of black vertices after phase 2 is at most $4 \cdot opt + 1 + ((4 \cdot opt + 1) - 2) = 8 \cdot opt$.

If the maximum number of black neighbors a gray vertex has is at most 2, then the size of the MIS computed in phase 1 is at most $2 \cdot opt$ since any vertex in opt connects to at most 2 vertices in the MIS. Thus from Lemma 3.4, total number of black hosts will be $2 \cdot opt + 2 \cdot opt - 1 < 4 \cdot opt$. ■

Note that from the proof of Theorem 3.6, if (id) instead of (d^*, id) is used in phase 1, our algorithm still has performance ratio at most 8.

4 Simulation

Table 1 in Section 1 compares our algorithms with others in [1], [7] and [15] theoretically. In this section, we will compare the size of the CDSs computed by different algorithms. As mentioned earlier, the virtual backbone is mainly used to disseminate control packets. Thus the most important parameter is the number of hosts in the virtual backbone after it is constructed. The bigger the size of a virtual backbone, the bigger the number of transmissions to broadcast a message to the whole network. Note that the message complexities of the algorithms in [7] and [15] are too high compared to other algorithms and they need 2-hop neighborhood information. Thus we will not consider them in the simulation study. We will compare our algorithm with the one given by [1].

We assume there are N hosts distributed randomly in a 100×100 square units. Transmission range R is chosen to be 25 or 50 units. Total hosts N is chosen to be 20 or 50 or 100. Also note that we only take connected graph into consideration. We run the algorithm 100 times on different set of parameters including N and R. The averaged results are reported in Table 2.

Table 2: The averaged simulation results (100 runs for each parameter set) for our algorithm (labeled by A) and the one in [1]

N	R	$Avg-degree$	[1]	A
20	25	3.48	11.9	9.59
20	50	9.17	5.54	3.61
50	25	7.64	17.94	13.16
50	50	23.59	6.34	4.11
100	25	15.5	20.44	13.47
100	50	47.39	6.68	4.55

From Table 2, we know that our algorithm is much better than the one given by Alzoubi et. al. in [1]. We also use charts (see Figures 1, 2, 3) to demonstrate the comparison of the 100 runs for some parameter sets. Note that we can not show graphs for all scenarios due to space limit. But in all cases, our algorithm performs consistently.

5 Conclusion

In this paper We provide a distributed algorithm which compute a connected dominating set with smaller size. Our algorithm has performance ratio at most 8 which is the best to our knowledge. Our future work is to study the problem of maintaining the connected dominating set in a mobility environment, thus study the performance of virtual backbone routing.

References

[1] K.M. Alzoubi, P.-J. Wan and O. Frieder, Message-efficient distributed algorithms for connected dominating set in wireless ad hoc networks, *manuscript*, 2001.

[2] H. Attiya and J. Welch, Distributed computing: fundamentals, simulations and advanced topics, *McGraw-Hill Publishing Company*, 1998.

[3] B. Awerbuch, 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*, pp. 230-240, 1987.

[4] V. Bharghavan and B. Das, Routing in ad hoc networks using minimum connected dominating sets, *ICC '97*, Montreal, Canada, June 1997.

[5] I. Cidon and O. Mokryn, Propagation and leader election in multihop broadcast environment, *12th International Symposium on Distributed Computing (DISC98)*, September 1998, Greece, pp. 104-119.

[6] B. N. Clark, C. J. Colbourn and D. S. Johnson, Unit disk graphs, *Discrete Mathematics*, Vol. 86, 1990, pp. 165-177.

[7] B. Das and V. Bharghavan, Routing in ad-hoc networks using minimum connected dominating sets, *Communications, 1997. ICC '97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on*, Vol. 1, 1997, pp. 376-380.

[8] S. Guha and S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica, Vol. 20(4), April 1998, pp. 374-387.

[9] M. Joa-Ng and I.-T. Lu, A Peer-to-Peer zone-based two-level link state routing for mobile Ad Hoc Networks, *IEEE Journal on Selected Areas in Communications*, Vol. 17(8), Aug. 1999, pp. 1415-1425.

[10] D.B. Johnson, D. A. Maltz, Y.-C. Hu and J. G. Jetcheva, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, Internet Draft draft-ietf-manet-dsr-05.txt, March 2001.

[11] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen and J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, *Proc. MOBICOM*, Seattle, Aug. 1999, pp. 151-162.

[12] G. Pei, M. Gerla and T.-W. Chen, Fisheye state routing: a routing scheme for ad hoc wireless networks, *ICC 2000*, Vol. 1, pp. 70-74.

[13] C. E. Perkins, E. M. Royer and S. R. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, Internet Draft draft-ietf-manet-aodv-08.txt, March 2001.

[14] P. Sinha, R. Sivakumar and V. Bharghavan, Enhancing ad hoc routing with dynamic virtual infrastructures, *INFOCOM 2001*, Vol. 3, pp. 1763-1772.

[15] J. Wu and H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, *Proc. of the 3rd International Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications*, 1999, Seattle, WA USA, pp.7-14.

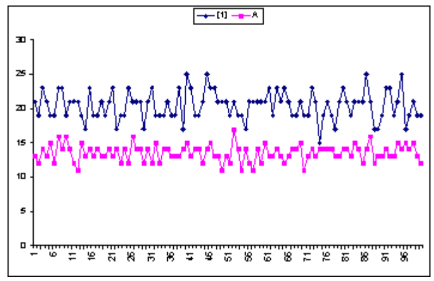

Figure 1: Simulation results for N=100, R=25

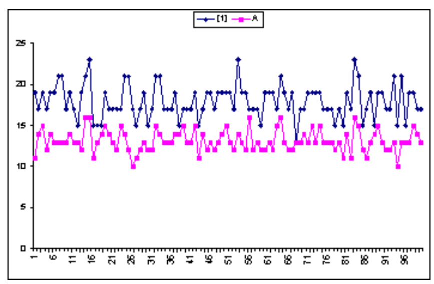

Figure 2: Simulation results for N=50, R=25

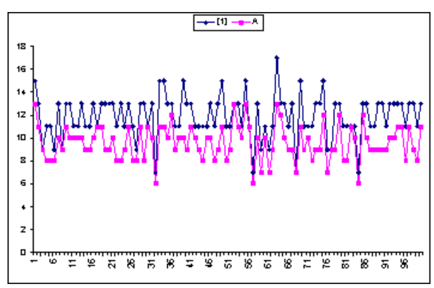

Figure 3: Simulation results for N=20, R=25