# Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 02-002

Virtual Backbone-Based Routing in Multihop Ad Hoc Wireless Networks

Xiuzhen Cheng and Ding-zhu Du

January 22, 2002

# Virtual Backbone-Based Routing in Multihop Ad Hoc Wireless Networks

Xiuzhen Cheng * Ding-Zhu Du *

**Abstract**

Recent research ([12][15][19]) shows that the flooding mechanism (for topology update or route request) used in existing protocols for ad hoc wireless networks greatly degrades the network capacity. If we force the effective broadcasts of control packets happen in a small subset of hosts in the network, the protocol overhead caused by global flooding can be substantially reduced. This is the basic idea of *virtual backbone-based routing* in ad hoc wireless networks, in which all hosts forming the virtual backbone are in charge of control packets dissemination. This strategy works quite effectively in decreasing message overhead. However, constructing a virtual backbone is not trivial. In our study, the virtual backbone is approximated by *minimum connected dominating set (MCDS)* in unit-disk graphs, which is NP-hard [9]. We propose two distributed time/message efficient approximation algorithms to compute MCDS. Both algorithms have linear message complexities. Algorithm I is cost-aware, which accommodates the strict network resources in virtual backbone construction. Algorithm II is degree-aware, which generates the best result in literature so far to our knowledge. We not only give theoretical performance analysis for both algorithms, but also conduct extensive simulation to compare our algorithms with those in literature. Simulation results and theoretical analysis show that both algorithms perform very well.

**Keywords:** Multihop ad hoc wireless network, virtual backbone-based routing, minimum connected dominating set, unit-disk graph.

## 1 Introduction

*Ad hoc wireless network* is an autonomous system consisting of (mobile) hosts (as routers) connected by wireless links. Ad hoc wireless network can be widely and quickly deployed.

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

Examples include *wireless sensor networks* and *military networks*. Ad hoc wireless network has no fixed infrastructure and central administration. Every host can move to any direction at any speed and any time. This induces a *dynamic topology*. Actually even in an environment without mobility, the very *unreliable* wireless links may cause the topology unstable. In an ad hoc wireless network, due to the *broadcast advantage* of wireless communication, the transmission of one host can be heard by all hosts in its communication range. If two hosts are not located in each other's transmission range, intermediate relay hosts must be employed as bridges to build communication paths. This is the *multihop* characteristic of the ad hoc wireless network, for which routing decisions must be made for far-away hosts to communicate. Ad hoc wireless network has a very strict *resource constraint*. Wireless mobiles are usually light-weight and battery-powered. Compared with wired lines, wireless links have much less available bandwidth.

These features — dynamic topology, multihop communication and strict resource limitation — make routing the most challenging problem in ad hoc wireless network. Existing routing protocols (see, [11][13][16][17]) can be classified into three categories: *proactive*, *reactive* and the combination of the two. Proactive routing protocols ask each host to maintain global topology information, thus a route can be provided immediately when requested. Protocols in this category suffer from lower scalability and high protocol overhead; Reactive routing protocols have the feature *on-demand*. Each host computes route for a specific destination only when necessary. Topology changes that do not influence active routes do not trigger any route maintenance function, thus communication overhead is lower compared to proactive routing protocol. The third category maintains partial topology information in some hosts. Routing decisions are made either proactively or reactively. One important observation on these protocols is that none of them can avoid the involvement of *flooding*. For example, proactive protocols rely on flooding for the dissemination of topology update packets; Reactive protocols rely on flooding for route discovery.

Flooding suffers from the notorious *broadcast storm problem* [15]. 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 ongoing traffic. On the other hand, flooding is very *unreliable* [12], which means that *not all* hosts get all the broadcast messages when free from collisions. Sinha et. al. in [19] claimed that "in moderately sparse graphs the expected number of nodes in the network that will receive a broadcast message was shown to be as low as 80%." In reactive protocols, the unreliability of flooding may obstruct the detection of the shortest path, or simply can't detect any path at all, even though there exists a path. In proactive protocols, the unreliability of flooding may cause the global topology information obsolete, thus cause the newly-computed path obsolete.

Recently an approach based on overlaying a virtual infrastructure on an ad hoc network is proposed in [19]. Routing protocols are operated over this infrastructure, which is termed *core*. All core hosts form a dominating set. The key feature in this approach is the new

*core broadcast mechanism* which uses unicast to replace the flooding mechanism used by most on-demand routing protocols. The unicast of route request packets are restricted to core nodes and a (small) subset of non-core nodes. Simulation results when running DSR (Dynamic Source Routing [13]) and AODV (Ad hoc On-demand Distance Vector routing [17]) over the 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 [2][8][20]).

In this paper, we will study the problem of efficiently constructing a virtual backbone for ad hoc wireless network. The basic motivation of our work is the substantial reduction of protocol overhead using the virtual backbone as compared with a pure flooding mechanism in existing protocols. The virtual backbone is mainly used for the dissemination of topology update packets in proactive protocols or route request packets in reactive protocols. It can also serve as a backup when route is unavailable temporarily. Other applications of virtual backbone in ad hoc wireless networks include the propagation of "link quality" information for route selection for multimedia traffic [18], location database servers for mobility management [14], etc. Note that in reactive protocols, the data path from source to destination should be computed separately to avoid overloading those links in the virtual backbone. Sivakumar et. al. in [18] provided an iterative strategy to compute the data path from source to destination with the help of the direction information obtained through the propagation of the route request packets along the core structure.

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 $R$, and the footprint of an ad hoc wireless 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 hosts $u$ and $v$ exists if and only if their distance is at most $R$. In a real 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, as done by most of the existing routing protocols today. We use a *minimum connected dominating set (MCDS)* in unit-disk graphs to approximate the virtual backbone in our study. MCDS has the following properties: (i) the number of hosts is minimized; (ii) all hosts are connected; (iii) all hosts not in the MCDS has a neighbor in MCDS. These considerations come from the application scenarios we are interested in ([7][8][14][18]). All these applications seek a connected dominating set with small size. Computing an MCDS in a unit-disk graph is NP-hard [9]. No exact efficient algorithm is available.

We proposed two distributed, message/time efficient algorithms to compute a virtual backbone, whose hosts form a *connected dominating set*. Both algorithms only need one-hop neighborhood information and both achieve constant performance ratio when cardinal-

ity is the considered parameter. We designate a host as the leader in both algorithm design, which is also an realistic assumption in ad hoc wireless networks (for example, the leader of a platoon of soldiers). The first algorithm is cost-aware. We associate each host with a local cost, which can be interpreted as the inverse of the residual power, the income bitrate, or the inverse of the velocity, etc. These considerations stem from our research on the cost-aware routing protocol design for ad hoc wireless networks [5]. To our best knowledge, this is the only cost-aware algorithm for MCDS in literature. The second algorithm explores the degree information. It has better performance with the cost of higher number of message exchanges. All messages involved in both algorithms have constant length.

Table 1: Performance comparison for the algorithms in [8], [20], [1] and in this paper. Here *opt* is the size of the given instance; $\Delta$ is the maximum degree; $|C|$ is the size of the generated connected dominating set; $n$ and $m$ are the number of hosts and edges respectively.

| | [8]-I | [8]-II | [20] | [1] | Algo-I | Algo-II |
|---|---|---|---|---|---|---|
| Cardinality | $\leq (2ln\Delta+3)opt$ | $\leq (2ln\Delta+2)opt$ | N/A | $\leq 8\,opt+1$ | $\leq 8\,opt+1$ | $< 8\,opt$ |
| Message | $O(n\|C\|+m+nlog\,n)$ | $O(n\|C\|)$ | $O(n\Delta)$ | $O(nlog\,n)$ | $O(n)$ | $O(n)$ |
| Time | $O((n+\|C\|)\Delta)$ | $O(\|C\|(\Delta+\|C\|))$ | $O(\Delta^2)$ | $O(n\Delta)$ | $O(n\Delta)$ | $O(n\Delta)$ |
| Msg length | $O(\Delta)$ | $O(\Delta)$ | $O(\Delta)$ | $O(1)$ | $O(1)$ | $O(1)$ |
| Information | 2-hop | 2-hop | 2-hop | 1-hop | 1-hop | 1-hop |

There exist several distributed algorithms ([1][8][20]) 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. The two algorithms in [8] are the distributed implementation of the two centralized algorithms given by Guha and Khuller [10]. Both implementations suffer from high message complexities. The one given by Wu and Li in [20] has no performance analysis. It needs at least two-hop neighborhood information. 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 the required neighborhood information.

From Table 1, we see our algorithms are superior over the two algorithms in [8] for all parameters. Algorithm in [20] takes less time than our algorithm but it has much higher message complexity and it uses more complicated message information. Algorithm in [1] is comparable with our algorithms in many parameters. But from simulation, we know that our algorithms perform better.

This paper is organized as follows. Section 2 introduces basic concepts and preliminary results needed in our algorithm elaboration. Sections 3 and 4 propose the two algorithms in detail. Simulation results are given in Section 5. Section 6 concludes the paper.

# 2 Preliminaries

Given graph $G = (V, E)$, two vertices are *independent* if they are not neighbors. For $\forall\, u, v \in V$, $hop\_count(u, v)$ is the number of edges (hops) in the shortest path from $u$ to $v$. For any vertex $v$, $N_1[v] = \{u \mid hop\_count(u, v) \leq 1\}$ is the *one-hop (close) neighbor set* of $v$; $N_2[v] = \{u \mid hop\_count(u, v) \leq 2\}$ is the *two-hop (close) neighbor set* of $v$. An *independent neighbor set* of $v$, denoted by $ins(v)$, is a subset of $N_1[v]$ such that any pair of vertices in $ins(v)$ is independent. An *independent set* $S$ of $G$ is a subset of $V$ such that for $\forall\, u, v \in S$, $(u, v) \notin E$. $S$ is *maximal* if any vertex not in $S$ has a neighbor in $S$.

A *dominating set* $D$ of $G$ is a subset of $V$ such that each node not in $D$ has at least one neighbor in $D$. For any edge $(u, v)$, if $u \in D$ and $v \notin D$, then $u$ is $v$'s *dominator* and $v$ is $u$'s *dominatee*. If both $u$ and $v$ are $\in D$, one can specify the other as its dominator. An optimal dominating set has minimum cardinality. 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)*. A maximal independent set is also a dominating set.

We will only consider connected unit-disk graphs in this paper. In any unit-disk graph, a host $u$ can have at most 5 independent neighbors in $N_1[u]$ and at most 11 independent neighbors in $N_2[u] - N_1[u]$. The following lemma relates the size of any maximal independent set $S$ of unit-disk graph $G$ to the size of its optimal connected dominating set $D$.

**Lemma 2.1** *The size of $S$ is at most $4 \times opt + 1$, where $opt$ is the size of $D$.*

Alzoubi et. al. gave a detailed proof of Lemma 2.1 in [1]. The idea is based on the observation that two neighboring nodes can dominate at most 9 independent vertices. Since $D$ is connected, at most one vertex in $D$ can dominate 5 nodes and each of the other vertices can dominate at most 4. Thus Lemma 2.1 holds.

# 3 Algorithm I

Our first algorithm is cost-aware. We assume each host has a local cost – host id, or the inverse of the residual power, or load. We also assume that each host $u$ knows $N_1[u]$ and their costs. These information can be achieved by periodic or event-driven $< hello >$ messages.

As mentioned earlier, we designate a host as the *leader*. If it is impossible to specify any leader, a distributed leader-election algorithm can be applied. This increases message and time complexities. The best leader-election algorithm in literature (see [4]) takes time $O(n)$ and message $O(n \log n)$ and these are the best-achievable results. Algorithm I computes a tree rooted at the leader.

Each host maintains the following parameters: *dom*, which is the dominator, or the parent of the host in the tree; *rank*, which defines a relative relationship among neighboring hosts; (In Algorithm I, rank is the level of the host in the tree.) *children*, which contains all dominated hosts, or the children of the host in the tree. We also associate a color with each host. Strictly speaking, *color* is not a parameter in our algorithm. It is retained in the algorithm description for the purpose of better easier elaboration. These parameters are updated by the exchange of the following 3 messages.

$< dominator(u, dom, l) >$ — host $u$, whose dominator is $dom$ and whose rank is $l$, broadcasts this message to all neighbors. $u$ is a dominator.

$< dominatee(u, dom, l) >$ — host $u$, whose dominator is $dom$ and whose rank is $l$, broadcasts this message to all neighbors. $u$ is a dominatee.

$< active(u) >$ — host $u$ broadcasts this message to all of its white neighbors when it becomes active. A white host becomes active after it receives the first $< dominatee >$ message from one of its neighbors.

## 3.1 Algorithm description

Now we propose our first algorithm. The state transition diagram is given in Figure 1.

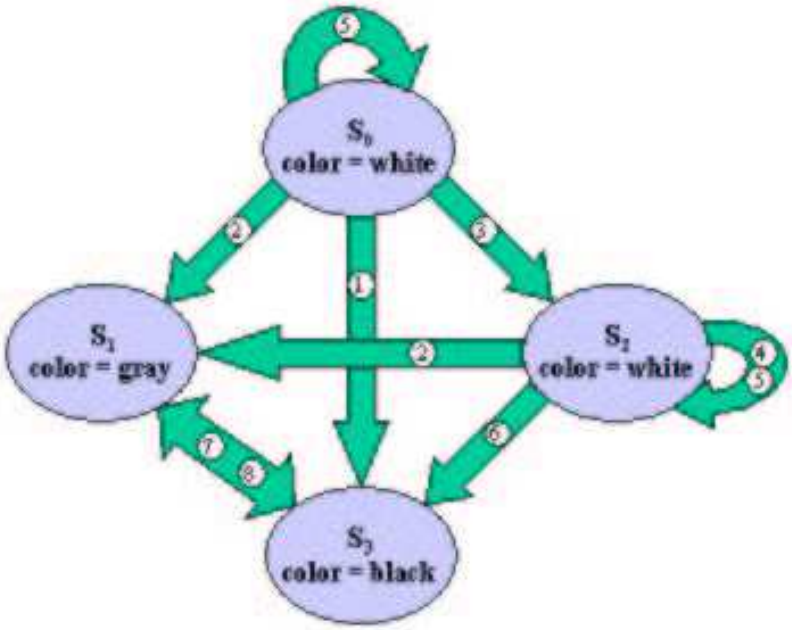


Figure 1: *The state transition diagram of Algorithm I for any host u.*

Each host $u$ runs a copy of the algorithm. At any time, $u$ can be in one of the 4 states:

$S_0$, $S_1$, $S_2$, and $S_3$. The directed arc from $S_i$ to $S_j$, where $i, j = 0, 1, 2, 3$, represents the transition from state $S_i$ to state $S_j$. Each transition is labeled by a number. These transitions will be explained latter.

State $S_0$ is the initial state. A host in this state has white color. All hosts are in $S_0$ at the beginning of the algorithm. State $S_1$ is the dominatee state. A host in this state is a dominatee and has gray color. State $S_2$ is the active state. A host in this state has at least one neighbor in $S_1$ and has white color. An active host is a candidate dominator in next step. State $S_3$ is the dominator state. A host in this state is a dominator and has black color. All hosts in $S_3$ form the connected dominating set.

Initially all hosts are in state $S_0$. If $u$ is the leader, it starts the algorithm and goes to state $S_3$. If $u$ is in state $S_0$ or $S_2$ and one of its neighbors becomes a dominator, $u$ will go to state $S_1$. If $u$ is in state $S_0$ and one of its neighbors becomes a dominatee, it will go to state $S_2$. In $S_2$, $u$ keeps track of the lowest cost gray neighbor, which serves as $u$'s candidate dominator. If $u$ has smallest $(cost, id)$ among all of its active neighbors, it will go to state $S_3$ from $S_2$. If $u$ is in state $S_3$ but its children list is empty, it will go to state $S_1$. If $u$ is in state $S_1$ but it has a non-empty children list, it will go to state $S_3$. These transitions are nontrivial. Each host in state $S_0$ or $S_2$ needs one more parameter $W$, which holds all active white neighbors. Now we are ready for the detailed description of all the transition steps.

1. $u$ is in state $S_0$. If $u$ is the leader, then $dom = u$, $rank = 0$. $u$ will broadcast message $< dominator(u, dom, rank) >$ and go to state $S_3$.

2. $u$ is in state $S_0$ or $S_2$ and receives message $< dominator(v, d, l) >$ from neighbor $v$. If $d$ is a neighbor of $u$, then $dom = d$, $rank = l$; otherwise, $dom = v$, $rank = l + 1$. $u$ will broadcast message $< dominatee(u, dom, rank) >$ and go to $S_1$.

3. $u$ is in state $S_0$ and receives message $< dominatee(v, d, l) >$ from neighbor $v$. $u$ will broadcast message $< active(u) >$ and go to state $S_2$. $u$ temporarily sets $dom$ to $v$ and $rank$ to $l + 1$. If $v$ is in $W$, it will be removed.

4. $u$ is in state $S_2$ and receives message $< dominatee(v, d, l) >$. Remove $v$ from $W$ if it is in $W$. If $cost(v) < cost(dom)$, then $dom = v$ and $rank = l + 1$. $u$ will go back to $S_2$.

5. $u$ is in state $S_0$ or $S_2$ and receives message $< active(v) >$. $W = W \cup \{v\}$. $u$ remains in the original state.

6. $u$ is in state $S_2$ and there is no broadcasting in $N_1[u]$ for $\tau$ time unit (which is a constant). If $u$ has smallest $(cost, id)$ compared with all hosts in $W$, it will broadcast message $< dominator(u, dom, rank) >$ and go to $S_3$.

7. $u$ is in state $S_1$ and receives message $< dominator(v, d, l) >$ from $v$. If $d = u$, $u$ will broadcast message $< dominator(u, dom, rank) >$ and go to state $S_3$.

8. $u$ is in state $S_3$. If $u$ found that none of its neighbors took $u$ as the dominator, $u$ will broadcast message $dominatee(u, dom, rank)$ and go to state $S_1$.

**Remarks:** (i) Algorithm I grows a tree from the leader in a step-by-step fashion. At any time, all the inner nodes of the tree are colored black while all the leaf nodes are colored gray. In the first step, the leader and all of its neighbors are added to the tree. In every other step, a leaf node $v$ and one of its white neighbors $u$ are colored black (added to the tree). All the white neighbors of $u$ and $v$ are colored gray and added to the tree as leaves. This algorithm terminates when no white host left. All the black nodes form the CDS. (ii) Parameter $\tau$ is used to force the start of next step in nearby environment be strictly after the end of the current step. (iii) Each reliable broadcast takes time $O(\Delta)$, where $\Delta$ is the maximum degree. For example, if each broadcast is protected by some kind of handshake protocol such as IEEE 802.11, then the broadcast consists of $\Delta$ consecutive unicasts in data link layer. (iv) When $u$ receives a message $< dominator(v, u, l >$ or $dominatee(v, u, l)$, $u$ will put $v$ into its children list if $v$ is not there. (v) In state $S_1$, a gray host will select the lowest rank black host as its dominator. This can help to optimize the generated CDS. For example, if a black host $u$'s dominated children are also dominated by $dom(u)$, then $u$ can become a dominatee and go to state $S_1$. (vi) If the cost is the inverse of the residual power of each host, the output CDS has higher power capacity; If the cost is the income bitrate (load), the output CDS has lower load; If the cost is the inverse of the host velocity, the induced graph by the output CDS has a more stable topology.

## 3.2 An example

Now we use an example to demonstrate how to apply Algorithm I to compute a connected dominating set. The given unit-disk graph $G = (V, E)$ is shown in Figure 2. There are 9 hosts and 12 links. Host 0 is the leader. We assume the cost is the host id.

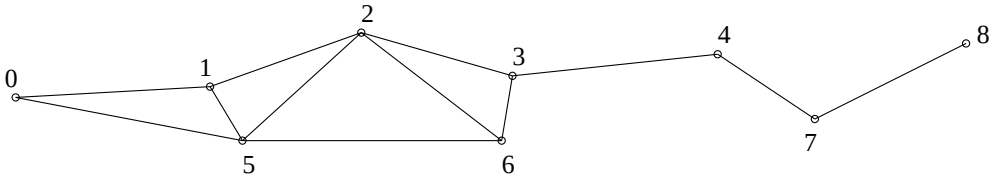


Figure 2: *An example. Unit-disk graph G contains 9 hosts and 12 links.*

For this example, 4 steps are needed to compute the connected dominated set. They are described in detail below.

0. Initially all hosts are in state $S_0$.

1. Host 0 broadcasts message $< dominator(0,0,0) >$ and goes to state $S_3$. After receiving $< dominator(0,0,0) >$, hosts 1 and 5 broadcast messages $< dominatee(1,0,1) >$ and $< dominatee(5,0,1) >$ respectively and go to state $S_1$. After receiving $< dominatee(1,0,1) >$ and/or $< dominatee(5,0,1) >$, hosts 2 and 6 broadcast messages $< active(2) >$ and $< active(6) >$ respectively and go to state $S_2$. After this step, the tree contains inner node $\{0\}$ and leaves $\{1,5\}$.

2. Host 2's $W$ contains only host 6 while host 6's $W$ contains only host 2. Host 2 has smaller $(cost, id)$ compared with host 6, thus host 2 broadcasts message $< dominator(2,1,2) >$ and goes to state $S_3$. Here host 2 selects host 1 as its dominator because host 1 has lower cost compared with host 5. After receiving $< dominator(2,1,2) >$, host 1 broadcasts message $< dominator(1,0,1) >$ and goes to state $S_3$; hosts 3 and 6 broadcast messages $< dominatee(3,2,3) >$ and $< dominatee(6,2,3) >$ respectively and go to state $S_1$. After receiving $< dominatee(3,2,3) >$, host 4 goes to state $S_2$. After this step, the tree contains inner nodes $\{0,1,2\}$ and leaves $\{5,3,6\}$.

3. Host 4 has no active neighbor and it is the only active host. Thus host 4 broadcasts message $< dominator(4,3,4) >$ and goes to state $S_3$. After receiving $< dominator(4,3,4) >$, host 3 broadcasts message $< dominator(3,2,3) >$ and goes to state $S_3$; host 7 broadcasts message $< dominatee(7,4,5) >$ and goes to state $S_1$. After receiving $< dominatee(7,4,5) >$, host 8 broadcasts message $< active(8) >$ and goes to state $S_2$. After this step, the tree contains inner nodes $\{0,1,2,3,4\}$ and leaves $\{5,6,7\}$.

4. Now host 8 is the only active host. Host 8 first broadcasts message $< dominator(8,7,6) >$ and then goes to state $S_3$. After receiving $< dominator(8,7,6) >$, host 7 broadcasts message $< dominator(7,4,5) >$ and goes to state $S_3$. Host 8 detects that its children list is empty, it will broadcast message $< dominatee(8,7,6) >$ and go to state $S_1$. The algorithm terminates. The final tree contains inner nodes $\{0,1,2,3,4,7\}$ and leaves $\{5,6,8\}$.

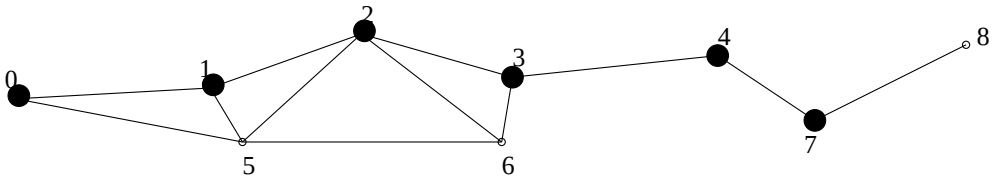


Figure 3: *The computed connected dominating set contains hosts* $\{0,1,2,3,4,7\}$. *The optimal solution contains* $\{1,2,3,4,7\}$.

These steps are summarized in Table 2. The result graph is shown in Figure 3.

Table 2: Summary of the steps applying Algorithm I to Figure 2. Entries contain the state, dominator and children set for each host after each step.

| Host | Step 0 | Step 1 | Step 2 | Step 3 | Step 4 |
|---|---|---|---|---|---|
| 0 | $S_0$,-,- | $S_3$,0,$\{1,5\}$ | | | |
| 1 | $S_0$,-,- | $S_1$,0,- | $S_3$,0,$\{2\}$ | | |
| 2 | $S_0$,-,- | $S_2$,1,- | $S_3$,1,$\{3,6\}$ | | |
| 3 | $S_0$,-,- | | $S_1$,2,- | $S_3$,2,$\{4\}$ | |
| 4 | $S_0$,-,- | | $S_2$,3,- | $S_3$,3,$\{7\}$ | |
| 5 | $S_0$,-,- | $S_1$,0,- | | | |
| 6 | $S_0$,-,- | $S_2$,5,- | $S_1$,2,- | | |
| 7 | $S_0$,-,- | | | $S_1$,4,- | $S_3$,4,$\{8\}$ |
| 8 | $S_0$,-,- | | | $S_2$,7,- | $S_1$,7,- |

## 3.3 Performance analysis

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

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

*Proof.* We have 3 kinds of messages: $< dominator >$, $< dominatee >$, and $< active >$. Each host broadcast each message at most once. Thus the total number of messages is at most $O(n)$.

Each broadcast takes time $O(\Delta)$. We assume that the running time is dominated by message passing instead of CPU processing. Thus the time complexity is $O(n \cdot \Delta)$. ■

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

*Proof.* We can partition all the dominators into two sets: $A$ and $B$. Set $A$ contains all vertices with color changing from *white to black directly* and $B$ contains all vertices with color changing from *white to gray then to black*. The first step adds the leader to $A$. Each of the other steps adds one host to $A$ and one host to $B$. Thus $|A| = |B| + 1$. Note that actually $|A| \leq |B| - 1$ because some host in $A$ may change color to gray if it does not dominate any other hosts.

Now we claim that $A$ is an independent set. This is obvious since each vertex $u$ in $A$ is colored black from white. This means $u$ has no black neighbors because each neighbor of a

black host has gray color. From Lemma 2.1, $|A| \leq 4 \cdot opt + 1$. Thus $|A| + |B| \leq 8 \cdot opt + 1$. ■

Note that theoretically Algorithm I has the same performance as the one given by Alzoubi et. al. [1] when cardinality of the output CDS is considered. But our algorithm has better message/time complexities. Simulation study in Section 5 shows that our algorithm works better.

# 4 Algorithm II

Algorithm II contains two phases. The first phase computes a maximal independent set (MIS) (see Lemma 4.1); The second phase makes the MIS connected (see Lemma 4.2).

## 4.1 Algorithm description

Algorithm II will explore degree information, thus intuitively it should have better performance. The state transition diagram for the first phase is shown in Figure 4.

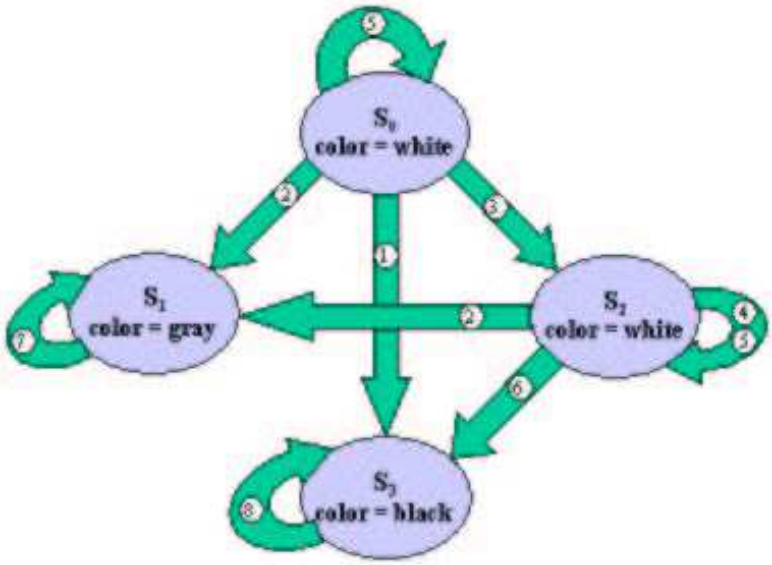


Figure 4: *The state transition diagram for the first phase of Algorithm II for any host.*

Initially all hosts are colored white. During the execution of phase 1, each white host keeps track of its effective degree $d^*$, which records the number of white neighbors. $d^*$ will be updated when a neighbor changes color. To keep the degree information up-to-date, we need a new message $< degree(u, d^*) >$, by which $u$ tells its neighbor that its effective

degree is $d^*$. We also need message $< blackdegree(u, d_b^*) >$, which is broadcasted by $u$, a dominatee, which is adjacent to $d_b^*$ number of *higher rank* black hosts. The "rank" information will be used in phase 2 to connect all members in the generated MIS. The transitions are explained below. Initially $d^* = d$, where $d$ is the degree whose value is $|N_1[u]| - 1$.

1. $u$ is the leader and is in state $S_0$. $u$ will broadcast message $< dominator(u, u, 0) >$ to its neighbors and go to state $S_3$.

2. $u$ is in state $S_0$ or $S_2$ and receives $< dominator(v, x, l) >$ from neighbor $v$. Here $x$ is either the leader or *null*. $u$ will broadcast $< dominatee(u, v, l) >$ and go to $S_1$.

3. $u$ is in state $S_0$ and receives message $< dominatee(v, d, l) >$. $u$ will update $d^*$ to $d^* - 1$, removes $v$ from $W$ if $v \in W$, set *rank* to $l + 1$, broadcast $< active(u) >$, and then go to $S_2$.

4. In $S_2$, $u$ will keep track of all $< dominatee(v, d, l) >$ messages broadcasted in $N_1[u]$ and remove those dominatees in $W$ from $W$. $u$ updates $d^*$ and *rank* accordingly ($rank = 1 + max_v\{l\}$). If there is no broadcasting in $N_1[u]$ for $\tau$ time unit, $u$ will broadcast message $< degree(u, d^*) >$ if $d^*$ is changed after last broadcasting of $< degree >$ message.

5. $u$ is in state $S_0$ or $S_2$. If $u$ receives message $< active(v) >$, then $W = W \cup \{v\}$; if $u$ receives message $< degree(v, d^*) >$, $u$ updates the local record of $v$'s effective degree.

6. $u$ is in state $S_2$ and there is no broadcasting in $N_1[u]$ for $\tau$ time unit. If $u$ has biggest $(d^*, id)$ compared with all hosts in $W$, $u$ will broadcast $< dominator(u, null, rank) >$ and go to state $S_3$.

7. $u$ is in state $S_1$. $u$ keeps track of the number of higher rank black neighbors (number of higher rank dominators in $N_1[u]$, denoted by $d_b^*$). When all of its neighbors are either in $S_1$ or in $S_3$, $u$ broadcasts message $< blackdegree(u, d_b^*) >$.

8. $u$ is in state $S_3$. $u$ keeps track of the gray neighbor with lower rank whose $(d_b^*, id)$ is the biggest, in lexicographic order. This is $u$'s dominator in phase 2.

**Remarks:** (i) Phase 1 generates many stars. Each star consists of one black host, which serves as the center, and many gray hosts dominated by the center. Each gray host in a star sets its *dom* to the center. All hosts in a star have the same rank, which is also the rank of the star. The dominator of a gray host is its first black neighbor. In other words, a gray host $u$ resides in the star centered at the black neighbor which is the first to go to

state $S_3$. (ii) No center has a dominator for all stars except the one the leader resides. The star centered at the leader has rank 0. The rank of any other star $s$ centered at $u$ is one plus the highest rank of $u$'s gray neighbors not in $s$. (iii) We can also understand phase 1 in the following way: phase 1 contains multiple steps. Each step generates a star. At each step except the first one, an active host (in $S_2$) $u$ with maximum effective degree compared with all of its active white neighbors is colored black. All of $u$'s 1-hop white neighbors (in state $S_1$ or $S_2$) are colored gray. All of $u$'s 2-hop white neighbors (in state $S_0$) will go to state $S_2$. (iv) $< degree >$ message is used to announce the number of white neighbors of host $u$ to all of $u$'s white neighbors. With this information, a dominator will be elected in next step. $< blackdegree >$ message is used by gray host $u$ to announce the number of black neighbors whose rank is higher than $u$ to all higher rank black neighbors of $u$. With this information, a black host $v$ can select a gray neighbor with lower rank than $v$ that connects to many stars.

After phase 1, each host is either in state $S_1$ or state $S_3$. A host in state $S_1$ has a dominator in $S_3$ while a host in $S_3$ does not have any dominator (except the leader). Phase 2 will designate a dominator for each black host $u$ (in $S_3$). $u$'s dominator will be the gray host $v$ in $S_1$ such that $rank(v) < rank(u)$ and $v$ has the largest $(d_b^*, id)$ among all lower rank gray neighbors of $u$.


Figure 5: *The state transition diagram for the second phase of Algorithm II for each host.*

The transition diagram for phase 2 is shown in Figure 5. We elaborate the details below.

1. After a black host $u$ (in $S_3$) received $< blackdegree >$ from all of its lower rank gray neighbors, it broadcasts message $< dominator(u, dom, rank) >$, where $dom$ is the lower rank gray neighbor with highest $(d_b^*, id)$. Here $rank$ was set in phase 1.

2. After a gray host $u$ (in $S_1$) received $< dominator(v, u, rank) >$ from $v$, it will broadcast $< dominator(u, dom, rank) >$ and go to state $S_3$. Here the $dom$ and $rank$ parameters for $u$ were set in phase 1.

3. A gray host $u$ in $S_1$ will keep track of the black neighbor $v$ with lowest rank by listening to all $< dominator >$ broadcastings. After all black neighbors (generated in phase 1) determined their dominators, $u$ will broadcast message $< dominatee(u, v, rank(v)) >$

if $v$ is different than its original dominator. Note that here $u$ will be a dominatee after phase 2.

4. $u$ is in state $S_3$ but no gray host selects $u$ as the dominator, $u$ will broadcast message $< dominatee(u, dom, rank) >$ and go to state $S_1$.

**Remarks:** (i) Phase 2 assigns a dominator $v$ to the center $u$ of each star (generated in phase 1) except the star centered at the leader (see Transitions 1 and 2). $v$ is a gray node satisfying the following conditions: $v$ is located in a star with rank lower than $u$ and $v$ is adjacent to maximum number of stars with higher rank than $v$. (ii) After phase 2, the dominator of a gray host $u$ is always the black neighbor with lowest rank. The purpose is to optimize the computed CDS (see Transitions 3 and 4). For example, assume phase 1 generates a star which contains exactly two hosts: black host $u$ and gray host $v$. If in phase 2, $u$ selects host $w$ to be its dominator and $v$ is a neighbor of $w$. Then $v$ will choose $w$ to be its dominator. Since $u$ does not dominate any host after $v$ changes dominator, it will become a dominatee. (iii) Phase 2 is a local process which happens among adjacent stars with different ranks. After a black host $u$ received $< blackdegree >$ from all of its lower rank gray neighbors, it can select its dominator and broadcast message $< dominator(u, dom, rank) >$. This means a black host can initiate phase 2 immediately after all necessary information is available. There is no explicit start time for phase 2. Phase 2 is used to connect all stars.

**Remarks:** (i) There are many similarities between Algorithm I and II. Both algorithms grow a maximal independent set (MIS) from the leader (see Theorem 3.2 and Lemma 4.1). An active host is added into this set if and only if it satisfies some condition. (ii) The major differences are: (a) The criteria to decide whether an active host can be elected to be a member of the MIS is different. In Algorithm I, an active host with minimum cost compared with all of its active neighbors will be added into the MIS. In Algorithm II, an active host with maximum effective degree compared with all of its active neighbors will be added into the MIS. (b) The time when the dominators of the hosts in the MIS are determined is different. Algorithm I assigns a dominator to each host in the MIS when the host satisfies the condition in (a); Algorithm II delays the determination of the dominator to phase 2. Thus Algorithm II can pick the gray host which connects to many number of elements in MIS. We will prove that Algorithm II has better performance than Algorithm I.

## 4.2 An example

Now we show how to apply Algorithm II to the example in subsection 3.2. Initially all hosts are in state $S_0$. Phase 1 contains 4 steps:

1. Host 0 broadcasts message $< dominator(0,0,0) >$ and goes to state $S_3$. After receiving $< dominator(0,0,0) >$, hosts 1 and 5 broadcast messages $< dominatee(1,0,0) >$

and $< dominatee(5,0,0) >$ respectively and go to state $S_1$. After receiving $< dominatee(1,0,0) >$ and/or $< dominatee(5,0,0) >$, hosts 2 and 6 broadcast messages $< active(2) >$ and $< active(6) >$ respectively and go to state $S_2$. After all these broadcasting, hosts 2 and 6 broadcast messages $< degree(2,2) >$ and $< degree(6,2) >$ respectively. This step generates star $(0,\{1,5\})$.

2. Both host 6 and host 2 have effective degree 2, but host 6 has higher $(d^*, id)$, thus host 6 broadcasts message $< dominator(6, null, 1) >$ and go to state $S_3$. After receiving $< dominator(6, null, 1) >$, hosts 2 and 3 broadcast messages $< dominatee(2,6,1) >$ and $< dominatee(3,6,1) >$ respectively and go to state $S_1$. After receiving $< dominatee(3,6,1) >$, host 4 broadcasts message $< active(4) >$ and goes to state $S_2$. After all these broadcasting, host 4 broadcasts message $< degree(4,1) >$. Host 5, 1, 2 broadcast messages $< blackdegree(5,1) >$, $< blackdegree(1,0) >$, $< blackdegree(2,0) >$ respectively. This step generates star $(6,\{2,3\})$.

3. Host 4 is the only active neighbor, thus it broadcasts message $< dominator(4, null, 2) >$ and goes to state $S_3$. After receiving $< dominator(4, null, 2) >$, host 7 broadcasts message $< dominatee(7,4,2) >$ and goes to state $S_1$. After receiving $< dominatee(7,4,2) >$, Host 8 broadcasts message $< active(8) >$ and goes to state $S_2$. After all these broadcasting, host 8 broadcasts message $< degree(8,0) >$. Host 3 broadcasts message $< blackdegree(3,1) >$. This step generates star $(4,\{7\})$.

4. Host 8 is the only active neighbor, thus it broadcasts message $< dominator(8, null, 3) >$ and goes to state $S_3$. After this broadcasting, host 7 broadcasts $< blackdegree(7,1) >$. This step generates star $(8,\{\})$.

After phase 1, hosts $\{0,6,4,8\}$ go to state $S_3$ and all other hosts go to state $S_1$. Phase 2 contains the following steps:

5. After receiving $< blackdegree >$ message from host 5, the only lower rank gray neighbor, host 6 broadcasts message $< dominator(6,5,1) >$. After receiving $< dominator(6,5,1) >$, host 5 broadcasts message $< dominator(5,0,0) >$ and goes to state $S_3$. After receiving $< dominator(5,0,0) >$, host 2 will selects host 5 as its dominator.

6. After receiving $< blackdegree >$ messages from 3, host 4 broadcasts message $< dominator(4,3,2) >$. After receiving $< dominator(4,3,2) >$, host 3 broadcasts message $< dominator(3,6,1) >$ and goes to state $S_3$.

7. After receiving $< blackdegree >$ message from 7, host 8 broadcasts message $< dominator(8,7,3) >$. After receiving $< dominator(8,7,3) >$, host 7 broadcasts message $< dominator(7,4,2) >$ and goes to state $S_3$. Since host 8 does not dominate any other host, it will go to state $S_1$. The algorithm terminates.

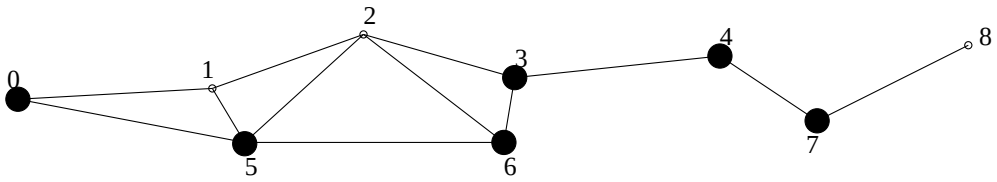


Figure 6: *The computed connected dominating set contains hosts* $\{0,5,6,3,4,7\}$.

The result graph is shown in Figure 6. The state transition are summarized in Table 3.

Table 3: Summary of the steps applying Algorithm II to Figure 2. Entries contain the state, dominator and children set for each host after each step.

| Host | Step 0 | Step 1 | Step 2 | Step 3 | Step 4 | Step 5 | Step 6 | Step 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | $S_0$,-,- | $S_3$,0,$\{1,5\}$ | | | | | | |
| 1 | $S_0$,-,- | $S_1$,0,- | | | | | | |
| 2 | $S_0$,-,- | $S_2$,-,- | $S_1$,6,- | | | $S_1$,5,- | | |
| 3 | $S_0$,-,- | | $S_1$,6,- | | | | $S_3$,6,$\{4\}$ | |
| 4 | $S_0$,-,- | | $S_2$,-,- | $S_3$,-,$\{7\}$ | | | $S_3$,3,$\{7\}$ | |
| 5 | $S_0$,-,- | $S_1$,0,- | | | | $S_3$,0,$\{2,6\}$ | | |
| 6 | $S_0$,-,- | $S_2$,-,- | $S_3$,-,$\{2,3\}$ | | | $S_3$,5,$\{3\}$ | | |
| 7 | $S_0$,-,- | | | $S_1$,4,- | | | | $S_3$,4,$\{8\}$ |
| 8 | $S_0$,-,- | | | $S_2$,-,- | $S_3$,-,$\{\}$ | | | $S_1$,7,- |

## 4.3 Performance analysis

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

**Lemma 4.1** *After phase 1, all hosts in state* $S_3$ *form an MIS. In other words, the centers of all stars form an MIS.*

*Proof.* A node is colored black only from white. No two white neighboring hosts could be colored black at the same time since they must have different $(d^*, id)$. Whenever 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 4.1, 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 4.2** *Phase 2 generates a connected dominating set.*

*Proof.* Note that phase 1 generates many stars whose centers form an MIS. Two stars are adjacent if a gray host in one star is a neighbor of the center of the other star. Since the input graph is connected, a star except the lowest rank one (only one lowest rank star) centered at the leader has at least one adjacent star. Each star has a rank which is higher than the rank of its adjacent stars. Phase 2 connects a star to a lower rank adjacent star by turning a gray vertex in the lower rank star to black. Thus after phase 2, each black vertex (the center of a star) has a path consisting of only black hosts to the leader. All these black vertices form a connected dominating set. ■

**Lemma 4.3** *In phase 2, the gray host with maximum $(d_b^*, id)$ will connect to $d_b^*$ number of lower rank stars.*

*Proof.* Let $u$ be the gray host with maximum $(d_b^*, id)$. Phase 1 ensures that $u$ belongs to the star whose center is the first black neighbor of $u$. For all other black neighbors (centers of some stars) of $u$, their ranks must be greater than that of $u$ since a black host in phase 1 always assigns its rank to be one plus the rank of its highest rank dominatee neighbor. Thus all these $d_b^*$ hosts will select $u$ as their dominators in phase 2. In other words, $u$ connects to $d_b^*$ number of lower rank stars. ■

**Theorem 4.4** *The connected dominating set generated in Algorithm II has size at most $8 \cdot opt$, where opt is the size of any optimal MCDS for the given instance.*

*Proof.* From Lemma 4.1, phase 1 computes an MIS. Let $A$ be this MIS with size $|A|$. From Lemma 2.1, $|A| \leq 4 \cdot opt + 1$. Note that in phase 2, at most $|A| - 1$ hosts in state $S_1$ will go to state $S_3$. Now we consider two cases here.

First, if there exist gray vertices with $d_b^* \geq 2$ at the beginning of phase 2, from Lemma 4.3, the gray vertex $u$ with maximum $(d_b^*, id)$ will connect $d_b^*$ stars to the higher rank star $u$ resides in phase 2. Therefor the number of hosts changing state from $S_1$ to $S_3$ in phase 2 is at most $|A| - 2$. Thus the total number of hosts in state $S_3$ is at most $|A| + |A| - 2 \leq 4 \cdot opt + 1 + 4 \cdot opt + 1 - 2 \leq 8opt$.

Secondly, if all gray vertices have $d_b^* \leq 1$ at the beginning of phase 2, then the number of hosts changing state from $S_1$ to $S_3$ in phase 2 is exactly $|A| - 1$. Since the dominator of any gray vertex $u$ is its first black neighbor, and all other black neighbors of $u$ have higher rank than $u$, thus the total number of black neighbors $u$ has is at most 2. In other words, any host in an optimal MCDS is either in $A$ or adjacent to at most 2 vertices in $A$ and any vertex in $A$ is dominated by a vertex in the MCDS. Therefor in this case, $|A| \leq 2 \cdot opt$. Thus

the total number of black hosts will be $2 \cdot opt + 2 \cdot opt - 1 < 4 \cdot opt$. ■

Our next theorem analyzes the time and message complexities of Algorithm II.

**Theorem 4.5** *Algorithm II has message complexity $O(n)$ and time complexity $O(n \cdot \Delta)$, where n is the total number of vertices and $\Delta$ is the maximum degree.*

*Proof.* In Algorithm II, the message complexity is dominated by the $< degree >$ messages broadcasted by white vertices in $S_2$ in phase 1 since each host broadcasts each of the other messages at most twice. But how many times does a host broadcast the $< degree >$ message? From Lemma 4.1, we know phase 1 builds an MIS. This tells us that at most 11 hosts in $N_2[u] - N_1[u]$ will go to state $S_3$ for any host $u$ in $S_2$. Note that parameter $\tau$ is introduced to force the start of next step be strictly after the end of the current step in the surrounding environment of $u$ and $< degree >$ message is broadcasted only after all neighbors of $u$ are silent for at least $\tau$ time unit, thus the maximum number of $< degree >$ broadcasting for $u$ is at most 11. From this analysis, we know the message complexity is $O(n)$.

The argument for time complexity is trivial. All messages are broadcasted in parallel or one after the other and each broadcasting takes time $O(\Delta)$, thus the time complexity is $O(n\Delta)$. ■

# 5 Simulation

Table 1 in Section 1 compares our algorithms with others in [1], [8] and [20] 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 and time complexities of the algorithms in [8] are too high compared to other algorithms. Thus we will not consider them in the simulation study. We will compare our algorithms with those given by [1] and [20]. In the simulation, we only take connected graph into consideration.

We assume there are $N$ hosts distributed randomly in a $100 \times 100$ square units. Transmission range $R$ is chosen to be 15 or 25 or 50 units. For each value of $R$, we run our algorithms 500 times for different values of $N$. The averaged results are reported in Figures 7, 8 and 9. Note that in all these figures, x-axis corresponds to $N$ while y-axis corresponds to the size of the generated connected dominating set. From these figures we know that in all of our simulation scenarios, Algorithm II performs better than Algorithm I and both per-

form better than the algorithms in [1] and [20]. For dense and large graphs, our algorithms perform even better.

Now we know that the averaged results of our algorithms are good. How about their behaviors for individual $N$ and $R$? As we can not report the results of all combinations of $N$ and $R$ used in the simulation, we show the charts for $N = 225$ and $R = 15$ in Figure 10, $N = 49$ and $R = 25$ in Figure 11, $N = 49$ and $R = 25$ in Figure 12. Other combinations of $N$ and $R$ perform similarly.

Note that in Figures 10, 11 and 12, to make the charts readable, we only show the results for 100 runs. From these charts, we know that our algorithms perform consistently for different random graphs when $N$ and $R$ are fixed.

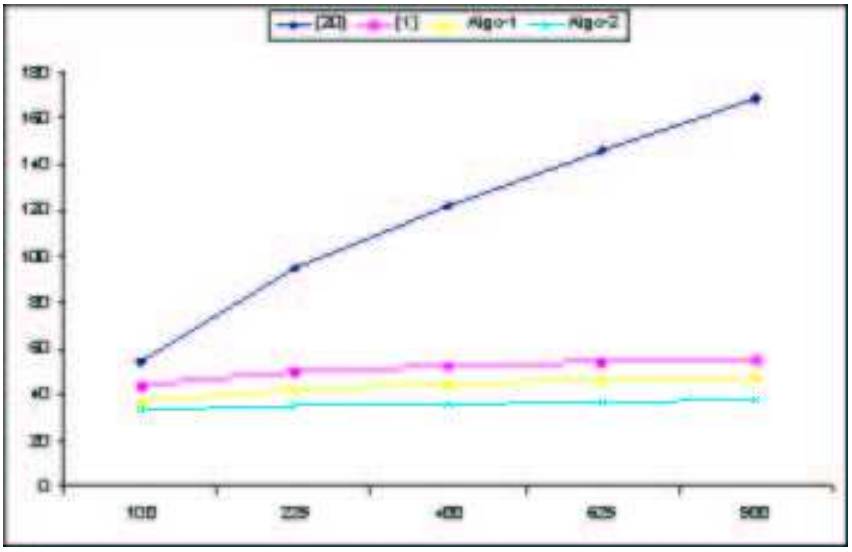


Figure 7: *Average result for R=15.*

# 6 Conclusion

Virtual backbone-based routing in ad hoc wireless networks has the advantage of substantially reducing the message overhead compared with pure flooding mechanism used in most existing protocols. we studied the problem of constructing a virtual backbone effectively and efficiently in an ad hoc wireless network environment. We use a minimum connected dominating set to approximate the virtual backbone. Two distributed message/time efficient algorithms were proposed. Their performance is witnessed by both theoretical analysis and simulation. These works are more significant when the underlying network is large and the

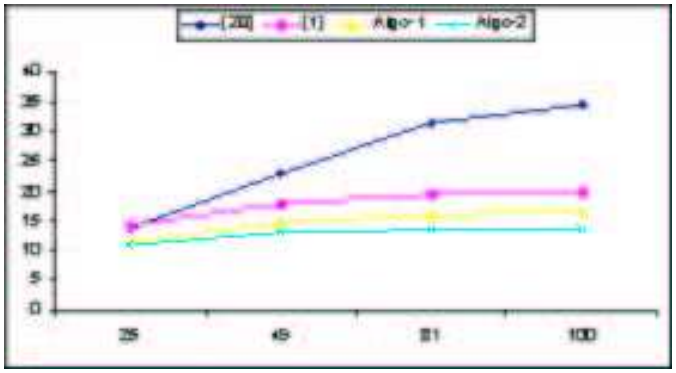


Figure 8: *Average result for R=25.*

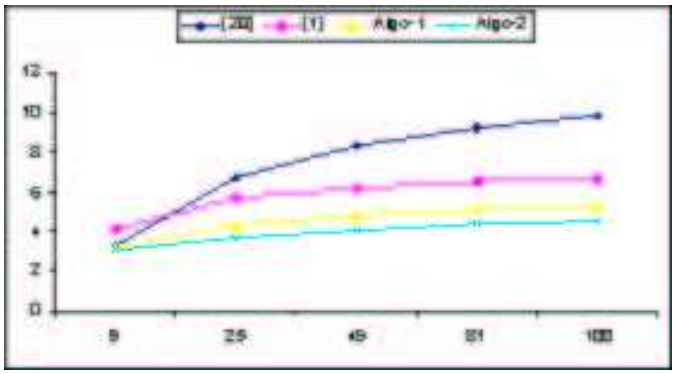


Figure 9: *Average result for R=50.*

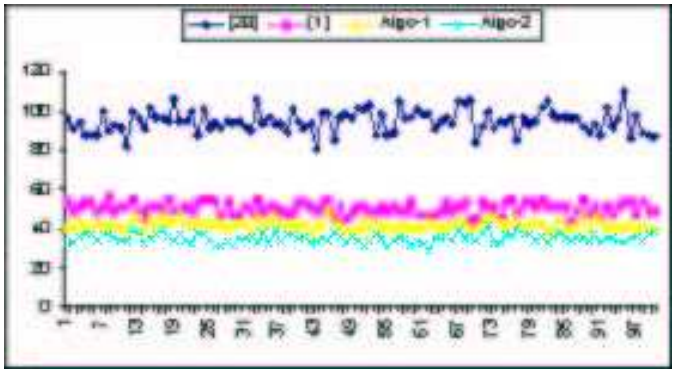


Figure 10: *Simulation results for N=225, R=15*

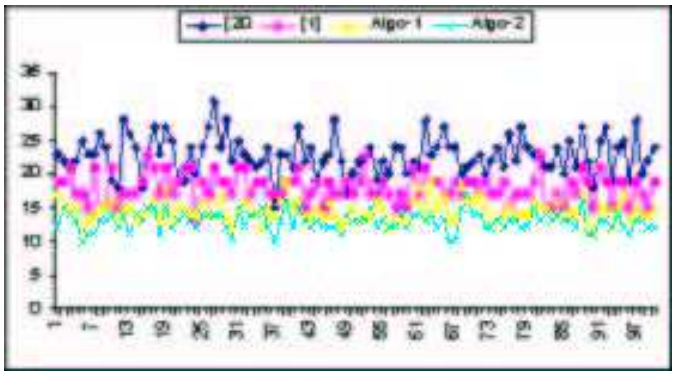


Figure 11: *Simulation results for N=49, R=25*

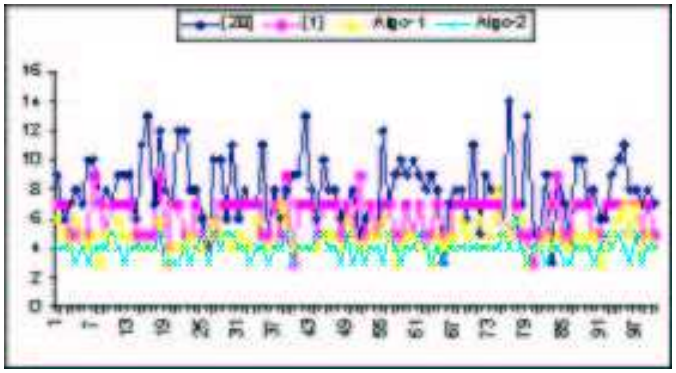


Figure 12: *Simulation results for N=49, R=50*

topology is dense. Based on these algorithms, currently we are designing a protocol which takes into account the rigid network resources [5].

# References


[1] K.M. Alzoubi, P.-J. Wan and O. Frieder, New distributed algorithm for connected dominating set in wireless ad hoc networks, to appear in *Proc. HICSS 2002*.

[2] A.D. Amis and R. Prakash, Load-balancing clusters in wireless ad hoc networks Application-Specific Systems and Software Engineering Technology, *Proc. 3rd IEEE Symposium on*, 2000, pp. 25-32.

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

[4] 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.

[5] X. Cheng and D.-Z. Du, Cost-aware routing protocol for ad hoc wireless networks, *in preparation*.

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

[7] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, and L. Viennot, Optimized link state routing protocol, IETF Internet Draft, draft-ietf-manet-olsr-05.txt, October 2001.

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

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

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

[11] 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.

[12] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek and M. Degermark, Scenario-based performance analysis of routing protocols for mobile ad hoc networks, *Proc. IEEE MOBICOM*, Seattle, Aug. 1999, pp. 195-206.

[13] 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.

[14] B. Liang and Z.J. Haas, Virtual backbone generation and maintenance in ad hoc network mobility management, *INFOCOM 2000*, Vol. 5, pp. 1293-1302.

[15] 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.

[16] 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.

[17] 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.

[18] R. Sivakumar, P. Sinha and V. Bharghavan, CEDAR: a core-extraction distributed ad hoc routing algorithm, *Selected Areas in Communications, IEEE Journal on*, Vol. 17(8), Aug. 1999, pp. 1454 -1465.

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

[20] 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.