On the Construction of Stable Virtual Backbones in Mobile Ad-Hoc Networks

Feng Wang
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55108

Manki Min
Dept. of Industrial Engineering
University of Florida
Gainesville, FL 32611

Yingshu Li
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55108

Dingzhu Du
Dept. of Computer Science
University of Minnesota
Minneapolis, MN 55108

Abstract

In mobile ad-hoc networks, hosts communicate with each other without the help of any physical infrastructure. Inevitably, the communication tends to be less efficient in terms of computational and communicational overhead. Recent studies have shown that virtual backbone can help reduce the communication overhead. However, the backbone structure is very vulnerable due to several reasons, e.g., node mobility and unstable links, etc. In this paper, we introduce a localized virtual backbone construction scheme, connected maximal independent set with multiple initiators(MCMIS), which takes node stability into consideration and can construct the backbone quickly. We design MCMIS aiming at three goals: small backbone size, fast construction, stable backbone. Through extensive simulations, we find that our scheme could obtain a better performance on stability and backbone size than other localized schemes.

1 Introduction

Mobile ad-hoc networks (MANET) are formed of mobile nodes without any underlying physical infrastructure. In order to enable data transfers in such networks, all the wireless nodes need to frequently *flooding* control messages thus causing a lot of redundancy, contentions and collisions (known as "broadcast storm problem" [6]). Inspired by the backbones in wired networks, e.g., today's Internet, many recent studies have focused on using *virtual backbone* as the routing infrastructure of wireless ad hoc networks [7]. With virtual backbones, routing messages are only exchanged between the backbone nodes, instead of being broadcasted to all the nodes. These studies have demonstrated through simulations that virtual backbones could dramatically reduce routing overhead. Moreover, virtual backbones can also be applied to (1) provide a backup route, (2) multi-cast or broadcast messages [12], (3) simplify connectivity management, and (4) reduce the overall energy consumption [11]. Hence, it is very important to construct a reliable and small-scale virtual backbone for MANET.

A virtual backbone with a small size is desirable in MANET since constructing and maintaining virtual backbone impose extra control overhead and such overhead increases as the size of virtual backbone increases. Furthermore, the role of virtual backbone requires that all backbone nodes are connected and dominate all the rest nodes. Hence a minimum connected dominating set (MCDS) can make a good candidate. We model the network topology with a unit disk graph (UDG) since we assume that every node has the same transmission range. Finding a MCDS in UDG is a NP-hard problem. Therefore, approximation of MCDS construction needs to be studied to construct virtual backbones.

In MANET, network topology is dynamic due to node mobility as well as instability of links. Therefore, constructing backbone should pro-actively take node mobility into consideration and a virtual backbone with a longer lifetime[1] is preferred. At the same time, localized algorithm can construct the backbone quickly and spatial reuse the wireless channel, thus suitable for MANET environment. In our previous work [4], we considered the mobility in the construction, while the algorithm assumes the existence of one initiator, which is not true for some wireless applications.

In this paper, we propose a localized backbone construction scheme, namely *connected maximal independent set with multiple initiators* (MCMIS), consist-

[1]The life time of a backbone refers to the earliest time that the backbone gets disconnected or an uncovered node appears

ing of two interleaved phases: forest construction and merging on conflicts. We design MCMIS aiming at three goals: small backbone size, fast construction and stable backbone. To address the small backbone size requirement, we propose a new connecting scheme. For fast construction, we design a completely localized algorithm, where every node makes local decisions, thus construction of virtual backbone can be conducted in parallel. To obtain a stable backbone, we define the node rank as a tuple of *(stability, effective_degree, id)*. Using this ranking scheme, we significantly prolonged the lifetime of the constructed backbone without increasing the backbone size.

Through extensive simulations, we found that compared with a localized algorithm in [10], the backbone constructed with MCMIS has a smaller size and longer average lifetime. Specifically, with 75 nodes in a $400 \times 400m^2$ area and the transmission range of each node as $100m$, the average backbone size generated by MCMIS is about 20% less than that generated by [10], while the backbone lifetime is about 60% longer. We have observed similar performance improvement for other network topologies during simulations.

The remainder of this paper is organized as follows. Section 2 surveys some existing schemes of virtual backbone constructions. In section 3, we explain and analyze our backbone scheme *MCMIS*. Then we present the simulation results indicating the backbone size and reliability of MCMIS in section 4. Finally, section 5 concludes our current studies and outlines the future work.

2 Related Work

In this section, we briefly survey some existing schemes of constructing connected dominating set in wireless networks. We classify the previous work into three categories: centralized, sequentially distributed and localized. Sequentially distributed algorithms differ with localized ones in that sequentially distributed algorithms require some global information even it is collected in a distributed way. For example, every node needs to know that there exists a leader or initiator and whether it is the leader or initiator. While localized ones do not need any global information. In most of the CDS construction algorithms, a coloring mechanism is used where initially all the nodes are colored white, a node in the CDS (dominator) is colored black or red and a node not in the CDS (dominatee) is colored gray after the execution of the algorithms.

2.1 Centralized algorithms

Guha and Khuller [5] first proposed two centralized greedy algorithms. The number of white neighbors of each node or a pair of nodes (a dominatee with one of its white neighbor) is the greedy function. The one with the largest such number will become dominator(s) at each step. Both of these algorithms require the global information which is not practical for wireless networks. Das *et al.* [7] first proposed using a CDS as a virtual backbone for routing and gave implementations of [5]. They also mentioned the maintenance of the CDS if nodes have mobility.

2.2 Sequentially distributed algorithms

Two main schemes [8] [4] are in this category. Both use the idea of constructing MIS and connecting it into a CDS. [8] requires a *leader election* [3] phase to elect a leader node and generate a spanning tree rooted on the leader node before constructing MIS.

The first phase of Wan et al's scheme [8] determines the level of each node (the number of hops between itself and the root of the spanning tree which is constructed by the leader election procedure) and finds MIS nodes. The ranking they used is an ordering of the level and ID pair. MIS nodes consist of independent nodes with higher ranking than their neighbors. Every pair of two complementary sets of MIS nodes are separated by exactly two hops. After an MIS is formed, a *dominating tree* is constructed and the tree is rooted at the grey neighbor of the leader node with maximum black degree. They proved that the scheme has a performance ratio of 8 based on the property of MIS stated above. Moreover, they established the lower bound $\Omega(n \log n)$ on message complexity of distributed algorithms for CDS construction which supports the tightness of the message complexity of their scheme.

Min *et. al.* [4] also adopts the approach of connecting a MIS set. It differs to [8] in three aspects: 1) they avoid the leader election phase which introduces additional time and message overhead. 2) they interleave the processes of creating MIS and interconnecting the set for construction efficiency. They propose a multiple rounds coloring process. Black nodes in round i will immediately connect to black nodes in round $i-1$ through gray nodes in round $i-1$. 3) they proactivly capture node mobility as a criteria of selecting backbone nodes, therefore improving the stability of the constructed backbone in mobile networks.

2.3 Localized algorithms

Alzoubi *et. al.* [9] propose a localized algorithm of constructing MIS and interconnecting them. An MIS is generated in a distributed fashion without building a tree or selecting a leader. Once a node knows that it has the smallest ID within its 1-hop neighborhood, this node becomes a dominator. After there are no white nodes, the dominators are responsible for iden-

tifying a path to connect all the dominators. They also prove the performance ratio as 192.

Wu and Li [10] proposed a localized algorithm called marking process where each node knows the connectivity information within the 2-hop neighborhood. If a node has two unconnected neighbors, it becomes a dominator. The generated CDS is easy to maintain. But the size of the CDS is large. Thus they designed two rules to prune the generated CDS. The performance ratio was not specified. In [8], the authors gave the performance ratio of this algorithm and correct the time complexity and message complexity.

Table 1 lists some measurement parameters of these algorithms where Δ is the maximum degree in the graph; $|C|$ is the size of the resultant CDS; n and m are the number of the vertices and edges in the graph and opt is the size of an optimal MCDS. PR stands for the performance ratio.

	PR	Time	Message	Scheme
[8]	$8opt + 1$	$O(n)$	$O(n \log(n))$	Non-localized
[4]	$8opt$	$O(n)$	$O(n\Delta)$	Non-localized
[9]	$192opt + 48$	$O(n)$	$O(n)$	Localized
[10]	$O(n)$	$O(\Delta^3)$	$\Theta(m)$	Localized

Table 1: Peroformance comparison of the CDS construction algorithms

Our algorithm belongs to the localized category. Two most related work are [9] and [4]. Compared with [9], we propose to construct multiple trees and then connect them into a whole virtual backbone for reducing the backbone size, also we proactivly take mobility into consideration when constructing a backbone. Our work is an extension of [4], which also proactively consider mobility during the CDS construction. We use effective_degree instead of coverage to measure the coverage of a node because calculating coverage is computation expensive, also we design a localized algorithm which is more suitable for wireless networks.

3 Connected Maximal Independent Set with Multiple Initiators(MCMIS)

A localized algorithm has the following features which suit wireless networks very well: first, it does not require global information. Every node makes decision only based on its local information(one or two-hop neighborhood); second, it can spatial reuse the wireless channel with utilizing the parallel channel capability in wireless networks.

In this paper, we propose an efficient localized algorithm, MCMIS, to construct a stable and small-scale virtual backbone.

MCMIS composes two phases:

1. Constructing a forest: a forest consisting of multiple dominating trees rooted at multiple initiators is constructed. A dominating tree is rooted at an initiator, and composes a subset of nodes in the topology. The construction of each dominating tree is started by its initiator, and multiple trees are constructed in parallel;

2. Merging on growth conflicts: the dominating trees in the forest are interconnected into a complete virtual backbone when branches of trees encounter with each other.

During the execution of the algorithm, a distributed coloring process is employed to color nodes in the network with three colors: black, gray, and red. After the execution of the algorithm, the node will be marked either black, red, or gray. All black nodes compose an Maximal Independent Set(MIS) of the network, all red nodes are the nodes connecting the MIS into a virtual backbone. Thus all black nodes and red nodes compose the virtual backbone. All gray nodes are dominatees.

In this section, we first introduce the ranking scheme, then present two phases in detail, finally we give the proof of correctness and the theoretical performance analysis.

3.1 Node Ranking

Node ranking is used for a node to decide whether it is an initiator, and for breaking ties when two candidate nodes can be chosen into the backbone, or two candidiate nodes can be chosen as the connectors.

We define the node ranking as an tuple of *(stability, effective_degree, id)*. Right parameter has higher order than the left one. Stability is for measuring mobility and effective_degree is for estimating the coverage of a node. For a white node, effective_degree is the number of its white neighbors. For nodes with other colors, effective_degree is its degree.

The stability of each node can be estimated using its previous location information since there is usually temporal and spatial locality in node movement.

The stability of a node v, s_v, is defined as the reciprocal of the sum of the distance between its initial location and the locations for 10 consecutive seconds, formally,

$$s_v = \frac{1}{\sum_{i=1}^{10} \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}}, \qquad (1)$$

where (x_0, y_0) is the initial location of the node, v, and (x_i, y_i) is the location of the node during the

ith second. Clearly, the more a node moves, the less stability it has.

3.2 Forest construction

A forest consists of multiple trees. Starting from multiple initiators, multiple dominating trees composing the forest are constructed in parallel.

To distinguish different dominating trees, we introduce two terms, i) *parent* of a node u: the node which is in the same dominating tree as u and decides the u's color; ii) *root* of a node u: The initiator of the dominating tree where u belongs. Note that every node has only one root and one parent.

The algorithm is executed through coloring the nodes in multiple rounds. The coloring procedure is illustrated in Fig. 1. Initially every node has white color. Let B_i and G_i be the set of nodes colored black and gray at the i-th round respectively. In the 1st round, the initiator (the node with the highest rank in its one-hop neighborhood) is colored black and all its direct neighbors (G_1) are colored gray. In the *ith* round ($i \geq 2$), among all the white nodes that have gray neighbors in G_{i-1}, denoted as W_i, the nodes with the highest rank among all its white neighbors in W_i will be colored black. The neighbors of newly colored black nodes(B_i) are colored gray(G_i). Nodes in B_i then select the gray neighbors in G_{i-1} with the highest rank as the connector which turns into red.

Each gray node in G_{i-1} must have at least one neighbor in B_{i-1} and 0 or more neighbors in B_i. Each black node in B_i must have at least one gray neighbor in G_{i-1}, thus it is guaranteed to be able to find gray nodes in G_{i-1} to connect B_i nodes to B_{i-1} nodes. In Fig. 1, we already know that every G_1 node is connected to the B_1 node which is the initiator, thus we can guarantee to form a tree (thick lines and nodes in the figure) using our algorithm. We call this tree dominating tree. Note that the leaves of the dominating tree can only be black or gray, but could not be red. Gray nodes can only be leave nodes. Every node records its parent information during the coloring process to maintain the tree structure R_1 in Fig. 1 represents connectors at the 1st round.

The dominating tree construction algorithm is described in Algorithm 1. Each node calculates its own rank as described in Section 3.1. Each node maintains its parent and root information and gray nodes keep a list of black neighbors.

There are five types of messages: *black, gray, blackto-be, red*, and *connect*. The first four are for a node to announce its own status, *connect* message is for a black node to choose a gray neighbor as connector. The formats of the messages are defined as follows:

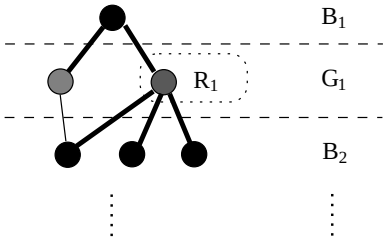


Figure 1: Interconnecting backbone nodes

- black: *black, id, stability, degree, rootid*
- blacktobe: *blacktobe, id, stability, effective_degree, parentid, rootid*
- gray: *gray, id, stability, degree, parentid, rootid*
- red: *red, id, stability, listofBlackNeighbors*
- connect: *connect, id*

Algorithm 1 Dominating tree construction

```
1: Initiator := the white node with highest rank among its one-hop neigbhorhood;
2: Initiator turns into black and broadcasts black message;
3: if a white node u receives black messages then
4:    u turns into gray and broadcasts gray message;
5: end if
6: if a white node u receives gray messages then
7:    u broadcasts blacktobe message;
8: end if
9: if a white node u has broadcasted blacktobe message then
10:   if u receives black message then
11:      u turns into gray and broadcasts gray message;
12:   end if
13:   if u has the highest rank among the senders of blacktobe messages it received or all its neighbors with higher ranks which have sent blacktobe message have turned gray then
14:      u turns into black and broadcasts black message;
15:      u picks its gray neighbor with highest rank and unicast connect message to the gray node;
16:   end if
17: end if
18: if a gray node u receives connect message then
19:   u turns into red and broadcasts red message;
20: end if
```

3.3 Merging on growth conflicts

In this section, we introduce the definition of growth conflicts and investigate all possible conflicts, then we present our scheme of resolving the conflicts to aggregate multiple dominating trees and illustrate the localized algorithm.

Growth Conflicts. In the forest construction phase, since multiple trees are constructed in parallel, some branches of some dominating trees will encounter

each other. We call it *growth conflicts*. Note that the dominating tree may continue growing at other branches. There are two possible conflicts: black_gray conflict and gray_gray conflict. Black_gray conflict happens when a black leaf node in one tree meets a gray leaf node in another tree; gray_gray conflict happens when a gray leaf node in one tree meets a gray leaf node in another tree. In both cases, one or two gray nodes need to be colored as connectors. There is no black_black conflict since: 1) any two initiators could not be neighbors, 2) after a white node broadcasts a black-to-be message and before it changes color, either it has black neighbors so it turns into gray instead of black, or if it turns into black, all its neighbors turn into gray. There are no conflicts involving red nodes either since in one dominating tree, red nodes are interconnecting nodes, it could not be a leaf node. So we only need to resolve black_gray conflict and gray_gray conflict.

Merging. To aggregate multiple dominating trees into one virtual backbone, more nodes need to be added to serve as connectors when two branches have growth conflicts. Our main idea is to let black nodes choose the connector because black nodes are already in the backbone. If there is a black_gray conflict, the black node requests a gray node to serve as the connector. If there is gray_gray conflicts, one of the two black parents of these two gray nodes requests the two gray nodes to serve as the connectors. In order to construct a small-sized virtual backbone, the number of connectors to connect multiple dominating trees is preferred to be small. The key ideas in our scheme for reducing the number of connectors are:

1. Before a black node requests gray node(s) as connector(s), it needs to first check whether it has connected to the other dominating tree through other connectors. Note we check for the connectivity of this black node to the other dominating tree, not the connectivity of this black node to the black parent of the gray node in the other tree.

2. Find the shortest path to interconnect two black nodes in two dominating trees. Due to the delay of exchanging messages among nodes, if there exists both a 2-hop path and a 3-hop path between two black nodes in different trees, the black nodes will learn the existence of the 2-hop path earlier than the existence of the 3-hop path. We let black node to choose the connector, so the black node will choose the node on the shorter path as connector;

3. choose only one of the multiple paths to connect two black nodes. After a black node chooses a connector to connect to another black node, it will delay the decision of whether to choose another connector until it receives the acknowledge from the connector it just chose and it remembers that it has been connected to the tree the black node in and ignores other connectors to connect to the same tree.

4. Let lower ranked tree connects to higher ranked tree. If there is a gray_gray conflict, only the parent with the lower rank initiates the connecting. The heuristic is to let the less stable tree connect to the more stable one.

Implementation for merging on conflicts. In addition to the five types of messages in section 3.2, two new messages, *gray2, connect2*, are defined to resolve the gray_gray conflict. The same *connect* message as defined in section 3.2 is used to resolve the black_gray conflict.

The formats of gray2 and connect2 messages are defined as below:

- gray2: *gray2, id, stability, parentofGrayNeighbor, rootofGrayNeighbor*

- connect2: *connect2, id*

In our design, each node maintains the id of its parent and root, a list of all black nodes that it has been connected to and their roots, and a list of all gray messages it has received but not processed. After the virtual backbone is constructed, a black node knows how to reach its backbone neighbors in 3-hop neighborhood, a red node knows its 1-hop backbone neighbors, and a gray node knows its dominators in the backbone. Algorithm 2 presents the localized implementation of the merging process.

3.4 An example

Fig. 2 illustrates the steps to mark 10 nodes with the given topology. The lower the number marked on the node, the higher the rank it has. There are two dominating trees rooted at 0 and 2, and there is a black_gray conflict between 0 and 7, and also gray_gray conflicts between 1 and 6, and 8 and 9. The reason why 7 is picked as a connector is due to the fact that 2 knows 7 as a connector one message-exchanging time before 2 learns that 6 and 9 can also be connectors. Further, 2 will only add new connectors after receiving a red message from 7 which tells it that it has been connected to tree 0, so 2 will not mark 6 or 9 as a connector. Because 0 has a lower rank than 2, it will

Algorithm 2 Merging on conflicts

1: **if** a gray node u in T_i receives gray msg for T_j **then**
2: u remember the sender id and unicasts gray2 msg to its parent;
3: **end if**
4: **if** a black node u in T_i receives gray msg for T_j and u has not been connected to T_j **then**
5: u unicast connect msg to the sender; insert root and parent of the sender into black node list and lock its black node list;
6: **end if**
7: **if** a black node u receives red msg **then**
8: unlock its black node list; add new root ids in the msg to black node list;
9: **end if**
10: **if** a black node u in T_i receives gray2 msg for T_j and u has not been connected to T_j and u has smaller rank than the parent of the other gray node **then**
11: u unicast connect2 msg to the sender; insert root and parent info kept in the message into black node list and lock its black node list;
12: **end if**
13: **if** a gray node u receives connect msg **then**
14: u turns into red and broadcasts red msg;
15: **end if**
16: **if** a gray node u receives connect2 msg **then**
17: u turns into red and broadcasts red msg; unicast connect msg to the gray msg sender it has remembered;
18: **end if**
19: **if** a red node u receives connect msg **then**
20: red node updates its black node list; encapsulates its new 2-hop black neighbor and its root id in red msg and broadcasts red msg;
21: **end if**

not mark 1 or 8 (thus 6 or 9) as a connector either. Note that tree 0 continues growing in the left direction while it merges with tree 2 in the right direction. The forest construction and merging on conflicts phases are interleaved.

3.5 Correctness and Performance Analysis

Lemma 1 *All black nodes B_i in a dominating tree T_i form an MIS, where $1 \leq i \leq$ the number of initiators.*

Proof. The dominating tree construction incrementally enlarges the black node set by adding black nodes 2 hops away from the previous black nodes set, also the newly colored black nodes could not be adjacent to each other, hence every black node is disjoint from other black nodes. This implies that B_i forms an independent set. Further, every gray or red node must have at least one black neighbor, so if coloring any gray or red node black, B_i will not be disjoint anymore. Hence B_i is a maximal independent set. □

Lemma 2 *All black nodes B generated by* MCMIS *form an MIS.*

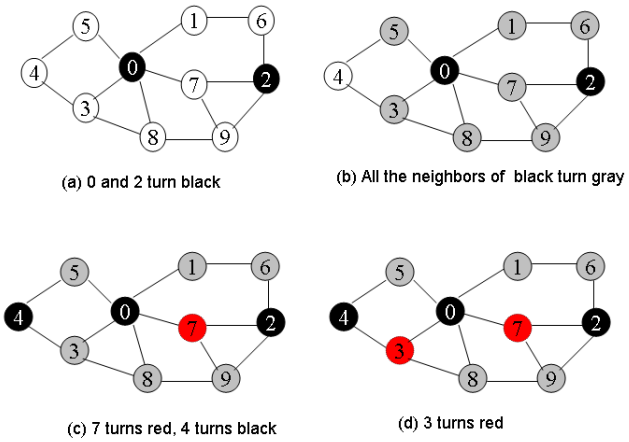


Figure 2: An Example of the coloring process of MCMIS

Proof. In MCMIS, all black nodes are decided in the forest construction. From lemma 1, all black nodes B_i in a dominating tree T_i form an MIS of T_i. There is no black_black conflict, and tree merging only introduces new red nodes, so all black nodes are disjoint, thus B is an independent set. Further, there can only exist black_gray and gray_gray conflicts between any two trees, thus the two black nodes in two different trees are separated by either 2 or 3-hops. This implies that B form a MIS. □

Theorem 1 *The resulting backbone is connected. The algorithm has performance ratio of 192, time complexity of $O(n)$, and message complexity of $O(n\Delta)$.*

Proof. In Lemma 2, we have proved that all black nodes form an MIS. In section 3.2, we have shown that dominating tree construction guarantees a tree, and all gray nodes are leaves, thus all black nodes and red nodes in a tree are connected. In the merging on conflicts phase, for a black_gray conflict, two black nodes in two trees are connected by coloring one gray node red if they have not been connected in other ways. For a gray_gray conflict, two black nodes in two trees are connected by coloring two gray nodes red if they have not been connected in other way, thus all black nodes and red nodes connected. The virtual backbone composed of all black and red nodes is also connected.

Since all black nodes form a MIS, the proof for performance ratio is the same as in [8], which is 192. The time complexity is O(n) since the worst case is that

all nodes are in either ascending or descending order. Each white node broadcasts 0 or one *blacktobe* message. Each black node broadcasts one *black* message and unicasts at most Δ *connect* messages, where Δ is the maximum nodal degree in the UDG. Each gray node broadcasts one *gray* message, unicasts at most Δ *gray*2 and *connect*2 messages. A red node broadcasts at most Δ *red* messages, thus the message complexity is $O(n\Delta)$ □

4 Simulation Results

In this section, we verify our algorithm by simulation and evaluate its performance in terms of backbone size and lifetime against Wu and Li's algorithm (WLA) [10] which is also a localized algorithm and has the longest backbone lifetime shown in [4]. To evaluate the effect of stability as a parameter, we implemented two versions of our algorithm, MCMIS-1 and MCMIS-2. The difference between these two versions is MCMIS-2 uses stability as a ranking parameter, while MCMIS-1 does not. The ranking of MCMIS-1 is $(effective\_degree, id)$. In the following, we first explain the mobility model we use and then present the simulation results.

4.1 Mobility Model

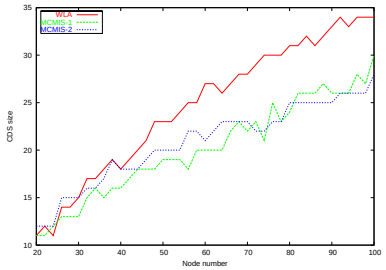
Random way-point model (RWP) [1] is widely used to model the movement of individual nodes. However, [1] [2] argued that the original RWP has two main problems: *border effect* causes the node distribution is not uniform and the average speed tends to reach 0 instead of around the middle point of minimum and maximum speeds.

We deploy a modified RWP model which uses static parameter as in [1] and modifies the destination selection so that a node selects its next destination from a bigger region beyond the border. When a node reaches a boundary, it bounces and moves back toward the center. It can be shown that this model is able to enter the steady state rapidly, i.e., the node distribution lasts nearly uniform and the average speed converges quickly. In our following simulation, the mobility parameters are set as follows: minimum speed is 1m/s, maximum speed is 5m/s, pause time varies from 0 to 10s, and the stability parameter is 0.5.

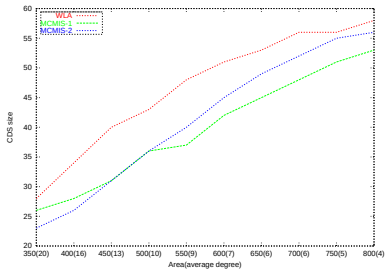
4.2 Simulation results

Three groups of experiments are conducted: 1) Both the area where a connected topology is generated and the transmission range of nodes are fixed. By varying the number of nodes in the area, we measure the scalability of our algorithm; 2) Both the node number and transmission range are fixed. By varying the area, we measure the performance of our algorithm under different network density; 3) The node number, transmission range, and the area are fixed. We run the simulation for 150 seconds, and measure the effect of the stability on the lifetime of the generated backbone.

Fixed area. In this experiment, the area is a $400m \times 400m$ square, transmission range of each node is $100m$, which is one-quarter of the area edge. This is a typical setting for multi-hop ad hoc networks. For each case as node number varies from 20 to 100, we run the simulation for 50 times. Fig. 3.(a) shows that both MCMIS-1&2 generate smaller backbone sizes than [10]. As the node number increases, the backbone size of our algorithm increases much slower than [10], which suggests our scheme has very good scalability. Another interesting observation is that when node number is greater than 75, MCMIS-2 performs slightly better than MCMIS-1. This indicates that when the network gets denser, the *effective_degree* parameter becomes less important in reducing the backbone size because most of the node's degrees are very high.



(a) The relationship between node number and CDS size when area size = $400 \times 400m^2$, transmission range = $100m$



(b) The relationship between area size and CDS size when node number = 100,transmission range = $100m$

Figure 3: Comparison of backbone size

Fixed node number. In this experiment, the node number is fixed as 100, and the transmission range is $100m$. For each case as the area increases from $200m \times 200m$ to $800m \times 800m$ (the edge is increased by $50m$) we run the simulations for 100 times. The number in the parenthesis is the average node degree. Fig. 3.(b) shows that both MCMIS schemes can generate a backbone about 10 nodes smaller than WLA when the average degree of the nodes is from 6 to 10, which are the most common and realistic cases. As we expected, the backbone size increases as the network gets sparser. We also observe that when the network is denser, for example the average degree of nodes is greater than 10, MCMIS-2 starts to outperform MCMIS-1. This confirms that *effective_degree* is not an important parameter when the network is very dense.

Lifetime of the virtual backbone. In this experiment, we randomly generated a connected topology for 75 nodes in a $400 \times 400m^2$ area, the transmission range is fixed at $100m$. We generated a movement history for 150 seconds and measures the average size and lifetime of the backbone constructed by our algorithm and [10] over 100 times.

Algorithm	*Average CDS size*	*Average Life time*
WLA	30.06	27.23
MCMIS-1	23.81	8.98
MCMIS-2	24.36	43.48

Table 2: Comparison of Backbone Size and Life Time

We can clearly see the effectiveness of adopting stability as one of the nodal ranking parameters from Table 2. The backbone lifetime of MCMIS-2 (which adopts nodal stability) is prolonged to almost five times longer than that of MCMIS-1, while the sizes of generated virtual backbone are comparable, which is only 0.5 difference. Compared with [10], MCMIS-2, with a smaller backbone, significantly improves the backbone lifetime from 27.23 seconds to 43.48 seconds. (Note that in [4], the [10] has the longest lifetime). Thus our scheme can generate a smaller and more reliable virtual backbone. Prolonged lifetime also implies less backbone reconstruction or maintenance overhead, which is a very desirable feature of virtual backbone schemes.

Overall, the simulation results demonstrate that MCMIS is able to obtain a stable virtual backbone with a small size in typical mobile network environments.

5 Conclusion and Future Work

In this paper, we presented a novel localized scheme to generate a connected dominating set, which can serve as a virtual backbone in mobile ad-hoc networks. We defined node ranking as a tuple of *(stability, effective_degree, id)* thus significantly prolonged the lifetime of the constructed backbone. We also designed a localized algorithm which can quickly construct a small-scaled backbone. Extensive simulation results demonstrate that the backbone constructed with our scheme is more stable and have smaller size than other localized scheme.

Our future work includes: 1) exploit how to interleave the construction of virtual backbones and route discovery. There are two possible directions. First, we could use the information collected for constructing virtual backbone to help reduce routing overhead. Secondly, we can identify useful information for route discovery and try to collect them during virtual backbone construction; 2) simulate mobile environments with other models, such as the group mobility model and the hybrid model; 3) handle the roaming of non-backbone nodes.

References

[1] C. Bettstetter, G. Resta and P. Santi, The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks, *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, 2003, pp. 257-269.

[2] J. Yoon, M. Liu and B. Noble, Random Waypoint Considered Harmful, *Proceedings of Infocom*, 2003.

[3] I. Cidon and O. Mokryn, Propagation and Leader Election in a Multihop Broadcast Environment, *Proceedings of 12th International Symposium on Distributed Computing*, 1998.

[4] Manki Min, Feng Wang, Ding-Zhu Du and Panos M. Pardalos, A Reliable Virtual Backbone Scheme in Mobile Ad-Hoc Networks, *Proceedings of the 1st IEEE International conference on Mobile Ad Hoc and Sensor Systems*, 2004.

[5] S. Guha and S. Khuller, Approximation Algorithms for Connected Dominating Sets, *Proceedings of European Symposium on Algorithms*, 1996.

[6] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen and J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, *Proceedings of MOBICOM*, 1999.

[7] P. Sinha, R. Sivakumar and V. Bharghavan, Enhancing Ad hoc Routing with Dynamic Virtual Infrastructures, *Proceedings of Infocom*, 2001.

[8] P.-J. Wan, K. M. Alzoubi and O. Frieder, Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks, *Proceedings of Infocom*, 2002.

[9] K.M. Alzoubi, P.-J. Wan and O. Frieder, Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks, *MOBIHOC*, 2002.

[10] J. Wu and H. L. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, *Proceedings of the 3rd ACM international workshop on Discrete algorithms and methods for mobile computing and communications*, 1999.

[11] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks, *ACM Wireless Networks Journal*, 2002.

[12] B. Williams and T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*,2002.