# Distributed Routing Schemes for Ad Hoc Networks Using d-SPR Sets

**Subhankar Dhar**
San José State University
San José, CA 95192 USA
+1 408 924 3499
dhar_s@cob.sjsu.edu

**Michael Q. Rieck**
Drake University
Des Moines, Iowa 50311 USA
+1 515 271 3795
michael.rieck@drake.edu

**Sukesh Pai**
Microsoft Corporation
Mountain View, CA 94043 USA
+1 650 693 3688
sukeshp@microsoft.com

**Eun Jik Kim**
San José State University
San José, CA 95192 USA
+1 408 924 3499
eunkim@sjsu.edu

## ABSTRACT

In this paper, we propose several new distributed algorithms for producing sets of nodes that can be used to form backbones of an ad hoc wireless network. Our focus is on producing small sets that are $d$-hop connected and $d$-dominating and have a desirable "$d$-shortest path property" which we call $d$-SPR sets. These algorithms produce sets that are considerably smaller than those produced by an algorithm previously introduced by the authors. Our proposed algorithms, except the greedy ones, have constant time complexity in the restricted sense that the time required is unaffected by the size of the network, assuming however that the node degrees are bounded by a constant. The performance of the new algorithms are compared, and also compared with the authors' earlier algorithm, and with an adaptation of an algorithm of J. Wu and H. Li.

## Keywords

dominating set, ad hoc wireless networks, routing algorithm, set covering problem, shortest path routing

## I. INTRODUCTION

Wireless ad hoc networks consists of a set of identical mobile devices (nodes) that communicate with each other via wireless links. The growing importance of ad hoc wireless networks can hardly be exaggerated, as portable wireless devices are now ubiquitous and continue to grow in popularity and in capabilities. In such networks, all of the nodes are mobile and so the infrastructure for message routing must be self-organizing and adaptive. Building an infrastructure for ad hoc network that guarantees reliable communication is an important problem.

In recent years, there have been prolific research activities in this regard. However, there are quite a number of challenging problems yet to be solved in the area of ad hoc networks. Finding efficient and effective routing schemes is just one example, and the one that we will focus on.

Ad hoc wireless networks are here represented by a connected graph where all the links are bi-directional. Several researchers have used minimum connected dominating sets to do routing in ad hoc wireless networks [2], [3], [4], [5], [15], [16], [17]. The dominating set induces a virtual connected backbone. The Connected Dominating Set (CDS) problem is described as follows: find a minimal subset $D$ of nodes, such that the subgraph induced by $D$ is connected and $D$ is a dominating set *i.e.*, it is a set in which each node is either in $D$ or adjacent to some node in $D$. It is well-known that finding a minimum connected dominating set is an NP-complete problem [6]. Some authors have proposed approximation algorithms for obtaining minimal connected dominating sets [6].

One of the earlier works was done by B. Das and V. Bharghavan who used minimum connected dominating sets (MCDS) as a virtual backbone to develop routing schemes for wireless ad hoc networks [5]. This virtual backbone may change with the movement of nodes and is used only for computing and updating routes. Their MCDS routing algorithm computes shortest possible paths for routes and updates routes soon after each node moves. Besides finding routes, their algorithm also supplies backup routes for temporary use while shortest paths are updated. Because their focus is on constructing a minimum connected dominating set, the overhead in setting up such a set is quite time consuming, when contrasted with other methods that merely settle for a reasonably small set.

B. Liang and Z. J. Haas use the greedy algorithm with redundancy elimination to obtain a $d$-dominating set, that is, a set of nodes such that every node in the network is within $d$ hops of a node in the set [13]. Rather than seeking an optimal solution, they apply the greedy algorithm to obtain a fairly small $d$-dominating set, and then apply redundancy elimination to reduce this slightly. Our approach in this paper is quite similar, as will be discussed in the next section. However, our set is in addition $d$-hop connected, and has a certain "$d$-shortest path property". Moreover, Liang and Haas's distributed method requires that each node maintain an awareness of all nodes within $2d$ hops of itself, whereas our method only requires an awareness of nodes within $d + 1$ hops.

More recently, J. Wu and H. Li developed a distributed algorithm [18] for constructing a connected dominating set in a connected graph that represents a wireless ad hoc net-

work. This set can be used to form a virtual backbone of a wireless ad hoc network. Their algorithm can be modified to produce a $d$-hop connected, $d$-dominating set. We do so in our experiments for benchmarking purposes.

In this paper, the methods of [8], [9] and [15] are modified in a number of ways and compared. The notion of an "$d$-SPR set" ("$d$-shortest path routing set") is introduced and justified as being quite useful as a virtual backbone for an ad hoc wireless network. We look at different ways of computing such a set by mapping the problem to the Set Covering Problem. We propose a general strategy to minimize the sets obtained through greedy refinement. We finally compare the results from these algorithms to results from previous work.

Such a set can be used to facilitate shortest path routing in a straightforward manner, as will now be explained. In [18, Subsection 5.1], Jie Wu and Hailan Li describe how to use their connected dominating set to route messages. The idea is that each of the nodes in this backbone maintains global routing tables, and whenever an ordinary node needs to send a message across the network, and is uncertain how to best route this message, it queries its neighboring backbone nodes. Each replies by indicating the cost (number of hops) of routing the message through that particular backbone node. The node sending the message then chooses to send the message along the path requiring the least number of hops. While the message will thus follow a shortest possible (*i.e.* fewest hops) backbone path, there is no guarantee that this path will be a shortest path for the network taken as a whole.

However, routing via a similar strategy using a $d$-SPR set does guarantee shortest path routing. An issue of concern in the Wu-Li method and in our methods is bottlenecking, since only certain paths are favored. Practical solutions to this problem have been considered, but will not be discussed here.

## II. RELATED WORK

Minimal and nearly minimal connected dominating sets have been studied for some time now, particularly as virtual backbones for routing in wireless ad hoc networks. Finding a minimum connected dominating set is an NP-complete problem. Even finding a nearly minimal one can be quite time consuming, and when implemented in a distributed manner, can involve a large amount of message passing between neighboring nodes. Several authors have proposed such methods.

S. Guha and S. Khuller [6] proposed two algorithms to construct small connected dominating sets. The first of these grows a tree so that at each stage, a large number of other nodes are dominated by the nodes of the tree. At first they consider doing this in a straightforward greedy manner, but then employ some look-ahead to refine the method. Their second approach grows several trees simultaneously, and then connects these together to form a connected dominating set.

B. Das and V. Bharghavan [5] proceed along similar lines. Their virtual backbone may change with the movement of nodes and is used only for computing and updating routes. Their MCDS routing algorithm computes shortest possible paths for routes and updates routes soon after each node moves. Besides finding routes, their algorithm also supplies backup routes for temporary use while shortest paths are updated. K. Alzoubi, P. Wan and O. Frieder [2] presented a distributed algorithm for finding a connected dominating set by first constructing a maximal independent set of nodes. Y. P. Chen and A. L. Liestman [4] introduce a notion that is weaker than the notion of a connected dominating set, and show how to alter Guha and Khuller's algorithms to produce such sets. This is just a sampling of the varied and interesting approaches that have been developed recently for constructing connected dominating sets.

Another direction for constructing virtual backbones that has been receiving increasing attention in recent years is based on $d$-dominating sets. The nodes of such a set constitute a virtual backbone, even though they are not in general directly connected. Each such node is responsible for all of the nodes in its $d$-hop neighborhood, and each maintains global routing information. Shay Kutten and David Peleg [10] introduce a method for finding a $d$-dominating set, first by partitioning the nodes of the graph based on their distance module $d$ from the root of a spanning tree. In connection with our work, the method of B. Liang and Z. J. Haas [13] is of special interest, and is discussed below.

### A. The algorithm of B. Liang and Z. J. Haas

B. Liang and Z. J. Haas [13] use a distributed greedy algorithm to produce a $d$-dominating set. To do so, they reduce the problem of finding this set to a special case of the Set Covering Problem. This is a well-known NP-complete problem. In fact, our approach in the present paper is quite similar in this regard. We also consider a certain Set Covering Problem, although a different one.

We too consider finding an approximately optimal solution to a covering problem via a distributed greedy algorithm. However, each node in our method only needs to maintain an awareness of its local $(d+1)$-hop neighborhood, as opposed to the $2d$-hop neighborhood required in [13]. Moreover, the resulting set is not only $d$-dominating, but is also $d$-hop connected and has the $d$-shortest path property discussed in the next section.

### B. The algorithm of J. Wu and H. Li

Jie Wu and Hailan Li proposed a basic distributed algorithm [18] for constructing a connected dominating set in a connected graph of radius at least two. The set produced by their basic algorithm is quite large in size. In order to produce smaller dominating set, they apply a refinement (rules 1 and 2) to produce a smaller set. They showed that this set is a connected dominating set. This set is used to form a virtual backbone of a wireless ad hoc network.

Recently, there have been some generalizations of the algorithm of Wu and Li. In [16], J. Wu and F. Dai presented a general framework for broadcasting in ad hoc wireless

networks through self-pruning. The set produced by their algorithm is based on certain coverage conditions (coverage conditions I and II). Their algorithm produces a connected dominating set. The Wu-Li algorithm is a special case of coverage condition II with 2 or 3-hop neighborhood information without any routing history. In addition, the size of the coverage set is less than or equal to 2.

Our research has been concerned with $d$-hop connected $d$-dominating sets, and was largely motivated by considering extensions/alterations of the Wu-Li method. One such alteration is quite simple, and just involves applying the Wu-Li method directly to the $d$-closure $G_d$ of the original graph $G$ (see next section). This produces a $d$-hop connected $d$-dominating set, as is clear from the definitions in the next section. It does not however have the $d$-shortest path property, as defined below.

## III. SHORTEST PATH ROUTING

### A. d-SPR Sets

Throughout this paper, a wireless ad hoc network will be represented by a graph $G$. We will further assume that this graph is connected. In [15], the authors presented an algorithm to construct a $d$-hop connected $d$-dominating set, used as a routing backbone, where $d$ is a fixed integer greater than one. "$d$-dominating" simply means that each node in the network is within $d$ hops of a node in the set.

Requiring $d$-dominating (for $d > 1$) rather than dominating results in fewer backbone nodes, each responsible for a wider region of neighboring nodes. "$d$-hop connected" means that given any two nodes $u$ and $v$ in the set, there is a path beginning with $u$ and ending with $v$, such that the hop count between consecutive nodes along the path that belong to the set never exceeds $d$.

In fact the set produced in [15] has the addition special property that there exists a path as just described between any two nodes $u$ and $v$, and that this path is a shortest (possible) path in the sense that any other path connecting $u$ and $v$ requires at least as many hops. We refer to this as the "$d$-shortest path property", whose proper definition is the following, where $\delta(u, v)$ denotes the graph-theoretic distance (number of hops) between two nodes $u$ and $v$.

*Definition:* A set of nodes $S$ has the *$d$-shortest path property* if, for any two nodes $u$ and $v$ in the graph, there exists a shortest path (*i.e.* a path whose graph-theoretic length is as small as possible) connecting $u$ and $v$ such that the nodes on this path that are also in $S$, together with $u$ and $v$, form a $d$-hop connected set.

A related notion which will prove to be useful as the basis of our routing scheme was introduced formally in [8], but earlier played the central role in [15]. It is as follows.

*Definition:* A set $S$ of nodes is *$d$-SPR set* if given any pair of nodes $u$ and $v$ of $G$ such that $\delta(u, v) = d+1$, there exists a $w \in S$ with $\delta(u, w) + \delta(w, v) = d + 1$ and $w \neq u, w \neq v$.

Here $d$-SPR stands for "$d$-shortest path routing", and under reasonable assumptions, a $d$-SPR set can be shown to be a $d$-hop connected, $d$-dominating set that has the $d$-shortest path property. Deciding whether or not a set is a $d$-SPR set is in some sense a local issue. To be more specific, it is possible to consider a certain proposition concerning the restriction of the set to each $(d+1)$-hop neighborhood. The proposition is true for each of these if and only if the set as a whole is a $d$-SPR set. This follows easily from the definition and [8, Corollary 1], which is also reproduced as Corollary 1 in this paper.

An alternative way to think about a $d$-hop connected $d$-dominating set for $G$ is simply as a connected dominating set in a graph $G_d$ derived from $G$ as follows:

*Definition:* The *$d$-closure* $G_d$ of $G$ is the graph obtained from the original graph $G$ by adding edges between any pair of nodes that are within a graph-theoretic distance $d$ in $G$.

This point of view is required in order to adapt the algorithm of Wu and Li in order to produce a $d$-hop connected $d$-dominating set. However, the resulting set does not in general satisfy the shortest path property for two reasons. In the first place, even when $d = 1$, routing via the Wu-Li algorithm only guarantees that routing will follow a shortest possible path through the backbone, but not necessarily a shortest path through the network as a whole. When $d > 1$ the situation is made worse by the fact that the graph $G_d$ does not retain the distance (hop count) information of the original graph $G$. Two nodes being adjacent in $G_d$ only means that these nodes are within $d$ hops of each other in $G$, but the actual hop count cannot be discerned.

In the next subsection, we will begin to consider the problem of finding small $d$-SPR sets. First, let us establish certain properties associated with such a set, which requires the following standard definition from graph theory.

*Definition:* The *radius* of a graph $G$ is the largest integer $r$ such that given any node $u$ in $G$, there exists a node $v$ in $G$ such that $\delta(u, v) = r$.

*Theorem 1: Assume that the connected graph $G$ has radius at least $d + 1$. Then any $d$-SPR set has the following properties:*

1. *It is $d$-dominating*
2. *It is $d$-hop connected*
3. *It has the $d$-shortest path property*

*Conversely, a set of nodes with the $d$-shortest path property is a $d$-SPR set.*

*Proof:* Fix a $d$-SPR set $S$. Consider a node $x$ in $G$. Since $G$ has a radius of at least $d + 1$, there exists a node $y \in V$ at a distance $d + 1$ from $x$. By definition of $d$-SPR, there exists a node $w$ in $S$ such that $\delta(x, w)+\delta(w, y) = d+1$, $w \neq x, w \neq y$, and so $w$ is within a distance $d$ of $x$. Hence $S$ is $d$-dominating. This proves item 1.

To show the $d$-shortest path property, fix any two nodes $u$ and $v$. Let $p$ be a shortest path in $G$ from $u$ to $v$. Let $u_j$ denote the node arrived at after taking $j$ steps along this path. ($j = 0, 1, 2, ..., m$, where $m = \delta(u, v)$). If $m \leq d$, then there is nothing to prove. Suppose $m > d$. Consider the nodes $u$ ($= u_0$) and $u_{d+1}$. Since they are a distance $d + 1$ apart, there exists a $w \in S$ and a path $q$ of length $d + 1$ between $u$ and $u_{d+1}$ that passes through $w$ in its interior. Now create another shortest path in $G$ from $u$ to $v$ by replacing the original subpath in $p$ from $u$ to $u_{d+1}$ with the path $q$. $w$ lies in this new path from $u$ to $v$. Repeat the procedure for the path from $w$ to $v$, assuming that the distance between these exceeds $d$. Continuing in this way, a suitable shortest path will eventually be produced from $u$ to $v$, one that exhibits the shortest path property. This proves item 3. The proof of item 2 follows immediately from this.

Conversely, let a set of nodes $S'$ be any subset of the node set of $G$ possessing the $d$-shortest path property. Consider any pair of nodes $u$ and $v$ such that $\delta(u, v) = d + 1$. Then there exists a shortest path connecting $u$ and $v$ as described in the definition of the $d$-shortest path property. Some node $w$ along this path, and strictly between $u$ and $v$ must be an element of $S'$. Clearly, $\delta(u, w) + \delta(w, v) = d + 1$. Hence, $S'$ is a $d$-SPR set. ■

### B. Set Covering Problem, Bipartite Graph

Our focus in this paper is to produce small $d$-SPR sets having the $d$-shortest path property. It was observed in [8] that our problem of finding minimal $d$-SPR sets reduces to the well-known Set Covering Problem. This problem can be described as follows. Given a set $U$ and a collection of subsets of $U$, $\{S_1, S_2, ....., S_n\}$, find the smallest subcollection of subsets $S_i$, such that the union of these $S_i$ equals $U$.

The Set Covering Problem is essentially a problem concerning bipartite graphs that can be stated as follows. Suppose that $H$ is a bipartite graph, consisting of two sets of nodes $A$ and $B$, where edges only make connections between $A$ and $B$. Also assume that for each node in $B$, there is at least one edge connecting it to a node in $A$.

The problem then is to find a smallest possible subset $C$ of $A$ that "covers" $B$. That is, for each node in $B$, there must be at least one edge connecting it to a node in $C$. While this is an NP-complete problem, it has been long known ([7], [12]) that the greedy algorithm results in a set whose size is bounded by the size of an optimal solution times $H(\beta)$, where $\beta$ is the maximum degree among the nodes in $B$, and $H(\beta) = 1 + 1/2 + 1/3 + .... + 1/\beta$.

The problem of finding a $d$-SPR set can be translated into the problem of finding a "covering set" $C$ as above, in the following bipartite graph $H$. Let $A$ be the set of all the nodes in the network. Let $B$ be the set of all unordered pairs $\{x, y\}$ of nodes in the network satisfying $\delta(x, y) = d + 1$. In $H$, put an edge between a node $v$ from $A$ and a pair $\{x, y\}$ from $B$ if $v$ does not equal $x$ or $y$, but $v$ does lie along some shortest (possible) path connecting $x$ and $y$. That is, $\delta(x, y) = \delta(x, v) + \delta(v, y)$. A subset $C$ of $A$ covers $B$ if and only if it is a $d$-SPR set, as is straightforward to check. For more details, see [8]. The following definition will be useful, although it really is just another name for "node degree" in the context of the bipartite graph.

*Definition:* The *covering number* of each node $w$ in $A$ is the number of $\{x, y\}$ pairs in $B$ that share an edge with $w$. Also, we say that $w$ *covers* the pair $\{x, y\}$ in $H$, and that $w$ is an *interior node* for $\{x, y\}$.

From the construction of $H$ and the definition of a $d$-SPR set, it is clear that a set $C \subset A$ is a $d$-SPR set if and only if $C$ covers all of the elements of $B$ in $H$. The problem of finding a minimal such $C$ is likely to be NP-hard, since the class of all possible bipartite graphs that could result from the above construction appears to be quite varied.

### C. Distributed approaches to obtaining $d$-SPR sets

In order to produce small $d$-SPR sets in a distributed approach, the following notion will be helpful.

*Definition:* The *$d$-local view* of a node $v$ consists of all the $d$-hop neighbors of $v$, together with all edges between these, except for the edges that connect two nodes that are both at a distance $d$ from $v$.

It is possible for the nodes in a network to learn about their $d$-local views after each transmits $d$ messages, containing local link-state information, to all of its neighbors. In our scheme though, each node will be required to maintain its $(d + 1)$-local view, which of course requires $d + 1$ transmissions per node. The following results allows each node to leverage the knowledge of its local $(d + 1)$-local view in order to assist in producing a $d$-SPR set.

*Lemma 1: For any nodes $x$, $y$, $u$, $v$ in $G$ such that $\delta(x, u) + \delta(y, u) \leq d + 1$ and $\delta(x, v) + \delta(y, v) \leq d + 1$, it follows that $\delta(u, v) \leq d + 1$.*

*Proof:* We have

$$\delta(x, u) + \delta(u, y) + \delta(y, v) + \delta(v, x) \leq 2(d + 1)$$

By the triangle inequality, we obtain

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v) \text{ and } \delta(u, v) \leq \delta(u, y) + \delta(y, v)$$

Adding the last two equations yields

$$2\delta(u, v) \leq \delta(u, x) + \delta(x, v) + \delta(u, y) + \delta(y, v)$$

Therefore,

$$\delta(u, v) \leq d + 1.$$

■

*Theorem 2: For any node $x$, $y$, $v$ in $G$ with $\delta(x, v) + \delta(y, v) = d + 1$, the distance $\delta(x, y)$ can be computed solely from a knowledge of the $(d + 1)$-local view of $v$. Moreover, all the shortest paths connecting $x$ to $y$ lie inside the $(d+1)$-local view of $v$.*

*Proof:* The first claim follows immediately from the second. To prove the second claim, consider any shortest path $p$ between $x$ and $y$. Let $w$ be any node on $p$. Then we have $\delta(x,w)+\delta(y,w) \leq d+1$ and $\delta(x,v)+\delta(y,v) \leq d+1$. It follows from Lemma 1 that $\delta(w,v) \leq d+1$. So all the nodes along shortest paths connecting $x$ to $y$ lie inside the $(d+1)$-local view of $v$.

From the definition of $(d+1)$-local view, the edges along such paths also appear in $v$' s $(d+1)$-local view, except possibly those which connect two nodes at a distance $d+1$ from $v$. However, such edges are impossible. To see this, assume such an edge exists and connects two nodes $u$ and $w$, with $u$ closer to $x$ and $w$ closer to $y$. Assume that this is part of a shortest path $p$ connecting $x$ and $y$. Then $\delta(v,x)+\delta(x,u) \geq \delta(v,u) = d+1$ and $\delta(v,y)+\delta(y,w) \geq \delta(v,w) = d+1$. But $\delta(v,x)+\delta(v,y) = d+1$ and $\text{length}(p) = \delta(x,u)+\delta(u,w)+\delta(w,y) = \delta(x,u)+1+\delta(w,y) \leq d+1$. This quickly leads to a contradiction. ■

*Corollary 1:* Given $u, v \in A$ (*i.e.* nodes in $G$), and a pair $\{x,y\} \in B$ such that $u$ and $v$ are both adjacent to $\{x,y\}$ in $H$, the four nodes $u, v, x, y$, as nodes of $G$, are within a distance $d+1$ of each other.

### D. The d-SPR-I Algorithm

It is henceforth assumed that every node in $G$ has somehow been assigned a unique positive integer ID. In order to blend the desirable features of cluster-based routing with those of backbone-based routing, one might consider applying the algorithm of Wu and Li to the $d$-hop closure $G_d$ of the original graph $G$. While this approach leads to a virtual backbone involving a small number of nodes, the message routing paths that result can be unnecessarily long, since the backbone will not have the $d$-shortest path property in general.

By contrast, the $d$-CDS algorithm [15], which we have decided to rename as $d$-SPR-I for clarity, produces a larger set of backbone nodes, but one which guarantees shortest path routing. In order to accomplish this, all of the shortest possible paths of length $d+1$ are discovered and the IDs of the nodes along such paths are considered. Further details on our original approach can be found in [15].

The algorithm can be recast in terms of the bipartite graph $H$ as follows. For each pair $\{x,y\}$ in $B$, let $A_{\{x,y\}} = \{w \mid \delta(x,w)+\delta(y,w) = d+1, w \neq x, w \neq y\}$. This consists of the interior nodes for the pair $\{x,y\}$. Now, using $d$-SPR-I, the node $v$ in $A_{\{x,y\}}$ with the highest ID is "elected" to cover this pair. The resulting set of elected nodes is clearly a $d$-SPR set.

This algorithm can be implemented in the following way. Each node learns about its own $(d+1)$-local view by using $d+1$ rounds of local broadcasting. By Corollary 1, each node is in a position to decide for itself whether it should join the $d$-SPR set or not. It decides to do so if it discovers some pair $\{x,y\}$ (in $B$) that it covers for which it has the highest ID among all the nodes that cover this pair. The resulting set produced by the $d$-SPR-I algorithm is a $d$-hop connected $d$-dominating set and has the desirable $d$-shortest path property.

Once a $d$-SPR set has been established, and assuming that nodes of this set have acquired the necessary global routing information, routing can be accomplished as follows. The routing procedure here is similar to one described by Wu and Li [18, Section 5]. A node wishing to send a message to a remote destination, queries all of the $d$-dominating nodes in its $d$-hop neighborhood. They inform the source node of the costs of sending the message along the optimal paths known to the $d$-dominating nodes, and the source chooses to send the message using the $d$-dominating node for which the cost is as small as possible. The nodes in the $d$-SPR set then coordinate the actual transmission of the message.

### E. d-SPR Based on Covering Number (d-SPR-C)

We designed a variant of $d$-SPR-I algorithm ($d$-SPR-C) that is an improvement of the $d$-CDS algorithm as discussed in [15].

The $d$-SPR-C algorithm requires an additional $d+1$ rounds of local broadcast. After the first $d+1$ rounds, each node is aware of its own $(d+1)$-local view, and is able to compute its own covering number. The subsequent $d+1$ rounds of broadcast are used to allow each node to transmit its covering number to each of its $(d+1)$-hop neighbors.

In the $d$-SPR-I algorithm, given a pair $\{x,y\}$ (from B), the node that covered $\{x,y\}$ with the highest ID was elected for inclusion into the $d$-SPR set. Here, instead we elect the node having the highest *priority*. The priority of each node is defined to be the ordered pair of numbers `(covering_number, ID)`, lexicographically ordered. This is analogous to the approach taken in [11, Subsection 2.1]. This variant of $d$-SPR-I will be referred to as $d$-SPR-C ("C" for "covering number").

## IV. Distributed Greedy Refinement of d-SPR Sets

### A. Greedy Refinement

In this subsection, we modify the greedy algorithm discussed in [8] in order to reduce the size of any $d$-SPR subset $A$ of all nodes in a given graph $G$. We refer to this reduction process as *greedy refinement*, applied to the set $A$. The initial set $A$ can be any $d$-SPR set obtained from among the various algorithms we have already discussed, or it can be simply the set of all nodes in $G$.

Greedy refinement can be described as follows. Beginning with a graph $G$ representing an ad hoc network and a $d$-SPR subset $A$ of $V(G)$ as input, the algorithm generates a $d$-SPR subset of $A$ as output. The first step is to construct the bipartite graph $H$ described in subsection III.B, using this set $A$ in place of the set of all nodes of $G$, and still using the set of all pairs of nodes in $G$ that are at a distance $d+1$ apart as the set $B$. $C$ will denote the output set, which is initially empty. The algorithm repeats the following steps while the set $B$ is non-empty.

*Step 1:* Compute the degree of all nodes in the set $A$ in

the bipartite graph $H$. (*i.e.* compute the covering numbers.)

*Step 2:* Elect a node $v$ from set $A$ such that $v$ has the highest degree and add it to the output set $C$. If there is a tie, then the highest ID is used to break the tie.

*Step 3:* Remove all nodes i.e. pairs $\{x, y\}$ in set $B$ in the bipartite graph $H$ which are covered by the node $v$ from Step 2. Also remove node $v$ from $A$.

### B. Distributed Greedy d-SPR Algorithm (d-SPR-G)

The set obtained by the sequential greedy algorithm described in the previous subsection can be achieved in a distributed way where each node maintains information about its $(d+1)$-local view. We call this the *greedy d-SPR (d-SPR-G)* algorithm. It takes the same input and generates the same output as non-distributed greedy refinement.

Each node begins in the "undecided state", maintains information about its $(d+1)$-local view, executes the following pseudo-code, and ultimately either ends up in the "elected" or "not elected" state. From Corollary 1 we know that a node $v$ can "see" all the node pairs $\{x, y\}$ it covers, and also the rest of the nodes that cover such pairs. The fact that the distributed algorithm below terminates and that the set of "elected" nodes is a $d$-SPR set follows immediately from the discussion of similar distributed greedy algorithms, as found in [11] and [13]. All of these, including ours, are in essence just distributed implementations of the greedy algorithm for the Set Covering Problem.

Our distributed greedy algorithm requires each node $v \in A$ to execute the following C-like pseudo-code, though they do not need to do so in lock-step. Other nodes only need to participate so as to route messages between members of $A$. The following pseudo-code is not the most efficient possible, but is comparatively straightforward to understand.

```
//       *** Greedy reduction pseudo-code ***

//       (Executed by each member v of set A)

// v exchanges local link-state information sufficient to
// determine its (d+1)-local view.  Also, it's assumed that
// v either already knows or will now be told which nodes
// in this local view belong to the initial d-SPR set A.

for (i=0; i<=d; i++)
  exchange_basic_info_with_neighbors(&local_view);

compute_distances_in_local_view(&local_view);

// Identify node pairs {x,y} in local view such that d+1 =
// dist(x,v) + dist(y,v) = dist(x,y) and v != x and v != y.
// These are the pairs covered by v.  Also obtain covering
// number for v and initialize some lists.

list_of_pairs = find_pairs(&local_views);
//          ( = the set B_v )
covering_number = list_of_pairs.size;
foreach (node u in local_view) covers_pair_list[u] = EMPTY;

// Identify nodes that cover a pair that v covers.

foreach ({x,y} in list_of_pairs) {
  interior_node_list[{x,y}] = EMPTY;
  foreach (u in local_view)
    if (dist(x,y) == dist(x,u)+dist(y,u)
    AND u != x AND u != y) {
      interior_node_list[{x,y}].insert(u);
      covers_pair_list[u].insert({x,y});
     }
}

if (covering_number == 0) status = NOT_ELECTED;
                     else status = UNDECIDED;

while (status == UNDECIDED)
{
  // Each node must now exchange messages to learn the
  // status and covering #s of its (d+1)-hop neighbors

  for (i=0; i<=d; i++)
    exchange_status_and_covering_info(&local_view);

  // Now update lists

  combined_interior_list = the union of all
    interior_node_list[{x,y}] , indexed over all
    {x,y} in list_of_pairs;  // ( = the set A_v )
  foreach (u in combined_interior_list)
    if (u.status != UNDECIDED) {
      for ({x,y} in list_of_pairs)
        if (interior_node_list[{x,y}].contains(u)) {
          interior_node_list[{x,y}].remove(u);
          if (u.status == ELECTED)
            list_of_pairs.remove({x,y});
        }
      combined_interior_list.remove(u);
    }

  covering_number = list_of_pairs.size;

  // Determine status now

  if (covering_number == 0) status = NOT_ELECTED;
  else {
    // Decide if elected
    found_higher_priority = false;
    foreach (u in combined_interior_list)
      if (covers_pair_list[u].size > covering_number
        OR (covers_pair_list[u].size == covering_number
        AND u.ID > v.ID)) found_higher_priority = true;
    if (!found_higher_priority) status = ELECTED;
  }
}
```

Note that there is not a synchronization problem in this distributed algorithm, even if each node fails wait for all the information to arrive when requesting the current covering numbers and status of each of its $(d+1)$-hop neighbors. This is due to the fact that a node's covering number never increases. So if a node $v$ is allowed to "time out" when waiting for the up-to-date information (*i.e.* during the `exchange_status_and_covering_info` call), it will behave conservatively, and will not prematurely change its status.

### C. Distributed Greedy Refinement of d-SPR-I with two covering nodes (d-SPR-$C_2$G)

This algorithm is a variation made on $d$-SPR-C. In $d$-SPR-C, for every pair of nodes with distance $(d+1)$, say $\{x, y\}$, a node in the set $A_{\{x,y\}}$ that has the highest priority is admitted to the $d$-SPR set. In the initial selection phase of $d$-SPR-$C_2$G, for each pair $\{x, y\}$ in set $B$, two such nodes

are admitted to the initial set.

Obviously, the initial set $A$ for $d$-SPR-$C_2$G is considerably bigger than the set obtained from $d$-SPR-C. However, having a smaller initial set limits the scope of optimization in the second phase. Thus, from the optimization point of view, it is more desirable to have a large initial set, although a large initial set requires more time and messages to process. $d$-SPR-$C_2$G was designed as to reduce the set size produced by $d$-SPR-C. The purpose of $d$-SPR-G is purely to produce a very small $d$-SPR set by using the largest initial set possible, i.e. the set of all nodes in the graph, but of course the processing time is great and many messages need to be passed.

### D. d-SPR Based on Weights Assigned by Node Pairs (d-SPR-PW)

Another interesting variant of $d$-SPR-I algorithm that produces a set somewhat better than the size produced by $d$-SPR-C is $d$-SPR-PW (for "Pair Weighted"). It is based on an approach similar to S. Rampone [14]. While this could be used to reduce any $d$-SPR set, just as greedy reduction for this purpose, we will only consider applying it to the set of all nodes in the network. It is worth mentioning, in the context of finding a $d$-SPR set in a distributed way, that no additional message passing is required when using this approach instead of the usual greedy approach. This is clear from Corollary 1. The $d$-SPR-PW method proceeds as follows.

Each node $v$ in $A$ is elected into the $d$-SPR set based on the weights assigned to it by each of the node pairs in $B_v$. For $v \in A$ and $\{x, y\} \in B$, define $I(v, \{x, y\})$ and $PW(v, \{x, y\})$ as follows:

$I(v, \{x, y\}) = 1$ if $v$ covers the pair $\{x, y\}$ and 0 otherwise,

$$PW(v, \{x, y\}) = I(v, \{x, y\}) \ / \ \Sigma_{k=1\ldots n} I(v_k, \{x, y\})$$

where $n$ is the number of nodes and $v_k$ denotes each node.

The total weight on any node $v$ is the sum of the weights assigned by each node pair it covers.

$$PW(v) = \Sigma_{\{x,y\}} PW(v, \{x, y\})$$

The $d$-SPR-PW algorithm requires an additional $d + 1$ rounds of local broadcast just like $d$-SPR-C. After the first $d+1$ rounds, each node is aware of its own $(d+1)$-local view, and is able to compute its own pair weighted number. The subsequent $d+1$ rounds of broadcast are used to allow each node to transmit its pair weighted number to each of its $(d + 1)$-hop neighbors. Then, we elect the node having the highest *priority*. Here, the priority of each node is defined to be the ordered pair of numbers (pair weighted number, ID), lexicographically ordered.

## V. Bound on the d-SPR set size

We are able to obtain an upper bound of the set produced by the $d$-SPR-G algorithm.

Fig. 1. Example graph representing an ad hoc network

*Theorem 3: For any $a$ in $A$, let $B_a$ be the set of pairs that $a$ covers. For any $b$ in $B$, let $A_b$ be the set of nodes that cover $b$. Also, let $\alpha$ denote the minimum value of $|A_b|$, and let $\beta$ denote the maximum value of $|B_a|$. Let $C$ be the subset of $A$ produced by d-SPR-G. For positive integers $j$, let $H(j) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{j}$. Then*

$$|C| \leq (\tfrac{|A|}{\alpha})(1 + \ln(\alpha \tfrac{|B|}{|A|}))H(\beta).$$

*Proof:* Let $O$ denote the optimal (minimal) $d$-SPR set. From [7] and [12], we have the well-known theorem of Johnson and Lovász: $|C| \leq H(\beta)|O|$. We now argue similar to Theorem 1.2.2 in [1]. Select a subset $X$ of $A$ randomly by including each $a$ in $A$ with a fixed probability $p$ (independent events). The expected size of $X$ is therefore $E(X) = |A|p$. Let $Y \subset B$ consist of all $b$ not covered by $X$. For $b$ in $B$, $\mathbf{Pr}[b \in Y] \leq (1-p)^\alpha$, since each of at least $\alpha$ elements of $A$ cannot be in $X$ if $b$ is in $Y$. It follows that: $E(|X|+|Y|) \leq |A|p+|B|(1-p)^\alpha \leq |A|p+|B|e^{-\alpha p}$. Taking $p$ to be $(\frac{1}{\alpha}) \ln(\alpha \frac{|B|}{|A|})$ here yields the bound $E(|X| + |Y|) \leq (\frac{|A|}{\alpha})(1 + \ln(\alpha \frac{|B|}{|A|})$. Now, there must be some subset $X$ for which $|X| + |Y|$ is less than or equal to the expected value. But this $X$ can clearly be extended to a covering set ($d$-SPR set) of size less than or equal to $|X| + |Y|$. Therefore, there exists a $d$-SPR set of size less than or equal to $(\frac{|A|}{\alpha})(1 + \ln(\alpha \frac{|B|}{|A|}))$. So this is an upper bound on $|O|$, and the desired result now follows. ■

## VI. Example

The example that will be considered involves the graph in Fig. 1. The nodes here together with the solid edges constitute the graph $G$. For the reader familiar with the Wu-Li algorithm, it is straightforward, though somewhat laborious to check that the set resulting from the Wu-Li algorithm for this example is {1, 18, 20, 27, 29, 30}.

Next, we consider applying the $d$-SPR-I algorithm to the graph $G$, using $d = 2$. The resulting set in this case consists of all the nodes whose ID is greater than or equal to 6. See-

ing this requires a detail inspection of all the pairs of nodes $x$ and $y$ that are a distance 3 apart in $G$. For example, consider nodes 8 and 20. Along shortest paths connecting these we encounter the nodes 14, 22 and 5. Since 22 is the largest of these, it is "elected" to be in the backbone set. Likewise, between 1 and 26, the only nodes are 15, 21, 25, and 30. So 30 is also elected to join the set. The backbone set produced in this way is {22, 30, 29, 15, 28, 16, 27, 20, 24, 6, 9, 12}.

The 2-SPR-C algorithm involves a consideration of the same pairs of nodes as in 2-SPR-I, and the nodes that lie in between along shortest paths. So for example, consider the pair of nodes $\{3, 13\}$. The nodes that lie between 3 and 13 along a shortest path are 7, 19 and 28. We say that these nodes "cover" the pair $\{3, 13\}$. The 2-SPR-I algorithm would elect 28 to be a backbone node, since 28 is larger than 7 and 19. The 2-SPR-C algorithm however makes its decision primarily based on "covering number". One checks that node 19 only covers two pairs, namely $\{3, 13\}, \{3, 29\}$. Likewise, node 28 has covering number ten, but node 7 has covering number twenty-two. So node 7 has higher "priority" than node 19 and node 28 and hence it is elected to be in the backbone. A careful check reveals that the backbone set produced by 2-SPR-C is {1, 3, 4, 7, 8, 10, 12, 14, 17, 20, 21, 23}.

When 2-SPR-G is applied to this graph, the set produced is {1, 30, 29, 2, 22, 7, 27}. The interested reader is encouraged to construct the bipartite graph $H$ described in subsection III.D. The set $A$ consists of the nodes of the graph $G$. The set $B$ consists of pairs of nodes that are a distance three apart in $G$, such as $\{3, 13\}$. A node from $A$ is connected to a pair from $B$ if the node covers the pair, as for example, 19 covers $\{3, 13\}$. The greedy algorithm is then applied to produce a small subset of $A$ that cover, namely the one specified above.

Note that 2-SPR-$C_2$G algorithm elects two nodes from the set $A$ with highest covering numbers in the initial selection phase. For example, in the intial selection phase, $\{3, 13\}$ selects 7 and 28 and not 15 because node 7 and node 28 has higher covering numbers than node 15. Then the Greedy refinement algorithm is applied to the set produced after the intial selection phase. The set produced by this algorithm after the intial selction phase is {15, 22, 30, 29, 28, 1 7, 16, 27, 5, 6, 2}. It can be verified that after the Greedy refinement is applied, the set produced by 2-SPR-$C_2$G is {1, 30, 22, 2, 27, 28}. In this example, it turns out that the set size is same as the set produced by the Wu-Li algorithm applied to $G_2$. However, the set produced by 2-SPR-$C_2$G has an additional property, namely the 2-shortest path property.

When 2-SPR-PW is applied to this graph, the resulting set is as follows: {22, 30, 29, 15, 28, 16, 27, 20, 24, 6, 9, 12} which is slightly larger than 2-SPR-$C_2$G.

## VII. PERFORMANCE EVALUATION OF THE ALGORITHMS

We implemented the $d$-SPR-C, $d$-SPR-G, $d$-SPR-$C_2$G, $d$-SPR-PW and compared them with Wu-Li algorithm with rules 1 and 2 and also the $d$-SPR-I. The implementation was run on a single machine while simulating the distributed nature of the algorithms. Each node gathers the information it needs from its neighboring nodes and declares its results. The above mentioned algorithms produced $d$-hop connected $d$-dominating sets. We compared the size of the dominating sets generated by each of these algorithms. We also compared the message costs for $d$-SPR-I, $d$-SPR-C and $d$-SPR-G.

Fig. 2. Set Size for $d = 3$

### A. Methodology

For each experiment, a random disk graph was generated and measurements were taken on it. A disk graph is a graph in which a node is connected to all other nodes within a geometric radius defined for the disk graph. This radius can be seen as the coverage radius of a wireless link in the ad-hoc network. A random disk graph with $n$ nodes was created by selecting random points in a 900 by 900 pixel 2-$D$ region. Each node is connected to all other nodes within its coverage radius. The node degree was kept constant irrespective of the number of nodes in the graph.

We ran the experiments on graphs with varying number of nodes to compare different algorithms for producing $d$-hop connected $d$-dominating sets, as the number of nodes were changed. The algorithms considered were the Wu-Li algorithm applied to the graph $G_d$, the $d$-SPR-I algorithm, $d$-SPR-C, $d$-SPR-G, $d$-SPR-$C_2$G and $d$-SPR-PW for different parameters. Note that the algorithms $d$-SPR-I, $d$-SPR-C and Wu-Li are constant time. For every experiment, we ensure that the random graph generated have a radius sufficient to run all variants of the algorithms we consider.

### B. Results

Overall, the $d$-SPR-G algorithm performed quite well compared to others in terms of set size having the $d$-shortest path property. The set size for $d$-SPR-C was larger than that for Wu-Li with Rules 1 and 2 turned on. This is expected since the $d$-SPR-C may add more nodes into the set to ensure the $d$-shortest path property.

Fig. 2 shows the average size of the sets produced by each

Fig. 3. Set Size for $d = 4$

Fig. 4. Set Size for $d = 5$

algorithm. We computed the sets for nodes equal to 100, 200, 300, 400 and 500 for $d = 3$ and transmission radius 100. In Fig. 3 and Fig. 4, we computed the average size of the sets for nodes equal 100, 200, 300, 400 and 500 for $d = 4$ and $d = 5$ respectively and transmission radius 100.

We notice that the sets produced by the $d$-SPR-G are slightly larger than the sets produced by Wu-Li algorithm. This is reasonable since our algorithms need to add more nodes into the set to ensure the $d$-shortest path property. Let us also note that the $d$-SPR-G-sets along with all the variants of $d$-SPR-I have the $d$-shortest path property which Wu-Li algorithm does not have. The $d$-SPR-C, $d$-SPR-$C_2$G, $d$-SPR-PW and $d$-SPR-G are improvements of the $d$-SPR-I algorithm from the set size perspective.

However, $d$-SPR-C requires $2(d + 1)$ rounds of flooding where as $d$-SPR-I requires $d + 1$ rounds of flooding and Wu-Li requires $2d$ rounds of flooding. The size of the set produced by $d$-SPR-G and $d$-SPR-PW are considerably smaller than those produced by $d$-SPR-C and $d$-SPR-I algorithms. We compared the size of the sets produced by $d$-SPR-G, $d$-SPR-$C_2$G and $d$-SPR-PW and we found that the set sizes are quite similar. However, we noticed that the size of the sets produced by the $d$-SPR-PW algorithm is slightly less than the distributed greedy algorithms $d$-SPR-G and $d$-SPR-$C_2$G.

Fig. 5. Message Cost for $d = 3$

Let us also mention that all the greedy algorithms involve larger number of overhead messages than $d$-SPR-I or $d$-SPR-C. Unlike the other methods, each node broadcasts a number of messages that is not simply a (linear) function of $d$, but instead depends on the size of the network. The time complexity for the distributed greedy algorithm in the case of $d$-SPR-G appears to be the same as in [13], which essentially only differs from $d$-SPR-G in that a somewhat different covering problem is addressed, and which [11] indicates is polynomial in the size of the network. We also noticed that $d$-SPR-PW is at least as expensive as $d$-SPR-G.

Fig. 5, Fig. 6, and Fig. 7 display the average message costs for $d = 3$, $d = 4$, and $d = 5$ with nodes varying from 10 to 60. It is evident that $d$-SPR-G is substantially more expensive than $d$-SPR-I. As indicated already, this is to be expected since $d$-SPR-G involved multiple iterations of the basic $(d + 1)$-local view information flooding, produced by $d + 1$ rounds of message passing on the part of each node. After each iteration, some nodes will drop out of active participation in the algorithm, having become either "elected" or "not elected", but even these may need to continue to participate in forwarding messages for a while. Further iterations of $(d + 1)$-local view flooding are required until every node has becoming either "elected" or "not elected". The cost ratio between $d$-SPR-G and $d$-SPR-I is bounded above by the number of such iterations.

## VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed several new distributed algorithms that produce $d$-hop connected $d$-hop dominating sets. These sets can be used to create a virtual backbone of a wireless ad hoc network. In addition, these sets have a $d$-shortest path property which works efficiently in low mobility environments. This is the basis of our routing scheme. These algorithms produce sets that are considerably smaller than those produced by an algorithm previ-

Fig. 6. Message Cost for $d = 4$

Fig. 7. Message Cost for $d = 5$

ously introduced by the authors. The $d$-SPR-I and $d$-SPR-C algorithms have constant-time complexity in the sense that the time required is unaffected by the size of the network, assuming however that the node degrees are bounded by a constant.

In order to further reduce the size of the set produced by our algorithms, we are currently exploring some distributed probabilistic algorithms. We have also developed a hierarchical version of our routing scheme based on a generalization of the notion of a $d$-SPR set to edge-weighted graphs. In addition, we are also exploring the possibility of developing these algorithms into energy-efficient routing protocols.

## References


[1] N. Alon and J. Spencer. *The Probabilistic Method*, John Wiely & Sons, New York, 2000.

[2] K.M. Alzoubi, P. Wan, O. Frieder. New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks. *Proceedings of 35th Hawaii International Conference on System Sciences*, Hawaii 2002.

[3] A.D. Amis, R. Prakash, T.H.P. Vuong and D.T. Huynh. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, March 2000.

[4] Yuanzhu P. Chen and Arthur L. Liestman. Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks. *Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), pp. 157-164*, Lausanne, Switzerland, June 2002.

[5] Bevan Das and Vaduvur Bharghavan. Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. *IEEE International Conference on Communications (ICC '97)*, (1) 1997: 376-380.

[6] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, Vol 20, 1998.

[7] D. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9:256-278, 1974.

[8] Subhankar Dhar, Michael Rieck and Sukesh Pai. On Shortest Path Routing Schemes For Wireless Ad-Hoc Networks. *Proceedings of the 10th International Conference on High Performance Computing (HiPC '03)*, Lecture Notes in Computer Science, Springer Verlag, 2003.

[9] S. Dhar, M. Q. Rieck, S. Pai and E. J. Kim. Various Distributed Shortest Path Routing Strategies for Wireless Ad Hoc Networks. *Proceedings of the 5th International Workshop on Distributed Computing (IWDC 2003)*, Lecture Notes in Computer Science, Springer Verlag, December 2003.

[10] S. Kutten, D. Peleg. Fast Distributed Construction of Small $k$-Dominating Sets and Applications. *Journal of Algorithms*, 28(1):40-66, July 1998.

[11] L. Jia, R Rajaraman, T Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing, pp 33-42, August 2001.*

[12] L. Lovász. On the Ratio of Optimal Integral and Fractional Covers. *Discrete Mathematics*, 13:383-390, 1975.

[13] B. Liang and Z.J. Haas. Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management. *Proc. 19th Ann. Joint Conf. IEEE Computer and Comm. Soc. INFOCOM*, Vol. 3, pp. 1293-1302, 2000.

[14] Salvatore Rampone, Probability-driven Greedy Algorithms for Set Cover. *VIII SIGEF Congress "New Logics for the New Economy"* Naples, Italy, September, 2001.

[15] Michael Q. Rieck, Sukesh Pai, Subhankar Dhar, Distributed Routing Algorithms for Wireless Ad Hoc Networks Using d-hop Connected d-hop Dominating Sets. *Proceedings of the 6th International Conference on High Performance Computing: Asia Pacific*, Bangalore, December 16-19, 2002.

[16] J. Wu and F. Dai. Broadcasting in Ad Hoc Networks Based on Self-Pruning. *IEEE INFOCOM*, April, 2003.

[17] J. Wu Extended Dominationg-Set-Based Routing in Ad Hoc Wireless Networks with Unidirectional Links. *IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 9*, September 2002.

[18] J. Wu and H. Li. A Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks. *Special Issue on Wireless Networks in the Telecommunication Systems Journal*, Vol. 3, 2001, 63-84.