Distributed Routing Algorithms for Multi-hop Ad Hoc Networks using d-Hop Connected d-Dominating Sets

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

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

ABSTRACT

This paper describes a distributed algorithm (generalized d-CDS) for producing a variety of d-dominating sets of nodes that can be used to form the backbone of an ad hoc wireless network. In special cases (ordinary d-CDS), these sets are also d-hop connected and has a desirable "shortest path property". Routing via the backbone created is also discussed. The algorithm has a "constant-time" complexity in the limited sense that it is unaffected by expanding the size of the network as long as the maximal node degree isn't allowed to increase too. The performances of this algorithm for various parameters are compared, and also compared with other algorithms.

Keywords

d-dominating set, ad hoc wireless networks, d-closure, routing algorithm.

INTRODUCTION

One of the important problems in ad hoc wireless networks is to find efficient and reliable routing algorithms. In such a network, the nodes are often mobile and routing requires a dynamic adaptation strategy. However, it is important to first study static ad hoc networks, such as sensor networks, and to devise good routing schemes for these. This static routing problem will be the focus of this article. There are several approaches to it. A commonly used general method is *cluster-based hierarchical* routing [3], [7], [8], [11]. The network is divided into several clusters and from each cluster, certain nodes are elected to be clusterheads. These clusterheads are responsible for maintaining the routing information [1], [4]. Each cluster can have one or more gateway nodes to connect to other clusters in the network. These gateway nodes ensure connectivity between all the clusters in the network.

Another approach, called *backbone-based* routing selects certain nodes from the ad hoc network which are similar to gateway nodes. These nodes form a connected dominating set and are responsible for routing within the network [5]. However, this backbone tends to be rather large. Our approach blends features of these two approaches with the intention of gaining the advantages of each. The set produced by our basic algorithm (d-CDS) is not connected and does not produce a traditional backbone. It is however a d-hop connected d-dominating set with certain properties.

Each node in the network will obtain an awareness of the other nodes (including at least one backbone node) within d hops of itself. This facilitates local routing. Once the backbone has been obtained, the backbone nodes are expected to exchange global routing information with each other, using local routing to communicate with each other through intermediary ordinary nodes.

After this, when an ordinary node wishes to send a message to another node that is more than d hops away, it first polls the backbone nodes that are within d hops of itself. These then indicate to the sender, the distances from themselves to the target. The sender can then determine which backbone node to use to send the message most expeditiously. The message will then be forwarded through the backbone, using local routing to move the message between consecutive backbone nodes, which will never be separated by a distance in excess of d. Moreover, if the backbone was formed using the d-CDS algorithm, then the path followed will be guaranteed to be a shortest possible path from the source to the target.

DEFINITIONS

Throughout this article, G will denote a connected graph, representing an ad hoc network. V denotes the set of all vertices in the graph G. The *distance* function in G will be denoted by δ. A vertex u in G is said to have *eccentricity* $e(u)$ if G has a vertex v such that $\delta(u,v) = e(u)$, and for all vertices w in G, $\delta(u,w) \leq e(u)$. The *radius* of G, $r(G)$, is the minimum of the eccentricities of its vertices.

G_d will denote the *d-closure* of G, by which we mean the graph whose vertices are the same as those of G, but which has an edge between two vertices u and v if and only if $0 < \delta(u,v) \leq d$. We call a subset D of the set of vertices of G a *d-dominating set* of G if it is a dominating set for G_d, that is, if every vertex of G is within a distance d of some vertex in D. A 1-dominating set is simply called a *dominating set*. We say that D is *d-hop connected* if it is connected in G_d.

We say a distributed algorithm is *constant-time* when its execution time does not depend on the number of nodes in the network, although it may depend on the maximum vertex degree. The algorithms to be introduced in this paper have this property.

RELATED WORK

Minimum connected dominating sets have been used for backbone routing in wireless ad hoc networks. One of the earlier studies 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. The dominating set induces a virtual backbone of connected vertices in the graph. Since it is connected and dominating, the set is likely to be very big for a network with a large number of nodes. Moreover, if some node in the backbone were to fail, it may partition the induced subgraph.

The Max-Min scheme for clustering nodes in a wireless ad hoc network is described in [2], which introduces the concept of d-dominating sets and proves that finding a minimum d-dominating set is NP-complete. They use the nodes selected in this set to divide the graph into a set of clusters. They assume unique IDs (identifiers) for each node and select a node for inclusion in the set if it has the highest ID in some d-hop neighborhood. They describe a distributed way of finding the dominating nodes by flooding the node ID information for d rounds to all the neighbors of the node. Further, they do another d rounds of flooding to determine the clusters dominated by each node in the dominating set. This algorithm is constant-time.

CEDAR [9] is another distributed routing algorithm for ad hoc networks, which dynamically establishes a core network. This core network is a dominating set and this set is constructed by a core computation algorithm that approximates the minimum dominating set for the nodes. So the size of the core network is minimal. CEDAR is based on the following principles:

1. *Core election*: From the nodes of the ad hoc network, a subset (core) is elected by means of an approximation algorithm to compute minimum dominating set. This computation is done locally. The members of the core are called *dominators*.

2. *Link state propagation*: CEDAR tries to achieve QoS by propagating the link state information of links that are stable and has high bandwidth to the members of the core network. The information about dynamic links and links that have low bandwidth are kept local.

3. *Route computation*: First the core path is computed from the dominator of the source to the dominator of the destination. After that, the QoS routing algorithm computes the path with maximum bandwidth to the final destination node using local link state information.

In [10], the authors of CEDAR further extended their algorithm and proposed MCEDAR, which allows multicasting capabilities and provides robustness of mesh based routing protocols.

Jie Wu and Hailan Li present a basic algorithm [10], [11] for constructing a connected dominating set in a connected graph of radius at least two. This algorithm is distributed in the sense that each node processes local information that it receives from its neighbors in order to decide whether or not it should join the dominating set, and it is constant-time. They then consider some ways to refine the basic algorithm in order to produce smaller connected dominating sets.

The basic Wu-Li algorithm [13] can be characterized as follows.

WuLi$_0$: For each node z, the following question is asked: Does z have neighbors x and y such that x and y are not adjacent? The vertex z is then admitted to a set which we will call $WuLi_0(G)$ if and only if the answer to this question is "yes". It is then possible to show that $WuLi_0(G)$ is a connected dominating set, unless G is complete (*i.e.* has radius one).

WuLi$_1$: Wu and Li then refine the above technique, by assuming that each vertex has a unique integer identifier. Their "Rule 1" amounts to asking a further question for each vertex z in $WuLi_0(G)$, as follows: Does z have a neighbor z' in $WuLi_0(G)$ whose ID is higher than that of z, and which is such that all of the neighbors of z are also neighbors of z'? If so, z is deemed to be superfluous. The set $WuLi_1(G)$ consists of all the vertices from $WuLi_0(G)$ for which the answer to the question is "no". It too is a connected dominating set.

WuLi$_2$: To further reduce the size of the set, Wu and Li also introduce "Rule 2". For each vertex z in $WuLi_1(G)$, the following question is asked: Does z have two neighbors from $WuLi_1(G)$, which are themselves adjacent, and which have IDs larger than that of z, and which are such that their combined neighbors include all of the neighbors of z? The set $WuLi_2(G)$ consists of all the vertices from $WuLi_1(G)$ for which the answer is "no". This too can be shown to be a connected dominating set.

PROPOSED ALGORITHMS

Altered Wu-Li

Let us consider the possibility of replacing Wu and Li's "Rule 1" with a stronger condition, and refer to the resulting algorithm as *altered Wu-Li*. The resulting set of vertices will be denoted by $D(G)$. Specifically, the algorithm we wish to consider proceeds as follows:

1. Consider each pair of vertices x and y which are separated by a distance 2 in G.

2. For such a pair, consider all of the common neighbors of x and y. Let $E(x, y)$ denote the vertex among these common neighbors whose ID is largest.

3. Admit a vertex to the set $D(G)$ if and only if it is $E(x, y)$ for some suitable pair x and y.

We will say that $E(x, y)$ was "elected" by the pair x and y to join the set. Notice that vertices elected by altered Wu-Li are also in the set $WuLi_0(G)$. Moreover, any vertex eliminated by Rule 1 from $WuLi_1(G)$ would not be elected to $D(G)$. Thus $D(G) \subseteq WuLi_1(G)$.

An advantage of this approach over that of Wu and Li is the "shortest path property" described in the following theorem. This is a special case ($d = 1$) of Theorem 2, which is stated and proved in the next section.

Theorem 1: Assume that the connected graph G has radius at least two. Then the set $D(G)$ constructed by the altered Wu-Li algorithm is a connected dominating set. Moreover, any two vertices in G can be connected by a shortest path consisting solely of vertices from $D(G)$ (apart from the endpoints).

The d-hop connected d-dominating set algorithm

There is a trivial way to apply the Wu-Li algorithm or altered Wu-Li algorithm in order to produce a d-hop connected d-dominating set for G. To do so, simply apply the algorithm to G_d instead of G. Then, from the standpoint of G, the resulting set is a d-hop connected d-dominating set. However, because the graph G_d obscures the sense of distance in G, we feel that this is not a desirable approach.

By contrast, our *d-hop Connected d-Dominating Set* algorithm (*d-CDS*), to be proposed next, works directly with the graph G, rather than G_d, and results in a set with this desirable shortest path property. Moreover, we will show that this algorithm has a more efficient implementation. It is described as follows:

1. For each pair of vertices x and y satisfying $\delta(x, y) = d + 1$, consider all of the shortest paths from x to y.

2. Consider the set of vertices that lie strictly between x and y along such a path. Let $E(x, y)$ be the vertex in this set with the highest ID. Call this vertex $E(x, y)$.

3. Construct the set $D_d(G)$ by including all such $E(x, y)$, and only these vertices.

Note that $D_1(G) = D(G)$. This algorithm also has a "shortest path property", as described in the following theorem.

Theorem 2: Assume that the connected graph G has radius at least d+1. Then the set $D_d(G)$ is a d-hop connected d-dominating set. Moreover, any two vertices u and v from G can be connected by a shortest path in G, with the property that the set of vertices which are on this path and also in $D_d(G)$, together with the vertices u and v, form a connected path between u and v in the d-closure G_d.

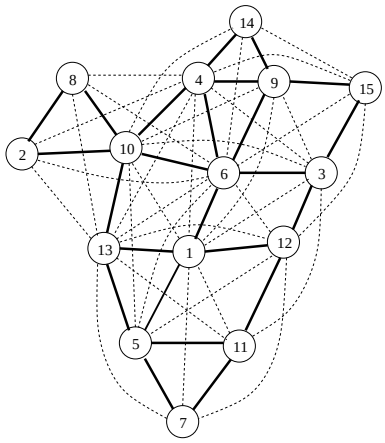

Figure 1 - Example G and G_2

Proof: Consider a vertex x in G. There exists a vertex y at a distance $d + 1$ from x. The vertex $E(x, y)$ is in $D_d(G)$ and is within a distance d of x. Hence $D_d(G)$ is d-dominating.

To show the rest, fix any two vertices u and v. Let p be a shortest path in G from u to v. Let u_j denote the vertex arrived at after taking j steps along this path. ($j = 0, 1, 2, ..., m$, where $m = \delta(u, v)$). Consider the vertices on p that are also in $D_d(G)$, together with the vertices u and v. Consider the subgraph of G_d induced by this set. If this is not a path from u to v in G_d, then let i be as large as possible so that u_i is either u or is in $D_d(G)$ and is connected to u in the induced subgraph. So $i < m - d$. So u_{i+d+1} is a vertex on the path at a distance $d + 1$ from u_i in G. Therefore, there is a path q of length $d+1$ from u_i to u_{i+d+1} which goes through an element of $D_d(G)$, namely $E(u_i, u_{i+d+1})$. Now create another shortest path in G from u to v by replacing the subpath of p from u_i to u_{i+d+1} with the path q. If the resulting path is still not satisfactory to establish the second claim in the theorem, then repeat the procedure. This time i will be larger. Continuing in this way, a suitable path will eventually be produced.

■

Example: Consider the example in Figure 1. The vertices here together with the solid edges constitute the graph G. By adding to this the dashed edges, which connect vertices separated by a distance two in G, the graph G_2 is obtained. In this example, when the Wu-Li algorithm is applied to the graph G_2, the set $WuLi_0(G_2)$ is found to consist of all the vertices except 2, 7, and 8. Each of these three vertices has the property that it forms a clique (in G_2) with its neighbors. That is, none of these three vertices has a pair of neighbors that are not adjacent to each other. Hence none of these three vertices is in $WuLi_0(G_2)$, while all the other vertices are in this set.

Rule 1 eliminates vertex 5 because all of its neighbors are also neighbors of vertex 13, and because $13 > 5$. Rule 2 eliminates several nodes. For example, vertex 1 is "covered by" vertices 6 and 13, together, in that the combined neigh-

$\{1,2\}$	$:\{4,6,10,\underline{13}\}$	$\{1,8\}$	$:\{4,6,10,\underline{13}\}$
$\{1,14\}$	$:\{4,6,9,\underline{10})\}$	$\{1,15\}$	$:\{3,4,6,9,\underline{12}\}$
$\{2,3\}$	$:\{4,6,\underline{10}\}$	$\{2,5\}$	$:\{6,10,\underline{13}\}$
$\{2,7\}$	$:\{\underline{13}\}$	$\{2,9\}$	$:\{4,6,\underline{10}\}$
$\{2,11\}$	$:\{\underline{13}\}$	$\{2,12\}$	$:\{6,\underline{13}\}$
$\{2,14\}$	$:\{4,6,\underline{10}\}$	$\{2,15\}$	$:\{4,\underline{6}\}$
$\{3,5\}$	$:\{1,6,10,11,\underline{12}\}$	$\{3,7\}$	$:\{1,11,\underline{12}\}$
$\{3,8\}$	$:\{4,6,\underline{10}\}$	$\{3,13\}$	$:\{1,4,6,10,11,\underline{12}\}$
$\{3,14\}$	$:\{4,6,9,10,\underline{15}\}$	$\{4,5\}$	$:\{1,6,10,\underline{13}\}$
$\{4,7\}$	$:\{1,\underline{13}\}$	$\{4,11\}$	$:\{1,3,\underline{13}\}$
$\{4,12\}$	$:\{1,3,6,13,\underline{15}\}$	$\{5,8\}$	$:\{6,10,\underline{13}\}$
$\{5,9\}$	$:\{1,6,\underline{10}\}$	$\{5,14\}$	$:\{6,\underline{10}\}$
$\{5,15\}$	$:\{6,\underline{12}\}$	$\{6,7\}$	$:\{1,5,12,\underline{13}\}$
$\{6,11\}$	$:\{1,3,5,12,\underline{13}\}$	$\{7,8\}$	$:\{\underline{13}\}$
$\{7,9\}$	$:\{\underline{1}\}$	$\{7,10\}$	$:\{1,5,\underline{13}\}$
$\{7,15\}$	$:\{\underline{12}\}$	$\{8,9\}$	$:\{4,6,\underline{10}\}$
$\{8,11\}$	$:\{\underline{13}\}$	$\{8,12\}$	$:\{6,\underline{13}\}$
$\{8,14\}$	$:\{4,6,\underline{10}\}$	$\{8,15\}$	$:\{4,\underline{6}\}$
$\{9,11\}$	$:\{1,\underline{3}\}$	$\{9,12\}$	$:\{1,3,6,\underline{15}\}$
$\{9,13\}$	$:\{1,4,6,\underline{10}\}$	$\{10,11\}$	$:\{1,3,5,\underline{13}\}$
$\{10,12\}$	$:\{1,3,5,6,\underline{13}\}$	$\{10,15\}$	$:\{3,4,6,9,\underline{14}\}$
$\{11,15\}$	$:\{3,\underline{12}\}$	$\{12,14\}$	$:\{6,\underline{15}\}$
$\{13,14\}$	$:\{4,6,\underline{10}\}$	$\{13,15\}$	$:\{4,6,\underline{12}\}$

Table 1 - Data for $D(G_2)$

$\{1,2\}$	$:\{6,10,\underline{13}\}$	$\{1,8\}$	$:\{6,10,\underline{13}\}$
$\{1,14\}$	$:\{4,6,\underline{9}\}$	$\{1,15\}$	$:\{3,6,9,\underline{12}\}$
$\{2,3\}$	$:\{6,\underline{10}\}$	$\{2,5\}$	$:\{10,\underline{13}\}$
$\{2,9\}$	$:\{4,6,\underline{10}\}$	$\{2,14\}$	$:\{4,\underline{10}\}$
$\{3,5\}$	$:\{1,6,11,\underline{12}\}$	$\{3,7\}$	$:\{11,\underline{12}\}$
$\{3,8\}$	$:\{6,\underline{10}\}$	$\{3,13\}$	$:\{1,6,10,\underline{12}\}$
$\{3,14\}$	$:\{4,6,9,\underline{15}\}$	$\{4,5\}$	$:\{1,6,10,\underline{13}\}$
$\{4,12\}$	$:\{1,3,\underline{6}\}$	$\{5,8\}$	$:\{10,\underline{13}\}$
$\{5,9\}$	$:\{1,\underline{6}\}$	$\{6,7\}$	$:\{1,\underline{5}\}$
$\{6,11\}$	$:\{1,3,5,\underline{12}\}$	$\{7,10\}$	$:\{5,\underline{13}\}$
$\{8,9\}$	$:\{4,6,\underline{10}\}$	$\{8,14\}$	$:\{4,\underline{10}\}$
$\{9,12\}$	$:\{1,3,6,\underline{15}\}$	$\{9,13\}$	$:\{1,4,6,\underline{10}\}$
$\{10,12\}$	$:\{1,3,6,\underline{13}\}$	$\{10,15\}$	$:\{3,4,6,\underline{9}\}$
$\{11,15\}$	$:\{3,\underline{12}\}$	$\{13,14\}$	$:\{4,\underline{10}\}$

Table 2 - Data for $D_2(G)$

bors of 6 and 13 include all the neighbors of 1. Moreover, vertices 1, 6 and 13 are pairwise adjacent, and of course $6 > 1$ and $13 > 1$. Therefore Rule 2 eliminates vertex 1. Likewise vertex 3 is covered by 6 and 12, vertex 4 is covered by 9 and 10, and vertex 11 is covered by 12 and 13. As a result, $WuLi_2(G_2) = \{6, 9, 10, 12, 13, 14, 15\}$. So $|WuLi_2(G_2)| = 7$.

Next consider applying altered Wu-Li to G_2. Two vertices are a distance two apart in G_2 if and only if they are a distance three or four apart in G. Table 1 is a list of such pairs of vertices $\{u, v\}$, together with the set of all the vertices that are common neighbors of u and v in G_2, with the largest of these neighbors underlined. Collecting all of the underlined numbers here, we see that $D(G_2) = \{1, 3, 6, 10, 12, 13, 14, 15\}$. So $|D(G_2)| = 8$.

Lastly, consider the 2-CDS algorithm applied to the graph G. Here only the pairs of vertices separated by a distance 3 in G are considered. Moreover, for any such pair, only vertices that lie between these two vertices along a path of length 3 are considered for inclusion into the set $D_2(G)$. The vertex among these paths with the highest ID will be included. Table 2 is similar to Table 1, but contains the data for forming $D_2(G)$ rather than $D(G_2)$.

So apparently $D_2(G) = \{5, 6, 9, 10, 12, 13, 15\}$ and $|D_2(G)| = 7$. Notice that this set contains the vertex 5, while $WuLi_2(G_2)$ does not. Also notice that the unique shortest path from 7 to 6 does not contain an intermediary node from $WuLi_2(G_2)$. Thus this set does not have the shortest path property. By Theorem 2, the set $D_2(G)$ must contain such a node, and have the shortest path property.

Generalization of d-CDS

The d-dominating set described in the previous section can be implemented in a distributed way discussed in the next section. But in fact, our method can be adjusted slightly to produce even more general d-dominating sets, and it is actually the implementation of this that will be presented later. A practical motivation for this generalization is the following. If we are willing to weaken somewhat the shortest path property requirement for the set $D_d(G)$ described in Theorem 2, then it is reasonable to expect that a smaller d-dominating set can be produced. In this section, we will consider how this might be achieved, with implementation details left until the next section.

Before presenting the technical details of the algorithm, let us look at an example. Consider the example graph G of the previous section and the set $D_2(G)$ produced by the 2-CDS algorithm. Now, suppose that we continue to use exactly the same pairs as in 2-CDS, namely the pairs of vertices separated by a distance 3 in G, but, for each such pair $\{u, v\}$, take as candidates for inclusion into the set of vertices being produced, all vertices that lie between u and v along a path whose length is either 3 or 4, as was the case in the altered Wu-Li algorithm example. This slight flexibility in the path lengths now means that for each pair, there are more candidates to consider. In a sense we are now blending desirable aspects of the 2-CDS algorithm and the altered Wu-Li algorithm: a small number of pairs to consider, as with 2-CDS, but for each pair, a large number of candidates, as with Altered Wu-Li. Both of these aspects help reduce the size of the set produced (although this is not evidenced by the small example presented here).

However, the properties of Theorem 2 have been compromised. The set need no longer be 2-dominating, nor have the shortest path property. 2-domination can be reclaimed by restricting, for each pair $\{u, v\}$, the associated candidate vertices to those that lie within a distance two of u and a distance two of v. So if a vertex lies along a path of length four connecting u and v, then it will only be considered a candidate based on this path if it lies midway between u and v.

This example is a particular case of the generalized d-CDS algorithm to be presented in this section, using the parameter values $d = e = 2, f = 3, g = 4$. The explanation for these values is that given any pair $\{u, v\}$ of vertices separated by a distance $f = 3$, we consider all paths that connect u and v, but whose length does not exceed $g = 4$, and along such paths, consider vertices w that are within a distance $d = 2$ of u and within a distance $e = 2$ of v. These vertices are the candidates for inclusion into the set being constructed. As usual, the vertex

$\{1,2\}$	$:\{6,10,\underline{13}\}$	$\{1,8\}$	$:\{4,6,10,\underline{13}\}$
$\{1,14\}$	$:\{4,6,7,\underline{9}\}$	$\{1,15\}$	$:\{3,4,6,9,\underline{12}\}$
$\{2,3\}$	$:\{4,6,\underline{10}\}$	$\{2,5\}$	$:\{10,\underline{13}\}$
$\{2,9\}$	$:\{4,6,\underline{10}\}$	$\{2,14\}$	$:\{4,6,\underline{10}\}$
$\{3,5\}$	$:\{1,6,10,11,\underline{12}\}$	$\{3,7\}$	$:\{1,11,\underline{12}\}$
$\{3,8\}$	$:\{4,6,9,10,\underline{15}\}$	$\{3,13\}$	$:\{1,6,10,11,\underline{12}\}$
$\{3,14\}$	$:\{4,6,9,10,\underline{15}\}$	$\{4,5\}$	$:\{1,6,10,\underline{13}\}$
$\{4,12\}$	$:\{1,3,6,13,\underline{15}\}$	$\{5,8\}$	$:\{6,10,\underline{13}\}$
$\{5,9\}$	$:\{1,6,\underline{10}\}$	$\{6,7\}$	$:\{1,5,12,\underline{13}\}$
$\{6,11\}$	$:\{1,3,5,12,\underline{13}\}$	$\{7,10\}$	$:\{1,5,\underline{13}\}$
$\{8,9\}$	$:\{4,6,\underline{10}\}$	$\{8,14\}$	$:\{4,6,\underline{10}\}$
$\{9,12\}$	$:\{1,3,6,\underline{15}\}$	$\{9,13\}$	$:\{1,4,6,\underline{10}\}$
$\{10,12\}$	$:\{1,3,6,\underline{13}\}$	$\{10,15\}$	$:\{3,4,6,9,\underline{14}\}$
$\{11,15\}$	$:\{3,\underline{12}\}$	$\{13,14\}$	$:\{4,6,\underline{10}\}$

Table 3 - Data for $D_{2,2,3,4}(G)$

with the highest ID is admitted into the set. In general the set is denoted $D_{d,e,f,g}(G)$, and so in the present example, the set is denoted $D_{2,2,3,4}(G)$. Table 3 shows the data when this algorithm is applied to the example graph G from the previous section. The resulting set is therefore $D_{2,2,3,4} = \{9, 10, 12, 13, 14, 15\}$, and so $|D_{2,2,3,4}| = 6$.

As suggested above, our general approach here is based on four non-negative integer parameters: d, e, f and g. We call it the *Generalized d-hop Connected d-Dominating Set* (*Generalized d-CDS*) algorithm, and it is described as follows:

1. For each pair of vertices x and y, a distance f apart, consider all paths from x to y whose length does not exceed g.

2. Consider the set $S_{d,e,g}(x, y)$ of all vertices that lie on at least one of these paths (including the endpoints), and which are within a distance d of x and within a distance e of y.

3. Define $E_{d,e,g}(x, y)$ to be the vertex with the largest ID among these vertices.

4. Define $D_{d,e,f,g}(G)$ to be the set of such $E_{d,e,g}(x, y)$ for all pairs x and y, as above.

The set $D_d(G)$ from the previous section is just the special case $D_{d,d,d+1,d+1}(G)$ here. Also, it should be noted that in general the set $S_{d,e,g}(x, y)$ and the vertex $E_{d,e,g}(x, y)$ can be defined for any vertices x and y, by simply taking $S_{d,e,g}(x, y) =$

$$\{\ z \mid \delta(x, z) \le d, \delta(y, z) \le e \text{ and } \delta(x, z) + \delta(y, z) \le g\ \},$$

and $E_{d,e,g}(x, y) = \max S_{d,e,g}(x, y)$, where "max" selects the vertex with the maximum ID from a set of vertices. In anticipation of the distributed algorithm in the next section for computing $D_{d,e,f,g}(G)$, we offer the following observations, where the notation $w \sim x$ means that the vertex w is adjacent to the vertex x.

Theorem 3: Fix non-negative integers d, e, f and g. Also, fix any two vertices x and y of G satisfying $f = \delta(x, y)$.

1. *If $0 < d$, $0 < g$, $f \le g$ and $e < f$, then*

$$S_{d,e,g}(x, y) = \bigcup_{w \sim x} S_{d-1,\, e,\, g-1}(w, y),$$

and so

$$E_{d,e,g}(x, y) = \max\{\ E_{d-1,\, e,\, g-1}(w, y) \mid w \sim x\ \}$$

2. *If $0 < d$, $0 < g$, $f \le g$ and $f \le e$, then*

$$S_{d,e,g}(x, y) = \{x\} \cup \bigcup_{w \sim x} S_{d-1,\, e,\, g-1}(w, y),$$

and so

$$E_{d,e,g}(x, y) =$$

$$\max\{\ x\ ,\ \max\{\ E_{d-1,\, e,\, g-1}(w, y) \mid w \sim x\ \}\ \}.$$

3. *If $d = 0$, $f \le e$ and $f \le g$, or if $f = 0$, then $S_{d,e,g}(x, y)$ is $\{x\}$, and so $E_{d,e,g}(x, y)$ is x.*

4. *In all other cases, $S_{d,e,g}(x, y)$ is empty, so that $E_{d,e,g}(x, y)$ is undefined.*

Proof: For item 1, consider first some $z \in S_{d,e,g}(x, y)$. This means that $\delta(x, z) \le d$, $\delta(y, z) \le e$, and there is a path from x to y that goes through z, and whose length does not exceed g. Since $e < f$, $z \ne x$. We may assume that the subpath from x to z is as short as possible, *i.e.* has length $\delta(x, z)$. Let w be the vertex immediately after x along this path. So $w \sim x$ and $\delta(w, z) = \delta(x, z) - 1 \le d - 1$. Consider the subpath from w to y that goes through z (*i.e.* the original path without x). This path demonstrates that $z \in S_{d-1,e,g-1}(w, y)$.

Conversely, let w be any neighbor of x. Let $z \in S_{d-1,e,g-1}(w, y)$. Consider a path from w to y through z with length less than or equal to $g - 1$. Extend this to a path (by adding one step) from x to y. Since $\delta(x, z) \le \delta(w, z) + 1 \le (d - 1) + 1 = d$, this path demonstrates that $z \in S_{d,e,g}(x, y)$. This establishes the first part of item 1. The second part is an immediate consequence of this.

Item 2 is similarly proved, taking note however that now x is an element of $S_{d,e,g}(x, y)$, but it might not be an element of any of the $S_{d-1,e,g-1}(w, y)$. Items 3 and 4 are straightforward to check.

■

Note that it may be assumed that $g \le d + e$, since $g > d + e$ implies that $S_{d,e,g}(x, y) = S_{d,e,d+e}(x, y)$.

Theorem 4: Fix non-negative integers d, e, f and g. Assume that the connected graph G has radius at least f, and that $0 < d < f \le g \le d + e$ and $0 < e < f$. Then the set $D_{d,e,f,g}(G)$ is a d-dominating set. Moveover, connecting any two vertices u and v, there exists a path $\tilde{p}$ with the following properties:

1. *The set of vertices on $\tilde{p}$ that are also in $D_{d,e,f,g}(G)$, together with u, form a path in G_d from u to a vertex whose distance in G to v is less than f.*

2. *The length of $\tilde{p}$ does not exceed*

$$\delta(u,v) + \max\{0, \lfloor 1 + \frac{\delta(u,v) - f}{f - e} \rfloor\} (g - f),$$

where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x.

Proof: The proof of the first part is essentially the same as in Theorem 2. For any vertex x, there is a vertex y with $\delta(x,y) = f$, and then the vertex $E_{d,e,g}(x,y)$ is both in the set $D_{d,e,f,g}(G)$ and within a distance d of x. Hence $D_{d,e,f,g}(G)$ is d-dominating.

The reasoning for the claim concerning a path $\tilde{p}$ from u to v is analogous to the proof of a less general version of this claim in Theorem 2. Begin with a shortest path p from u to v. The idea is to successively alter this path until a suitable path from u to v is obtained. If the length of p (*i.e.* $\delta(u,v)$) is less than f, then simply take $\tilde{p} = p$ and observe that nothing needs to be shown in this case. So assume otherwise. Let w be the vertex that is f steps along p away from u, so that $\delta(u,w) = f$. Let r be a path from u to w that passes through $E_{d,e,g}(u,w)$, and whose length does not exceed g. Let $u' = E_{d,e,g}(u,w)$, which of course is an element of $D_{d,e,f,g}(G)$. Note that $\delta(u,u') \leq d$ and that $\delta(u',v) \leq \delta(u',w) + \delta(w,v) \leq e + [\delta(u,v) - f] \leq \delta(u,v) - (f - e)$. Let q be the subpath of r from u to u'. Let p' be any shortest path from u' to v. Its length does not exceed $\delta(u,v) - (f - e)$. The path from u to v obtained by composing q with p' has length bounded by $\delta(u,v) + (g - f)$. This is because the length of r does not exceed g, and $\delta(w,v) = \delta(u,v) - f$.

Now, if the length of p' is not less than f, then treat p' as p was just treated, and hence obtain a vertex $u'' \in D_{d,e,f,g}(G)$, a path q' from u' to u'' whose length is bounded by d, and a shortest path p'' from u'' to v whose length is bounded by $\delta(u,v) - 2(f - e)$. The path from u to v obtained by composing q, followed by q', followed by p'', has length bounded by $\delta(u,v) + 2(g - f)$. And so forth.

This process iterates until, after say m iterations, we finally obtain a path $p^{(m)}$ from a vertex $u^{(m)}$ to v, whose length is less than f. Now m cannot exceed m_0, where this is defined to be the minimum integer for which $\delta(u,v) - m_0(f - e) < f$. It is thus seen that $m \leq m_0 = \lfloor 1 + (\delta(u,v) - f)/(f - e) \rfloor$. Now consider the path $\tilde{p}$ defined as the concatenation of $q, q', q'' \cdots q^{(m-1)}$ and $p^{(m)}$. This path has length bounded by $\delta(u,v) + m(g - f)$ which is bounded by $\delta(u,v) + \lfloor 1 + (\delta(u,v) - f)/(f - e) \rfloor (g - f)$.

■

The second part of Theorem 4 is a worse case analysis of the path lengths for acceptable routing paths. The average case for these lengths is more difficult to analyze, but one would expect that the average lengths are far shorter than the bound given in Theorem 4. This is supported by our experiments (see Chart 5).

In the next section, in connection with the application of Theorem 3 as the basis of a distributed algorithm, the following lemma will also be required.

Lemma 1: Fix two vertices x and y of the connected graph G. Suppose that $v_0, v_1, v_2, ..., v_k$ are vertices with $x = v_0 \sim v_1 \sim v_2 \sim \cdots \sim v_k = y$. Then for $0 \leq i \leq k$,

$$\delta(x,y) - i \leq \delta(v_i, y) \leq k - i.$$

Proof: $\delta(x,y) \leq \delta(x,v_i) + \delta(v_i,y) \leq i + \delta(v_i,y)$. This establishes the lower bound. The upper bound is immediate.

■

A DISTRIBUTED IMPLEMENTATION

The nodes in an ad hoc network, described by a connected graph G with uniquely labeled vertices (the IDs), can be coordinated in order to compute the set $D_{d,e,f,g}(G)$, where it will be assumed that d, e, f, g are integers with $0 < d \leq e < f \leq g \leq d + e$. In fact, assuming that their communications can be synchronized, each node only needs to transmit g times, and simultaneously receive the corresponding messages from its neighbors, and then process these messages. Theorem 3 and Lemma 1 provide the basis for the approach to be taken for computing $D_{d,e,f,g}(G)$.

In addition, each node x will learn about all of the nodes within a distance g of itself, and (by means of an array `next_node_to`) for each such node, y, will also know a neighbor of x which is closer to y than x is. This can then be used to route messages locally, *i.e.* within a distance g, without the need to use the network backbone.

In the following implementation, each message will consist of a number of ordered pairs or ordered triples of node IDs. For the first $g - d$ rounds of message passing, ordered pairs will be transmitted. For the remaining d rounds, ordered triples will be transmitted. To simplify the discussion, given a node x in the network, the integer ID(x) will simply be denoted as "x". Thus "x" must be read in context. The algorithm is as follows.

Initialization: Each node x establishes two (possibly associative) arrays `next_node_to` and `selected_node`, both indexed by node IDs, and containing node IDs, initially all `NULL` (the null node ID). Each node also maintains an (ordinary) array `nodes_at_a_distance` of lists (or pointers to lists) of node IDs. These are initialized so that `nodes_at_a_distance[0]` is a list consisting only of the given node x's own ID, while the other lists are empty.

After the k-th round of message passing, which could occur either in phase 1 or phase 2, the list `nodes_at_a_distance[`k`]` will contain the nodes at a distance k from x. If y is such a node, then `next_node_to[`y`]` will be the ID of a neighbor of x that is closer to y than x is. These two arrays ultimately facilitate local routing to nodes within g hops, as described later in this section.

Also, after the j-th round of phase 2, if a vertex y has a distance from x in the range $f - d + j$ to $g - d + j$, then `selected_node[`y`]` will be equal to the ID of the node $E_{j,\,e,\,g-d+j}(x,y)$. This array ultimately holds $E_{d,e,g}(x,y)$, and helps decide the set $D_{d,e,f,g}$.

Phase 1: For $g-d$ rounds ($j = 1, 2, ..., g-d$), each node x broadcasts to its neighbors, a message consisting of pairs of the form: (x, s). On the j-th round x will broadcast such pairs for vertices s satisfying $\delta(x, s) = j-1$. These are the nodes included in its list `nodes_at_a_distance[j-1]`.

Upon receiving a similar pair (w, y) from one of its neighbors, a node x checks to see whether or not its `next_node_to[y]` is `NULL`. If so, then `next_node_to[y]` is changed to w, `selected_node[y]` is set to x, and y is added to the list `nodes_at_a_distance[j]`. The reason for this is to initialize the array `nodes_at_a_distance` prior to phase 2. As indicated above, setting $j = 0$, this should initially equal $E_{0,e,g-d}(x, y)$, which in turn equals x since it must be a node within distance zero of x.

Phase 2: For d rounds ($j = 1, 2, ..., d$), each node x now broadcasts all triples (x, s, t) such that

1. $f - d + j - 1 \ \leq \ \delta(x, s) \leq g - d + j - 1$, and

2. $t = E_{j-1,\, e,\, g-d+j-1}(x, s)$.

The first of these two conditions can be managed via the array `nodes_at_a_distance`. The second condition amounts to t equaling `selected_node[s]` (as maintained by x during round j). Note that when $j = 1$, the second condition reads $t = E_{0,\, e,\, g-d}(x, s)$, which, assuming the first condition, means $t = x$ because $\delta(x, s) \leq g - d \leq e$.

Upon receiving all such triples from its neighbors for a given round, a node x considers collections of triples that share a common second entry y. Among these triples, let (w, y, z) denote the one with the largest third entry. Note that w must be adjacent to x. The node x now conditionally updates (if y is new to x) the entries `next_node_to[y]` and `nodes_at_a_distance[`$g - d + j$`]`, essentially as was done in phase 1, adjusting here to the fact that if y is a newly discovered vertex, then its distance from x is $g-d+j$, not j .

The ultimate goal is to compute $E_{d,e,g}(x, y)$ for pairs $\{x, y\}$ with $\delta(x, y) = f$. A subgoal during the j-th round of phase 2, for each node x, is to compute $E_{j,\, e,\, g-d+j}(x, y)$ for relevant choices of y. Toward this end, Theorem 3 may be iteratively applied. Lemma 1, setting i to $d - j$ and v_i to the x here, implies that on the j-th round it is only necessary to consider those y that satisfy

$$f - d + j \leq \delta(x, y) \leq g - d + j,$$

which can be checked via `nodes_at_a_distance` (as maintained by x).

Consider such a node y. If any triples having y as a second entry have been received by x from transmissions made during the previous round, then let (w, y, z) be as described earlier, *i.e.* it has the largest third entry among triples whose second entry is y. Otherwise, let $z =$ `NULL`. Define z' to be x if $\delta(x, y) \leq e$, corresponding to case 2 of Theorem 3. Otherwise, let $z' =$ `NULL`, corresponding to case 1. Let $z'' = \max\{z, z'\}$, where it is understood that `NULL` is less than any actual vertex. Using Theorem 3, it can be checked that z'' is in fact the vertex $E_{j,e,g-d+j}(x, y)$. This value is now stored in `selected_node[y]` (as maintained by x).

Once this has been done for all appropriate nodes y, the node x broadcasts a message consisting of the triples (x, y, z'') for which $z'' \neq$ `NULL`. After d rounds of this process, each vertex x will have stored the value $E_{d,e,g}(x, y)$ in `selected_node[y]`, for each vertex y whose distance from x falls in the range from f to g. Those whose distance is f determine the set $D_{d,e,f,g}$.

Once the set $D_{d,e,f,g}$ has been selected, routing information can be gathered and maintained by the nodes of this set. However, every node in the network will have already learned about all of the other nodes in its g-hop neighborhood, and so local messages can easily be passed between nodes within a distance g of each other without involving the backbone. This is achieved by means of `next_node_to`.

To manage general routing through the network, a routing process that involves only the d-dominating nodes in the network can be implemented. Link state information can be flowed from each dominating node to other d-dominating nodes in its d-hop neighborhood. A d-dominating node can keep information about the shortest path length from it to the other d-dominating nodes in its d-hop neighborhood. Upon receiving link state information, each dominating node can build a weighted graph for the whole network with each link in the graph having a weight equal to the length of the shortest path between the two d-dominating nodes. This graph can be used to compute the shortest path between any two d-dominating nodes.

Of course, each d-dominating node knows about all of the nodes within a distance g of itself. When a shortest path needs to be found from a non-dominating node to another, the first node can query all the d-hop neighbors that are d-dominating and find the best route to the other node by comparing the path lengths returned by each after adding the cost of the shortest path to that dominating node.

PERFORMANCE EVALUATION OF THE ALGORITHMS

We implemented the *Generalized d-CDS* algorithm from the previous section. This was used to evaluate the ordinary d-CDS algorithm ($d+1 = e+1 = f = g$), as well as a variation of this that admits longer paths for the backbone candidate pools ($d+1 = e+1 = f < g$). We compared these to the Wu-Li algorithm with optional use of rules 1 and 2, and also to the altered Wu-Li algorithm. Wu-Li and altered Wu-Li were applied to the d-closure graph to allow for sensible comparison with the d-CDS and generalized d-CDS algorithms. In this way, all of the algorithms produced d-dominating sets.

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. While the above mentioned algorithms generate d-hop connected d-dominating sets, they were also compared to the Max-Min algorithm, which computes a d-dominating set.

Performance Metrics Used

1. **Message cost:** All messages are sent across the network for a given algorithm until completion. At every step of any algorithm, each node sends at most one message to each of its neighbors.

Message cost is calculated as 1 packet whenever a node transmits some information to a neighboring node at the end of each step. There might be multiple set of information that a node needs to transfer to a neighbor in any given step. We assume that all that information can be packed in a single message packet. Since the degree of each node is bounded within a small value, we believe this is a reasonable assumption to make.

Generalized d-CDS: Each message is sent to all the neighboring nodes g steps. So the cost per node is $g * \delta$ where δ is the node degree. Add to that the cost of sending a selected message to every node in the d neighborhood of the node selected as dominating node at the end of the process.

d-CDS: Each message is sent to all neighboring nodes for f steps. So the cost per node is $f * \delta$ where δ is the node degree. Add to that the cost of sending a selected message to every node in the d-hop neighborhood of the node selected as dominating.

MaxMin: Each message is sent to all neighboring nodes d floodmax round and then d floodmin round. So the cost per node is $2 * d * \delta$ where δ is the node degree.

WuLi (Rules 1 and 2): First a node sends messages to its neighbors for d rounds to determine all the nodes to which it is adjacent in the d-closure of the original graph. Once every node knows its d-hop neighbors, it passes this information to its d-hop neighbors so that each node can compute rule 1 and rule 2 locally. This takes another d rounds of messages to each neighbor. Thus totally, it takes $2 * d * \delta$ messages per node.

altered WuLi: Similar to WuLi plus extra cost to let each winner that has been selected in the d-hop neighborhood of each node.

2. **d-dominating set size:** The number of nodes selected in the d-dominating set by each algorithm.

3. **Cumulative routing path length:** For every pair of nodes, the shortest paths connecting these in such a way that backbone nodes are routinely encountered within d hops, are determined. The length of all these paths is averaged for each pair of nodes for the whole graph. This determines the cumulative routing path length. Because of the shortest path property, in the cases of Altered Wu-Li and d-CDS, the paths of interest here will also be shortest paths through the graph G from the source to the target. In general, this is not the case for the other algorithms.

Each node, whether or not it is in the d-dominating set, maintains information about all the nodes in a d-hop neighborhood. (In the case of *Generalized d-CDS*, each node must keep track of the nodes that are within $f - 1$ hops of itself, not just d hops.) Since a local d-hop neighborhood graph is available at each node, a node can figure out the shortest path to any other node in its d-hop neighborhood. To find the shortest path to any node not in the d-hop neighborhood, a node contacts all the backbone nodes in the d-hop neighborhood and queries for the shortest path to the required node through that backbone node.

A routing process that involves only the backbone nodes in the network can be implemented that flows link state information from each backbone node to other backbone nodes in its d-hop neighborhood. A backbone node keeps information about the shortest path length from it to the other backbone nodes in its d-hop neighborhood. Upon receiving link state information, each dominating node can build a weighted graph for the whole network with each link in the graph having a weight equal to the length of the shortest path between the two backbone nodes. This graph can be used to compute the shortest path between any two backbone nodes.

Each d-dominating set node also keeps information about the network as a whole. When a shortest path needs to be found from a non-backbone node to another node, the first node queries all the backbone nodes in its d-hop neighborhood and finds the best route to the other node by comparing the path lengths returned by each after adding the cost of the shortest path to that backbone node.

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 300×400 pixel 2-D region. Each node is connected to all other nodes within its coverage radius. As the number of nodes in the graph increases, the degree of each node increases as there are more nodes in the vicinity of any node.

We ran the experiments on graphs with varying number of nodes to compare different algorithms for producing d-dominating sets, as the number of nodes were changed. The algorithms considered were the Max-Min algorithm of [2], different versions of the Wu-Li algorithm and the altered Wu-Li algorithm applied to the graph G_d, as well as the *Generalized d-CDS* algorithm with different values of f. Note that all these algorithms are distributed and constant-time. Hence, increasing the number of nodes has no bearing on the cost per node. But, the cost of computation and message costs depend on the degree of each node in the graph. Our intention here is to understand the

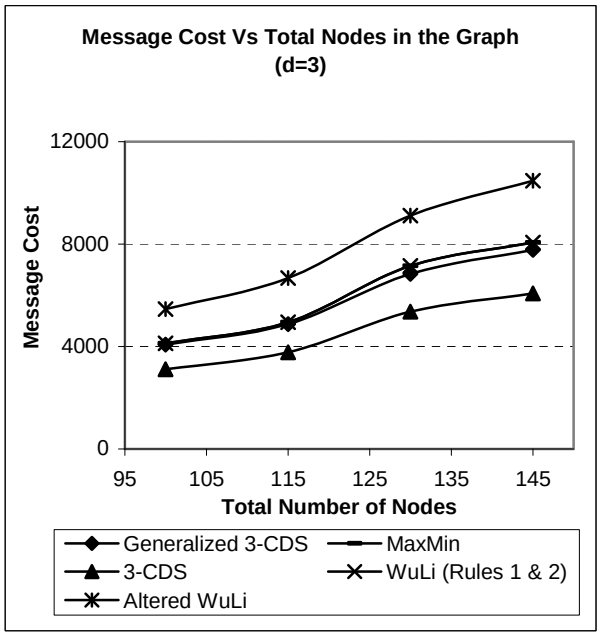

Chart 1: Message cost for $d = 3$

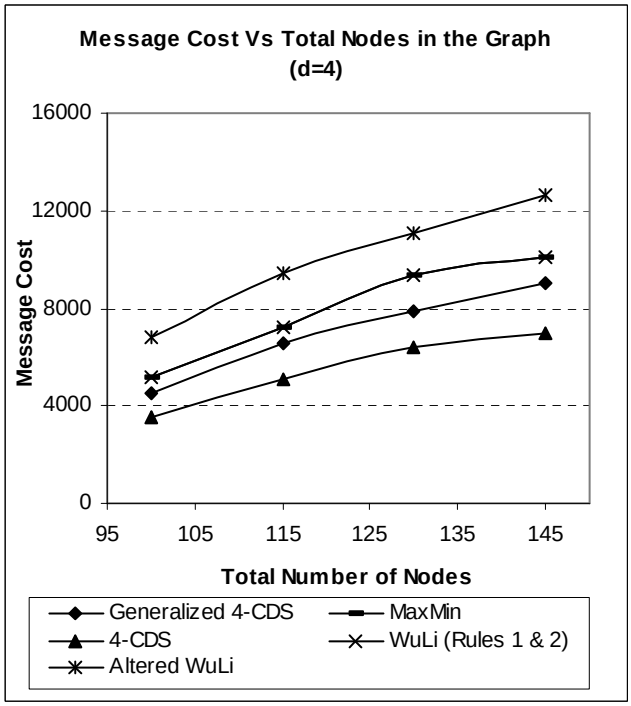

Chart 2: Message cost for $d = 4$

behavior of the algorithms as the density of the nodes in a given area increases. In our setup, we achieve this by simply increasing the number of nodes in the same pixel 2-D region. So, when we say we increase the number of nodes or we increase the density of nodes, we imply we are increasing the average degree of each node in the graph.

For every experiment, we ensure that the random graph generated have a radius sufficient to run all variants of the algorithms we consider. Specifically, we had the radius of the graphs to be at least $2d$ for a given value of d.

Results

Overall, the *Generalized d-CDS* algorithm performed very well compared to others in terms of the message costs and cumulative routing path lengths. The d-dominating set size for *Generalized d-CDS* was a little larger than that for Wu-Li with rules 1 and 2 turned on. This is expected since the *Generalized d-CDS* may add more nodes into the set to ensure the shortest path property or an approximation of this property (when $f < g$).

Chart 1 shows the message costs for each algorithm for $d = 3$ averaged over a few steps of perturbations for some graph. The basic Wu-Li with no rules applied has obviously the least cost as it only involves the cost of finding the d neighborhood for each node and then sharing neighborhood information with each d neighbors. As rule 1 and rule 2 are applied, the cost increases. Hence, Wu-Li with both rules has the higher cost when compared to *d-CDS* and *Generalized d-CDS*. For the *Generalized d-CDS*, where only cases with $g = f$ have been investigated, the cost increases as the value of f approaches $2d$. So, among the *Generalized d-CDS* versions, the one with $f = d + 1$, i.e., *d-CDS* has the least cost and it is very much close to basic Wu-Li. Min-Max has a cost that is equal to the cost of WuLi with both rules as can be seen from their message cost calculations. In both cases, each node sends information to its neighbors $2d$ times.

Comparing Chart 1 to Chart 2, we can see as we increase the value of d, the *Generalized d-CDS* gets more close to basic Wu-Li in terms of messages exchanged. All the *Generalized d-CDS* variants are upper bounded in cost by the cost for Max-Min. The *altered Wu-Li* now incurs more messages as it has to do more comparisons for each selection into the d-dominating set. Note that Wu-Li with both the rules turned on has the same message cost as MaxMin. So in Charts 1 and 2, we see that the legends for WuLi (rules 1 and 2) and MaxMin overlapped each other.

Charts 3 and 4 show the the d-dominating set size for the various algorithms for d=3 and d=4 respectively. Wu-Li with Rule 1 applied has the biggest dominating set. The *altered Wu-Li* set is slightly better than this. The *Generalized d-CDS* d-dominating sets follow the same curve as the previous ones but are better than them. This shows that the d-dominating set is affected by the connectivity of each node. As the number of nodes increase, there are more paths to be selected from and this increases the chance of a node getting into the dominating set. On the other hand, the Wu-Li with both rules has a much better d-dominating set as does the one for Max-Min. The set size remains the same as the node density increases. Recall that in our experiments, the nodes are constrained to a certain region, so that increasing their number increases their density and so too the vertex degrees. As the number of nodes increase, since the connectivity increases proportionally, there is an equal chance of eliminating some nodes based on Rule 2.

Cumulative path lengths for Wu-Li with both rules is

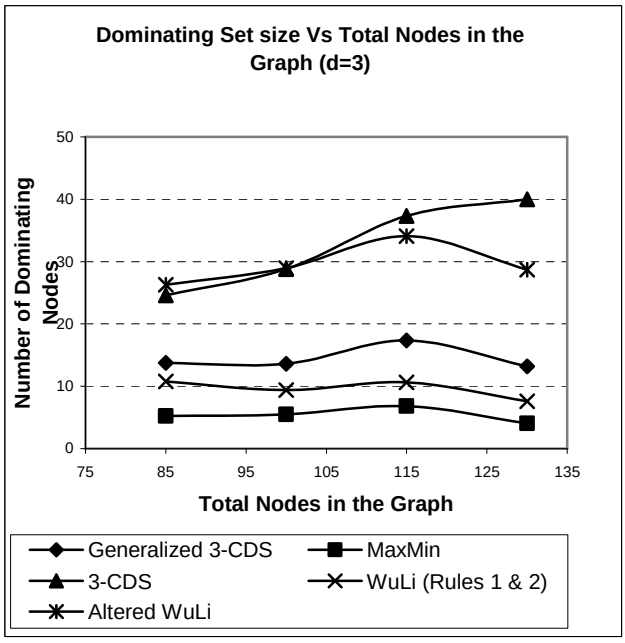

Chart 3: d-dominating set size for $d = 3$

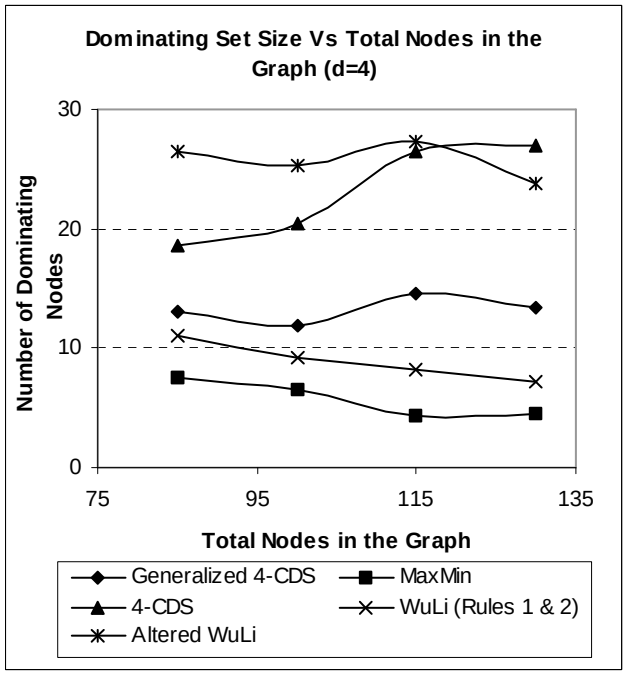

Chart 4: d-dominating set size for $d = 4$

compared with the cumulative path lengths for d-CDS. As expected, d-CDS always finds the shortest path between any two nodes. However, Wu-Li with both rules applied misses the shortest path for quite a few pair of nodes. Chart 5 shows the how worse the cumulative path lengths found by Wu-Li were as compared to the d-CDS ones. The y-axis represents the the ratio of the difference in the cumulative path lengths. If L_{WL} is the cumulative path length for Wu-Li and L_{Gen} is the cumulative path length for d-CDS, then the y-axis shows $(L_{WL} - L_{Gen})/L_{Gen}$.

We see that as the value for d increases the percentage

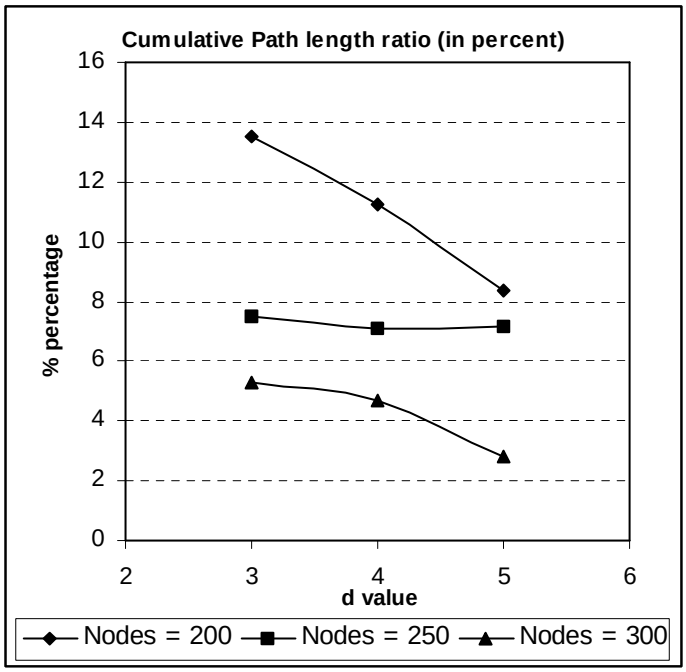

Chart 5: Cumulative path length ratio comparisons

difference in the cumulative path lengths go down. This is because, more nodes are now directly connected to other nodes. As the number of nodes increases, the nodes are more connected (as discussed in Methodology section) and consequently, more nodes are directly connected to other nodes. Hence, the percentage difference decreases.

CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel approach of finding a d-dominating set in an ad hoc wireless network that is also d-hop connected and has a certain shortest path property in some special cases. This is the basis of our routing scheme which is also very efficient from a cost perspective.

We are exploring cost efficient alternatives to Rule 2 in the Wu-Li algorithm. While we recognize that Rule 2 plays a very useful role in controlling the size of the set, it also sacrifices the shortest path property, and is costly to compute. One of several approaches which we are currently examining is the possibility of letting g exceed f in the *generalized d-CDS* algorithm. We are also considering the idea of changing the parameters used based on dynamically obtained information about the network, like vertex degree.

References

[1] A.D. Amis and R. Prakash. L. Load-Balancing Clusters in Wireless Ad Hoc Networks. *Proceedings of ASSET 2000*, Richardson, Texas, March 2000.

[2] 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.

[3] S. Bannerjee and S. Khuller. A Clustering Scheme for Higherarchical Control in Multi-hop Wireless Networks. *IEEE Infocom 2001*, Anchorage, Alaska, April 2001.

[4] M. Chatterjee, S. Das and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, Vol. 5, No. 2, April 2002, 193-204.

[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] Charles E. Perkins. Ad Hoc Networking. Addison-Wesley, Upper Saddle River, NJ, 2001.

[8] R. Ramanathan and M. Streenstrup. Hierarchically-organized multihop mobile wireless networks for quality-of-service support. *Mobile Networks and Applications*, Vol. 3, pp. 101-119, June 1998.

[9] R. Sivakumar, P. Sinha and V. Bharghavan. CEDAR: a Core-Extraction Distributed Ad Hoc Routing Algorithm *MIEE Journal of Selected Areas of Communication*, Vol. 17, No. 8, August 1999.

[10] P. Sinha, R. Sivakumar and V. Bharghavan. MCEDAR: Multicast Core-Extraction Distributed Ad Hoc Routing Algorithm *Proceedings of IEEE WCNC '99*, pp. 1313-1317, September 1999.

[11] C-K Toh. Ad Hoc Wireless Mobile Networks. Prentice Hall Inc, Upper Saddle River, NJ, 2002.

[12] J. Wu and H. Li. Domination and Its Applications in Ad Hoc Wireless Networks with Unidirectional Links. *Proc. of International Conference on Parallel Processing (ICPP)*, Aug. 2000, 189-200.

[13] Jie Wu and Hailian 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.