Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks *

Yuanzhu Peter Chen
School of Computing Science
Simon Fraser University
British Columbia, V5A 1S6, Canada
yzchen@cs.sfu.ca

Arthur L. Liestman
School of Computing Science
Simon Fraser University
British Columbia, V5A 1S6, Canada
art@cs.sfu.ca

ABSTRACT

We present a series of approximation algorithms for finding a small weakly-connected dominating set (WCDS) in a given graph to be used in clustering mobile ad hoc networks. The structure of a graph can be simplified using WCDS's and made more succinct for routing in ad hoc networks. The theoretical performance ratio of these algorithms is $O(\ln \Delta)$ compared to the minimum size WCDS, where Δ is the maximum degree of the input graph. The first two algorithms are based on the centralized approximation algorithms of Guha and Khuller [14] for finding small connected dominating sets (CDS's). The main contribution of this work is a completely distributed algorithm for finding small WCDS's and the performance of this algorithm is shown to be very close to that of the centralized approach. Comparisons between our work and some previous work (CDS-based) are also given in terms of the size of resultant dominating sets and graph connectivity degradation.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design, Wireless Communication*; G.1.2 [**Mathematics in Computing**]: Approximation—*Minimax Approximation and Algorithms*

General Terms

Algorithms, Experimentation

Keywords

ad hoc network, clustering, dominating set, weakly-connected dominating set, distributed algorithm

*This work is supported by The Natural Sciences and Engineering Research Council of Canada.

MOBIHOC'02, June 9-11, 2002, EPFL Lausanne, Switzerland.

1. INTRODUCTION AND RELATED WORK

1.1 Mobile Ad hoc Networks

A *mobile ad hoc network* is a type of infrastructureless wireless communication network, that requires only subscriber units to form the network. Ad hoc networks can involve hundreds of thousands of mobile subscriber units, each with a transmission range that is small relative to the network size. A subscriber unit can only directly communicate with other subscriber units that are within range. For users that are not within range to communicate, other subscriber units must relay messages. Thus, subscriber units may be able to communicate even though they may not be within their mutual transmission ranges. When not considering the physical layer or the data link layer issues, it is natural to represent an ad hoc network by a graph $G = (V, E)$ where the vertices represent the individual subscriber units and there is an edge between two vertices if the corresponding subscriber units are within range of each other.

An ad hoc network is a *multihop* network in which messages may travel along several links from the source to the destination via a certain path. The conventional pre-calculated routing algorithms for wired communication networks are not readily applicable to the ad hoc network since the subscriber units are mobile and the network topology is continuously changing. Ad hoc networks are also called *reconfigurable networks*. In wired networks, the network structure is essentially static and link failure is not frequent. In contrast, ad hoc networks allow total mobility which permits rapid topology change. Thus, pre-calculated routing information can become stale quickly. Routes must be calculated much more frequently to maintain the same response level as in wired networks. Royer and Toh have a very good survey [17] on the routing algorithms for ad hoc networks. Broch, et al. have a more experiment-oriented view of the comparison of the routing protocols in ad hoc networks [6].

1.2 Clustering ad hoc networks

Both the size of the network and the level of mobility of subscriber units affect the performance of the ad hoc network. Note that as subscriber units move the graph changes. We will assume, however, that the graph is always connected. In ad hoc networks, most topology changes are localized within a small area of the network. Therefore, it is desirable to abstract the network structure such that local changes need not be seen by the entire network. This is done using logical substructures called *clusters*. The pro-

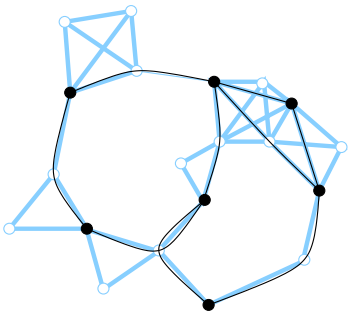


Figure 1: Abstracting a network topology

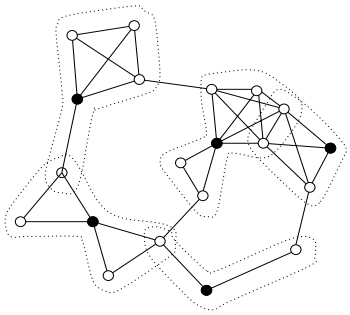


Figure 2: A dominating set.

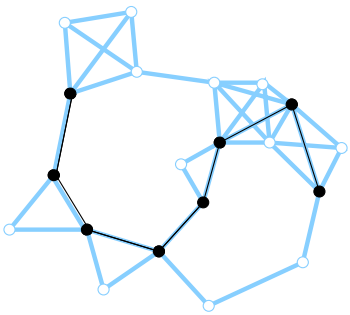


Figure 3: A connected dominating set.

cess of defining these substructures within the complete network topology is called *clustering*. In some cases, particular vertices called *clusterheads* may be designated to oversee channel allocation and message routing within each cluster. From a higher level view of the network, each of these clusters (perhaps, represented by its clusterhead), is a vertex and virtual connections between clusters are edges. Running a routing algorithm on this abstracted structure has lower cost, and local topology changes in the original network do not necessarily affect the higher level structure. Figure 1 gives an example of abstracting an ad hoc network. In the figure, the network structure is indicated in gray. The black vertices represent clusterheads and the black edges represent the virtual connections between clusters.

Baker, Ephremides and Flynn [2] began the study of the clustering problem in ad hoc networks by proposing a *linked cluster algorithm*. Several other authors have studied clustering and devised different approaches. Gerla and Tsai [13] presented two distributed algorithms to choose clusterheads. Their algorithms are based on lowest-ID and highest degree, respectively. Their approaches designates the vertices with the lowest-ID or highest degree within the neighborhoods as clusterheads and their neighborhoods as clusters. Basagni [4] generalized the work of Gerla and Tsai, and used a generic weight as a criterion for clustering. Bettstetter and Krausser [5] investigated the stability of the results presented in [4] by experiments. Krishna, Vaidya, Chatterjee and Pradham [16] presented another type of clustering which does not use clusterheads. In their model, a cluster corresponds to a clique in G and the resultant clusters are usually heavily overlapped. Banerjee and Khuller [3] used spanning trees to form clusters. In their model, a cluster is a subset of the vertices whose induced subgraph is connected in the spanning tree. Awerbuch, Du, Khan and Shavitt [1] studied clustering which they referred to as aggregated network routing. Another method used for clustering is based on *domination* in graphs, which is described in detail in the next subsection. Chen and Stojmenovic provide a more detailed review of these clustering techniques in [7]. Chlamtac and Farago [8] presented a random clustering which is essentially finding a small independent dominating set. Another recent clustering algorithm based on finding connected dominating sets has been proposed by Wu and Li [18].

1.3 Graph domination

A natural method for forming clusters is based on the idea of graph domination. A *dominating set* of a graph $G = (V, E)$ is a vertex subset $S \subseteq V$, such that every vertex $v \in V$ is either in S or adjacent to a vertex of S. A vertex of S is said to *dominate* itself and all adjacent vertices. For example, in Figure 2, the black vertices form a dominating set. We can use the vertices of a dominating set as clusterheads and assign each vertex to a cluster corresponding to a vertex that dominates it. In general, one wishes to find a small number of clusterheads, that is, a small dominating set, in order to simplify the network structure as much as possible. Unfortunately, finding a minimum size dominating set in a general graph is NP-complete [12]. Haynes, Hedetniemi and Slater present a detailed survey of the area of domination in graphs [15].

By constraining the dominating set, we can ensure that the clusterheads are adjacent to each other or are at least reasonably close to each other. This simplifies the construction of a higher level view of the network on which to route messages. A *connected dominating set* (CDS) of a given graph G is a dominating set whose induced subgraph is connected. For example, the black vertices of Figure 3 form a connected dominating set of the graph and their induced subgraph is indicated by the black lines. Such an induced subgraph can easily be used for routing messages between clusters. Unfortunately, finding a minimum size connected dominating set is also NP-complete [12]. Guha and Khuller [14] proposed two approximation algorithms for finding small connected dominating sets. Das and Bharghavan [9] implemented distributed versions of the above two algorithms.

Although a connected dominating set provides an obvious routing for messages, the connectivity requirement causes the number of clusters to be rather large. The number can be reduced by relaxing this requirement. The subgraph *weakly induced* by S ($S \subseteq V$) is the graph $\langle S \rangle_w = (N[S], E \cap (N[S] \times S))$. $\langle S \rangle_w$ includes the vertices in S and all of their

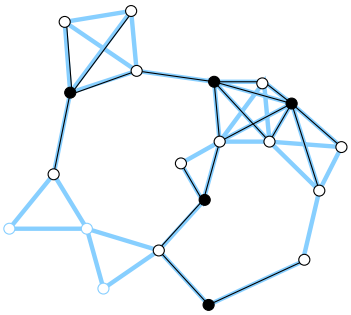


Figure 4: Weakly induced subgraph

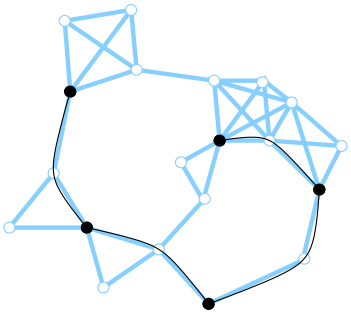


Figure 5: Network structure from the weakly-connected dominating set.

neighbors as the vertex set. The edges of $\langle S \rangle_w$ are all edges of G which have at least one end point in S. Figure 4 shows a subset S of vertices in black with the edges of $\langle S \rangle_w$ indicated by black lines. A vertex subset S is a *weakly-connected dominating set (WCDS)*, if S is dominating and $\langle S \rangle_w$ is connected. The black vertices in Figure 5 are an example of a weakly-connected dominating set S for the graph pictured in gray. The black edges show the corresponding abstracted network structure $\langle S \rangle_w$. Dunbar, et al. studied weakly-connected domination in graphs and, among other things, showed that it is NP-complete to find a minimum size dominating set of a given graph [10].

The vertices of a WCDS S can be used as clusterheads, and $\langle S \rangle_w$ can be used to route messages between clusters. In general, a WCDS can be smaller than a CDS, and yield fewer clusters. Computing a small WCDS is not more complicated than computing a small CDS. Thus, we feel that the WCDS is a better method for clustering than the CDS.

In this paper, we focus on the initial construction of a set of clusterheads for a graph. In future papers, we will address the issues arising from the change of network structure as subscriber units move. We propose a series of approximation algorithms for finding a small weakly-connected dominating set for a given graph. Section 2 introduces two sequential algorithms, Algorithm I and II, with asymptotically optimal approximation ratios. We then describe distributed implementations, Algorithm III and IV, respectively, of the sequential algorithms. In Section 3 we present a completely distributed algorithm for asynchronous networks, Algorithm V. We have designed experiments for comparing our clustering approach with the previous connected dominating sets. We report results comparing the size of the dominating sets and vertex pair distance/capacity degradation in Section 4.

2. CENTRALIZED ALGORITHMS

In this section, we present two centralized algorithms for finding small weakly-connected dominating sets in an arbitrary graph. For each algorithm we analyze its performance by determining the size of the resultant weakly-connected dominating sets compared to a minimum weakly-connected dominating set. These algorithms form the basis for the upcoming distributed algorithms.

2.1 Algorithm I

The first algorithm is based on Algorithm 2 of Guha and Khuller [14]. Given a graph $G = (V, E)$, we associate a color (*white*, *gray*, or *black*) with each vertex. All vertices are initially white and change color as the algorithm progresses. The algorithm is essentially an iteration of the process of choosing a white or gray vertex to dye black. When any vertex is dyed black, any neighboring white vertices are changed to gray. At the end of the algorithm, the black vertices constitute a weakly-connected dominating set.

The term *piece* is used to refer to a particular substructure of the graph. A *white piece* is simply a white vertex. A *black piece* contains a maximal set of black vertices whose weakly induced subgraph is connected plus any gray vertices that are adjacent to at least one of the black vertices of the piece. Figure 6 illustrates the definitions. The pieces are indicated by the dotted regions. Vertices 4, 5, 6, and 7 are each white pieces. The other vertices are divided among the two black pieces, one containing the black vertex 1 and the other containing the black vertices 2 and 3.

We define the *improvement* of a (non-black) vertex u to be the number of distinct pieces within the closed neighborhood of u. That is, the improvement of u is the number of pieces that would be merged into a single black piece if u were to be dyed black.

In each iteration, the algorithm chooses a single white or gray vertex to dye black. The vertex is chosen greedily so as to reduce the number of pieces as much as possible until there is only one piece left. In particular, a vertex with maximum improvement value is chosen (with ties broken arbitrarily). The black vertices are the required weakly-connected dominating set S.

Figure 6 illustrates the situation after the third iteration of the algorithm for the given graph. Dyeing vertex 5 black would merge 4 pieces, reducing the number of pieces by 3. Dyeing any of the other vertices black would merge at most 3 pieces. Thus, we choose vertex 5 to dye black in the next iteration.

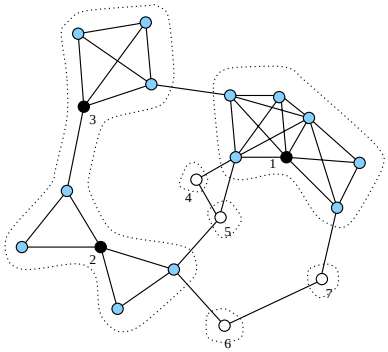


Figure 6: A snap shot of pieces.

Let OPT denote a minimum size weakly-connected dominating set for G and let Δ denote the maximum degree of G. We can bound the size of the WCDS found by Algorithm I as follows:

THEOREM 2.1. *The size of the weakly-connected dominating set found by Algorithm I is at most* $(\ln \Delta + 1) \cdot |OPT|$.

Proof: Let u_1 be an arbitrary vertex of OPT. As u_1 is of degree at most Δ, at most $\Delta + 1$ distinct vertices can be dominated by u_1 (including u_1 itself). As OPT is a weakly-connected dominating set of G, there must be another element of OPT at distance at most 2 from u_1. Let u_2 be such a vertex. As at least one vertex in u_2's closed neighborhood is also in the closed neighborhood of u_1, at most Δ new distinct vertices are dominated by u_2. Again, as OPT is a weakly-connected dominating set of G, there must be another element of OPT at distance at most 2 from either u_1 or u_2. This vertex, called u_3, dominates at most Δ new distinct vertices. We repeat this argument until we have included all $|OPT|$ elements of OPT. Thus, G can contain at most $n \leq (\Delta + 1) + \Delta(|OPT| - 1)$ vertices. It follows that $|OPT| \geq \frac{n-1}{\Delta}$.

In each iteration of the algorithm, we dye a vertex black and put it in set S. Observe that the improvement value of any vertex is monotonically non-increasing over time. At the start of the algorithm, the improvement of every vertex u is one more than its degree. When a neighboring vertex is colored black, u becomes gray and its improvement decreases by at least one. When a neighboring vertex is colored gray, u's improvement may decrease but will not increase. When u is dyed black, it no longer has an improvement value.

Let a_i be the number of pieces left after the i^{th} iteration and let $a_0 = n$. Consider the $i + 1^{st}$ iteration. Since the addition of the (non-black) vertices of OPT would join all of the remaining a_i pieces, decreasing the number of pieces by $a_i - 1$, there is at least one non-black vertex of OPT which would decrease the number of pieces by at least $\lceil \frac{a_i - 1}{|OPT|} \rceil$.

Figure 7 depicts the situation after 2 iterations on the given graph. In the figure, the five circled vertices are a minimum weakly-connected dominating set (OPT). At this point, one vertex u in OPT has already been dyed black. Picking the remaining 4 vertices of OPT would join the remaining $a_2 = 10$ pieces. Therefore, our greedy algorithm is guaranteed to decrease the number of pieces at least by $\lceil \frac{a_2 - 1}{|OPT|} \rceil = 2$ in the 3rd iteration.

This gives us the recurrence relation,

$$a_{i+1} \leq a_i - \lceil \frac{a_i - 1}{|OPT|} \rceil \leq a_i(1 - \frac{1}{|OPT|}) + \frac{1}{|OPT|}.$$

Solving it, we get the following bound:

$$\begin{aligned} a_{i+1} &\leq a_0(1 - \frac{1}{|OPT|})^{i+1} + \frac{1}{|OPT|} \sum_{j=0}^{i} (1 - \frac{1}{|OPT|})^j \\ &= (a_0 - 1)(1 - \frac{1}{|OPT|})^{i+1} + 1 \end{aligned}$$

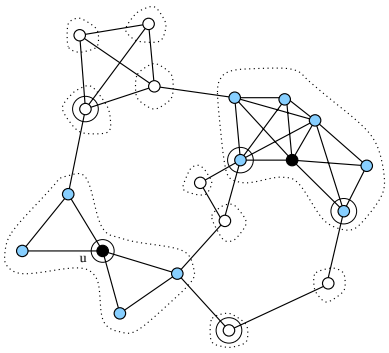


Figure 7: Greedy scenario.

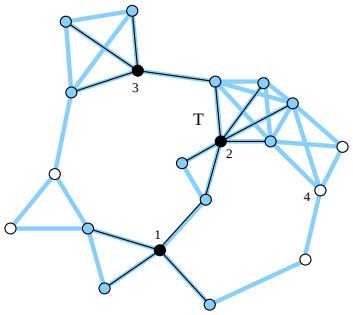


Figure 8: Growing the black piece.

Letting $i + 1 = |OPT| \cdot \ln \frac{a_0 - 1}{|OPT|}$, we have:

$$\begin{aligned} a_{i+1} &\leq (a_0 - 1)(1 - \frac{1}{|OPT|})^{i+1} + 1 \\ &= (a_0 - 1)(1 - \frac{1}{|OPT|})^{|OPT| \cdot \ln \frac{a_0 - 1}{|OPT|}} + 1 \\ &\leq (a_0 - 1)(\frac{1}{e})^{\ln \frac{a_0 - 1}{|OPT|}} + 1 \\ &= (a_0 - 1) \cdot \frac{|OPT|}{a_0 - 1} + 1 \\ &= |OPT| + 1 \end{aligned}$$

That is, after $|OPT| \cdot \ln \frac{a_0 - 1}{|OPT|}$ iterations, the number of pieces remaining is at most $|OPT| + 1$. For each additional vertex we choose, we will decrease the number of pieces by at least one. Thus, we need only pick at most $|OPT|$ additional vertices to reduce the number of pieces to one. The total number of vertices that we choose is no more than $|OPT| \cdot \ln \frac{a_0 - 1}{|OPT|} + |OPT|$. Since $|OPT| \geq \frac{n-1}{\Delta}$, the solution found by our algorithm has at most $|OPT| \cdot (\ln \Delta + 1)$ vertices. □

2.2 Algorithm II

Our second algorithm for constructing a weakly-connected dominating set is based on growing a single black piece T in the input graph $G = (V, E)$. As with the previous algorithm, the vertices of G are colored *white*, *gray*, or *black*. Initially, all vertices are white and T is empty. When a white or gray vertex is dyed black, all white vertices adjacent to it are colored gray. When a vertex is dyed black, it is placed into T along with all of its newly gray neighbors.

The algorithm starts by choosing an arbitrary vertex in G to dye black. In each subsequent iteration, a *candidate*

vertex is chosen to dye black. These candidates consist of the gray vertices of T and the white vertices adjacent to those gray vertices. In these later iterations, the algorithm considers all of these candidate vertices. For each candidate vertex u it counts the number of white vertices that are in u's closed neighborhood. By coloring u black, this number of vertices will be added to T. The candidate vertex with the maximum such value is chosen to be dyed black.

Figure 8 illustrates the situation after the 3rd iteration in the given graph. The black vertices 1, 2, and 3 are the vertices dyed in the first 3 iterations, respectively. Vertex 4, which is at distance two from the black vertex 2, is chosen in the 4th iteration, adding three vertices to T.

This iterative process continues until all vertices in G are non-white. The set of black vertices at the termination of the algorithm constitute the desired weakly-connected dominating set S.

Let S_b denote the set of black vertices at some point during the execution of the algorithm. As $\langle S_b \rangle_w$ is always connected and the algorithm terminates when there are no white vertices remaining, S is a weakly-connected dominating set of G. It can be shown that the size of the weakly-connected dominating set S found by Algorithm II is at most $(\lg \Delta + 2) \cdot |OPT|$.

2.3 Algorithms III and IV: Distributed Implementation of Algorithm I and II

In an ad hoc network, a subscriber unit does not know the structure of the network beyond its own neighborhood. Thus, in this section, we propose distributed versions of Algorithm I and II, called Algorithm III and IV respectively.

Algorithm I and II grow their weakly-connected dominating sets S by including a globally best vertex for each iteration. In order to achieve the same performance ratio in the distributed scenario, we use a special vertex to determine the global optimum.

To implement Algorithm III, we build a rooted spanning tree within the network and use the tree root as the arbitrator. The root starts an iteration by broadcasting a request throughout the network along the tree edges to search for the vertex with the maximum improvement value. The result is convergecast back to the root along the tree edges (with ties broken arbitrarily). Once the root has determined the global maximum, it unicasts a message to that vertex to color it black. The route for sending the unicast can be found in the convergecast search result.

Algorithm III generates the same weakly-connected dominating set as the its centralized counterpart on any input graph. Excluding building the spanning tree, the distributed version of Algorithm I can be done in $O(|V| \times |S|)$ time with $O(|V| \times |S|)$ messages. Note that if the spanning tree is balanced, i.e. the depth of the tree is at the order of th graph diameter $Diam(G)$, then the running time can be reduced to $O(Diam(G) \times |S|)$. The classic GHS algorithm for finding a minimum-weight spanning tree (Gallager, Humblet and Spira [11]) adds $O(|V| \log |V|)$ and $O(|V| \log |V| + |E|)$ to these costs, respectively, though any spanning tree algorithm will help.

A similar modification of Algorithm II results in the distributed Algorithm IV. In this case, we do not need to build a spanning tree beforehand, but can do it while growing the single black piece. A straightforward implementation can be done using $O(|S|^2)$ time and $O(|V| \times |S|)$ messages.

3. DISTRIBUTED ALGORITHM FOR FINDING A SMALL WCDS

Algorithms III and IV are still inherently sequential since only one vertex can be colored black in each iteration. Therefore, it is desirable to be able to color multiple vertices black in each iteration. In other words, it would be useful for different parts of G to have vertices colored black in parallel. In this section we propose Algorithm V, that finds a small weakly-connected dominating set in an asynchronous ad hoc network.

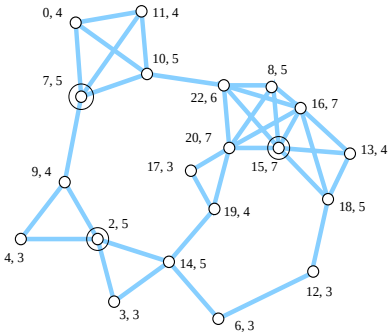
3.1 Overview

In a fully distributed approach, we grow multiple black pieces in parallel, each of which has its own internal decision mechanism to determine its own (local) best candidate. Here, any gray or white vertex may be a candidate. (Recall that in Algorithm II a candidate vertex was either a gray vertex or a white vertex adjacent to some gray vertex.) To be considered as a candidate, a gray or white vertex must also have the largest improvement among any vertex in its closed neighborhood. In each iteration, each piece calculates its own candidate(s). A black piece may have more than one candidate, while a white piece may have at most one candidate (itself). Each piece selects from its own candidate vertices the candidate with maximum improvement. The chosen candidate vertex is then colored black, causing neighboring white vertices to be colored gray and tangent pieces to be merged into a large one. A tie is broken arbitrarily, say by using the vertex ID. The *piece ID*, unique to each piece, of the new large piece is broadcast to all vertices in the new piece and the new piece is ready for next iteration. We assume every node knows the color and piece ID information of all its neighbors at any time. The algorithm terminates when no piece has a candidate with improvement greater than one.

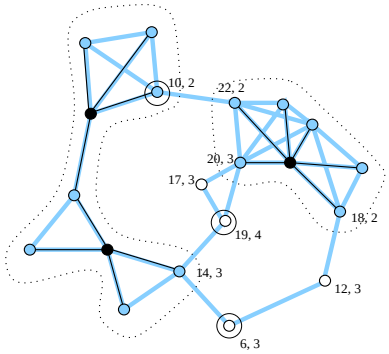
Figure 9 gives an example of how these best candidates are found in each iteration. The numbers beside each vertex are vertex ID, improvement pairs. We omit the vertex labels for which the improvement is 1. Figure 9(a) depicts the scenario of the first iteration when 3 white vertices, vertices 2, 7, and 15, find themselves candidates by comparing their improvements with those of their neighbors. When two neighboring vertices, for example vertex 7 and 10, have the same improvement, that with the lower ID is the winner. The 3 candidates are automatically the best candidates as they are the only vertices in their (white) pieces, and they decide to color themselves black. At the end of the first iteration, two larger black pieces are formed, as shown in Figure 9(b). During the second iteration, another two white vertices, vertex 6 and 19, and one gray vertex, vertex 10, are the best candidates. Note that in Figure 9(b), vertex 14 is not a candidate since it does not have the largest improvement in its closed neighborhood.

3.2 Algorithm V — Distributed Asynchronous Approach

Here, we use an asynchronous model where all vertices in the network are autonomous machines using their own clocks and there is no agreement on interlocked timing. We assume that there is a bound on how much time a link can delay message delivery and how much time a vertex can spend on a single step of execution.



(a) Iteration 1, 3 best candidates.



(b) Iteration 2, 3 best candidates.

Figure 9: First 2 iterations.

The basic process of our algorithm is as described above. Whenever a vertex is colored black (from white or gray), it causes all its white neighbors to be colored gray by sending an update message. Whenever a vertex changes its colors or piece ID, it informs all of its neighbors about the change.

The piece ID is used to help calculate the improvement of vertices. However, due to propagation delay, nodes in the same piece may appear to have different piece ID's. By calculating an improvement value without waiting for the piece ID's to be fully propagated, the improvement value may be calculated incorrectly. In our experiments, this situation does not occur very often and the size of the weakly-connected dominating set generated by Algorithm V is very close to that produced by Algorithm I (see Section 4). A *root vertex* of a piece is the vertex whose vertex ID is used as the piece ID. The algorithm proceeds as follows: a root vertex broadcasts a "best-candidate-inquiry" message within its piece. When other vertices forward this inquiry they only use the "black-black", "black-gray", or "gray-black" edges to make sure that the message does not propagate beyond the piece border. These out going messages form a broadcast tree in the piece. Leaves of this tree reply with their improvement values and internal vertices of the tree collect results from their children and forward the best value back up the tree. When the root knows that it has a best candidate with improvement of more than 1, it sends a "please-color-black" message to the corresponding vertex. When that vertex receives the "please-color-black" message, it colors itself black and tells all neighbors about this message. The white neighbors color themselves gray, causing some tangent pieces to merge. The newly colored black vertex becomes the root of the new piece and sends a "new-piece-ID" message to all vertices within the larger piece. The root also piggybacks the "best-candidate-inquiry" with the "new-piece-ID" message, starting the next iteration.

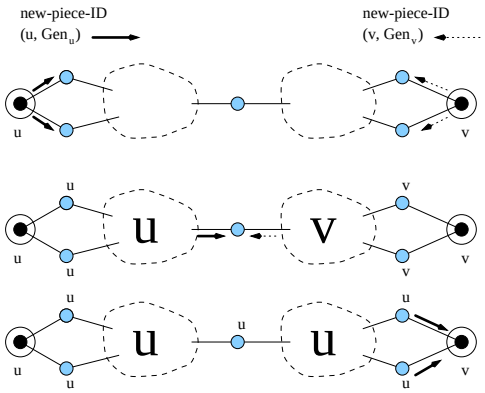


Figure 10: When two frontlines meet.

Due to the asynchrony and the possibility of multiple simultaneous changes to the structure of the pieces, it is possible that multiple vertices will declare themselves to be roots of a single piece and initiate a new search process. We need a mechanism to choose among the roots as their "new-piece-ID" messages encounter each other. We define the notion of the *generation* of a black vertex. A black vertex that has just been colored from white is a first generation black vertex. When a root (black) vertex chooses the best candidate of its piece, the old root assigns the new root to be of generation one greater than its own. The "new-piece-ID" messages carry the generation number of the root. When two "new-piece-ID" messages meet each other, later generation (with larger generation number) continues to broadcast while the earlier generation message is dropped. When two "new-piece-ID" messages with the same generation number meet, we favor the smaller piece ID to break the tie.

In Figure 10, we assume that two new black vertices, u and v, broadcast their "new-piece-ID" messages within the new black piece and u has a greater generation number. Before the messages meet, the left side portion of the piece accepts u as the new piece ID and the right side takes v. Once the two messages meet, the message of u overtakes that of v and pushes towards v so that the entire piece accepts u as piece ID.

The algorithm ends when a new black finds that its best candidate has an improvement value of one.

4. COMPARISONS

Using the members of a weakly-connected dominating set as clusterheads and the closed neighborhood of each clusterhead as a cluster, we can regard the network as a graph of clusters from a higher level. Clusterheads can be directly joined by a link or may share a neighbor that can relay communications between them. Note that links between non-clusterheads are not included in the higher level graph. Using a weakly-connected dominating set to define clusters helps to simplify the network structure. However, it may compromise the network structure to some degree as some edges are not included.

To measure the performance of our clustering algorithm, we consider three parameters. First we consider the size of the dominating set produced. Since our goal is to find

a small weakly-connected dominating set in order to abstract the network structure as much as possible, we prefer smaller values for this parameter. The second parameter is the average distance between pairs of vertices in the *clustered network* (subgraph weakly induced by the dominating set). Clustering using weakly-connected dominating sets removes edges not incident to vertices of the dominating set resulting in a sparser structure. This sparser structure dilates the network so that the distance between pairs of vertices in the clustered network may be longer than the corresponding distance in the original network. We do not wish the dilation to be too large, so smaller average distance is preferred. The third parameter is the average number of edge-disjoint paths between pairs of vertices in the clustered network. This parameter measures, in some sense, the capacity of the network. As we are simplifying the structure and deleting both edges and vertices, we can expect that this value will be less than in the original graph. However, we would prefer that it does not drop significantly.

In our experiment we generate random graphs repeatedly and run our algorithms and those of [9], measuring the above parameters. The random graphs have expected average degree of 6 and 12, providing two levels of density. The size of the graphs ranges from 20 to 200 vertices and 40 to 200 vertices, respectively. To simulate the structure of ad hoc networks, we place vertices (subscriber units) randomly in a rectangular area in a 2D-plane. The coordinates of the vertices are chosen uniformly in each dimension. We assign each subscriber unit a transmission range according to a normal distribution centered at a predefined expected value. When two subscriber units are placed within range of each other, an edge is added between the vertices simulating a reliable link between them. By changing the number of vertices in the plane and the expected transmission range, one can adjust the network size and density.

For each randomly generated network, we measure the dominating set size resulting from the two algorithms of [9] and our Algorithms I and V. Figure 11 shows the results obtained by the four algorithms when the graph is sparse (a) and dense (b). In the figure, CDS-SBP is "single-black-piece" (Algorithm II of [9]), CDS-MBP is "multiple-black-piece" (Algorithm I of [9]), WCDS-CTR is our centralized algorithm (Algorithm I), and WCDS-DST is our distributed algorithm (Algorithm V). It is apparent that our algorithms generate smaller dominating sets than those of Das and Bharghavan [9].

The average distance and number of edge-disjoint paths between vertex pairs are compared between the original graph, Algorithm I of [9] and our Algorithm V. Both sparse and dense graphs were considered as shown in Figure 12 and Figure 13. These results indicate that the average distance between vertex pairs increases by about 20% when connected dominating sets are used for clustering and increases a further 10% when weakly-connected dominating sets are used. These increases are not unreasonable. The average number of edge-disjoint paths is reduced by about 25% when connected dominating sets are used for clustering and by a further 20% with weakly-connected dominating sets. Again, these decreases are as expected.

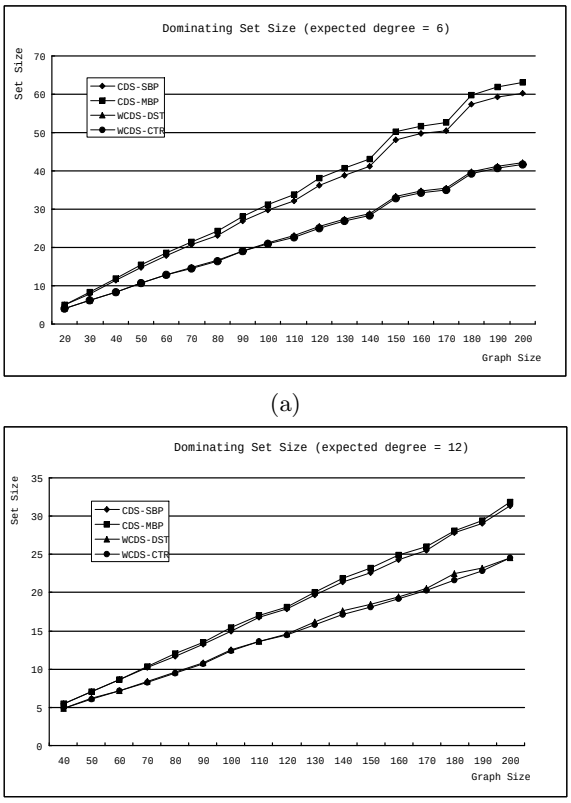


Figure 11: Dominating set size.

5. CONCLUSION

Ad hoc networks are a type of highly dynamic wireless networks, in which routing and clustering are important in determining the network performance. In this work we presented algorithms for finding small weakly-connected dominating sets including an asynchronous distributed algorithm. When using weakly-connected dominating sets to cluster ad hoc networks, the network structure can be greatly simplified with reasonable degradation of connectivity. Due to the dynamic nature of ad hoc networks, it is desirable to be able to maintain the clustering information at a reasonably low cost. Our next step is to devise a mechanism for maintaining the WCDS structure as the network changes.

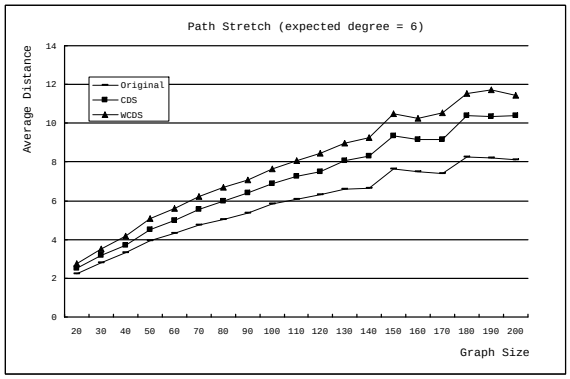
6. REFERENCES

[1] Baruch Awerbuch, Yi Du, Bilal Khan, and Yuval Shavitt. Routing through networks with hierarchical topology aggregation. In *IEEE ISCC'98*, pages 406–412, Athens, Greece, June 1998.

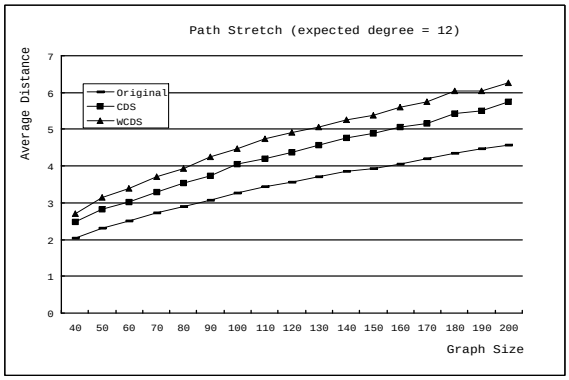
[2] Dennis Baker, Anthony Ephremides, and Julia A. Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE Journal on Selected Areas in Communications*, SAC-2(1):226–237, 1984.

[3] S. Banerjee and S. Khuller. A clustering scheme for hierarchical routing in wireless networks. Technical Report CS-TR-4103, University of Maryland, College Park, February 2000.

[4] B. Basagni. Distributed clustering for ad hoc networks. In *Proc. ISPAN'99 Int. Symp. on Parallel Architectures, Algorithms, and Networks*, pages

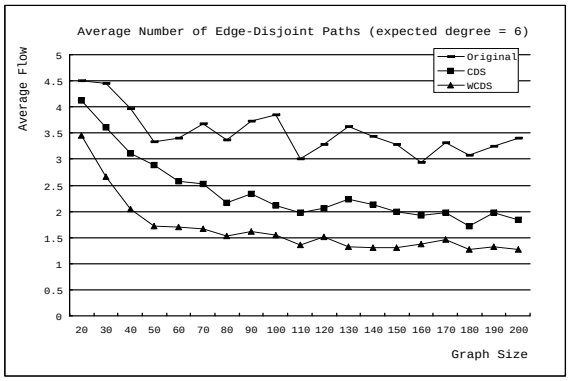


(a)

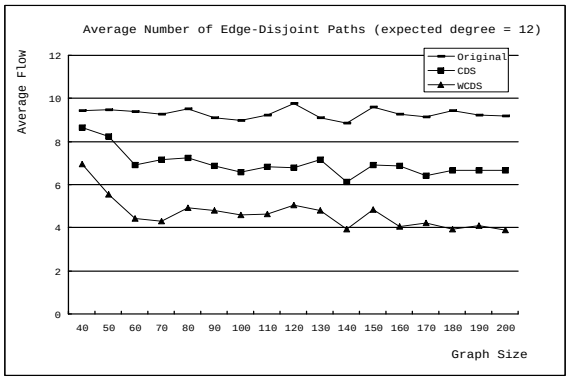


(b)

Figure 12: Average vertex distance.



(a)



(b)

Figure 13: Average number of edge-disjoint paths.

310–315, 1999.

[5] C. Bettstetter and R. Krausser. Scenario-based stability analysis of the distributed mobility-adaptive clustering (dmac) algorithm. In *Proc. ACM MobiHoc 2001*, pages 232–241, 2001.

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM, Dallas, TX*, October 1998.

[7] Geng Chen and Ivan Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, SITE, June 1999.

[8] I. Chlamtac and A. Farago. A new approach to the design and analysis of peer-to-peer mobile networks. *Wireless Networks*, 5:149–156, 1999.

[9] Bevan Das and Vaduvur Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications (ICC'97)*, June 1997.

[10] J. E. Dunbar, J. W. Grossman, J. H. Hattingh, S. T. Hedetniemi, and A. A. McRae. On weakly-connected domination in graphs. *Discrete Math.*, 167/168:261–269, 1997.

[11] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning tree. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, January 1983.

[12] M. L. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.

[13] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, mobile, multimedia radio network. *Wireless Networks*, 1:255–265, 1995.

[14] Sudipto Guha and Samir Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[15] Teresa W. Haynes, Stephen T. Hedetniemi, and Peter J. Slater. *Fundamentals of Domination in graphs*. Marcel Dekker, Inc., 1998.

[16] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *Computer Communication Review*, 49:49–64, 1997.

[17] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, 4 1999.

[18] Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIAL M'99*, Seattle, 1999.