International Journal of Foundations of Computer Science
© World Scientific Publishing Company

# A ZONAL ALGORITHM FOR CLUSTERING AD HOC NETWORKS [*]

YUANZHU PETER CHEN
*School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, British Columbia, V5A 1S6, Canada*

and

ARTHUR L. LIESTMAN
*School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, British Columbia, V5A 1S6, Canada*

Received 10 July 2002
Revised 15 October 2002
Communicated by S. Olariu

ABSTRACT

A Mobile Ad Hoc Network (MANET) is an infrastructureless wireless network that can support highly dynamic mobile units. The multi-hop feature of a MANET suggests the use of clustering to simplify routing. Graph domination can be used in defining clusters in MANETs. A variant of dominating set which is more suitable for clustering MANETs is the weakly-connected dominating set. A cluster is defined to be the set of vertices dominated by a particular vertex in the dominating set. As it is NP-complete to determine whether a given graph has a weakly-connected dominating set of a particular size, we present a zonal distributed algorithm for finding small weakly-connected dominating sets. In this new approach, we divide the graph into regions, construct a weakly-connected dominating set for each region, and make adjustments along the borders of the regions to produce a weakly-connected dominating set of the entire graph. We present experimental evidence that this zonal algorithm has similar performance to and provides better cluster connectivity than previous algorithms.

*Keywords:* Zonal algorithm; Ad hoc network; Clustering; Weakly-connected dominating set.

## 1. Introduction and Related Work

### *1.1. Preliminaries*

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 a great number of mobile subscriber units, each with a transmission

[*]This work is supported by The National Sciences and Engineering Research Council of Canada.

range that is small relative to the network size. Each subscriber unit can only communicate directly 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.

The conventional pre-calculated routing algorithms for wired communication networks are not readily applicable to ad hoc networks since the subscriber units are mobile and the network topology is continuously changing. In wired networks, the network structure is mostly static and link failure is not frequent. In contrast, ad hoc networks allow higher mobility which permits rapid topology changes. 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 on the routing algorithms for ad hoc networks [18]. Broch, et al. present a more experiment-oriented view of the comparison of the routing protocols in ad hoc networks [6].

In large MANETs, route calculation can be expensive. To simplify routing, it may be convenient to confine the routes to a substructure of the network. Informally a routing scheme can be such that a "local" message is sent on short paths, while a "long-distance" message travels a longer distance. Only a subset of the vertices are knowledgeable about routing. When a vertex not in that subset wishes to send a message, it first sends the message to a neighbor that is in the subset. From this vertex, the message can either be sent directly to its destination (local) or routed to another knowledgeable vertex within one hop of the destination and from there to the destination (long-distance). If the number of knowledgeable vertices is small, route calculation in the substructure will not be expensive.

One way to realize this local/long-distance notion is to introduce clustering. A cluster is a connected subnetwork usually with small diameter. We partition the vertices into clusters and select the edges of a spanning subgraph. The clusters along with the selected edges form the substructure of the network that we will use for routing. In fact, multiple layers of this kind of routing can be used, giving a hierarchical solution. In the majority of the clustering literature, a subset of the vertices of the graph are designated as clusterheads. Each clusterhead is the representative of its cluster and is responsible for calculating the routes for long-distance messages and forwarding inter-cluster packets.

Calls from any source vertex are first directed to its clusterhead. If the destination is in the same cluster (a local call), the clusterhead simply forwards the call to the destination. Otherwise, (for a long-distance call) the clusterhead routes the call, within the substructure, to the clusterhead associated with the destination, which will in turn forward it to the destination.

To simplify routing, we want the number of clusters generated for a given network to be as small as possible.

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.

Each edge is given a weight which is the distance between the corresponding subscriber units. 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. A dominating set can also be independent, called *independent dominating set*, in which no two vertices are adjacent. A *connected dominating set* (CDS) of a given graph $G$ is a dominating set whose induced subgraph is connected. Another variation is the weakly-connected dominating set. 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 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 $S$. A vertex subset $S$ is a *weakly-connected dominating set* (WCDS), if $S$ is dominating and $\langle S\rangle_w$ is connected. In general, one wishes to find the smallest possible dominating set, independent dominating set, connected dominating set, or weakly-connected dominating set of a graph for use in clustering MANETs. Unfortunately, the decision versions of the dominating set, independent dominating set, connected dominating set, and weakly-connected dominating set problems in general graphs are all NP-complete [10, 12]. Haynes, Hedetniemi and Slater present a detailed survey of the area of domination in graphs [17].

### *1.2. Related Work — Clustering MANETs*

One class of existing clustering algorithms is based on independent dominating sets of graphs. An independent set of a graph is a subset of the vertex such that no two vertices in this subset are adjacent. Gerla and Tsai [14] proposed two weight based clustering algorithms, using node ID and node degree for weights respectively. A vertex with optimal weight (lowest node ID or highest node degree, respectively) within its neighborhood is a clusterhead. The neighborhood of a clusterhead is a cluster. These algorithms are localized since a vertex only needs to know the information of 1-hop neighbors. Chen and Stojmenovic [7] proposed a clustering algorithm using both degree and ID. Basagni [4] generalized the weight idea such that any meaningful parameter can be used as the weight in order to best exploit the network properties.

Maintaining clusters as the topology changes is an issue that arises in ad hoc networks. A local change in weights may result in a different vertex becoming a clusterhead. If we insist on independent dominating sets, when two clusterheads become adjacent, one of them has to abdicate. All these local cluster maintenance may cause other changes to propagate through the network. This is commonly referred to as the *chain effect* [13].

Connected dominating sets are often used to determine clusterheads in MANETs. Guha and Khuller [16] proposed centralized approximation algorithms for finding small connected dominating sets in arbitrary connected graphs, which have asymptotically optimal approximation ratios of $O(\lg \Delta)$, where $\Delta$ is the maximum vertex degree of the input graph. Das and Bharghavan [9] provided distributed implementations of these algorithms, and applied them to ad hoc networks. These distributed algorithms have exactly the same approximation ratios as their centralized counterparts, since they utilize central coordinators to oversee the entire execution, but the computation is not localized. To address the problem of non-localized computation, Wu and Li presented a localized distributed algorithm in [20] for finding small connected dominating sets, in which each node only needs to know the information of vertices within two-hop distance. The execution time is $O(\Delta^3)$. However, the algorithm by Wu and Li does not guarantee a good approximation ratio. It was shown by Alzoubi et al. [19] that the approximation ratio for this algorithm can be as large as $\frac{n}{2}$, where $n$ is the number of nodes in the network.

Alzoubi, Wan and Frieder [1, 2, 3, 19] proposed a series of results on heuristics for finding small connected dominating sets in Unit Disk Graphs (UDGs). In [1, 3, 19], Alzoubi et al. proposed two distributed algorithms for finding small connected dominating sets to construct the *virtual backbone* of ad hoc networks (modeled with UDGs) with approximation ratios of 8 and 12 respectively. Although the algorithms proposed in [1, 3, 19] are distributed, they are not localized. In [2], Alzoubi et al. proposed a localized distributed heuristic. In this algorithm, the approximation ratio can be bounded by 192.

The weakly-connected dominating set was first proposed for clustering ad hoc networks by Chen and Liestman [8], The non-localized approximation algorithms in [8] were shown to find weakly-connected dominating sets consistently smaller than the connected dominating sets found by the algorithms of Guha and Khuller [16].

In this paper, we present a zonal distributed algorithm to find a small weakly-connected dominating set of the input graph $G = (V, E)$. The graph is first partitioned into non-overlapping regions. Then a greedy approximation algorithm is executed to find a small weakly-connected dominating set of each region. Taking the union of these weakly-connected dominating sets we obtain a dominating set of $G$. Some additional vertices from region borders are added to the dominating set to ensure that the final dominating set of $G$ is weakly-connected. Intuitively, this algorithm should have a good approximation ratio due to the computation of weakly-connected dominating sets in the regions. Note that although this algorithm is for finding weakly-connected dominating sets, minor modifications can be made to extend to find a small connected dominating set.

Section 2 describes the partitioning of the graph into regions. Section 3 presents a zonal algorithm for finding small weakly-connected dominating sets in the regions. The border fixing phase is investigated in Section 4. Experiments and results are presented in Section 5.

## 2. Graph partitioning using minimum spanning forests

In this section, we concentrate on the first phase of our zonal distributed clustering algorithm — the process of partitioning a given graph $G = (V, E)$ into non-overlapping regions. This is done by growing a spanning forest of the graph. At the end of this phase, the subgraph induced by each tree defines a region. We adopt the commonly used model in which a sender $v$ of degree $d(v)$ can broadcast to its neighbors in constant time with $O(d(v))$ messages. A vertex can also send a message to any adjacent vertex in constant time.

Our algorithm for this phase is based on an algorithm of Gallager, Humblet, and Spira [11] that is based on Kruskal's classic centralized algorithm for Minimum Spanning Tree (MST), an iterative greedy algorithm. We assume that all edge weights are distinct, breaking ties using the vertex IDs of the endpoints. The MST is unique for a given graph with distinct edge weights. The algorithm maintains a spanning forest. Initially, the spanning forest is a collection of trees of single vertices. At each step the algorithm merges two trees by including an edge in the spanning forest. During the process of the algorithm, an edge can be in any of the three states: tree edge, rejected edge, or candidate edge. All edges are candidate edges at the beginning of the algorithm. When an edge is included in the spanning forest, it becomes a tree edge. If the addition of a particular edge would create a cycle in the spanning forest, the edge is called a rejected edge and will not be considered further. In each iteration, the algorithm looks for the candidate edge with minimum weight, and changes it to a tree edge merging two trees into one. During the algorithm, the tree edges and all the vertices form a spanning forest. The algorithm terminates when the forest becomes a single spanning tree. This greedy algorithm finds the optimal solution, i.e., the MST.

Gallager, Humblet, and Spira [11] presented an asynchronous distributed algorithm, called the GHS algorithm, for finding the MST in a graph. In their algorithm, vertices in the graph grow their own tree in the spanning forest in parallel. Each tree in the spanning forest is called a *component*. Initially, every vertex is a component by itself. The central idea of the GHS algorithm is that each component locates its minimum weight outgoing edge, an edge of minimum weight which connects the component to some other component. Each vertex PROBEs each of its neighbors to see if the edge connecting them is an outgoing edge. The root of a component collects the weights of all outgoing edges to locate the minimum weight outgoing edge, and attempts to merge using the minimum weight outgoing edge. Some restrictions are applied during the merge process to guarantee balanced growth among components. The algorithm terminates when no outgoing edge can be found for any component, which means that the MST has been found.

Our partitioning process consists of a partial execution of the GHS algorithm, which terminates before the MST is fully formed. We control the size of components by picking a value $x$. Once a component has exceeded size $x$, it no longer participates. In particular, it will not send or accept PROBEs. We call such a component *full*. Since every merge joins two components of size of no more than $x$, the size of a full component is at most $2x$. Note that a component that has not exceeded the threshold size by the time that all of the surrounding components become full may be of size $x$ or less. To implement this size control, the root of each component keeps track of the number of vertices in its component. When the size $x$ is exceeded, the root broadcasts a FULL message to all vertices of the component. Having received the FULL message, a vertex refuses the PROBEs from any neighbor.

The threshold size $x$ provides a mechanism to control the tradeoff between the locality of the algorithm and size of the output weakly-connected dominating sets. When $x$ is small, the algorithm is fairly localized and its behavior is similar to that of the localized algorithms for connected dominating sets of Wu and Li [20] and of Alzoubi, et al. [2]. When $x$ is large, the algorithm is not as localized but tends to produce smaller weakly-connected dominating sets, as we will see in Section 5. The selection of the value of $x$ is a problem of interest. The threshold size $x$ can be preset to a particular value before the network is setup, if the scale and behavior of the network is relatively predictable. In particular, the value of $x$ may be determined by the size of the network, the density of the nodes, and the locality requirements of the administrator. Experiments can be used to fine tune the value given the above parameters.

In the future, we plan to develop algorithms for maintaining the cluster structure as the network changes. In this case, the value of $x$ will also need to be changed dynamically. However, this is beyond the scope of this paper.

Note that in the centralized greedy algorithm and the original GHS algorithm for MST, the spanning forest maintained at any time is part of some MST. This condition does not necessarily hold after adding the size control, since a component only considers edges leading to non-full components when locating the minimum weight outgoing edge. But this modification is acceptable as our goal is simply to partition a graph into regions using some spanning forest with the size control. We do not strictly require a minimum spanning tree.

To calculate the costs of the partitioning phase, let $m_i$ denote the number of edges incident on the vertices within component $i$. The time complexity to form this component using the modified GHS algorithm is $O(x \log x)$ and the message cost is $O(m_i + x \log x)$. (See the analysis of Gallager, Humblet and Spira [11].) Therefore the total time required is the same as that of each individual component, $O(x \log x)$, and the message cost is the sum over all the components, $O(m + n \log x)$. The threshold number $x$ is a variable used to control the approximate size of each partition. As $x$ increases, the partitions become larger which, intuitively, allows a smaller weakly-connected dominating set, as we can run the asymptotically optimal approximation algorithm (as we will see in next section) within larger partitions of the graph.

In principal, any distributed algorithm for constructing spanning trees can be modified for use here. Among algorithms for minimum spanning trees, the GHS algorithm does not have optimal time or message costs. However, the GHS algorithm is localized during the early stages of its execution and thus is easy to extend to our problem. Further, using a minimum spanning tree algorithm with edge weights obtained from subscriber unit distance (rather than an arbitrary spanning tree algorithm) tends to give us spanning trees of smaller heights and thus regions of smaller diameters.

## 3. Computing Weakly-Connected Dominating Sets of the Regions

Once the graph $G$ is partitioned into regions and a spanning tree has been determined for each region, we run the following algorithm within each region. This color-based algorithm is a distributed implementation of the centralized greedy algorithm for finding small weakly-connected dominating sets in graphs (see [8]). We begin by describing the centralized greedy algorithm. We then describe a distributed implementation and its regional version.

The centralized algorithm (previously presented by Chen and Liestman [8]) is based on Algorithm 2 of Guha and Khuller [16]. 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. The *improvement* of a (non-black) vertex $u$ is 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 $S$.

Let $OPT$ denote a minimum size weakly-connected dominating set for $G$ and let $\Delta$ denote the maximum degree of $G$. We can bound the size of the WCDS found by this algorithm as follows (see also [8]):

**Theorem 1** *The size of the weakly-connected dominating set found by the above centralized algorithm 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 $\Delta$, 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 $\Delta$ 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 $\Delta$ 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

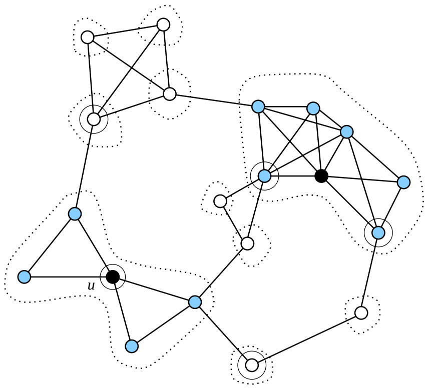

Fig. 1. Greedy scenario.

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 1 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,

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

Solving it, we get the following bound:

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

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

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

$$\begin{aligned} &= (a_0 - 1) \cdot \frac{|OPT|}{a_0 - 1} + 1 \\ &= |OPT| + 1 \end{aligned}$$

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. □

The above algorithm grows its weakly-connected dominating set $S$ by including a globally optimal vertex for each iteration. In order to achieve the same performance ratio in the distributed scenario, the following algorithm uses a special vertex to determine the global optimum.

To implement the distributed algorithm, 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 sends a message to that vertex to color it black. The route for this message can be found in the convergecast search result.

This algorithm generates the same weakly-connected dominating set as its centralized counterpart on any input graph. Excluding building the spanning tree, the distributed algorithm 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 on the order of the graph diameter $Diam(G)$, then the running time can be reduced to $O(Diam(G) \times |S|)$.

To compute weakly-connected dominating sets for each region, we simply run the above distributed algorithm within each region, ignoring the edges to other regions, using the spanning tree of the region defined during the partitioning phase. For a given region, the root broadcasts a request throughout the region along the spanning tree edges to search for the vertex with 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 maximum improvement value of the region, it sends a message to the chosen vertex to color it black. After the new vertex is colored black, pieces are merged and we are ready for the execution of the next iteration. This regional version of the algorithm is terminated when there is only one piece in the region.

This algorithm can be run in any region $R_i$ in time $O(x \times |S_i|)$, using $O(x \times |S_i|)$ messages, where $S_i$ is the weakly-connected dominating set produced by $R_i$. Therefore, within the entire graph, the total time is $O(x \times |S_{max}|)$ and the message cost is $O(n \times |S_{max}|)$, where $S_{max}$ is the largest weakly-connected dominating set produced by all regions.

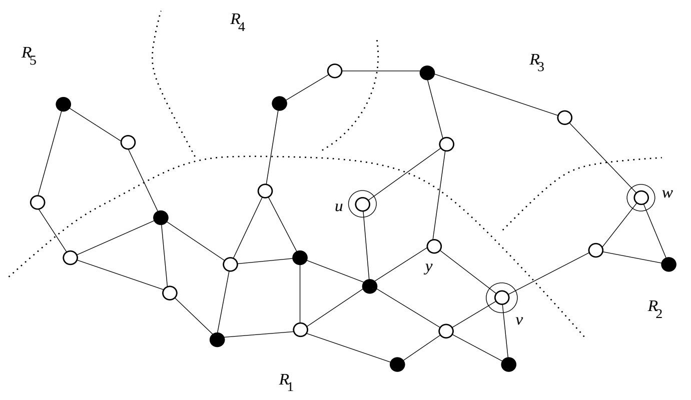

Fig. 2. Fixing borders.

## 4. Fixing the Borders

After we have calculated a small weakly-connected dominating set $S_i$ for each region $R_i$ of $G$, combining these solutions does not necessarily give us a weakly-connected dominating set of $G$. We will likely need to include some additional vertices from the borders of the regions in order to obtain a weakly-connected dominating set of $G$.

The edges of $G$ are either *dominated* (that is, they have either endpoint in some dominating set $S_i$) or *free* (in which case neither endpoint is in a dominating set). Two regions $R_i$ and $R_j$ joined by a dominated edge can comprise a single region with dominating set $S_i \cup S_j$, and do not need to have their shared border fixed.

The root of region $R$ can learn, by polling all the vertices in its region, which regions are adjacent and can determine which neighboring regions are not joined by a dominated edge. For each such pair of adjacent regions, one of the regions must "fix the border". To break ties, we assume that the region with lower region ID takes control of this process, where the region ID is the vertex ID of the region root. In other words, if neighboring regions $R_i$ and $R_j$ are not joined by a shared dominated edge, the region with the lower subscript adds a new vertex from the $R_i/R_j$ border into the dominating set.

For example, in Figure 2, the five regions have weakly-connected dominating sets indicated by the solid black vertices. Region $R_1$ is adjacent to regions $R_2$, $R_3$, $R_4$, and $R_5$. Among these, regions $R_2$ and $R_3$ do not share dominated edges with $R_1$. As $R_1$ has a lower region ID than either $R_2$ or $R_3$, $R_1$ is responsible for fixing these borders. The root of $R_1$ adds $u$ and $v$ into the dominating set. $R_2$ is adjacent to two regions, $R_1$ and $R_3$, but it is only responsible for fixing the $R_2/R_3$ border, due to the region IDs. The root of $R_2$ adds $w$ to the dominating set.

A detailed description of this process for a given region $R$ follows. The goal is for the root $r$ to find a small number of dominated vertices within $R$ to add to the dominating set.

Assume that every vertex knows the vertex ID, color, and region ID of all of its neighbors. (This can be done with a single round of information exchange.) Root

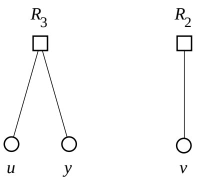

Fig. 3. Bipartite graph for the problem of fixing a border.

$r$ collects the above neighborhood information from all of the border vertices of $R$. We define a *problem region* with regard to $R$ to be any region $R'$ that is adjacent to $R$, does not share dominated edges with $R$, and has a higher region ID than $R$. Region $R$ is responsible for fixing its border with each problem region.

We construct a bipartite graph $B(X, Y, E)$ using the collected information for root $r$. Vertex set $X$ contains a vertex for each problem region with regard to $R$, and vertex set $Y$ contains a vertex for every border vertex in $R$. There is an edge between vertices $y_i$ and $x_j$ iff $y_i$ is adjacent to a vertex in problem region $x_i$ in the ordinal graph. Figure 3 shows the bipartite graph constructed by region $R_1$ in the example of Figure 2. In this bipartite graph, $X = \{R_2, R_3\}$ and $Y = \{u, y, v\}$. In this case, $\{u, v\}$ is a possible solution for $R_1$ to add to the weakly-connected dominating set in order to fix its borders with $R_2$ and $R_3$. To find the smallest possible set of vertices to add to the dominating set, $r$ must find a minimum size subset of $Y$ to dominate $X$.

The decision version of this problem is equivalent to the dominating set problem in split graphs, which is NP-complete (see [15] for definition of split graph and [5] for NP-completeness result). As an optimal solution is not likely, we use a greedy procedure to find an approximate solution. In each iteration of this greedy procedure, we remove a maximum degree vertex from $Y$. Since the calculation for each region is done by the region root, the time and message costs are those by the polling process, i.e. a broadcast and a convergecast along the tree edges of the spanning tree of each region. Both costs can be bounded by $O(x)$ given the size of regions are within $O(x)$. Therefore, the time and message costs for the entire graph are $O(x)$ and $O(n)$, respectively.

After the root of each region has executed the above algorithm, the resulting dominating set is a weakly-connected dominating set of the entire graph $G$.

## 5. Simulation

At the point, we have not been able to obtain a theoretical bound on the size of the weakly-connected dominating set produced. Therefore we have done simulations to determine the quality of this zonal distributed algorithm for finding small weakly-connected dominating sets, measuring the dominating set size, the path length, and the network capacity. We compare this algorithm to the non-localized distributed algorithm from Section 3 and to Das and Bharghavan's algorithm for connected dominating sets [9].

To measure the performance of our zonal 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, we prefer smaller values for this parameter. The second parameter that we are interested in is the average path length, that is, the average pairwise vertex distance. In each of the clustered graphs, we compute these distances using only the edges in subgraph weakly induced by the dominating set. Small increases of this parameter are expected, as some edges in the clustered graph are removed. The third parameter is the average number of edge-disjoint paths between pairs of vertices. This parameter measures, in some sense, the capacity of the network. As we delete edges, we expect that this value will decrease, but we do not wish it to decrease too much.

Originally we planned to include the localized algorithms of Wu and Li [20] and of Alzoubi, et al. [2] in our comparison, but our preliminary experiments showed that for our data set these localized algorithms produced much larger dominating sets than the non-localized distributed algorithm of Das and Bharghavan [9]. Therefore, in this paper we focus on comparing our algorithm with the non-localized distributed algorithm presented by Chen and Liestman [8] and that by Das and Bharghavan [9].

In our experiment we generate random graphs repeatedly and run our zonal algorithm, the non-localized distributed algorithm from Section 3, and the connected dominating set algorithm [9], measuring the above three 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 pair of number of vertices and expected degree, we generated 100 random graphs to test the algorithms. In order to investigate the influence of the size of the regions on the performance of the algorithm, we configured the experiment to run with parameter $x$ set to the values 10, 20, 40, 80 and 160. Given the sizes of the graphs in our simulations (up to 200), values 20 and 80 represent two typical locality levels, and these are the results presented in this paper. Intuitively, when $x = 20$, the algorithm behaves more locally. In particular, with average degrees either 6 or 12, we could expect the region diameter to be 4 or less. Raising the threshold to 80, we would expect the region diameter to be closer to 6. As the region diameter increases, the locality of the algorithm decreases and it begins to behave more like the non-localized algorithm of [8].

Figures 4 and 5 show the size of the weakly-connected dominating sets produced by the zonal algorithm with region parameter $x$ set to 20 and 80 (LOC-20 and LOC-80, respectively) and by the non-localized algorithm from Section 3

(WCDS), and size of the connected dominating sets produced by the CDS algorithm, Algorithm II of [9], (CDS). Connected dominating sets are generally denser than weakly-connected dominating sets for the same graph, so we expect the size of connected dominating sets produced by the CDS algorithm to be larger than the others. In Figure 4, where the expected degree is 6, we can see that when the regions contains approximately 80 vertices (LOC-80) the size of the weakly-connected dominating set produced by the zonal algorithm is just slightly larger than that produced by the non-localized algorithm (WCDS). For smaller regions (LOC-20), the size of the dominating set is again larger, but still smaller than the size of the connected dominating sets (CDS). In Figure 5 the expected degree is 12. The difference from Figure 4 is that when using small regions (LOC-20), the size of the weakly-connected dominating sets is larger than that of the connected dominating sets (CDS), though LOC-80 is still smaller than CDS. In this case, we believe that the graph is fragmented by small regions because the threshold size is not much larger than average degree.

Figures 6 and 7 display the average path length in the original graph (Original), that in the graph clustered by weakly-connected dominating sets produced by the zonal algorithm (LOC-20 and LOC-80), that in the graph clustered by weakly-connected dominating sets produced by the non-localized algorithm (WCDS), and that in the graph clustered by the connected dominating sets produced by the CDS algorithm (CDS). In both Figures 6 and 7, we can see that LOC-20 and LOC-80 yield a smaller average path length than WCDS. Especially, for small regions (LOC-20), the path length in the graph clustered by weakly-connected dominating sets as sparser dominating sets, is even smaller than that in the graph clustered by the connected dominating sets.

Figures 8 and 9 show the average number of edge-disjoint paths in the original graph (Original), that in the graph clustered by weakly-connected dominating sets produced by the zonal algorithm (LOC-20 and LOC-80), that in the graph clustered by weakly-connected dominating sets produced by the non-localized algorithm (WCDS), and that in the graph clustered by the connected dominating sets produced by the CDS algorithm (CDS). In both Figures 8 and 9, we can see that LOC-20 and LOC-80 have a larger average number of edge-disjoint paths than WCDS (WCDS). Again counter-intuitively, for small regions (LOC-20), the average number of edge-disjoint paths in the graph clustered by weakly-connected dominating sets is smaller than that in the graph clustered by the connected dominating sets.

## 6. Conclusion

We presented a zonal distributed algorithm for finding small weakly-connected dominating sets for use in clustering MANETs. The algorithm consists of three phases: partitioning, regional WCDS, and border fixing. We first partition the input graph into regions of sizes approximately $x$ using a modified GHS algorithm for constructing spanning forests. Then the distributed algorithm for weakly-connected dominating sets is executed in each region. In the last phase, some additional ver-

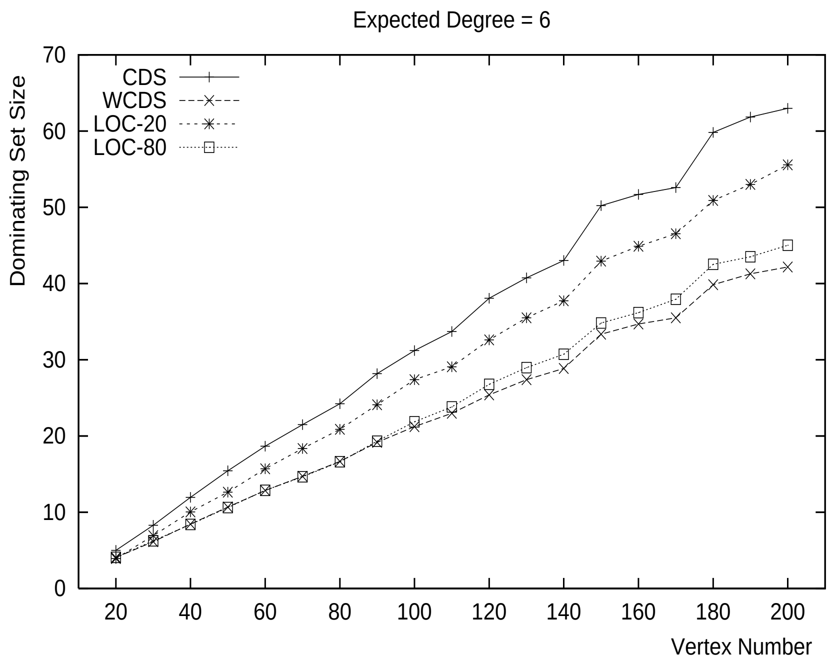

Fig. 4. Dominating set size (a).

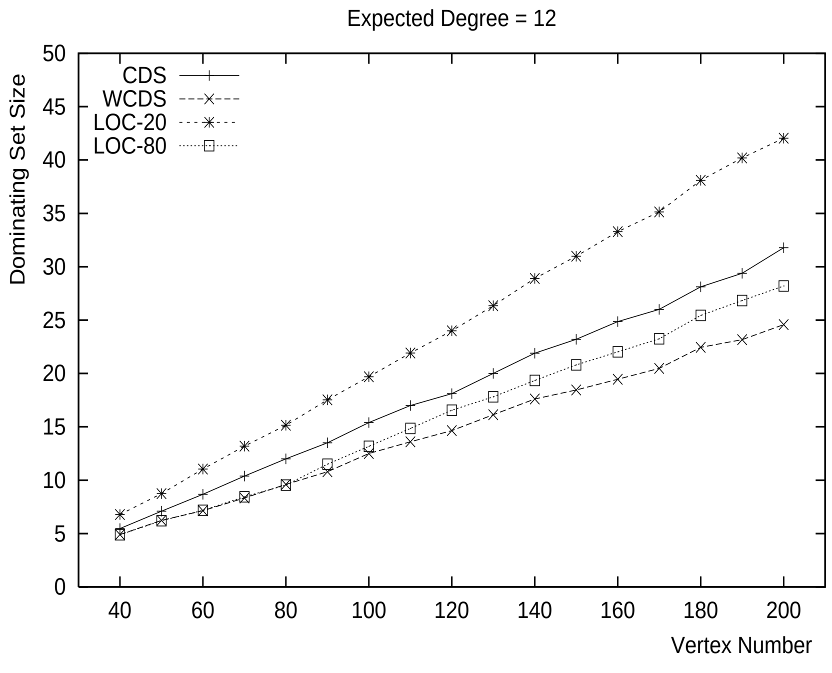

Fig. 5. Dominating set size (b).

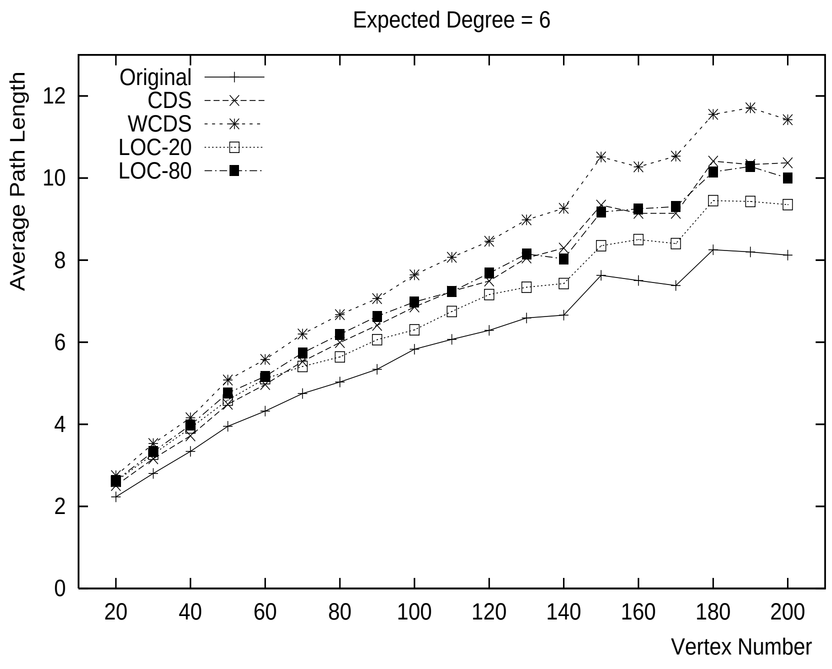

Fig. 6. Average path length (a).

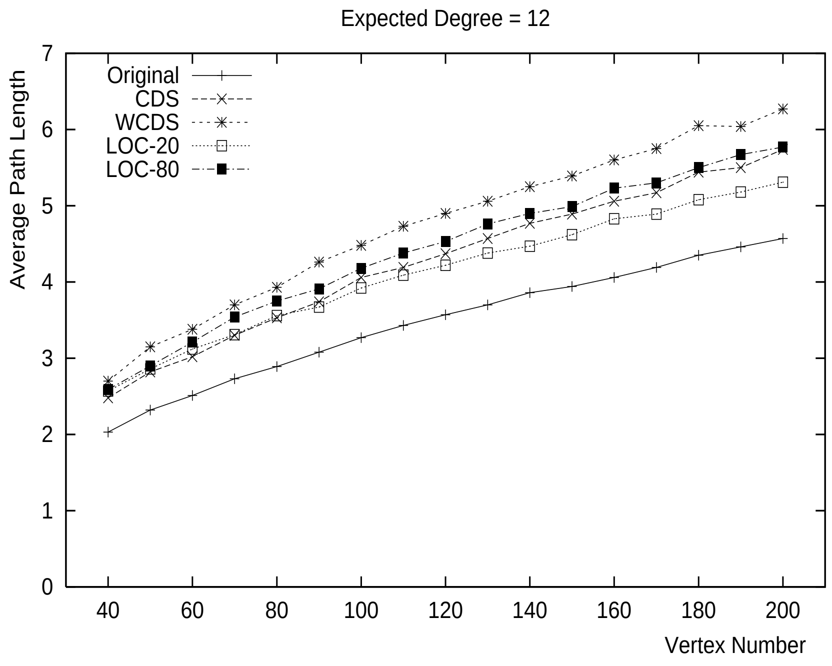

Fig. 7. Average path length (b).

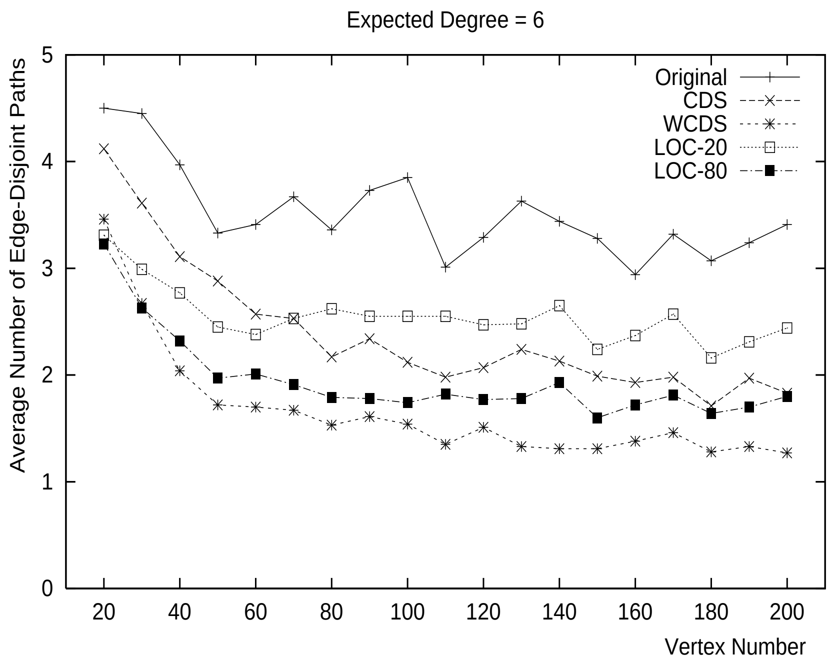

Fig. 8. Average number of edge-disjoint paths (a).

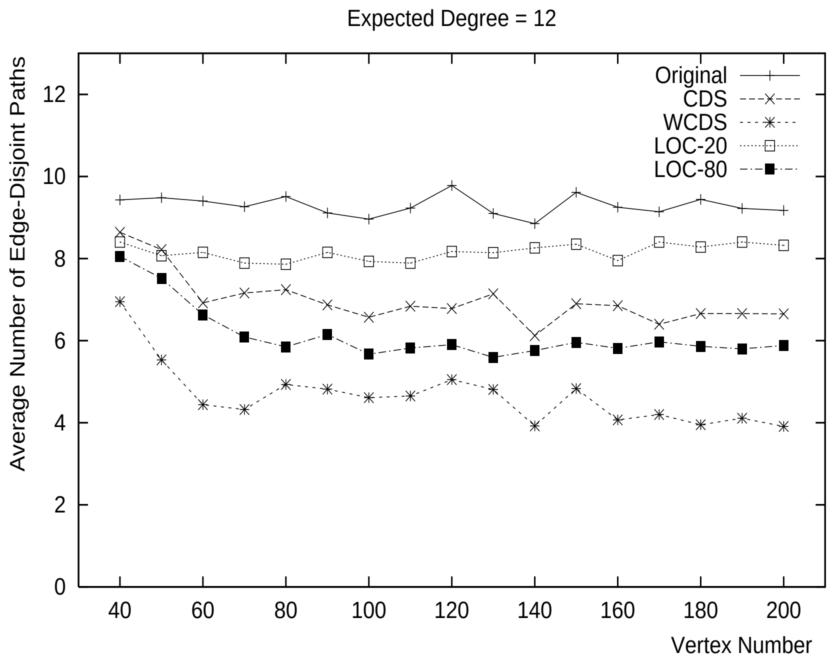

Fig. 9. Average number of edge-disjoint paths (b).

tices from region borders are added in order to obtain a weakly-connected dominating set of the graph. The execution time of this algorithm is $O(x(\log x+|S_{max}|))$ and it generates $O(m+n(\log x+|S_{max}|))$ messages, where $S_{max}$ is the largest weakly-connected dominating set generated by all regions and can be trivially bounded by $O(x)$ from above. This zonal algorithm is regulated by a single parameter $x$, which controls the size of regions. When $x$ is small, the algorithm finishes quickly with a large weakly-connected dominating set. When it is large, it behaves more like the non-localized algorithm and generates smaller weakly-connected dominating sets. Experiments were designed to test the quality of this zonal algorithm. Comparisons are made to the non-localized algorithm for weakly-connected dominating sets and the algorithm for finding small connected dominating sets.

## References

1. K. M. Alzoubi, P.-J. Wan, and O. Frieder. Distributed heuristics for connected dominating set in wireless ad hoc networks. *KICS Journal of Communications and Networks*, 4(1), March 2002.
2. K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 157–164, June 2002.
3. K. M. Alzoubi, P.-J. Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. In *IEEE HICSS35*, January 2002.
4. S. Basagni. Distributed clustering for ad hoc networks. In *Proc. ISPAN'99 Int. Symp. on Parallel Architectures, Algorithms, and Networks*, pages 310–315, 1999.
5. A. A. Bertonssi. Dominating sets for split and bipartite graphs. *Information processing letters*, 19:37–40, 1984.
6. J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. 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. G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, SITE, June 1999.
8. Y. P. Chen and A. L. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 165–172, June 2002.
9. B. Das and V. 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. M. Gerla, T. J. Kwon, and G. Pei. On demand routing in large ad hoc wireless

networks with passive clustering. In *Proceedings of IEEE WCNC*, September 2000.

14. M. Gerla and J. T.-C. Tsai. Multicluster, mobile, multimedia radio network. *Wireless Networks*, 1:255–265, 1995.
15. M. C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, San Diego, CA, 1980.
16. S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.
17. T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in graphs.* Marcel Dekker, Inc., 1998.
18. E. M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, 4 1999.
19. P.-J. Wan, K. M. Alzoubi, and O. Frieder. Distributed construction of connected dominating set in wireless ad hoc networks. In *IEEE INFOCOM*, June 2002.
20. J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIAL M'99*, Seattle, 1999.