Available online at www.sciencedirect.com

SCIENCE DIRECT®

Ad Hoc Networks 3 (2005) 629–642

www.elsevier.com/locate/adhoc

Maintaining weakly-connected dominating sets for clustering ad hoc networks

Yuanzhu Peter Chen [*], Arthur L. Liestman

School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada V5A 1S6

Available online 15 September 2004

Abstract

An ad hoc network is a multihop wireless communication network supporting mobile users. Network performance degradation is a major problem as the network becomes larger. Clustering is an approach to simplify the network structure and thus alleviate the scalability problem. One method that has been proposed to form clusters is to use weakly-connected dominating sets [Y.P. Chen, 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), 2002, pp. 165–172; Y.P. Chen, A.L. Liestman, A zonal algorithm for clustering ad hoc networks, International Journal of Foundations of Computer Science 14(2) (2003) 305–322]. Here, we present a zonal distributed algorithm to maintain weakly-connected dominating sets as the network structure changes. When the zones are small, the algorithm is essentially localized; when the zones are large, it behaves more globally. The size of the weakly-connected dominating set obtained also varies depending on the choice of zone size, with larger zones generally resulting in smaller weakly-connected dominating sets. Experiments provide evidence that this maintenance algorithm keeps the size of the weakly-connected dominating set approximately the same as its initial size and does not compromise the network connectivity.
© 2004 Published by Elsevier B.V.

Keywords: Ad hoc networks; Clustering; Weakly-connected dominating sets; Maintenance

1. Introduction

An *ad hoc network* is a type of wireless communication network that does not rely on any existing infrastructure, is multihop, and supports mobile vertices. As a local region of an ad hoc network may have relatively stable internal structure, one can abstract the network to obtain a simpler topology. In this abstraction, local portions of the network are represented by super-vertices in the abstracted topology and connections between these local portions are super-edges. The process

[*] Corresponding author.
E-mail address: yzxchen@cs.sfu.ca (Y.P. Chen).

of defining such an abstracted structure of a network is referred to as *clustering*.

A natural way to cluster an ad hoc network is to use the notion of graph domination or one of its variants (see Section 1.1 for formal definitions). A dominating set is chosen and the neighborhood of each vertex of the set comprises a cluster. The network may be viewed as a simpler network of these clusters and routing can be done more efficiently. In this work, we focus on weakly-connected dominating sets, which models the scatternet configuration of Bluetooth [1]. The zonal method of maintaining a weakly-connected dominating set can be easily adapted to other graph domination variants, such as connected dominating sets. Further, this zonal clustering algorithm can be combined with the Zone Routing Protocol (ZRP) [2] to improve routing efficiency.

In the Bluetooth specification, devices can form two types of master–slave structures: piconet and scatternet. A piconet has a star topology with a master device at the center and a set of slave devices around. Several piconets can be joined in certain ways to form a scatternet. To do that, a slave device can have multiple masters and a master device can be a slave of another master. As we will see, the weakly-connected dominating set of a graph faithfully captures the scatternet topology with the vertices in the dominating set being the master devices.

The maintenance algorithm proposed here is zonal with a tunable *zone size control parameter* x. When x is small, the algorithm is localized and suitable for fast moving devices and rapidly changing network topologies. When x is large, the algorithm behaves more like a centralized greedy algorithm and generally maintains a small weakly-connected dominating set. Thus, large values for x are suitable for networks with almost static topology such as wireless sensor networks. The parameter allows variation over the full spectrum from local to global and, therefore, it enables a trade-off between localized and centralized computations. The value of the zone size control parameter can be fine-tuned given specific network characteristics.

1.1. Preliminaries—graph domination

We represent an ad hoc network by a graph $G = (V, E)$ where the vertices represent the individual subscriber units and an edge connects two vertices if the corresponding subscriber units are within transmission range of each other. The *closed neighborhood* $N_G[v]$ *of a vertex* v in graph G consists of the vertices adjacent to v plus vertex v itself. The *closed neighborhood* $N_G[S]$ *of the set* S is the union $\bigcup_{v\in S}^{k} N_G[v]$. The subscript G can be omitted if the meaning is clear from the context.

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. We say that an edge is *dominated* if either of its endpoints is in S and refer to other edges as *free*. A dominating set is an *independent dominating set*, if no two vertices in the set are adjacent. A *connected dominating set* S of a given graph G is a dominating set whose induced subgraph, denoted $\langle S\rangle$, is connected.

Of primary interest here is the weakly-connected dominating set. For any subset $S \subseteq V$, the subgraph *weakly induced* by S is the graph $(N[S], E \cap (N[S] \times S))$, which we denote by $\langle S\rangle_{\mathrm{w}}$. That is, the weakly induced subgraph $\langle S\rangle_{\mathrm{w}}$ contains the vertices of S, their neighbors, and all edges with at least one endpoint in S. Fig. 1 shows a subset S of vertices in black with the edges of $\langle S\rangle_{\mathrm{w}}$ indicated by black lines. Note that in this example, $\langle S\rangle_{\mathrm{w}}$ is not a spanning subgraph. A vertex subset S is a *weakly-connected dominating set*, if S is dominating and $\langle S\rangle_{\mathrm{w}}$ is connected. The black vertices in Fig. 2 are a weakly-connected dominating set S for the graph G.

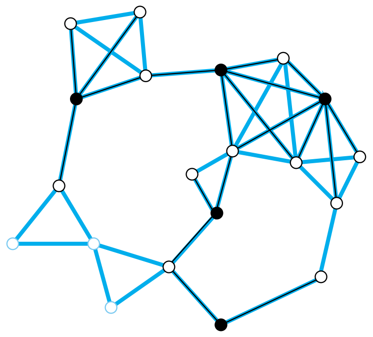
Fig. 1. Weakly induced subgraph.

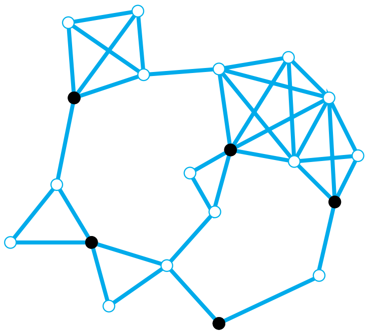
Fig. 2. A weakly-connected dominating set.

In the network context, the vertices of the dominating set are called *clusterheads*. And their neighborhoods are the *clusters*. These clusters can be viewed as super-vertices in an abstracted network, and the connections between them are super-edges. In general, one wishes to find the smallest possible set of clusterheads for the network. Unfortunately, the related decision problems on dominating set variants for general graphs are all NP-complete. See Haynes, Hedetniemi and Slater's comprehensive monograph on graph domination [3] for more on this topic.

1.2. Related work

One class of existing clustering algorithms is based on independent dominating sets of graphs. Gerla and Tsai [4] proposed two weight-based clustering algorithms, using vertex ID and vertex degree for weights, respectively. Chen and Stojmenovic [5] proposed a clustering algorithm using both vertex ID and vertex degree combined. Basagni [6] showed that any meaningful parameter can be used as the weight in order to best exploit the network properties.

Connected dominating sets may be used as clusterheads in ad hoc networks. Guha and Khuller [7] 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 Δ is the maximum vertex degree of the input graph. Das and Bharghavan [8] provided distributed implementations of these algorithms for ad hoc networks. These distributed algorithms have the same approximation ratios as their centralized counterparts, as they utilize central coordinators to oversee the entire execution. Wu and Li presented a localized distributed algorithm [9] for finding small connected dominating sets in which each vertex only needs to know the information of vertices within a two-hop distance.

In an earlier paper [10], we proposed the use of weakly-connected dominating sets for clustering ad hoc networks as an alternative to connected dominating sets. By comparison, weakly-connected dominating sets will, in general, be smaller than connected dominating sets and the resulting induced graph will have fewer edges. This corresponds to fewer clusters and a sparser abstracted network. To decentralize the algorithms in [10], we presented a zonal distributed algorithm to find a small weakly-connected dominating set [11]. The zonal idea can be easily adapted to the connected dominating set-based approaches.

Randomized algorithms were also introduced to cluster ad hoc networks using graph domination. Jia et al. [12] proposed a randomized distributed algorithm for constructing dominating sets in polylogarithmic time with high probability. Dubhashi et al. [13] showed how to use the algorithm of Jia et al. to build connected dominating sets and weakly-connected dominating sets.

1.3. Our work

The zonal algorithm for constructing weakly-connected dominating sets [11] is good for static ad hoc networks, such as sensor networks. In this work, we present an algorithm that updates the weakly-connected dominating set when the topology changes. Here, cluster maintenance is logically divided into two layers—intrazonal (within zones) and inter-zonal (along zone borders).

The size of a zone is controlled by a single global parameter x, so that the size of a zone does not exceed $2x$. Using such a parameter to control the zone granularity, we obtain a trade-off between the extent of network structure simplification and the locality of algorithm execution. When x is large, the algorithm maintains a small weakly-connected dominating set and behaves like a global greedy algorithm [10]; when x is small, the algorithm behaves more like the localized algorithm of Wu and Li [9].

For simplicity, we assume that all of the changes occur sequentially and that we have sufficient time to restructure the network before the next topology change occurs. Further, we assume that the underlying graph always remains connected despite the topology changes. In an actual implementation of the protocol, various timeout mechanisms are used to avoid "infinite" waiting from simultaneous topology changes (see Section 4.1 for an example).

For our algorithms below, we are given the network as a graph $G = (V, E)$ and a subset $S \subseteq V$ that is the weakly-connected dominating set to be maintained. In the figures, solid black vertices represent vertices of the dominating set and white vertices represent dominated vertices. We assume that each vertex has a unique ID, such as the hardware address of the device.

In Section 2, we review the basic ideas of the zonal construction algorithm for weakly-connected dominating sets [11], since these ideas will be used in later sections. In Section 3, we present a non-zonal update algorithm that can be used to maintain the weakly-connected dominating set structure resulting from the static algorithms in [10]. In Section 4, we describe a zonal maintenance algorithm utilizing the algorithm of Section 3. Here, we restrict the execution of the earlier algorithm to zones to accomplish intrazonal maintenance and add inter-zonal maintenance procedures to maintain the weakly-connected dominating set of the entire network. We present our experimental settings and results to test the performance and efficiency of the maintenance algorithm in Section 5. The results show that both the size of the weakly-connected dominating set maintained and the connectivity of the abstracted network stabilize after a short simulation time and these values remain approximately the same as initially. Furthermore, the different zone sizes also show different emphases between algorithm locality and dominating set size.

2. Zonal weakly-connected dominating set construction

In this section, we review the zonal weakly-connected dominating set construction algorithm [11], which is to be extended in this work.

In the zonal clustering algorithm, a zone is a connected subgraph of the input network that has no more than $2x$ vertices, where x is a preset parameter. A zone has a dedicated vertex known by all zone members as *root*. The zonal construction algorithm has two levels:

- Intrazonal—a weakly-connected dominating set is independently constructed for each zone.
- Inter-zonal—the root of a zone determines an additional set to its weakly-connected dominating set to guarantee that the unioned dominating set is a weakly-connected dominating set for the entire network.

The zonal construction algorithm consists of three phases. In the first phase, the graph is partitioned into non-overlapping zones by constructing a spanning forest. At the end of this phase, the root of each tree in the spanning forest is the *zone root* and its ID is known by all vertices of the zone as the *zone ID*.

In the second phase, the greedy weakly-connected dominating set algorithm [10] is independently executed in each zone to find a small weakly-connected dominating set within the zone. The zone root coordinates this process by collecting zonal progress information and then making and disseminating decisions throughout the zone. Specifically, 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 greedy process of choosing a white or gray vertex to dye black. When any vertex is dyed black, the neighboring white vertices in the zone are changed to gray. A *piece* in a zone is either 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 or a white vertex by itself. The *improvement* of a (non-black) vertex is the number of distinct pieces within its closed neighborhood. A vertex with the maximum improvement value is chosen to be colored black in each iteration. At the end of the algorithm, the black vertices constitute a weakly-connected dominating set and the zone contains a single piece.

In the third phase, a weakly-connected dominating set of the entire graph is formed by adding more vertices (if required) to the union of the weakly-connected dominating sets of all the zones. In particular, when two zones are adjacent but not connected by a dominated edge, a dominated vertex is added to the dominating set. To describe this process, we first define some terms. A *problem zone* with regard to zone Z is any zone Z' that is adjacent to zone Z, does not share dominated edges with Z, and has a higher zone ID than Z. A *border vertex of a zone* is a vertex that is adjacent to vertices of other zones. Zone Z is responsible for "fixing" its border to each problem zone. The root of Z maintains a list of its border vertices that are adjacent to problem zone. Thus, a small number of additional vertices can be chosen by the root to fix the borders with the problem zones. Specifically, the root constructs a bipartite graph $B(X, Y, F)$ representing this information. Vertex set X contains a vertex for each problem zone with regard to Z and vertex set Y contains a vertex for every border vertex in Z. There is an edge in F between vertices $y_i \in Y$ and $x_j \in X$ iff border vertex y_i is adjacent to a vertex in problem zone x_i.

Fig. 3 depicts an example of the zonal clustering scheme. It is a snapshot when the clustering is initially completed in the given network. The network is partitioned into three zones, Z_1, Z_2, and Z_3, as indicated by the dashed contours. The solid black vertices belong to each zone's weakly-connected dominating set after the intrazonal phase. The hollow black vertex can be added to the weakly-connected dominating set of zone Z_1 to "fix" the border to Z_3. Note that the border between zones Z_1 and Z_2 and that between zones Z_2 and Z_3 need not be fixed because these borders contain dominated edges.

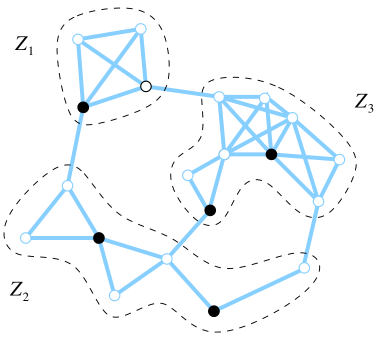

Fig. 3. Zonal clustering scheme.

As shown in [11], this algorithm has a good approximation ratio due to the asymptotically optimal computation of weakly-connected dominating sets in each zone. The choice of the parameter x provides a controllable granularity of local execution. Thus, we choose this construction algorithm as a basis to solve the cluster maintenance problem.

3. Non-zonal maintenance

In this section, we present a non-zonal algorithm to maintain existing weakly-connected dominating sets in arbitrary graphs. The algorithm can easily be adapted to a zonal algorithm for intrazonal maintenance by restricting the execution to a single zone. The algorithm uses message flooding along dominated edges to propagate the control messages. We classify the possible topological changes into four primitives: edge-down (the loss of an edge), vertex-down (the loss of a vertex), edge-up (the addition of an edge), and vertex-up (the addition of a vertex).

3.1. Edge-down

We first deal with the loss of an edge. We have assumed that the root is known to every other vertex. The root is responsible for *fixing* the weakly-connected dominating set if the loss of an edge breaks the piece into *fragments*. Before an edge-down event, the graph contains a single black piece because it has a weakly-connected dominating set. The loss of a free edge (such as edge (w, v) in Fig. 5) does not change the subgraph weakly induced by the dominating set and the maintenance procedure need not be triggered. The loss of a dominated edge, however, must be reported by the vertices incident upon it, because the piece may have been broken into two fragments. In particular, when a vertex loses an incident dominated edge, the root vertex will be notified and will coordinate the procedure to add a dominated vertex to the dominating set if necessary. The response to an edge-down event consists of two logical parts. In the first part, *piece integrity test*, the two vertices that have detected the edge-down event each broadcast an `edge_loss` message to determine if the piece is broken. If so, the root initiates *breach suturing* to add a dominated vertex to the dominating set of the zone (Fig. 4).

To determine if the piece has been broken, both endpoints broadcast an `edge_loss` message along the dominated edges. The `edge_loss` message must be acknowledged by all vertices that receive it. If the piece is not broken, each of the endpoints is able to detect the existence of the other. In this case, the endpoint with lower vertex ID sends a message to the root to indicate that the piece is still intact. If the endpoints do not detect the existence of the other, one of them is in the same fragment as the root and can notify the root that fixing is needed.

After the root decides that the piece needs to be fixed, it broadcasts an `improvement_inquiry` message along the dominated edges to find a vertex that has the greatest *improvement* value. (Here, each vertex knows its improvement value, that is, the number of distinct fragments within its closed neighborhood.) Fragments are identified by the different origins of the `edge_loss` message that a vertex has received for the corresponding event. For an edge-down event, the maximum improvement value will be two. When such a vertex is located, the root instructs it to change to black. This results in a new weakly-connected dominating set for the entire graph.

3.2. Vertex-down

The vertex-down event handling can be regarded as a generalized edge-down event. In the implementation of the distributed algorithm, they are treated identically since a detecting vertex cannot really distinguish the loss of an edge and the loss of a neighboring vertex. The vertex-down event triggers a response consisting of a piece integrity test and breach suturing, if needed.

We define the *clustered degree of a vertex v*, denoted $c_{v,S}$, to be the number of neighbors of v in the subgraph weakly induced by a dominating set S. In essence, this is the number of incident dominated edges of a vertex. If $v \in S$, $c_{v,S}$ is simply v's degree in the original graph. When S is understood from context, we write c_v without confusion. For example, in Fig. 5, $c_v = 4$. As a contrast to the edge-down event, the loss of vertex v can break the piece as many as $c_{v,S}$ fragments. When detecting the loss of a neighbor v, each *detecting* vertex broadcasts an `edge_loss` message along the dominated edges. Again, each vertex receiving this message sends an acknowledgment. The way that a detecting vertex decides if the original piece has been broken or not is slightly different here. Each detecting vertex checks to see if there are $c_{v,S}$ detecting vertices in the fragments from the acknowledgments (rather than two as in the edge-down event). If so, the piece has not been broken and the detecting vertex with the lowest

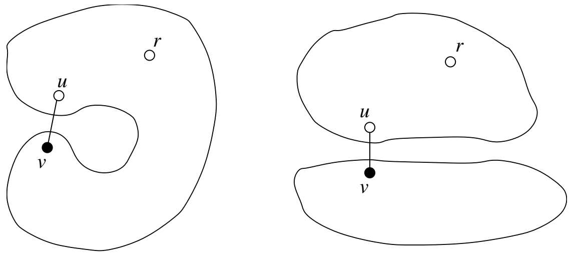

Fig. 4. Edge-down.

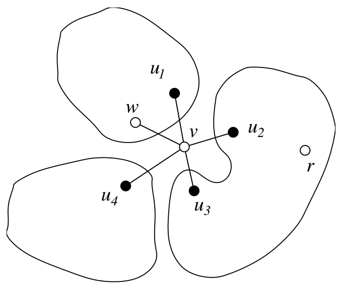

Fig. 5. Vertex-down.

ID sends a message to the root vertex that the piece is still intact. Otherwise, the detecting vertex with the lowest ID that has been acknowledged by the root vertex sends a message to the root along the dominated edges that fixing is needed.

If the piece has been broken into two or more fragments due to the loss of vertex v, the root decides that more dominated vertices will be added to the dominating set. Unlike the case of the edge-down event, the root may need multiple iterations to locate these vertices. In each iteration, the root broadcasts an `improvement_inquiry` message along the dominated edges to find a vertex that has the greatest improvement value. This process stops when the maximum vertex improvement value is one, and the dominating set is weakly-connected.

The above assumes that the lost vertex is not the root. For the special case that the root is down, another vertex takes on the role of the root. To achieve that, every root has a neighbor as a backup. If the loss is discovered by the backup, the backup assumes the role of the root. If the root was not lost but just the edge between the root and its backup is lost, both "roots" will try to add a vertex to the dominating set. For example, the graphs in Fig. 6 illustrate these situations, where vertex r is the root and u is its backup in each graph. If the root and its backup are in the same fragment, as in the left graph of Fig. 6, vertex u will learn that the original root r is still alive during the piece integrity test, and thus, will stop acting as a root. If the two vertices in different fragments, as in the right graph of Fig. 6, either of vertices u and r will broadcast the `improvement_inquiry` message in its fragment. However, when a vertex x in u's fragment that is also adjacent to a vertex y in r's fragment detects that r's ID is used as fragment ID of y, then x learns that the old root r is still alive. Vertex x will then inform u using the returning `improvement_inquiry` message so that u will stop acting as a root. Root r will continue to join these two fragments.

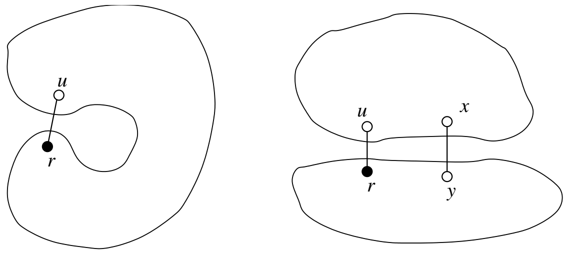

Fig. 6. Possible scenarios for the presumed loss of the root.

3.3. Edge-up

When a new edge is inserted due to, for example, the endpoints moving closer, the weak connectivity of the dominating set will not be affected. However, adding an edge may cause some vertices in the dominating set to be redundant. That is, we can remove these vertices from the dominating set and still have a weakly-connected dominating set. Thus, we focus on localized procedures for eliminating redundancies caused by the edge-up event.

In event of edge-up, we only consider the case of adding a dominated edge. The addition of a free edge does not change the neighborhood of any dominating vertex and thus does not cause any redundancy. For example, in both graphs of Fig. 7, when a dominated edge (indicated as dashed lines) is added, vertex 1 becomes redundant because its closed neighborhood is a subset of that of the neighboring dominating vertex 2. Thus, vertex 1 can be removed from the dominating set. A vertex v in the dominating set may also become redundant if its closed neighborhood is a subset of the union of the closed neighborhoods of v's dominating neighbors. The addition of an edge can also cause this to occur. For example, in each of the two graphs in Fig. 8, vertex 1 becomes redundant when the respective indicated edge is added.

We say that a dominating vertex v is *redundant* if it has a set T of dominating neighbors, such that:

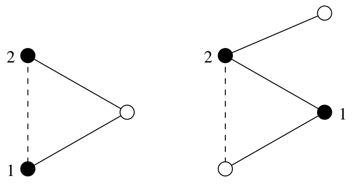

Fig. 7. Edge-up—single coverage.

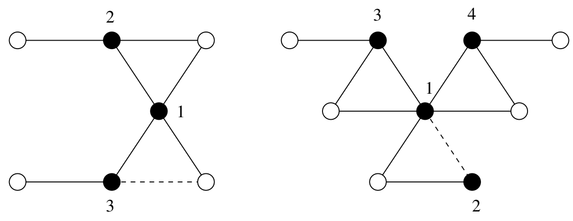

Fig. 8. Edge-up—combined coverage.

1. $N[v] \subseteq N[T]$, and
2. $id(v) < id(u)\ \forall u \in T$.

Note that the weak connectivity of the dominating set is not violated with the removal of redundant vertices. The second requirement is used to avoid simultaneous abdications of multiple dominating vertices when they have identical neighborhoods.

The edge-up message is sent to "nearby" dominating vertices to determine if there is redundancy. Only the endpoints of the new edge and their immediate neighbors need to consider redundancy. The necessity can be observed from the previous examples. For sufficiency, we see that a vertex at distance two or more from the new edge does not have any neighbor whose neighborhood is changed and thus there is no need to calculate redundancies. Therefore, when a vertex has observed an edge-up event, it only needs to notify its immediate neighbors. Assuming that every vertex knows the members of its neighborhood, a single round of information exchange among neighbors will suffice to determine redundancy.

3.4. Vertex-up

The handling of the vertex-up event is very simple. When a new vertex is added, if it has a neighbor in the dominating set, it is dominated; otherwise, it changes its status to be dominating. Note that no dominating vertices will become redundant in the latter case as the newly added dominating vertex is at least two hops away from any other dominating vertex. Therefore, no redundancy calculation is needed in handling the vertex-up event.

4. Zonal maintenance

In order to maintain the weakly-connected dominating set structure in a more localized fashion, we divide the maintenance process into two levels, i.e. intra-zonal and inter-zonal. Intra-zonal maintenance can be accomplished by restricting the execution of the above algorithm within each zone, assuming that each zone's algorithm only takes care of vertices within the zone.

4.1. Intra-zonal maintenance

Occasionally, vertices with the same zone ID may form a disconnected induced subgraph even though the whole network is connected. Thus, we modify the non-zonal maintenance process so that connected components of the induced subgraph that do not contain the zone root can automatically generate a root and form a new zone. To do that, the detecting vertex with the lowest ID of each fragment waits to be annexed after determining that a breach suturing is needed. If nothing happens within a *time-out interval* chosen to be proportional to the zone granularity threshold x, these timed-out vertices of the connected component elect a root simply by broadcasting their ID's within the connected component. On the other hand, when two neighboring zones are both smaller than the threshold size, they merge with each other to maintain relatively stable zone sizes.

4.2. Inter-zonal maintenance

Recall that in the zonal construction algorithm a zone deals only with some neighboring zones, called *problem zones*, and this suffices to generate a weakly-connected dominating set of the entire network. As in the non-zonal scenario, when a border vertex (a vertex adjacent to vertices of other zones) v loses an edge to a foreign vertex (a vertex of other zones) u, it cannot distinguish whether it was an edge-down event or vertex-down event. Thus, we treat them identically.

From v's perspective, if it has a zone ID higher than u does, it does nothing. Otherwise, v will first check to see if it still has another dominated edge incident to u's zone. If so, it does nothing.

Otherwise, it sends a zone_loss message to its zone root r. Upon receiving this message, r will modify its bipartite graph for the border fix procedure and will add a dominated vertex to the dominating set of the zone if needed.

Note that if a border vertex is lost, its zone root r must take special care in addition to the usual intrazonal maintenance. In particular, if the lost border vertex was in the bipartite graph for border fix maintained by r, then it is removed from the bipartite graph. As a consequence, some other vertices may be added to the dominating set by r to fix the broken border. See Fig. 9 for an example of updating the bipartite graph for a zone rooted at r. Before vertex y is lost, y and v were added to fix the borders with zones Z_1 and Z_2. After y is removed, u is added to the dominating set.

In the edge-up or vertex-up event, when a vertex v detects a new edge to a problem zone, v does nothing except to notify its zone root so that it can update the bipartite graph for future border maintenance.

4.3. Complexity analysis

We assume that zone Z has $O(x)$ vertices and maintains a weakly-connected dominating set S_Z. Also, we assume that the maximum vertex degree (number of neighbors a vertex has) is Δ. We use the classic multi-port distributed computing model for the analysis. In particular, in each step, a vertex receives one or zero message from each incident link, carries out internal computation based on its current state and input messages, and sends out one or zero message to each incident link. All complexity analysis is done from the perspective of zone Z.

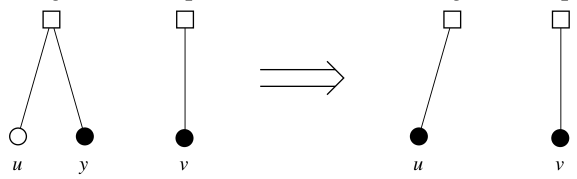

Fig. 9. Updating the bipartite graph.

We first analyze the time and communication complexities of handling the edge-down event. In either part of the process, each broadcast message takes $O(|S_Z|)$ time and $O(|S_Z|\Delta)$ messages to finish, since it always propagates through dominated edges. As there are only constant number of such broadcasts involved, the time complexity of this process is $O(|S_Z|)$ and the communication complexity is $O(|S_Z|\Delta)$. To handle the vertex-down event, both parts can be more complicated than the edge-down event. In the piece integrity test, there can be as many as Δ detecting vertices, while breach suturing may take $\Delta - 1$ iterations in the worst case. Therefore, the process can take up to $O(|S_Z|\Delta)$ time and use $O(|S_Z|\Delta^2)$ messages to complete.

The most costly operation in the edge-up event handling is the redundancy calculation that is done when a dominated edge is inserted. In this case, at most 2Δ vertices need to calculate redundancies, each of which exchanges the neighbor list with all its neighbors using at most 2Δ messages in constant time. Thus, the total communication cost for each (dominated) edge-up event is bounded above by $4\Delta^2$ and it can be done in $O(1)$ time. The vertex-up event involves only incurs $O(\Delta)$ messages and constant time.

The most costly event at the inter-zonal level occurs if a border vertex in a problem zone of zone Z is down. In that case, at most $\Delta - 1$ vertices in Z may send the zone_loss message to the root of Z. Thus, the communication cost is $O(x\Delta)$ and time cost is $O(x)$.

Note that $|S_Z| = O(x)$, so these costs indeed increase as x does. However, when x is small, the algorithm is essentially localized. If, on the other hand, the network topology is expected to be less dynamic, larger x can be used to obtain smaller weakly-connected dominating sets.

5. Simulation

The goals of simulating the proposed maintenance algorithm are to verify its stability and to determine how the zone size control parameter x affects the size of the abstracted network and the locality of the algorithm execution. Super-vertices of the abstracted graph represent the neighborhoods of the vertices in the weakly-connected dominating set. Dominated edges of the graph correspond to virtual edges between the super-vertices. The abstracted graph may have lower connectivity than the original network. Thus, we design simulations to show that, under some typical mobility assumptions, the algorithm keeps the size of the weakly-connected dominating set approximately the same as the beginning, does not reduce the network connectivity, and does not cause frequent cluster changes. (Note that the tests and results for static weakly-connected dominating sets and other domination variants can be found in earlier papers [10,11].)

5.1. Settings

Our simulation is done in an $800\,\text{m} \times 600\,\text{m}$ rectangular area. The simulated network initially has 200 vertices with average transmission ranges of 100 m and 200 m to represent two network density levels. There is an edge between two vertices if and only if their distance is no more than the smaller of the transmission ranges. We set the zone size threshold x (see Section 2) to be 10, 20 and 40 for different zone localities.

Vertices move according to certain mobility models. (Readers are referred to Camp et al. [14] for a survey on mobility models used for ad hoc networking research.) In an entity mobility model, mobile vertices move independently from each other. One entity mobility model is the Random Waypoint model. In this model, each vertex chooses a random destination, moves there at a randomly chosen speed, and then pauses for a random length of time before choosing the next destination. By contrast, in a group mobility model, vertices are divided into groups, the vertices within a group remain close to each other, but the groups move independently. One group mobility model is the Reference Point Group model. Each group has a moving logical center, called the *reference point*. Each vertex within the group may wander within a certain range of its reference point.

We use the Random Waypoint and the Reference Point Group mobility models, as representative entity and group mobility models, respectively. We add random vertex-down and vertex-up events to these mobility models in order to generate the four event primitives. To do that, we independently delete and/or insert a vertex at an expected period of 6 s. Thus, the expected vertex life span is 20 min and the network size remains at approximately 200 despite the dynamic nature of the network.

In our experiments, we set the pause time of the Random Waypoint model to zero. When the Reference Point Group model is used, mobile vertices are initially divided into groups of five. Group reference points move within the rectangular simulation area as entities in the Random Waypoint model. The vertices of a group are confined within a circular region of radius 100 m centered at the group reference point. Their movements also follow the Random Waypoint model relative to the group reference point.

In our first experiments, we fixed the average vertex speed at 5 m/s. With this setting, we determine how several measures (weakly-connected dominating set size, pairwise average vertex distance, and pairwise number of edge-disjoint paths) vary over time. We observed that these values stabilize after about 50 s of simulated time. In our second experiments, we considered the same measures under speed settings 1 m/s, 5 m/s, 10 m/s, and 20 m/s, and let each simulation run until the measured value stabilized. In addition, when the variable speed settings were considered, we noted the rates at which vertices change their roles as clusterheads and their cluster membership.

5.2. Results

We first consider the stability of the maintenance algorithm, that is, whether the weakly-connected dominating set remains roughly the same size over time. We record the set size under both mobility models, setting the average vertex speed and reference point speed to 5 m/s, for the first 200 s of the simulation. The test is done with graphs of initial size of 200 vertices and average device transmission ranges of 100 and 200, respectively (Figs. 10 and 11). Three different zone size control parameters, $x = 10$, 20 and 40, are considered. As shown in the figures, during

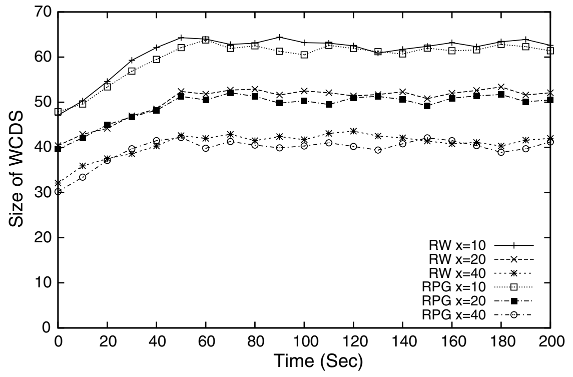

Fig. 10. Maintenance stability (range = 100).

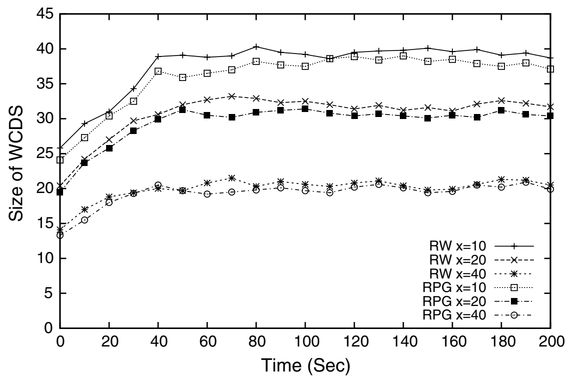

Fig. 11. Maintenance stability (range = 200).

approximately the first 50 s of the simulation, the size of the weakly-connected dominating set slightly increases. Then the size of the dominating set stabilizes for the rest of the time. Note that the smaller the zone size control parameter x is, the larger the dominating set. This shows a trade-off between locality of the algorithm execution and succinctness of the abstracted network. The idea of controllable zone sizes is a generalization of pure centralized greedy algorithms and pure localized algorithms. When larger zones and less locality are affordable, smaller abstracted networks can be generated. It is known that the network density increases to a higher level when vertices are moving than in the initial distribution [14]. We have observed this in our data. But the maintenance algorithm pays the price of larger dominating sets to avoid constructing a weakly-connected dominating set from scratch. The benefit of having a larger dominating set, though, is to have better connectivity properties as we will see. One other observation is that the maintenance algorithm behaves similarly under both mobility models. Therefore, from this point on, we only show the data generated by the Random Waypoint mobility model.

We also consider how the network connectivity of the abstracted graph (using only dominated edges) changes over time. We test this by measuring the average distance and average number of edge-disjoint paths between all pairs of vertices in the network under both network density levels (transmission ranges of 100 and 200). In order to have better network connectivity, it is desirable to have small average distances and large numbers of edge-disjoint paths. We compare these values to the same values for the original network. Figs. 12 and 13 show that, for both network density levels and all zone size control parameters, the average pairwise vertex distance decreases first and then stabilizes as the network density does. The spikes on the curves in Figs. 12 and 13 are due to the delay of the algorithm in rejoining the weakly-connected components after the abstracted graph has become disconnected. Because our algorithm reacts to topology changes in the network, these rare disconnections are inevitable. Figs. 14 and 15 show that the average number of edge-disjoint paths increases slightly at the beginning and then

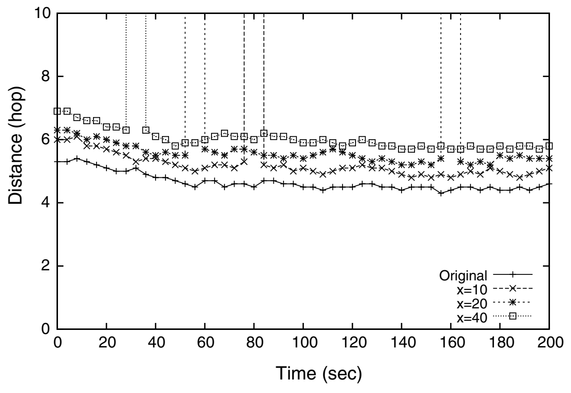

Fig. 12. Pairwise vertex distance vs. time (range = 100).

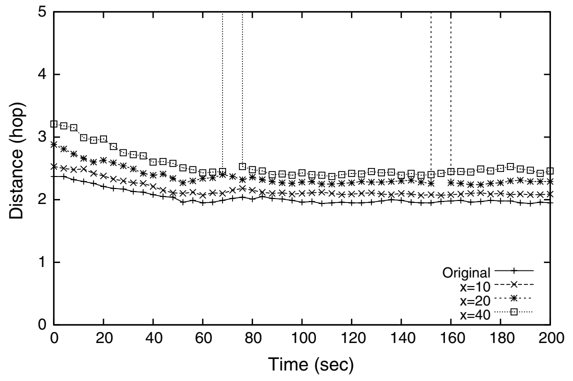

Fig. 13. Pairwise vertex distance vs. time (range = 200).

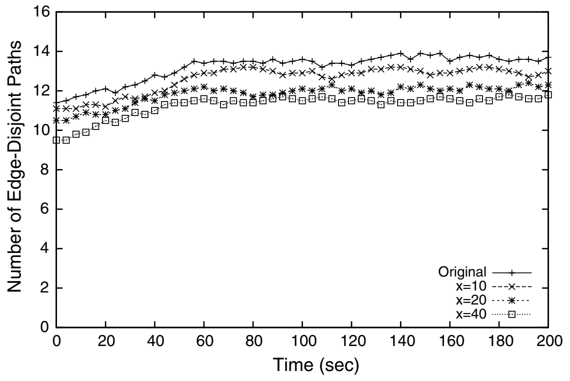

Fig. 15. Number of edge-disjoint paths vs. time (range = 200).

remains roughly stable as the network density stabilizes.

To determine how the average vertex speed affects the weakly-connected dominating set size and the connectivity of the abstracted network, we record the same measures after the network stabilizes, for both network density levels with average vertex moving speeds of 1, 5, 10, and 20 m/s. From Figs. 16–18, we can see that the weakly-connected dominating set size increases slightly and the network connectivity decreases slightly as the average vertex speed increases as was expected.

As the network topology changes, some vertices change status from dominating to dominated, or vice versa (clusterhead changes). Other vertices may move from one cluster to another (cluster membership changes). These changes affect routing and addressing in a hierarchical routing

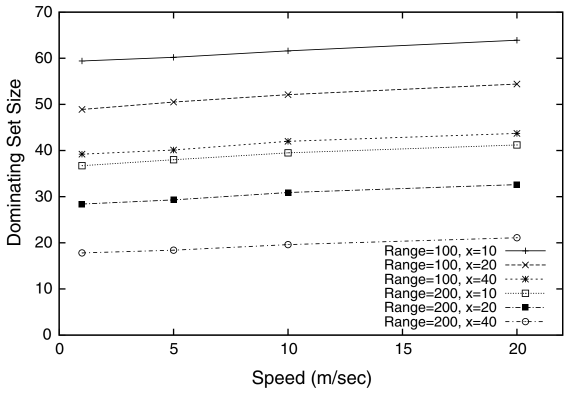

Fig. 16. Dominating set size vs. speed.

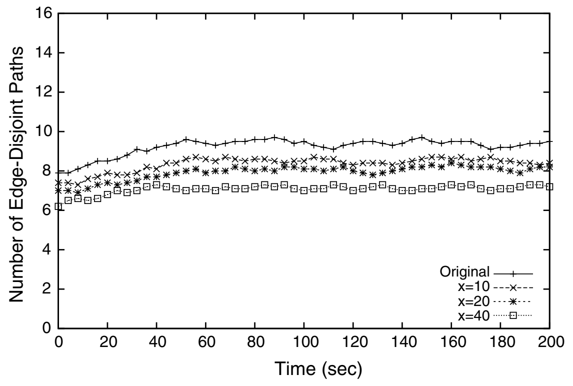

Fig. 14. Number of edge-disjoint paths vs. time (range = 100).

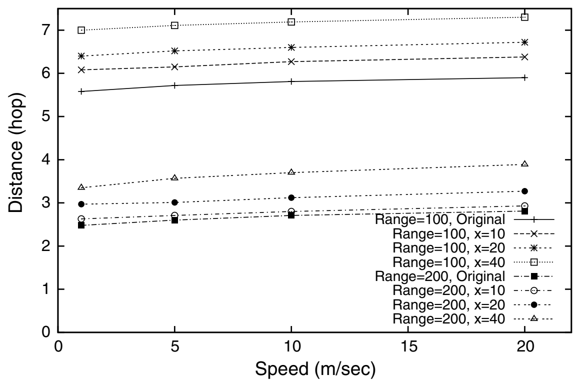

Fig. 17. Pairwise vertex distance vs. speed.

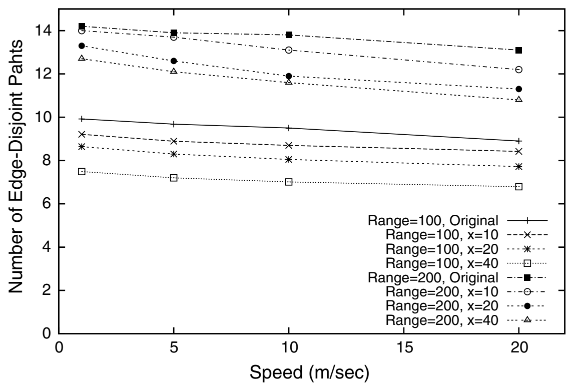

Fig. 18. Number of edge-disjoint paths vs. speed.

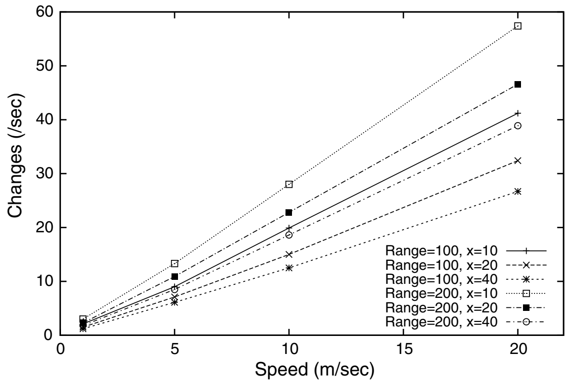

Fig. 20. Rate of cluster membership change.

scheme. We consider the rates of these changes in the weakly-connected dominating set as the vertex speed varies. We record all the above changes occurring in each second with different average vertex speeds of 1, 5, 10, and 20 m/s. The results are shown in Figs. 19 and 20. We find that the change rates increase when vertices move faster as might be expected and that denser networks generate more such changes than sparser networks.

In summary, clustering ad hoc networks by using weakly-connected dominating sets helps to simplify the network structure and, thus, alleviates the scalability problem. The algorithm presented in this work maintains a weakly-connected dominating set. We have shown that the weakly-connected dominating set size remains approximately the same as its initial size. Further, the abstracted network connectivity does not vary significantly. Most importantly, the zone size control parameter provides a simple mechanism to control the locality of the algorithm execution and the size of the abstracted network. By varying this parameter we obtain a trade-off between a purely localized approach and a purely centralized approach.

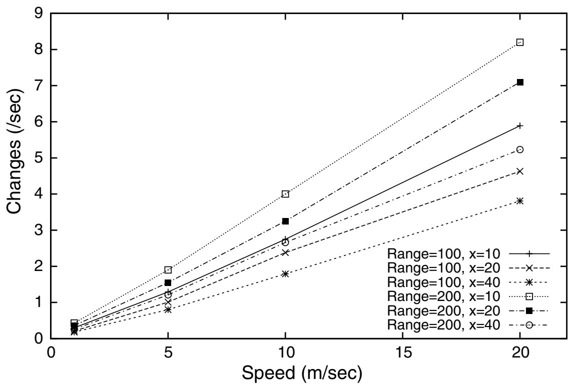

Fig. 19. Rate of clusterhead change.

References

[1] J. Haartsen, Bluetooth—the universal radio interface for ad hoc, wireless connectivity, Ericsson Reviews (3) (1998) 110–117.
[2] Z.J. Haas, M.R. Pearlman, ZRP: a hybrid framework for routing in ad hoc networks, in: C. Perkins (Ed.), Ad Hoc Networking, Addison-Wesley, Reading, MA, 2001.
[3] T.W. Haynes, S.T. Hedetniemi, P.J. Slater, Fundamentals of Domination in Graphs, Marcel Dekker, New York, 1998.
[4] M. Gerla, J.T.-C. Tsai, Multicluster, mobile, multimedia radio network, Wireless Networks 1 (3) (1995) 255–265.
[5] G. Chen, I. Stojmenovic, Clustering and routing in mobile wireless networks, Tech. Rep. TR-99-05, SITE, June 1999.
[6] S. Basagni, Distributed clustering for ad hoc networks, in: Proc. ISPAN'99 Int. Symp. on Parallel Architectures, Algorithms, and Networks, 1999, pp. 310–315.
[7] S. Guha, S. Khuller, Approximation algorithms for connected dominating sets, Algorithmica 20 (4) (1998) 374–387.
[8] B. Das, V. Bharghavan, Routing in ad-hoc networks using minimum connected dominating sets, in: IEEE

International Conference on Communications (ICC'97), vol. 1, 1997, pp. 376–380.

[9] J. Wu, H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in: DIAL-M'99, Seattle, 1999, pp. 7–14.

[10] Y.P. Chen, 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), 2002, pp. 165–172.

[11] Y.P. Chen, A.L. Liestman, A zonal algorithm for clustering ad hoc networks, International Journal of Foundations of Computer Science 14 (2) (2003) 305–322.

[12] L. Jia, R. Rajaraman, T. Suel, An efficient distributed algorithm for constructing small dominating sets, Distributed Computing 15 (4) (2002) 193–205.

[13] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, A. Srinivasan, Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons, in: Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003, pp. 717–724.

[14] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Communications and Mobile Computing 2 (5) (2002) 483–502.

Yuanzhu Peter Chen received his Ph.D. from Simon Fraser University in 2004 and B.Sc. from Peking University in 1999. He is currently a post-doctoral researcher at Simon Fraser University.

Arthur L. Liestman received his undergraduate degree from the University of Kansas and his M.S. and Ph.D. degrees from the University of Illinois. He is a Professor in the School of Computing Science at Simon Fraser University in Burnaby, British Columbia. His research interests involve the interplay between network communication and network structure.