A DISTRIBUTED TOPOLOGY CONTROL ALGORITHM FOR MANETS

S. Venkatesan
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083-0688
venky@utdallas.edu

C. David Young
Rockwell Collins, Inc
3200 E Renner Road
Richardson, TX 75083
cdyoung@rockwellcollins.com

ABSTRACT

Topology control is an important problem in mobile ad hoc networks (MANETs), where the underlying communication topology may constantly change. One variation of this well-studied problem is that of finding a minimum connected subgraph (of the network topology) that serves as a backbone for routing messages. This problem, known as the minimum connected dominating set (MCDS) in the literature, is NP-hard. Therefore, heuristics are used to find a solution close to the optimal. In this paper, we present a distributed algorithm to find a minimal connected dominating set (mCDS) of a network. Our algorithm is simple to implement, executes fast, and has low time complexity. Our algorithm has been implemented and tested extensively.

INTRODUCTION

The problem of topology control has been studied by many researchers in the past and several sequential algorithms have been proposed. Topology control is an important problem in mobile ad hoc networks (MANETs) and has many important uses. In this paper, we consider the problem of finding a minimal connected dominating set and present a distributed algorithm for solving this problem. The problem of distributed construction of minimal connected dominating set has been considered in the past and previous solutions have been implemented as part of Rockwell Collins's ongoing contracts. In this paper, we develop a new algorithm for this problem.

Let G be the topology of a communication network, such as a mobile ad hoc network. We are interested in finding a subgraph (or skeletal graph) SG of G with the following properties:

1. Node set of SG is a subset of the node set of G. The link set of SG is a subset of the link set of G and each link of SG connects two nodes of SG.
2. SG provides shortest paths between every pair of nodes of G. Thus, from any source S in G to any destination D in G, there exits a shortest path P from S to D using only nodes of SG such that P is a shortest path between S and D in G, the original topology. Thus, nodes of SG can be used for finding routes (instead of using G). If S and D are neighbors, then the (only) shortest path between D and D is the direct 1-hop path and nodes of SG are not needed.
3. SG is connected if G is connected.
4. The subgraph SG is minimal. Thus, we require that no proper subset of SG satisfies the above properties.

Note that the nodes of SG form a non-empty minimal subset of the nodes of G and the links of SG consist of all links of G whose both end points are in SG. We will show that SG is a minimal connected dominating set (mCDS). The nodes of SG are called ***skeletal*** nodes. In addition to providing

shortest paths, skeletal nodes improve the efficient use of channel resources for bursty traffic, especially in dense networks. For example, the skeletal nodes can be assigned more resources (such as time slots in TDMA based channel access scheme) than they need and these additional resources (time slots) can be loaned to adjacent nodes that need them.

For example, consider the network show in Figure 1 consisting of seven nodes. There are two possible mCDSs for this topology. The set {B,D,F,G} forms one mCDS (shown as mCDS 1 in Figure 1(a)). Note that {B,D,F,G} is minimal and provides the shortest paths property. (For example, the shortest paths of length 2 or 3 between any pair of nodes go via nodes of the mCDS.) Also, note that {B,D,F,G} is connected. Similarly, the set {A,B,C,D,E,F} forms another mCDS (shown in as mCDS 2 in Figure 1(b)). Note that mCDS 1 has a smaller number of nodes than mCDS 2. In general, a topology may have many mCDSs and any of the possible mCDSs is acceptable. (Note again that finding the mCDS of minimum cardinality is NP-hard.)

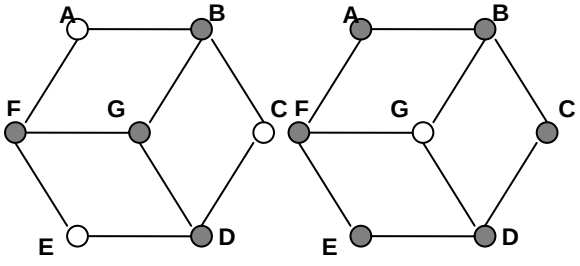

Fig 1(a); mCDS 1 Fig 1(b); mCDS 2

We present a distributed algorithm to find at least one of the acceptable SGs. We assume that the underlying topology does not change when the algorithm executes. (If it does, all the nodes can initiate the algorithm again, from the beginning, if needed.) Nodes exchange messages with only their neighbors and no node knows the complete topology.

At any time during the execution of our algorithm, each node can be in any one of the six possible states: (a) Stable skeletal node, (b) stable non-skeletal node, (c) bi-stable skeletal node, (d) bi-stable non-skeletal node, (e) bi-stable undecided, or (f) undecided node. Initially each node is in the undecided state. Among the nodes of the topology, there are some nodes that will be part of every mCDS (for the given topology). These nodes are stable skeletal nodes. There are some nodes will not part of any mCDS (for the given topology). These are stable non-skeletal nodes. The rest of the nodes are bi-stable nodes. Each bi-stable node is a node that may either be a skeletal node in one or more mCDS(s) or a non-skeletal node in other mCDSs. The nodes of SG (or mCDS) consist of stable skeletal nodes and bi-stable skeletal nodes. This can be proved and the proof will appear in the full paper. The algorithm consists of the following three stages.

In the first stage, each node decides if it can change its state to stable skeletal node or stable non-skeletal node (after exchanging its neighborhood information over 2-hops. (Details follow.) Some of the nodes may find that they can change their states, and the other nodes may continue to be in the undecided state. Nodes in the undecided state after the first stage proceed to the next stage and these nodes are in the bi-stable undecided state.

In the second stage, nodes decide if each undecided node (at the end of the first stage) can change to either bi-stable skeletal or bi-stable non-skeletal state using simple rules. The outcome of this stage depends only on the topology and each node is able to make a decision by itself knowing only the 2-hop neighborhood information. The remaining nodes, which remain in the undecided state at the end of second stage, proceed to the third stage and these nodes are said to be in bi-stable undecided state.

In the third stage, all the nodes in the bi-stable undecided state change to either the skeletal or non-skeletal state. The time

complexity of this step is proportional to the diameter of the sub-network that consists only of those nodes that remain in the undecided state (after stage two) that are at most 2-hops from each other. The algorithm converges fairly quickly. For many randomly generated graphs, the convergence is fast.

ALGORITHM DESCRIPTION

Node classification rule:

1. Stable Skeletal Node: A node u is said to be a stable skeletal node if it has two neighbors v and w such that v and w are not neighbors of each other and u is the only common neighbor for both v and w. In other words, node u is the mid point of the only shortest path (that is 2-hops) between v and w. Since SG is required to provide shortest paths between every node pair and since u is the only intermediate node on the shortest path between v and w, u must be included in any SG that is an mCDS. In Figure 1(a) B is a stable skeletal node since the (only) shortest path between A and C goes through B. Similarly, nodes D and F in Figure 1(a) are stable skeletal nodes.
2. Stable Non-Skeletal Node: Node x is said to be a stable non-skeletal node if, for each pair of neighbors y and z of node x, nodes y and z are either (1) neighbors of each other (directly connected) or (2) y and z are connected by a 2-hop path going through a stable skeletal node. In Figure 2, node 0 (which has only one neighbor) is a stable non-skeletal node. Also, node 6 is a stable non-skeletal node since node 6 has three neighbors (nodes 5, 7, and 8) and each pair of these three neighbors is directly connected. Similarly, nodes 7 and 8 are stable non-skeletal nodes.
3. Bi-stable Skeletal and Bi-stable non-Skeletal nodes. All other nodes that are not classified as either a stable skeletal node or a stable non-skeletal node are bi-stable undecided nodes. The bi-stable nodes can become skeletal nodes or non-skeletal nodes. The bi-stable nodes are in "undecided" state. We will change the state of each bi-stable node into either a skeletal node or a non-skeletal node after further information exchange.

Stable Skeletal Node Rule: Nodes exchange their 1-hop neighborhood information (list of neighbors) over one hop. Let Nbrs(u) be the set of neighbors of node u. Node u sends Nbrs(u) to all of its neighbors. At the end of this step, each node u knows, for each neighbor v of u, all the neighbors of v. Now, u is a stable skeletal node if u has two neighbors v and w such that Nbrs(v) and Nbrs(w) have only one common node, namely, node u. If u becomes a stable skeletal node, it notifies its neighbors that are 1-hop and 2-hops away.

Stable non-Skeletal Node Rule: Classifying a node as a stable non-skeletal node is easy after all nodes in the 2-hop neighborhood have determined if they are stable skeletal nodes and notified their 1-hop and 2-hop neighbors. Node w is a stable non-skeletal node if, for each pair of neighbors, say x and y (both in Nbrs(w)), x and y are neighbors of each other or x and y have a common neighbor (of both x and y), say z, which is a stable skeletal node. Intuitively, as far as providing a shortest path between its neighbors is concerned, a stable non-skeletal node w is of no use. It is possible to construct a mCDS without w. In the full paper, we prove that no SG will contain node w.

Once a node applies the two rules above (to find out if it is a stable skeletal node or a stable non-skeletal node) and determines that it does not belong to either of the two types, it is in the bi-stable undecided state. In such a mode, it does not know if it is going to be part of mCDS or not. Consider the sample graph shown in the following figure:

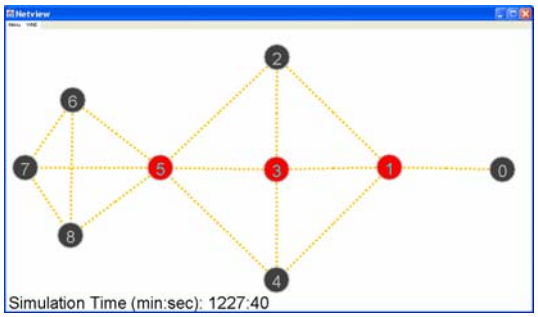

Figure 2

In Figure 2, (i) nodes 0, 6, 7, and 8 are stable non-skeletal nodes, (ii) nodes 1 and 5 are stable skeletal nodes, and (iii) nodes 2, 3, and 4 are in bi-stable undecided state.

Using the following rules, we can classify some (but not necessarily all) of the bi-stable undecided nodes as either bi-stable skeletal nodes or bi-stable non-skeletal nodes:

Bi-Stable Non-Skeletal Node Rule: A bi-stable undecided node v changes its state to bi-stable non-skeletal if (i) Nbrs(v) is a proper subset of Nbrs(w) where w is any node in the 1-hop or 2-hop neighbor set of v or (ii) there exists a neighbor x (either 1-hop or 2-hop neighbor of v) such that Nbrs(v) = Nbrs(x) and v > x . In Figure 2, by applying this rule, nodes 2 and 4 change from bi-stable undecided state to bi-stable non-skeletal node state. (Note that Nbrs(3) ={1,2,4,5} and both Nbrs(2) and Nbrs(4) are subsets of Nbrs(3).)

Bi-stable Skeletal Node Rule: A bi-stable undecided node u is said to be a bi-stable skeletal node if Nbrs(u) is (1) the superset of each of the Nbrs(v) for each node v in the 1-hop and 2-hop neighborhood of u or (2) (i) is not the proper subset of any of the Nbrs(v) and (ii)(a) Nbrs(u) is a superset of Nbrs(v) for some of the (1-hop and 2-hop) neighbors of u or (b) u < v for each neighbor v of u such that Nbrs(u)=Nbrs(v). In figure 2, by applying this rule node 3 becomes a bi-stable skeletal node because Nbrs(3) is a superset of Nbrs(2) and Nbrs(4).

There are two ways to classify each node that is still in the bi-stable undecided state into either a bi-stable skeletal node or a bi-stable non-skeletal node by using (i) a deterministic way or (ii) a randomized way.

Deterministic Tie-breaking Algorithm: Define SPS(u) to be the set of 2-hop shortest paths provided by u that are not provided by any skeletal nodes (either stable or bi-stable). Thus, SPS(u) = $\{x_1y_1, x_2y_2, \ldots, x_ny_n\}$ such that (1) x_i and y_i are neighbors of u, (2) x_i and y_i are not neighbors of each other and (3) x_i and y_i do not have a common neighbor that is a skeletal node (either stable skeletal node or bi-stable skeletal node) for each 1 <= i <= n. Note that SPS(u) shrinks if nodes in the vicinity of u change from the (bi-stable) undecided state to the (bi-stable) skeletal state. Consider all bi-stable undecided nodes. Now, node u constructs SPS(u) and sends |SPS(u)| (the cardinality of SPS(u)) to all 1-hop and 2-hop neighbors everytime |SPS(u)| changes.

Bi-Stable Dynamic Skeletal Node Rule: A bi-stable undecided node u becomes a bi-stable skeletal node (a) if |SPS(u)| > |SPS(v)| for each 1-hop and 2-hop bi-stable (undecided) neighbor v of u or (b) |SPS(u)| >= |SPS(v)| for each 1-hop and 2-hop bi-stable (undecided) neighbor v of u and u has the smallest id among these neighbors whose SPS value is equal to SPS(u). If |SPS(w)|=0 for some bi-stable undecided node w, then w becomes a bi-stable non-skeletal node. Note that the time it takes for all bi-stable undecided nodes to change to bi-stable skeletal or bi-stable non-skeletal nodes is proportional to the diameter of the network consisting of all bi-stable nodes that are no more than 2 hops from each other.

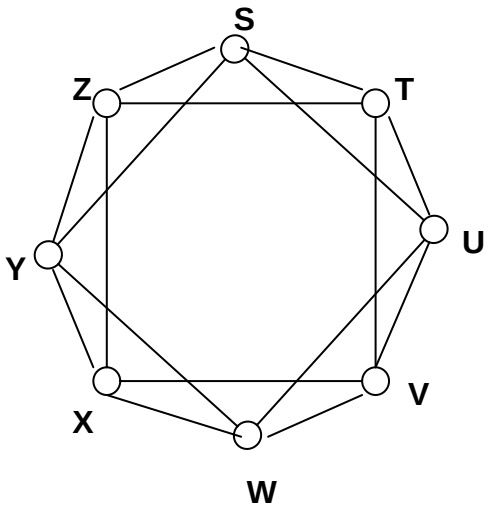

Figure 3

Consider the topology in Figure 3. All the eight nodes are in the bi-stable undecided state after applying all the rules of the first two stages. Consider node S. SPS(S) = {TY, UY,UZ} and |SPS(S)| = 3, In fact |SPS| is 3 for all the eight nodes. Now, all the nodes exchange their SPS values over their 1 and 2-hop neighborhoods. Clearly, on applying the deterministic tie-breaking algorithm, node S becomes a bi-stable skeletal node and sends this change of state to all nodes. On receiving this, the three nodes T, W, and Z decreases their |SPS| value by 1 (and the |SPS| values of U, V, X and Y remain unchanged at 3). Among the bi-stable undecided nodes, node U has the smallest id among all nodes with the highest |SPS| value. Thus, U becomes a bi-stable skeletal node. U informs all the 2-hop neighbors about its state change. In response, nodes T, V, and Y decrease their |SPS| value by one each. Now, node X is the only undecided node whose |SPS| value is 3 and it becomes a bi-stable skeletal node, causing T, W, and Y to decrease their |SPS| value by one each. Likewise, node V becomes a bi-stable skeletal node next. At this point nodes T and W find their |SPS(W)| at zero and become bi-stable non-skeletal nodes. Then node T becomes a bi-stable skeletal node and the remaining node Z becomes a bi-stable non-skeletal node (after finding that |SPS(Z)| =0).

Randomized Tie-breaking Algorithm: In this discussion, we use the term artery nodes as a synonym for skeletal nodes. Define the **Artery Node Neighbors (ANN)** of a node as the set of neighbors of that node (including itself) that have announced themselves as artery (or skeletal) nodes. Each node announces its ANN periodically. A node will then examine the artery node status of its neighbors and the ANNs they are announcing, after excluding this node, to decide if it should become an artery (skeletal) node.

The **Artery Node Rule** is as follows:

If (no artery node neighbors)
OR (two ANNs are disjoint)
be an artery node.

Else
do not be an artery node.

The random tie-breaking consists of applying the Artery Node Rule after a random delay. We have determined that the Artery Node Rule by itself can be used in place of the skeletal node rules but that the convergence time is typically substantially longer due to the need for random delays.

RELATED WORK

The problem of finding connected dominating set and variations of this problem have been studied extensively. Guha and Khuller [GK98] present centralized algorithms based on the idea of growing spanning tree and a distributed implementation of this idea is presented by Das et al [DB97, SDB98,SDB98a] . A central coordinator is used to ensure that the centralized and distributed implementations select the same number of nodes for the backbone. Cardei et al [CCCD02] and Wan et al [WAF02] present distributed

algorithms that uses a maximal independent set (MIS) as a starting point and connect the nodes of the MIS using additional nodes and links. Another approach is to start with a set of cluster heads or weakly connected set and connect the cluster heads/weakly connected sets by adding additional nodes [AWF02,DMPRS03,CL03]. In contrast, we start with a null graph as the mCDS and add nodes distributively. Our algorithm produces a minimal connected set and each node is either part of mCDS or a neighbor of a node of mCDS. The ability for some of the nodes to locally identify if they are stable skeletal, stable non-skeletal, bi-stable skeletal, or bi-stable non-skeletal nodes by examining local neighborhood reduces the termination time. The worst case for our algorithm consists of all nodes being arranged symmetrically and the degree of shortest paths connectivity provided by each node is the same. In this case, we break ties using node id (preferring small node ids). These are likely very contrived examples and are unlikely to occur very often in real life.

CONCLUSIONS

In this paper, we have presented a distributed algorithm for finding a minimal connected dominating set. Our algorithm has been implemented and numerous simulation experiments have been conducted. In the simulated topologies, the algorithm converges very quickly. In the future, it will be worth investigating how far away the result of our algorithm is from the optimum (in terms of the number of nodes in mCDS in relation to the smallest number of nodes in the minimum connected dominating set).

ACKNOWLEDGEMENTS

S. Venkatesan would like to thank Rockwell Collins, Inc., where he was working in summer of 2004. We would like to thank Alan Amis and Jim Stevens for several stimulating discussions.

REFERENCES

[AWF02] K. Alzoubi, P.-J. Wan, and Frieder, “Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks,” Proc. of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc ’02), June 2002, pp. 157-164.

[CCCD02] M. Cardei, X. Cheng, X. Cheng, and D. Du, “Connected Domination in Multihop Ad Hoc Wireless Networks,” 6th International Conference on Computer Science and Infomatics (CS&I 2002), March 2002.

[CL03] Y.P. Chen and A.L. Liestman, “A Zonal Algorithm for Clustering Ad Hoc Networks,” International Journal on Foundations of Computer Science, 14(2), Apr. 2003, pp. 305-322.

[DB97] B. Das and V. Bharghavan, “Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets,” IEEE International Conference on Communications (ICC ’97), June 1997, pp. 376-380.

[DMPRS03] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan and A. Srinivasan, “Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons,” Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003, pp. 717-724.

[GK98] S. Guha and S. Khuller, “Approximation Algorithms for Connected Dominating Sets,” Algorithmica, 20(4), Springer-Verlag, Apr. 1998 pp. 374-387.

[SDB98] R. Sivakumar, B. Das, and V. Bharghavan, “Spine Routing in Ad hoc Networks,” ACM/Baltzer Publications Cluster Computing Journal, Special Issue on Mobile Computing, 1998.

[SDB98a] R. Sivakumar and B. Das and V. Bharghavan, “The Clade Vertebrata: Spines and Routing in Ad Hoc Networks,” Proc. of the IEEE Symposium on Computers and Communications, 1998.

[WAF02] P.-J. Wan, K.M. Alzoubi, and O. Frieder, “Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks,” IEEE INFOCOM, June 2002.