Distributed Center-Location Algorithms

David G. Thaler and Chinya V. Ravishankar, *Member, IEEE*

***Abstract*— Recent multicast routing protocol proposals such as protocol independent multicast (PIM) and core-based trees (CBT) have been based on the notion of group-shared trees. Since construction of a minimal-cost tree spanning all members of a group is difficult, they rely on center-based trees and distribute packets from all sources over a single shortest-path tree rooted at some center. PIM and CBT provisionally use administrative selection or simple heuristics for locating the center of a group but do not preclude the use of other methods that provide an ordered list of centers. Other previously proposed heuristics typically require knowledge of the complete network topology, a requirement which is not always practical for a distributed problem such as Internet routing. In this paper we investigate the problem of finding a good center in distributed fashion, study various heuristics for automating center selection, and examine their applicability to real-world networks. We also propose several new algorithms which we feel to be more practical than existing methods. We present simulation results on hierarchical and nonhierarchical networks showing that of the methods potentially feasible in the Internet multicast backbone, ours offer the best results in terms of cost and delay, and they incur low overhead.**

***Index Terms*— Communication system routing, layout, multicast channels, multicast communication, networks, trees (graphs).**

I. Introduction

MULTICAST technology allows point-to-multipoint communication and enables the use of multimedia applications such as voice and video transmission over the Internet. Multicast methods typically use spanning trees and minimize delay by distributing packets along the shortest path between a receiver and a sender. The collection of shortest paths from a data source to all receivers is known as a *source-specific tree*.

The collection of routers in today's Internet with multicast capability form the multicast backbone (MBone) [1], in which multicast groups consist of dynamic sets of receivers (also called members) and senders to a group are not required to be members of the group. A group may have a single data source, as for a video broadcast, but in the general case there can be many sources per group.

As the number of multicast groups and sources grows, the amount of state required at each multicast router grows. One method to reduce this state uses *group-shared trees*, in which data from all sources in a multicast group is distributed along a single shared tree, rather than a separate tree for each source. This obviates the need to keep per-source information for the multicast group at each intermediate router.

Manuscript received January 23, 1996; revised August 14, 1996. This work was supported in part by the National Science Foundation under Grant NCR-9417032.

The authors are with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109-2122 USA.

Publisher Item Identifier S 0733-8716(97)02261-0.

Ideally, a group-shared tree would use a minimal spanning tree to minimize total bandwidth usage, at the expense of end-to-end delay. Finding this tree for some subset of nodes in a graph is known as the minimal Steiner tree problem and is known to be NP-complete [2]. Previously proposed heuristics, surveyed in [3], typically require knowledge of the complete network topology, which is impractical for the Internet.

A simpler approach to constructing a group-shared tree, proposed by Wall [4], is to use a *center-specific tree*. In this approach, a single node is chosen near the center of the group. The group-shared tree then becomes the shortest-path tree rooted at that node. Wall shows that a topologically centered tree gives a delay bound of twice that of source-specific trees. If the root is moved to a group member, the bound becomes three times that of source-specific trees.

The advantages of a center-specific tree over a minimal Steiner tree thus include bounded delay and simpler implementation. Wei and Estrin [3] show that in terms of total bandwidth usage, center-specific trees lie somewhere between the minimal Steiner tree and source-specific trees.

Recent multicast routing protocol efforts, such as protocol independent multicast (PIM) [5] and and core-based trees (CBT) [6], rely on the notion of center-specific trees. In CBT, group-shared trees have centers called "cores." In PIM, a group-shared tree is rooted at a rendezvous point (RP). Both terms are conceptually equivalent, and we will refer to the root of a center-specific tree as simply a *center*. In both protocols, the mechanism for distributing the center's identity is orthogonal to the method for choosing a center. An analysis of distribution mechanisms and their overheads and convergence times is outside the scope of this paper.

While locating the best center is simple given complete topological information, such information is not always available in distributed routing protocols. Current approaches typically use either administrative selection of centers or some simple heuristic.

In this paper we investigate the problem of finding a good center in a distributed fashion and examine various heuristics for automating center selection. We also propose new heuristics and center-location protocols.

The remainder of this paper is organized as follows. Section II details several previous proposals. In Section III, we present new center-location algorithms. Section IV describes our simulation results, and Section V covers conclusions and the future.

II. Previous Work

A number of methods have already been proposed for center location. In this section we present a brief overview of such methods and their performance. As a reference for comparison,

we use an "optimal" center-based tree (OCBT) chosen by calculating the actual cost of the tree rooted at each node in the network and picking one which gives the lowest maximum delay over all those with the lowest cost.

In the random source-specific tree (RSST) heuristic, the center is chosen randomly among the sources and does not move. Doar and Leslie [7] found the ratio of the costs of this approach to the optimal minimal Steiner tree cost to be typically between one and two in random graphs of average node degree three to six. The RSST approach is also equivalent to selecting the first source or the initiator of the multicast group, as suggested by PIM [5] and CBT [6]. Note that this approach only gives a single center, rather than a list of possible centers which is required for fault tolerance.

Wei and Estrin [3] show that the minimum shortest path tree (MSPT) approach performs almost as well as OCBT and suggest that it is adequate for use with center-based trees. This approach requires calculating the actual costs for the trees rooted at each group member and chooses the member with the lowest cost. Wall [4] shows that such a tree has a delay bound of three times that of a source-specific tree for each source (whereas a topologically centered tree has a delay bound of two times that of a source-specific tree for each source). We observe that the MSPT approach reduces to OCBT when all nodes are group members.

Wall presents the following three center-location algorithms in [4], which operate on all nodes in the network.

The maximum-centered tree (MCT) algorithm picks the node with the lowest maximum distance to any group member. The average-centered tree (ACT) algorithm picks the node with the lowest average distance to all group members. The diameter-centered tree (DCT) algorithm finds the node which is the midpoint of the lowest maximum diameter, defined as the sum of the distances to the two furthest away nodes.

The tournament-based center-location protocol (TOURNEY), proposed by Shukla *et al.* [8], [9], runs a tournament between nodes to determine a center. Sources are initially paired with group members in decreasing order of distance, and remaining nodes are paired randomly with byes inserted appropriately. The winner of a pairing is determined by finding the node intermediate on a path between the pair. This requires either knowledge of the network topology or an exchange of route tracing messages for each pair in order to discover the necessary topological information. The tournament continues for $\lceil \log_2 M \rceil$ rounds until one winner remains, where M is the number of sources and members in the multicast group. It thus potentially involves cooperation between $2M - 1$ nodes.

Finally, we introduce for comparison a globally centered tree (GCT) algorithm, which picks the node with the lowest average distance to all other nodes in the network regardless of group membership. This is intended to model *a priori* administrative selection of a center at some central network location.

III. Issues and Alternatives

In a distributed environment, topological information is often distributed across all nodes, so that no single node has complete topological information. Thus, an ideal algorithm to locate the center of a multicast group should require only a small amount of information at each node and minimal interaction between neighboring nodes. We emphasize that multicast groups have dynamic memberships, and thus the optimal center will change over time.

This paper studies the problem of finding good centers in distributed fashion. We will examine some previously proposed heuristics and then propose several new ones. To support reliability, we extend the problem to that of finding the n best nodes to use as centers. We can then construct an ordered list of centers to use as backups, should the best center fail.

We define the *cost* of a tree to be the sum of the costs of the links in the tree. If the cost of every link is one, the tree cost is the number of links in the tree. The cost for a group-shared tree currently in use by a multicast group can be determined with a simple algorithm. "Leaf" nodes would report a subtree cost of one to their parent, while intermediate nodes would add up the subtree costs reported by child nodes and report the sum (plus one for itself) up to its own parent.

Such a method is less useful in finding the best root to use for a center-specific tree in a network of N nodes. In practice, it is not feasible to construct all N trees for a given multicast group in a distributed environment. Also, subtree costs can only be calculated in this manner for a functioning multicast group. Other off-tree nodes may not have the necessary information to do this calculation.

To calculate the actual cost of a tree for an arbitrary center, we must know the complete network topology and the list of group members. While link-state routing protocols such as open shortest path first (OSPF) [10] maintain topological information for a local domain, complete global network knowledge is not available. Any algorithm which requires complete knowledge is not useful across the MBone, as we require. Algorithms which compute actual tree costs may thus not be practical.

The list of multicast group members may also be unknown. PIM, for example, assumes that a rendezvous point (i.e., center) has (at best) a list of sources only, rather than a list of all group members. On the other hand, it may be possible to modify the routing protocol to maintain membership lists or perhaps to obtain the list of group members from some external protocol or application. For example, existing MBone applications such as `vat, wb`, and `vic` all maintain lists of group members.

Finally, the question arises as to when or how often a center-location algorithm should be executed. Overhead arises both from the cost of protocol messages and, where required by applications, retransmissions due to loss of data. Currently, it is not well understood how much of an effect changing the center of an active group in PIM or CBT will have on the loss of data, but we believe that the effects could be made arbitrarily small (at the expense of performance) by reducing the frequency at which the algorithm is executed. When applications know the sources *a priori* but not necessarily the receivers, techniques which only require knowledge of sources would be useful. In such cases, an algorithm could be run once at the outset and never again.

The optimal center is unlikely to move very much for groups with a relatively large number of members at steady state, with members leaving and joining randomly. On the other hand, once the center has been determined for small groups with dynamic membership, the tree will gradually degrade toward a randomly-centered tree as nodes join and leave the group, until the center-location algorithm is run again. Thus, there exists a tradeoff between overhead and maintaining a good tree.

A. New Proposals

Although some approaches such as RSST or TOURNEY are exceptions, many center-location algorithms operate by picking a node with minimum *weight*, where the weight is some function of measures such as cost or delay. Existing algorithms of this type generally fall into one of the following two classes.

Class A: All network nodes participate, using a list of group members. This includes algorithms such as OCBT, MCT, ACT, and DCT.

Class B: All group members participate, using a list of group members. This includes algorithms such as MSPT.

In addition to these classes, we propose for study the following four new classes of algorithms in this paper.

Class C: All network nodes participate, using only a list of sources.

Class D: All group members participate, using only a list of sources.

Class E: A hill-climbing algorithm (detailed below in Section III-A1) is used to find a local minimum, using a list of group members.

Class F: A hill-climbing algorithm is used to find a local minimum, using a list of sources.

Classes C, D, and F may be appropriate for routing protocols such as PIM which avoid enumerating all group members but do require centers to enumerate all sources. We should expect that these classes will pick a node near the center of all the sources, rather than the receivers.

1) Two New Minimization Protocols: Distributed algorithms which require all nodes in the network to participate (as Classes A and C do), typically work by having all nodes exchange information with their neighbors, keeping the best costs thus far. However, in a large network such as the MBone, it is infeasible to require that every node in the network have a list of all members for every multicast group. It is quickly becoming impractical even to require every node to maintain a list of sources for each multicast group, which was a strong motivation for center-specific trees in the first place. Thus, while Classes A and C are not practical for general use, we include them for comparison.

For Classes B and D, we propose the following protocol to find the n best nodes which minimize a weight function using a list of group members/sources.

Minimal Member Protocol (MIN-MEMB):

1) When the multicast group is created, it has only one member, which becomes the center. As nodes join or leave the multicast group, the center can migrate as group members execute the following steps periodically.
2) The center starts a timer T_1 with a fixed duration and waits until it expires. This timer determines how frequently the center location algorithm runs, and thus how much overhead the protocol will incur.
3) The center calculates its own weight according to some predefined function such as the ones described below in Sections III-A2 and III-A3. It then multicasts its own weight plus the list of group members/sources if necessary, to all group members and starts a second timer T_2 with a fixed duration.
4) Any group member which is willing to become a center then computes its own weight using the given list and waits a random amount of time (T_3) during which it listens for replies from other group members.
5) When a member's timer T_3 expires, it multicasts its own weight to all group members if its own weight is less than the nth lowest weight heard so far.
6) Once the initiator's timer T_2 expires, the node reporting the lowest weight is chosen as the next center. The process then repeats from step 2). To avoid frequent center migrations, the center's timers can be set to some reasonably high values, and the center can refuse to relinquish the position of center unless the weight improvement is above some threshold. (Note that if the threshold is infinite, this reduces to the simple RSST model.)

Each round thus requires an average of between $(m+1)/2$ and m messages (for $n = 1$ and $n = m$, respectively) to determine the list of n best nodes, where m is the number of members in the group. The associated overhead of constructing a center list is therefore $O(m)$. We note that the time T_3 must also be scaled proportional to m, decreasing the timeliness of the information when there are a large number of members. We do not expect this to be a problem, however, since, as we will later see, individual changes in the group membership have a far less significant effect on center location accuracy when the number of members is large.

The majority of the algorithms previously described which are actually feasible limit the center to be one of the group members or one of the sources. We now present a method to relax this restriction.

For Classes E and F, we propose the following protocol to construct a list of up to n best nodes which minimize a weight function using a list of group members/sources.

Hill-Climbing Protocol (HILLCLIMB): A path list holds the list of nodes in the "path" formed by traversing toward neighbors with better weights. This path list is used to ensure that the algorithm terminates. It is also trivial to impose a maximum path length so that the algorithm terminates after a certain number of hops. The protocol works as follows, starting with an empty path list and a weight function known to all nodes.

1) When the multicast group is created, it initially has only one member, which becomes the only center in the list of possible centers. The following steps then occur periodically.

2) The center starts a timer T_1 with a fixed duration and waits until it expires. It then starts a probe.
3) The probing node queries its neighbors for their weights by sending them the list of group members/sources. It then restarts the timer T_1 so the algorithm will eventually resume from this step if a message below is lost.
4) Each neighbor calculates its own weight according to the weight function and responds.
5) The probing node then updates the list of n best centers to account for the new information.
6) If the probing node's own weight is lower than the lowest neighbor weight, we proceed from step 11).
7) If all best neighbors are already in the path list, we go to step 11).
8) The probing node adds itself to the list of visited nodes.
9) The probing node picks an unvisited best neighbor to be the next probing node.
10) The old probing node sends the path list and group member/source list to the new probing node, which then proceeds from step 3).
11) The final probing node sends a message back to the center, which is the first node in the path list, informing the center of its weight.
12) The final probing node then becomes the new center and repeats the process from step 2). Again, to avoid frequent center migrations, the center's timer can be set to some reasonably high value, and the center can refuse to relinquish the position of center to another node unless the weight improvement is above some threshold. Thus, the long-term overhead can again be made arbitrarily low.

The average number of messages exchanged in determining the center list with this algorithm is equal to $(2d+1)H$, where d is the average node degree, and H is the number of hops in the path list. The overhead of constructing a center list is therefore $O(Hd)$. As we will later see, H is typically very small, so that the overhead of HILLCLIMB is much less than that of MIN-MEMB. However, the centers in the resulting list generated by HILLCLIMB tend to be located in the same general vicinity, whereas MIN-MEMB yields a more geographically-distributed list. We would expect HILLCLIMB to exhibit less data loss than MIN-MEMB when a center fails, since the new tree may have many links in common with the old tree. On the other hand, when a network partition occurs, it is less likely that all centers chosen by MIN-MEMB will be unreachable than those chosen by HILLCLIMB.

2) Weight Functions Studied: Functions proposed by others for minimizing include the actual tree cost [3], the average delay, the maximum delay, and the maximum diameter [4]. Although previous work has only dealt with functions for a single algorithm class, we will generalize the functions to apply to any of the six classes described in Section III-A.

Let S be the node list used by nodes participating in the algorithm. Thus, S is the set of multicast group members for Classes A, B, and E. For the remaining classes, S is the set of sources. We then define the following weight functions for a given set S and root

$$\text{Actual Cost} = \text{number of links in tree rooted at } root \text{ and extending to all of } S \tag{1}$$

$$\text{Max Dist} = \max_{u \in S} d(root, u) \tag{2}$$

$$\text{Avg Dist} = \frac{1}{|S|} \sum_{u \in S} d(root, u) \tag{3}$$

$$\text{Max Diam} = \max_{u \in S} d(root, u) + \max_{v \in S,\, v \neq u} d(root, v) \tag{4}$$

where $d(a, b)$ is the distance from a to b.

To reiterate, *Actual Cost* does not lend itself well to distributed computation for a large number of groups. However, the other weight functions all rely on local distance information and are thus applicable to routing domains where the distance to other nodes is known (as is true in today's MBone). In Section IV-E, we evaluate how these weight functions perform with various algorithms.

3) The Estimated Cost Function: Our work suggests that it is useful to define another function describing an *estimated* tree cost, calculated by taking the average of the maximum and minimum bounds on tree cost. To estimate costs, we will again use the distance for each possible destination, information which is already available to routers.

To get a lower bound on the cost of a tree rooted at some node, we observe that the best-case tree is linear. In this case, all group members lie on the path from the root to the farthest member, so that the cost of the tree is simply the maximum distance from the root to any group member. When the distances are given as hop counts, we can get a slightly tighter bound. Specifically, if two group members are at an equal distance, the distribution tree cannot be completely linear, but must have at least one additional link. Thus

$$\text{Est Cost}_{\min} = \max_{u \in S} d(root, u) + \text{number of duplicate distance nodes in } S.$$

To get an upper bound on the cost of the tree rooted at some node, we note that in the worst case, no links are shared among the paths to each member. Thus, the maximum tree cost is the sum of the member distances. If the number of group members (other than the root, if it is a member) is greater than the node degree, we may tighten the bound by subtracting the difference to account for the knowledge of sharing those links. If distances are given in hop counts, we get

$$\text{Est Cost}_{\max} = \begin{cases} \displaystyle\sum_{u \in S} d(root, u) & \text{if } |S| \leq \deg(root), \\ \left\lceil \displaystyle\sum_{u \in S} d(root, u) \right\rceil & \\ \quad -[|S| - \deg(root)] & \text{otherwise.} \end{cases}$$

We now define

$$\text{Est Cost} = \frac{\text{Est Cost}_{\min} + \text{Est Cost}_{\max}}{2}. \tag{5}$$

Although routers also keep the identity of the next hop neighbor used to reach each destination, in general, one cannot use this information to draw conclusions about distant nodes on the actual multicast tree. This is because the actual tree may be using reverse paths (shortest path from each group member to the root) rather than forward paths (from the root to each group member), so that a member may be on a subtree rooted at a different neighbor than the listed next hop. This typically occurs when multiple equal paths exist.

B. Using the Estimated Cost Function

We now present several new algorithms, corresponding to several of the classes from Section III-A, that use the estimated cost heuristic of Section III-A3.

Class B: The minimum estimated member-member tree (MEMMT) heuristic uses the MIN-MEMB protocol with the list of all multicast group members to find the member with the lowest estimated tree cost. This is equivalent to MSPT except that tree costs are estimates only. This approach may be feasible since, as has already been mentioned, group members may already *have* a list of all other members.

Class D: The minimum estimated member-source tree (MEMST) heuristic is motivated by the fact that in the existing PIM specification, a rendezvous point (center) knows only the list of sources, rather than the list of all group members. MEMST uses the member whose tree to all sources (only) contains the least number of estimated links, thus choosing a node closest to the center of all sources. Note that this reduces to RSST for a single source which is a member and to MEMMT when all members are sources.

MEMST again uses the MIN-MEMB protocol, except that the list of sources is used in place of the list of group members. This approach is feasible in light of the fact that the current center may already maintain a list of sources, as in PIM.

Class E: The member-based hill-climbing algorithm (HC-M) uses the HILLCLIMB protocol with estimated cost as its weight function. It requires a list of all members in the multicast group to be passed along the path.

Class F: The sender-based hill-climbing algorithm (HC-S) functions like HC-M except it uses only a list of sources.

Table I summarizes the requirements of the various algorithms which have been described above.

TABLE I
REQUIREMENTS OF CENTER-LOCATION ALGORITHMS

Algorithm	Source list	Member list	Actual tree costs	Knowledge of topology[1]	Distance information
OCBT	No	No	Yes	No	No
RSST	No	No	No	No	No
MCT	No	Yes	No	No	Yes
ACT	No	Yes	No	No	Yes
DCT	No	Yes	No	No	Yes
MSPT	No	Yes	Yes	No	No
TOURNEY	Yes	Yes	No	Yes	No
MEMMT	No	Yes	No	No	Yes
MEMST	Yes	No	No	No	Yes
HC-M	No	Yes	No	No	Yes
HC-S	Yes	No	No	No	Yes
GCT	No	No	No	No	Yes

[1] Although knowledge of the underlying topology is not explicitly assumed by OCBT and MSPT, some knowledge is necessary for computing the actual tree costs.

IV. PERFORMANCE STUDIES

In our simulations, all links were symmetric with unit cost, so that tree cost is simply the total number of links in the tree. For the purpose of constructing trees, we also assume all sources are also group members. Each simulation point reflects an average over 500 runs, using an average node degree of four unless otherwise specified.

A. Generating Random Graphs

To avoid limiting ourselves to any specific network, we generate random network topologies which exhibit connectivity characteristics approximating real-world networks.

We use the random graph model presented by Waxman [11], where nodes are randomly distributed over a Cartesian coordinate system. The probability that an edge exists between any two nodes u and v is given by the probability function

$$P(\{u, v\}) = \beta \exp \frac{-d(u, v)}{L\alpha}$$

where $d(u, v)$ is the distance between the two nodes, L is the maximum possible distance, and α and β are parameters in the range $0 < \alpha, \beta \leq 1$. Larger values of α increases the proportion of longer edges to shorter edges, while larger values of β increase the average node degree.

Graphs are then generated until one is found which has a single connected component.

B. Generating Hierarchical Random Graphs

To address scalability issues in the MBone, Thyagarajan and Deering [12] have recently proposed using a hierarchy. We therefore examine the performance of center-location algorithms in hierarchical networks, where nodes are divided into regions connected by a backbone.

To generate random hierarchical graphs, we used eight regions of 50 nodes each. A random graph with average node degree four was generated for each region, and a random graph with average node degree 2.5 was generated for the backbone. Two nodes in each region were randomly selected as interregion gateways. Since each backbone node represented an entire region, each backbone link for a region was randomly assigned to one of the two interregion gateways.

C. Parameters of Interest

We will analyze the performance of various center-location algorithms according to two criteria: actual tree cost and maximum source-to-destination delay. Actual tree cost is measured using the *Actual Cost* metric defined in Section III-A2. For delay, we measure the maximum distance between any source and any other multicast group member over a tree rooted at a given center. We use the following definition, given a root, a set of sources S, and a set of group members M:

$$\text{Max Delay} = \max_{s \in S, m \in M} \text{TreeDist}(root, s, m) \qquad (6)$$

where TreeDist$(root, s, m)$ is the length of the shortest path between s and m along links in the tree rooted at $root$.

We must bear in mind that this concept is fundamentally different from the Max Dist weight function defined in Section III-A2, which only measures the maximum distance from the *root* to any group member, rather than from a source.

In practice, a lower tree cost reduces overall bandwidth requirements and effectively raises the number of multicast groups that can be supported by the network. This is especially important since it is expected that the number of multicast groups will become very large in the future. A lower maximum delay, on the other hand, means that packets from sources will tend to arrive at their destinations sooner. A tradeoff exists between these two goals, as we will see in the following sections. We note, however, that delay is much less critical than tree cost when the option exists, as in PIM, to use shortest-path trees for delay-sensitive applications.

We now examine the performance of the various classes of algorithms and weight functions according to these criteria. Two other parameters that we expect to significantly affect the performance are the fraction of network nodes which are group members and the number of sources per group. These are important because in practice, we require center-location algorithms to scale well for groups with many members and sources.

D. Analysis of Group Dynamics

To determine how often a center-location algorithm should be run, we must determine how quickly a tree degrades as the group membership changes. We start with an optimal tree as determined by OCBT and let the membership change through events corresponding to nodes joining or leaving. For the probability that a given event corresponds to a node joining the group, we will use the function

$$P_c(k) = \frac{\gamma(n-k)}{\gamma(n-k) + (1-\gamma)k}$$

where n is the number of nodes in the network, k is the current number of group members, and γ is a parameter in the range $0 < \gamma < 1$ representing the fraction of nodes which are members at equilibrium [7], [13]. Thus, $P_c(k) > \frac{1}{2}$ when $k < \gamma n$, $P_c(k) = \frac{1}{2}$ when $k = \gamma n$, and $P_c(k) < \frac{1}{2}$ when $k > \gamma n$.

Fig. 1 gives the results of this simulation, using 400-node graphs with ten members and five senders. The line *Random Center* indicates the ratio of the expected cost of a tree rooted at a randomly-selected center to that of an optimal tree. The line *Original Center* indicates the cost ratio between the new tree rooted at the original center and an optimal tree. The parameter Δ indicates the amount by which the group has changed, as given by

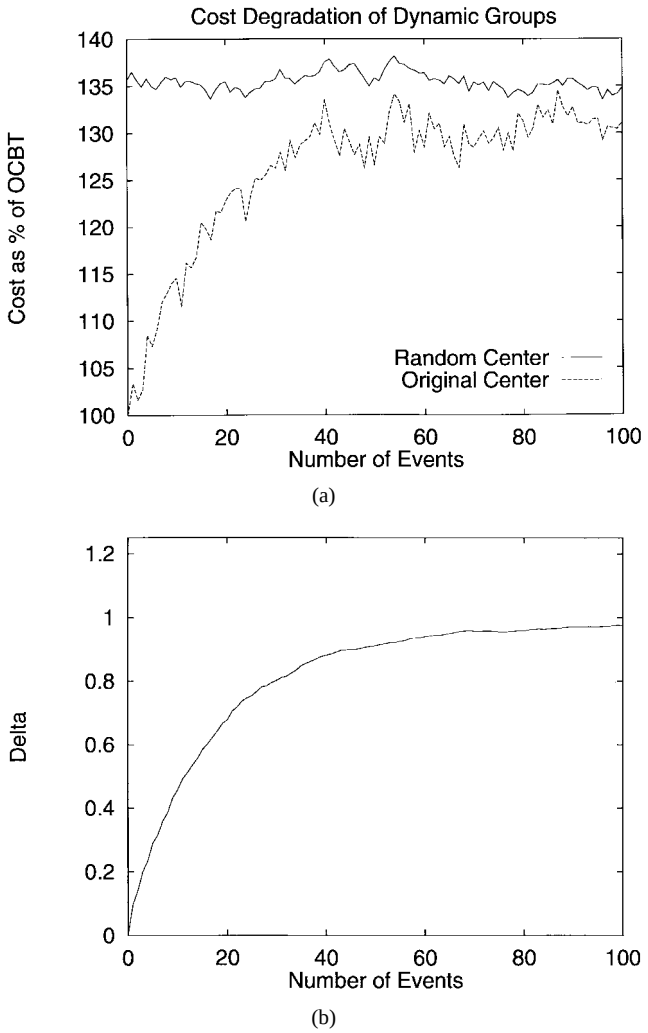


Fig. 1. Cost degradation of dynamic group.

$$\Delta = 1 - \frac{|G_0 \cap G_i|}{\max(|G_0|, |G_i|)}$$

where G_0 is the original group membership and G_i is the current group membership. From these graphs, we see that the cost of keeping the original center as the membership changes approaches that of using a randomly-centered tree. We also observe that a strong correlation exists between Δ and the tree cost, suggesting that a threshold for recomputing the center could be based on Δ, which is much easier to compute locally than tree cost. For this simulation, we see that after about 40 events, 90% of the membership had changed, and tree cost had likewise degraded about 90% of the way toward a randomly centered tree.

Almeroth and Ammar [14] describe one study of observed group dynamics on the MBone. For example, a Space Shuttle broadcast over an 11-day period saw 5055 member connections of average duration 300 min, giving an average group

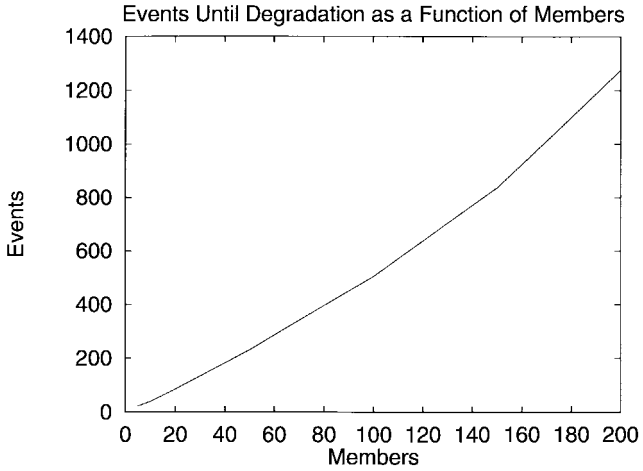


Fig. 2. Events until 90% membership change.

size of 96 and an average event rate of 0.64 events/min. From our simulations, we estimate that this type of group would take over 12.75 hours to reach a 90% membership change (assuming 2500 subnets in the MBone and at most one member per subnet).

Fig. 2 shows the average number of events (E) needed to reach a 90% change in group membership for networks (hierarchical or not) of 2500 nodes. An average over 100 000 trials was calculated for group sizes of 5, 10, 20, 50, 100, 150, and 200 members, giving values for E ranging from approximately 22 to 1275 events. The average time until 90% membership change (denoted $T_{0.9}$) can be determined as E/(mean event rate). Assuming that group membership does not grow without bound, we have (mean join rate) $=$ (mean leave rate) $=$ (mean event rate)/2. Little's theorem tells us that (average group size) $=$ (mean join rate) $\cdot$ (average connection duration). Putting these together, we obtain the following formula:

$$T_{0.9} = \frac{E \cdot (\text{average connection duration})}{2 \cdot (\text{average group size})}.$$

Thus, the 90% membership change time typically lies between two and three times the average connection duration in our simulations. Since global membership changes may not be observable at individual nodes, the criteria above can be used to determine an appropriate period for running a center selection algorithm based on expected characteristics.

E. Analysis of Weight Functions

First, we will compare the effects of using weight functions (1)–(5) from Sections III-A2 and III-A3. We start by running Class A algorithms which will choose the network node which minimizes each of Actual Cost, Est Cost, Avg Dist, Max Dist, and Max Diam. Fig. 3 shows the results of 100 trials using five senders in a 50-node network as the number of members in the group varies. Each weight function was used on the same set of 100 graphs. The Y-axis plots the ratio between the average Actual Cost at a center located using each weight function and the optimal center location as determined by minimizing Actual Cost. Several facts are apparent from these two plots.

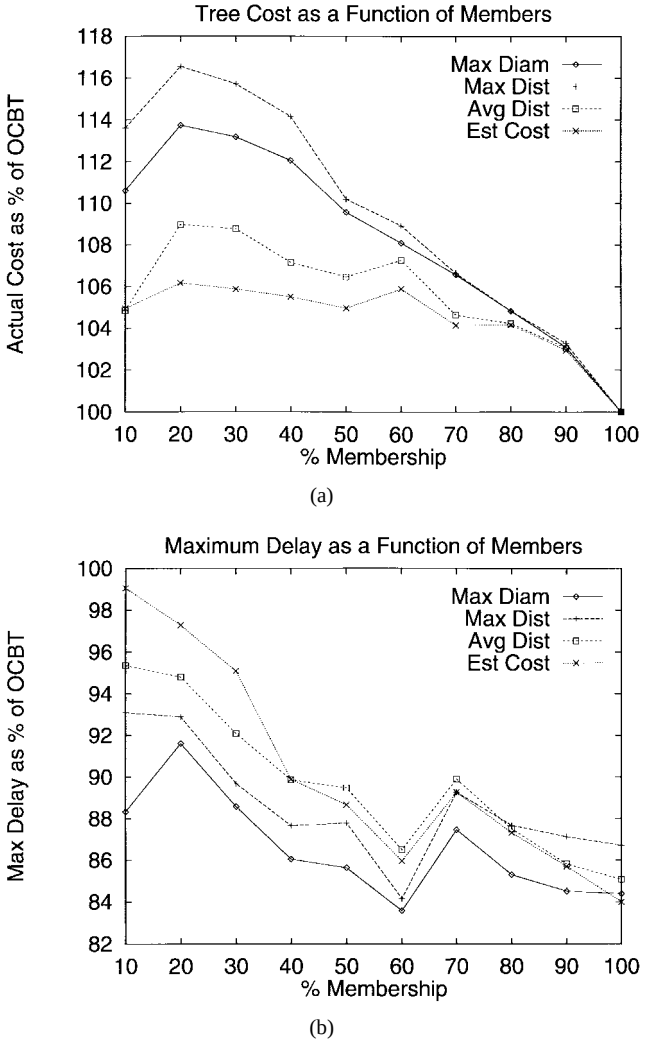


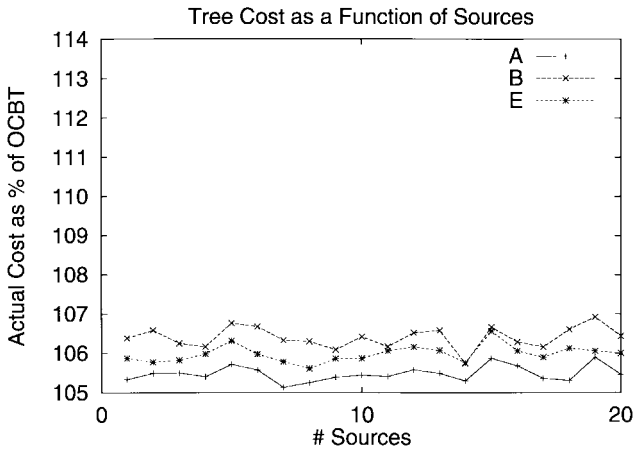
Fig. 3. Cost versus delay of functions.

We see that the weight functions which give the best actual cost typically give the worst average delay, showing that a cost versus delay tradeoff exists. The Max Diam weight function gave the best maximum delay, while our Est Cost function gave the best tree cost.

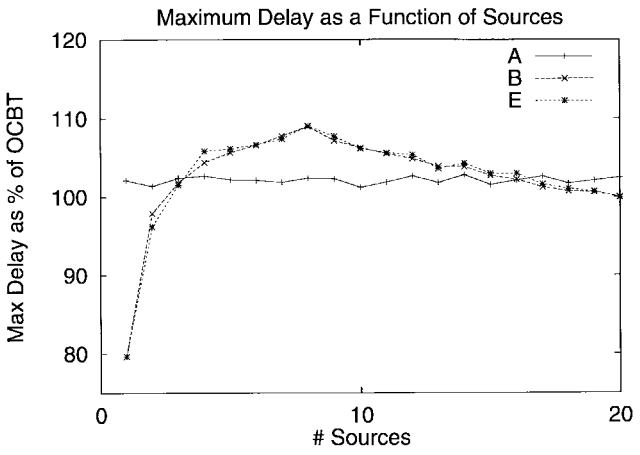
Interestingly enough, the Max Dist measure provided worse maximum delay than did Max Diam. This is due to the fundamental difference between the Max Delay measured, which is from a *source* to a group member, and the Max Dist function, which minimizes the maximum delay between the *root* and the group members. Max Diam, on the other hand, is not as biased toward a single distant member.

The Avg Dist function did not perform as well since it tries to provide a lower *average* delay and may yield higher maximum delays. While the actual values in all cases depend on parameters such as the number of nodes and senders, the relative positions of points remained relatively constant under different conditions in our simulations.

Finally, when all nodes are members of the multicast group, the tree will include every network node. In this case, every tree will have exactly $N - 1$ links. The location of the center has no effect on Actual Cost, and all algorithms converge as shown.

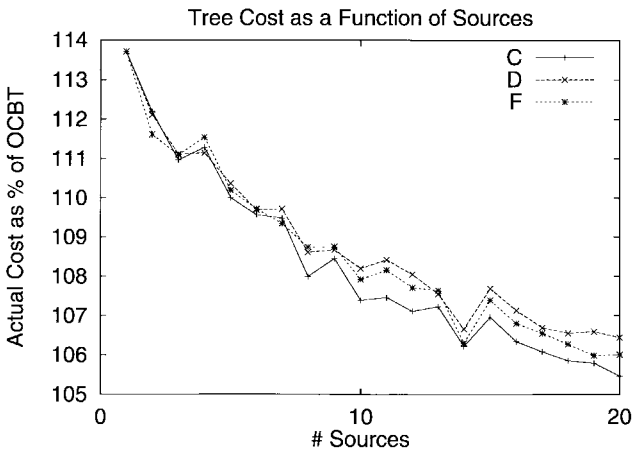


(a)

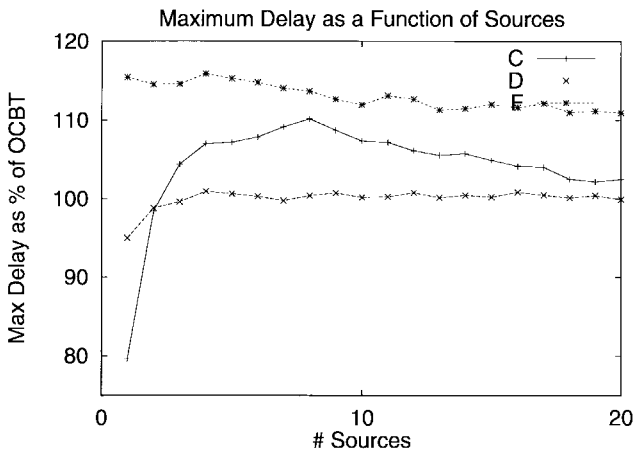


(b)

Fig. 4. Performance of classes using a list of members.



(a)



(b)

Fig. 5. Performance of classes using a list of sources.

F. Analysis of Algorithm Classes

Next, we wish to see where the various algorithm classes lie in terms of cost versus delay. For this analysis, we pick Est Cost as the weight function and run the algorithm for each class using this function. Figs. 4 and 5 show the results of 500 trials using 20 members in a 50-node network, as the number of sources in the group varied. The Y-axis again plots the average ratio between the Actual Cost at a center located using each class of algorithm and the optimal center location as determined by minimizing Actual Cost. The hill-climbing algorithm for Classes E and F used a random sender as the initial location.[2] Each class of algorithm was run on the same set of 500 graphs.

For the relationships between Classes A–D, these results only confirm what was already intuitive: A and C give better tree costs than B and D since they find the best node in the network, while B and D are limited to the best node which is a group member. Similarly, A and B give better tree costs than C and D since they use more complete information to compute weights.

However, what is interesting from Figs. 4 and 5 is the performance of the hill-climbing Classes E and F. These results indicate that they provide better performance than Classes B and D which locate the center at a group member. As a reminder, Classes A and C are infeasible in real world networks but are shown simply for comparison.

[2]Simulation showed that hill-climbing was relatively insensitive to the location of the initial node.

G. Analysis of Proposed Algorithms

Having analyzed the performance of the various algorithm classes and weight functions, we now compare the actual center-location methods which have been proposed, since several of them do not fall into the category of algorithms analyzed above.

Fig. 6 shows the effects of varying the group size on the proposed algorithms. For simplicity, we have limited these plots to the algorithms which may be feasible. For reference, we include MSPT, which is feasible only in a limited domain. This simulation was run on 50-node graphs with five senders. The results for other values of these parameters were similar.

RSST performed the worst in terms of both cost and delay, which is hardly surprising. Although none of the algorithms performed as well as MSPT in terms of cost, they each provide better performance than RSST, with HC-M being the best overall, followed closely by MEMMT.

When there are few members in the group, the percent difference in delay is higher simply because the tree cost is

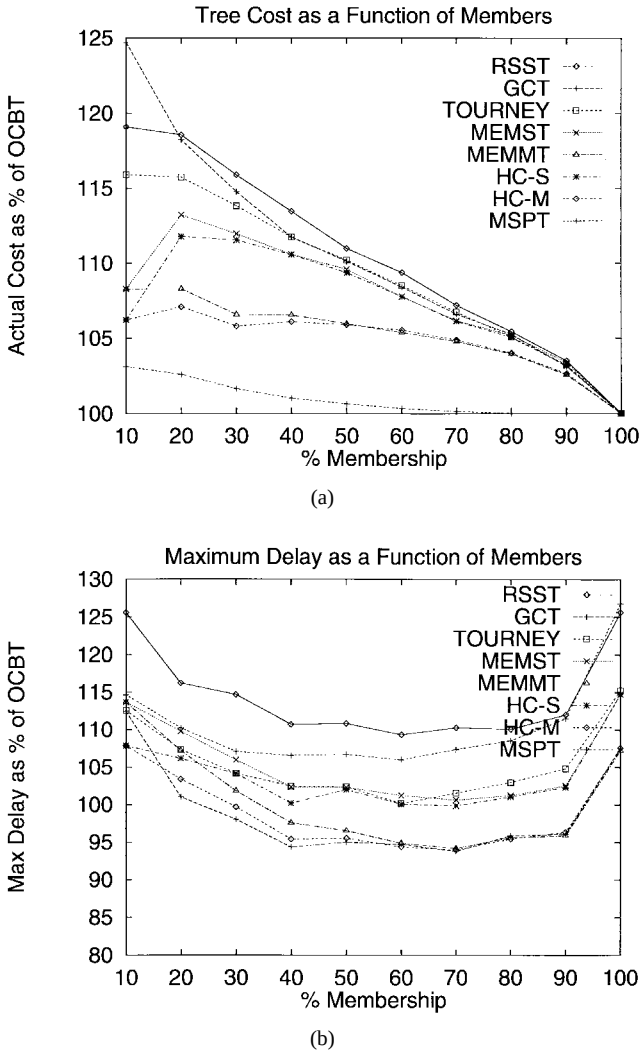


Fig. 6. Effects of group size on proposed algorithms.

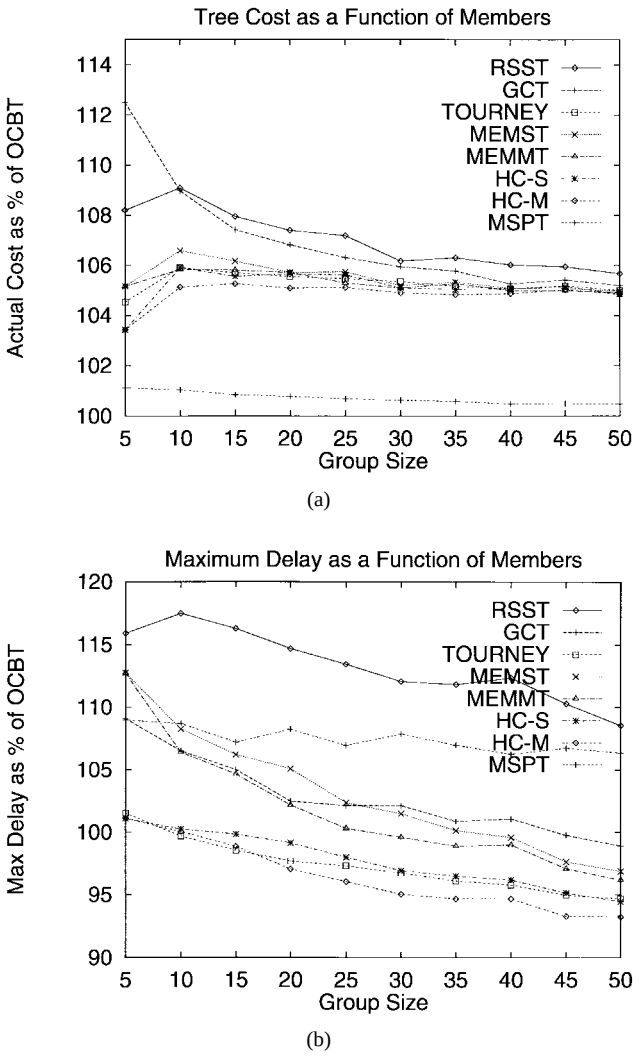


Fig. 7. Effects of group size on proposed algorithms in hierarchical graphs.

much lower. Therefore, the difference in maximum delay is a higher proportion of the actual value.

It is interesting to note that near 100% membership, the algorithms give worse delay. This is because when all nodes are members, every tree has exactly $N-1$ links regardless of the center location. Thus, the algorithms essentially become more random since they do not attempt to optimize for source-to-destination delay.

Fig. 7 shows similar effects in our hierarchical network model. While Fig. 6 showed various group sizes within a single 50-node region, Fig. 7 uses the same group sizes spread out over eight 50-node regions. This corresponds to a membership range of 1.25–12.5%, with the number of senders remaining constant at five.

From these graphs, we can see that the relative performance of the various algorithms remained relatively stable, although they were closer to each other in terms of tree costs. For five-member groups in this simulation, the hill-climbing algorithms gave trees with a cost of 92% of those used by GCT, rather than only 85% of the cost as seen in Fig. 6. The hierarchical topology allows fewer degrees of freedom in constructing trees, since gateway nodes and backbone links tend to be in the tree for most groups and centers. Thus, trees in a hierarchical network tend to be more similar than in a nonhierarchical network.

To investigate the effects of varying the number of sources, we simulated the performance of each algorithm on 50-node graphs with ten members and one to ten sources. Again, the results were similar for other parameters. Fig. 8 gives the results from this simulation.

MSPT's requirement to compute actual tree costs is not feasible, but its performance is again shown for comparison. We see that HC-M, followed by MEMMT, give the best tree costs since they use a list of all group members. HC-S and MEMST reduce to the simple RSST for only one sender, and to HC-M and MEMMT, respectively, when all members are sources. This is because they use a list of sources, and hence locate the center near the center of all sources, rather than the center of all group members.

From the plot on the right, we notice that RSST, MEMST, and HC-S provide lower maximum delays than the others when there are few sources. For a single source, this is because the center will always be located at the source. Thus, all

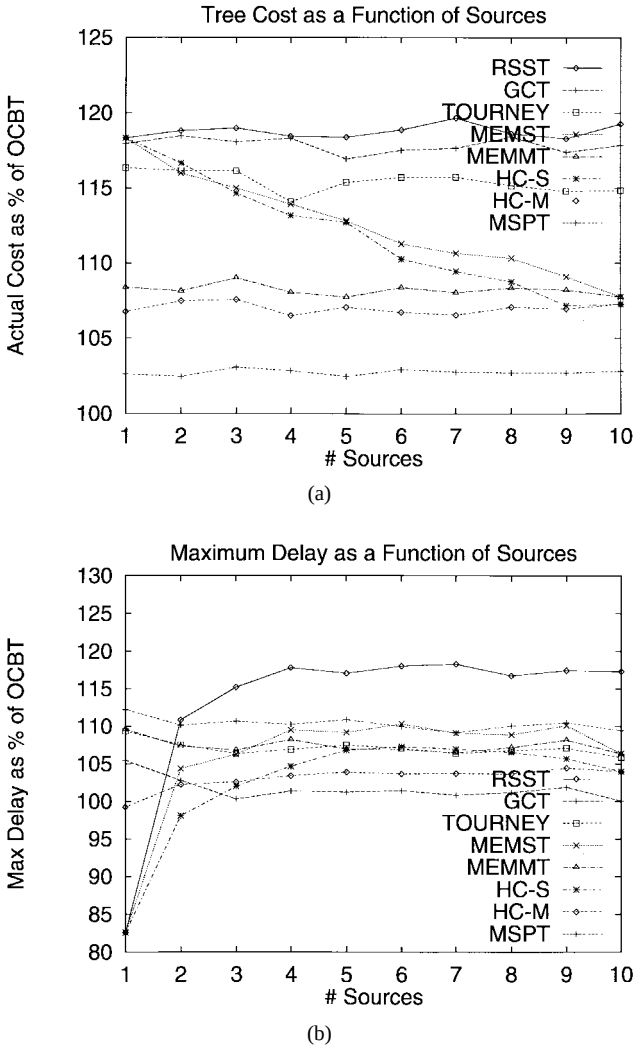


Fig. 8. Effects of number of sources on proposed algorithms.

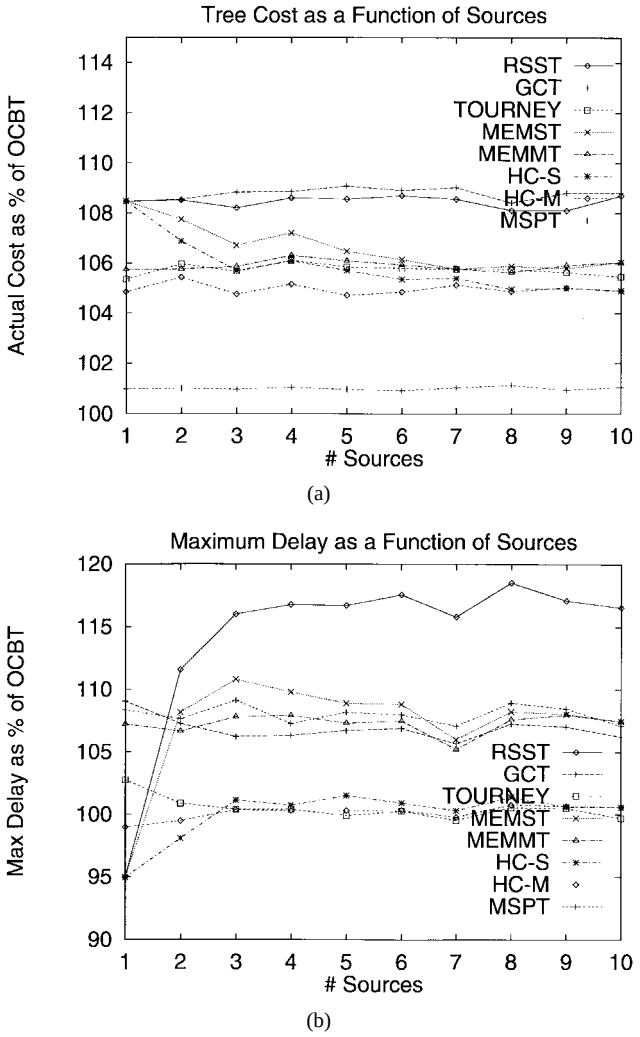


Fig. 9. Effects of number of sources on proposed algorithms in hierarchical graphs.

packets will follow the shortest path tree, providing the least delay. As the number of sources increases, the center of the sources becomes closer to the center of all group members, and the distance from each source to the center increases. This latter fact explains the increase in maximum delay.

Fig. 9 shows similar effects in our eight-region hierarchical network model. As before, the relative performance of the algorithms remains relatively constant, and the smaller differences between them are due to the topological constraints imposed by the hierarchical model.

For further analysis of the overhead incurred by the hill-climbing algorithms, we ran another simulation to determine the average path length traversed in determining the list of centers. Fig. 10 shows the effects of varying the network size and size of the member/sender list used. For Fig. 10(a), 400-node graphs were used, while varying the list size. For Fig. 10(b), 20-node and five-node lists were used as the network size varied. In each case, hill-climbing began from a random sender. From these plots we see that the path length is relatively insensitive to group size and grows slowly with the size of the network. For these reasons, the hill-climbing algorithms exhibit good scaling properties.

Fig. 11 shows the average path length in our hierarchical network model for various list sizes between 5 and 350. Again, the results are similar.

Now that we have seen the effects of varying the network parameters on the performance of the algorithms, it is interesting to see where each lies on the cost versus delay axes. Fig. 12 elaborates on the results in Fig. 6 corresponding to 50 nodes, five senders, and ten members in the multicast group (20% membership). We also include the performance of GCT and Wall's ACT and MCT algorithms for comparison. We recall that OCBT, MSPT, ACT, and MCT are all infeasible for general usage in the MBone today because they require either the ability to compute actual tree costs or that every node in the network have a list of group members. From this plot, we see that of the potentially feasible algorithms, our algorithms HC-M, MEMMT, HC-S, and MEMST provide better overall performance than the others.

As pointed out in Section III, once the center location has been determined for small groups with dynamic membership, the cost and delay will degrade toward random center placement until the center-location algorithm is run again. This effect may be less significant for groups at steady state with a large number of members.

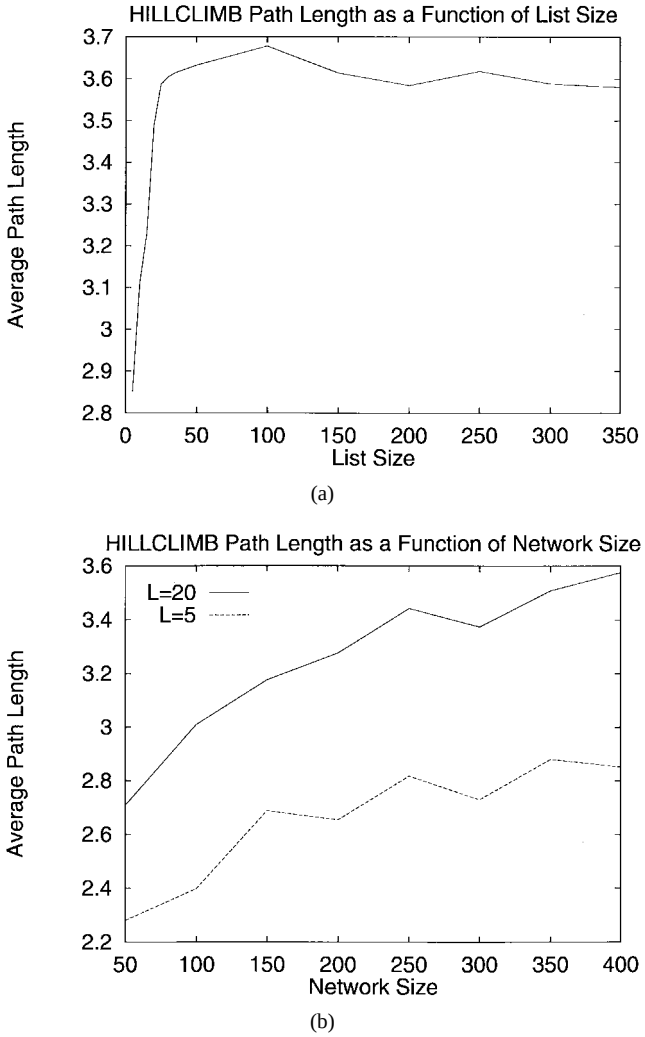


Fig. 10. Overhead of HILLCLIMB algorithms.

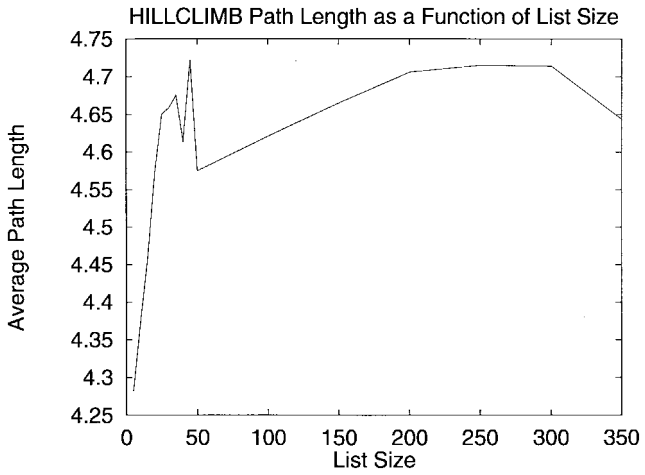


Fig. 11. HILLCLIMB overhead in hierarchical graphs.

H. Analysis Using Hierarchical Routing

Hierarchical routing can be used to reduce the size of routing tables needed in a hierarchical network. We assume a two-level hierarchical network composed of a collection of interconnected nonhierarchical regions, as in the HDVMRP proposal [12] for the MBone. Nodes inside regions maintain complete routing tables for the region, but know nothing about the structure of other regions. Each interregion gateway is preconfigured with a metric for a default route it advertises within a region. This route is then used by internal nodes to determine routes to nodes in other regions.

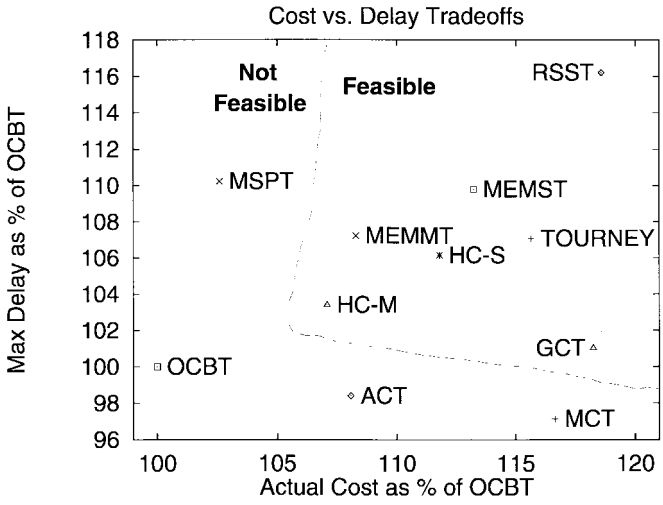


Fig. 12. Relative performance of proposed algorithms.

Figs. 13 and 14 show the effects of varying the group size and the number of senders on the proposed algorithms under hierarchical routing. We used the same hierarchical model described in Section IV-B, but with modified routing tables. To set a default route metric at each gateway, we found the average distance to all nodes in other regions and used the closest integer value.

From these two figures, we see that the differences between the proposed algorithms become almost negligible for groups with more than a few members. This is because algorithms which rely on distance information are not able to estimate distances as well in the presence of default routes. This problem could be solved by determining distance information from methods other than the use of routing tables. For example, traceroute-like messages could determine actual distances between each potential center and the group members/senders, but only by incurring a much higher overhead.

Another method of improving performance would be to use a hierarchy of trees. Each region with members would select its own internal center for a tree extending to all internal members as well as to its interregion gateways. Another tree would be similarly constructed for the backbone which would extend to all regions having group members. If nodes inside regions keep complete routing tables for the region, this model would then give performance similar to the original nonhierarchical model within each region.

V. Conclusions

Recent multicast routing protocol proposals such as PIM and CBT have been based on the notion of center-specific trees and distribute packets from all sources over a single shortest-path tree rooted at some center. For locating the center of a group, they provisionally use administrative selection of centers or

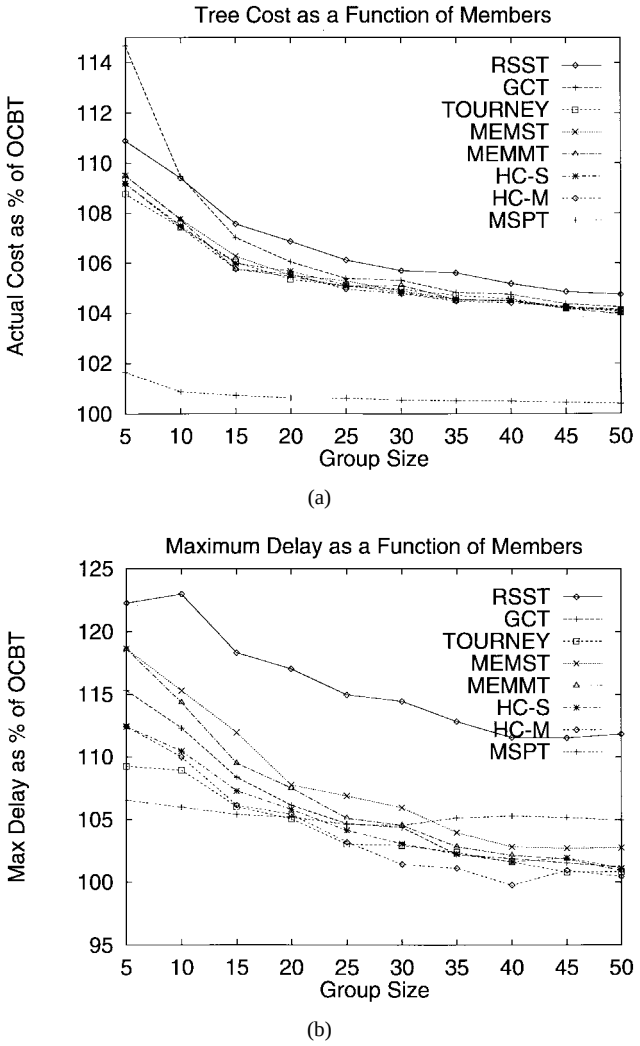


Fig. 13. Effects of group size with hierarchical routing.

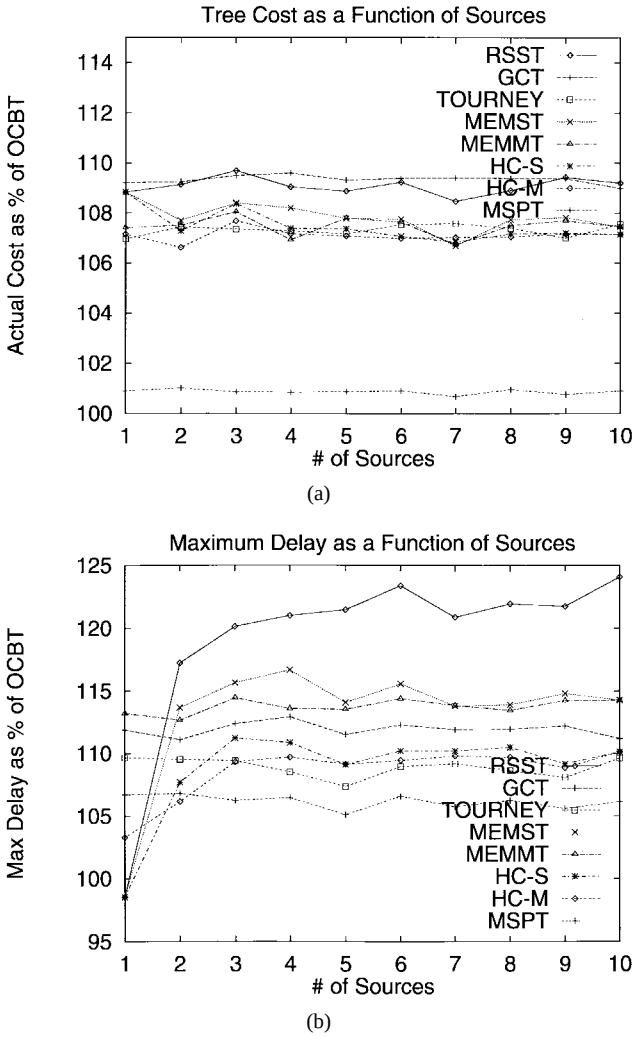


Fig. 14. Effects of number of sources with hierarchical routing.

trivial heuristics but do not preclude the use of other methods as long as they provide an ordered list of centers.

In this paper we have investigated the problem of finding a good center in a distributed fashion and examined various heuristics for automating center selection. We have also proposed several new algorithms, including MEMMT, MEMST, HC-M, and HC-S, which we feel to be more applicable to real-world networks than existing heuristics which require knowledge of the complete network topology.

Simulation results for all the algorithms show that of the ones which may be technically feasible in the MBone, HC-M offers the best results in terms of tree cost. Of those using a list of sources, HC-S provides the best results. These two algorithms also exhibit favorable scaling properties. In hierarchical networks, the relative performance of the algorithms remain relatively constant, but the differences between them are smaller due to the more constrained nature of the network. This suggests that center-location algorithms (as opposed to random or administrative selection of centers) are more useful for groups with an intradomain scope than for those with an interdomain scope, although some improvement in performance would always occur. On the other hand, if each domain had its own center for its portion of the tree, the simulation results using flat networks would again apply, yielding favorable results.

In the presence of hierarchical routing with default routes, the differences in trees given by each algorithm become negligible for groups with more than a few members. A better scheme in this case would be to construct separate trees for each region, connected by another tree at the backbone level.

A more difficult problem results when only a subset of nodes are willing to become centers. This may occur, for example, if only a subset of the routers have been upgraded to use a new center-location algorithm. In this situation, MEMMT and MEMST will both work without modification. Since only members willing to become centers will respond to a multicast request, the best site will be chosen from among the candidates for center. HC-M and HC-S, on the other hand, must be modified so that each node keeps the closest candidate center for each interface. The HILLCLIMB protocol would then use the list of closest candidate centers in place of the list of neighbors. When all nodes are candidates, this becomes equivalent to the existing HILLCLIMB protocol specification. If this problem proves to be of practical significance, it will require further investigation.

In conclusion, the choice of whether to use a dynamic center-location algorithm depends on the importance of minimizing tree cost versus the time and complexity required. When minimizing tree cost is critical, a hill-climbing strategy works best. When the magnitude of improvement (e.g., tens of percents) is not significant, globally-centered trees are most appropriate.

REFERENCES

[1] M. R. Macedonia and D. P. Brutzman, "MBone provides audio and video across the internet," *IEEE Comput.*, pp. 30–36, Apr. 1994.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, June 1988.

[3] L. Wei and D. Estrin, "A comparison of multicast trees and algorithms," Comput. Sci. Dept., Univ. Southern California, Tech. Rep. USC-CS-93-560, Sept. 1993.

[4] D. W. Wall, "Mechanisms for broadcast and selective broadcast," Ph.D. dissertation, Stanford University, June 1980.

[5] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An architecture for wide-area multicast routing," in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 126–135.

[6] T. Ballardie, P. Francis, and J. Crowcroft, "An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, Sept. 1993, pp. 85–95.

[7] M. Doar and I. Leslie, "How bad is naive multicast routing," in *Proc. IEEE Infocom'93*, pp. 82–89.

[8] S. B. Shukla, E. B. Boyer, and J. E. Klinker, "Multicast tree construction in network topologies with asymmetric link loads," Naval Postgraduate School, Tech. Rep. NPS-EC-94-012, Sept. 1994.

[9] R. Voigt, R. Barton, and S. B. Shukla, "A tool for configuring multicast data distribution over global networks," in *Proc. INET*, 1995, pp. 455–463.

[10] J. Moy, *OSPF, Version 2*, Oct. 1991, RFC-1247.

[11] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.*, vol. SAC-6, pp. 1617–1622, Dec. 1988.

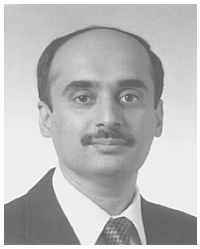
[12] A. S. Thyagarajan and S. E. Deering, "Hierarchical distance-vector multicast routing for the MBone," in *Proc. ACM SIGCOMM*, 1995, pp. 60–66.

[13] B. M. Waxman, "Performance evaluation of multipoint routing algorithms," in *Proc. IEEE Infocom'93*, pp. 980–986.

[14] K. C. Almeroth and M. H. Ammar, "Characterization of Mbone session dynamics: Developing and applying a measurement tool," Georgia Inst. Technol., Tech. Rep. GIT-CC-95-22, June 1995, Available as ftp://ftp.cc.gatech.edu/pub/coc/tech_reports/1995/GIT-CC-95-22.ps.Z.

David G. Thaler received the M.S. degree from the University of Michigan, Ann Arbor, in 1994 and the B.S. degree from Michigan State University, East Lansing, in 1992. He is currently pursuing the Ph.D. degree at the University of Michigan.

His research interests include network management and multicast routing. He is also an active participant in the Internet Engineering Task Force (IETF).



Chinya V. Ravishankar (S'82–M'84) received the B.Tech. degree in chemical engineering from the Indian Institute of Technology, Bombay, and the M.S. and Ph.D. degrees in computer science from the University of Wisconsin, Madison, in 1986 and 1987, respectively.

He has been on the faculty of the Electrical Engineering and Computer Science Department at the University of Michigan, Ann Arbor, since 1986. His teaching and research at the University of Michigan have been in the areas of databases, distributed systems, networking, and programming languages. He founded the Software Systems Research Laboratory at the University of Michigan, where he is also a member of the Real-Time Computing Laboratory.

Dr. Ravishankar is a member of the IEEE Computer Society and of the Association for Computing Machinery.