Distributed Center-Location Algorithms: Proposals and Comparisons*

David G. Thaler and Chinya V. Ravishankar
Electrical Engineering and Computer Science Department
The University of Michigan, Ann Arbor, Michigan 48109-2122
thalerd@eecs.umich.edu ravi@eecs.umich.edu

Abstract

Recent multicast routing protocol proposals such as PIM and 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. While PIM and CBT provisionally use administrative selection of centers or trivial heuristics for locating the center of a group, they do not preclude the use of other methods as long as they 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 showing that of the methods potentially feasible in the Internet Multicast Backbone, ours offer the best results in terms of cost and delay.

1 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, 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.

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 group-shared trees, 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 center. 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 PIM [5] and 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.*

While locating the best center is simple given complete topological information, such information is not

*This work was supported in part by National Science Foundation Grant NCR-9417032.

always available in distributed routing protocols. Current approaches typically use either administrative selection of centers or some trivial 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 2 details several previous proposals. In section 3, we present new center-location algorithms. Section 4 describes our simulation results, and section 5 covers conclusions and the future.

2 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 1 and 2 in random graphs of average node degree 3 to 6. 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 3 times that of a source-specific tree for each source (whereas a topologically centered tree has a delay bound of 2 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, Boyer, and Klinker[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.

3 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 location will evolve over time.

This paper studies the problem of finding good centers in distributed fashion. We will examine a few previously proposed heuristics below, and then propose a few new ones. To insure reliability, we extend the problem to that of finding the n best nodes to use as centers. By finding the n best nodes, we can 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 1, 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 1 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 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 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 can be made arbitrarily small by controlling 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 relatively large groups at steady state, with members leaving and joining randomly. On the other hand, once the center location has been determined for small groups with dynamic membership, the tree will gradually degrade towards a randomly-centered tree as nodes join and leave the group until the center-location algorithm is performed again. Thus there exists a tradeoff between overhead and maintaining a good tree.

3.1 New Approaches

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 3.1.1) finds a local minimum, using a list of group members.

Class F: A hill-climbing algorithm finds 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.

3.1.1 Two 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 3.1.2 and 3.1.3. 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.)

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 towards 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 as follows.

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.

3.1.2 Weight Functions

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 3.1.

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.

$$\begin{aligned} \textit{Actual Cost} = &\quad \text{number of links in tree rooted at} \\ &\quad \textit{root} \text{ and extending to all of } S \quad (1) \\ \textit{Max Dist} = &\quad \max_{u \in S} d(root, u) \quad (2) \\ \textit{Avg Dist} = &\quad \frac{1}{|S|} \sum_{u \in S} d(root, u) \quad (3) \\ \textit{Max Diam} = &\quad max_{u \in S} d(root, u) + \\ &\quad max_{v \in S, v \neq u} d(root, v) \quad (4) \end{aligned}$$

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 available to each router.

3.1.3 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,

$$\begin{aligned} \textit{Est Cost}_{min} = &\quad \max_{u \in S} d(root, u) + \text{number of} \\ &\quad \text{duplicate distance nodes in } S \end{aligned}$$

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 number of interfaces, 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 $\textit{Est Cost}_{max} = \sum_{u \in S} d(root, u)$ if $|S| \leq deg(root)$, and $\textit{Est Cost}_{max} = (\sum_{u \in S} d(root, u)) - (|S| - deg(root))$ otherwise.

Averaging the two bounds, we obtain:

$$\textit{Est Cost} = \frac{\textit{Est Cost}_{min} + \textit{Est Cost}_{max}}{2} \quad (5)$$

Although routers also keep the identity of the next hop neighbor used to reach each destination, in general one cannot make use of 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.

3.2 Using the Estimated Cost Function

We now present several algorithms, corresponding to several of the classes from Section 3.1, that use the estimated cost heuristic of Section 3.1.3.

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

4 Performance Studies

In our simulations, all links were symmetric with unit cost, so that tree cost is simply the total number

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

Table 1: Requirements of center-location algorithms

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 4 unless otherwise specified.

4.1 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.

4.2 Parameters of Interest

We wish to 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 3.1.2. For delay, we wish to 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:

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

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 3.1.2, which only measures the maximum distance from the *root* to any group member, rather than from a source.

In practice, a better tree cost lowers 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 better maximum delay, on the other hand, means that packets from sources will arrive at their destinations sooner. A tradeoff exists between these two goals, as we will see in the following sections. It must be pointed out, however, that delay is much less critical than tree cost when the option exists to use shortest-path trees for delay-sensitive applications, as allowed by PIM.

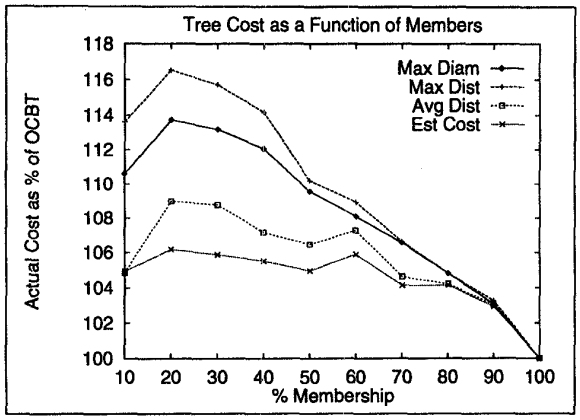
We now examine the performance of the various classes of algorithms and weight functions according to these criteria. Two other parameters that we expect to significantly affect the performance are the fraction of network nodes which are group members, and the number of sources per group. These are important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

4.3 Analysis of Weight Functions

First, we will compare the effects of using weight functions (1)-(5) from Sections 3.1.2 and 3.1.3. 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*. Figure 1 shows the results of 100 trials using 5 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.

We see that the weight functions which give the best actual cost typically give the worst average delay,

[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.



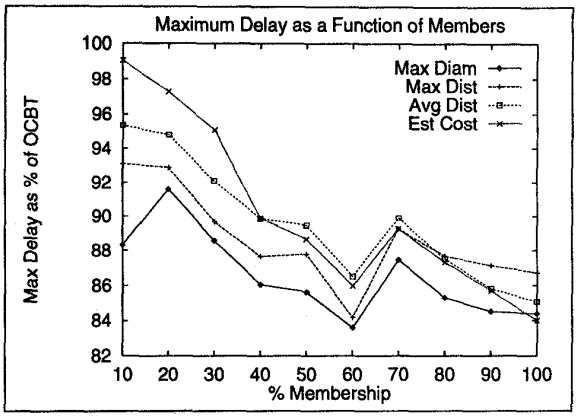
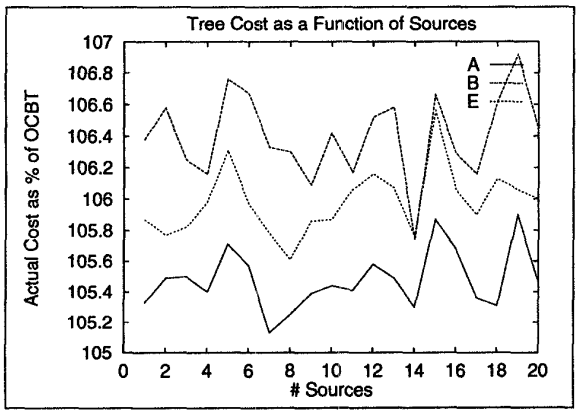


Figure 1: Cost vs. Delay of Functions



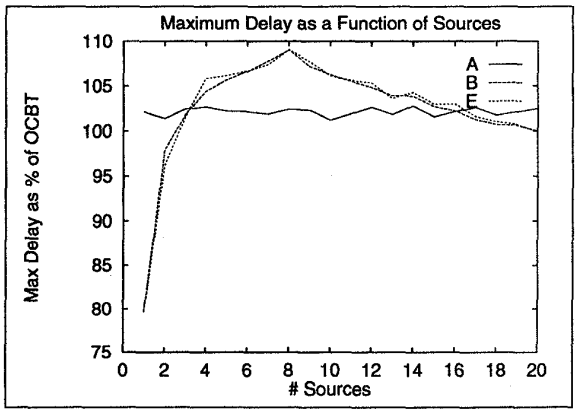


Figure 2: Performance of Classes using a List of Members

showing that a cost vs. 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 towards a single distant member.

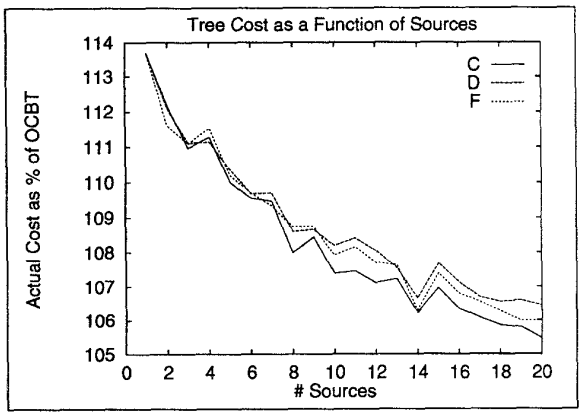
The *Avg Dist* function did not perform as well since it tries to provide a lower *average* delay at the expense of the maximum. While the actual values in all cases depend on parameters such as the number of nodes and senders, the relative positions of points remained relatively constant under different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

4.4 Analysis of Algorithm Classes

Next, we wish to see where the various algorithm classes lie in terms of cost vs. delay. For this analysis, we pick *Est Cost* as the weight function, and run the algorithm for each class using this function. Figures 2 and 3 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 node as the initial location[2]. Each class of algorithm was run on the same set of 500 graphs.

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



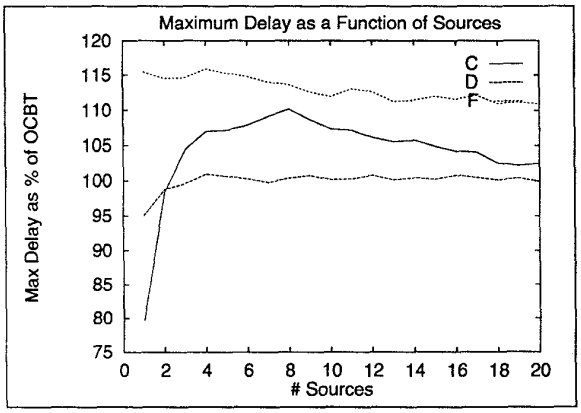
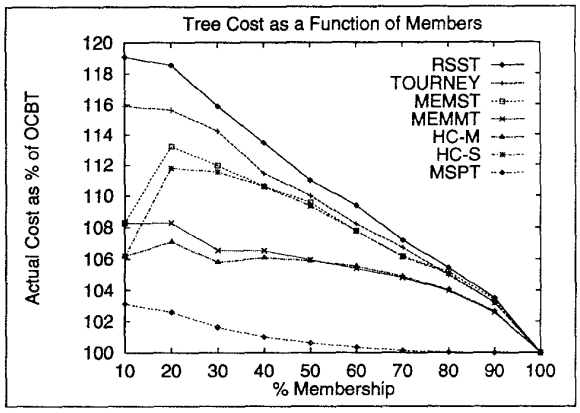


Figure 3: Performance of Classes using a List of Sources



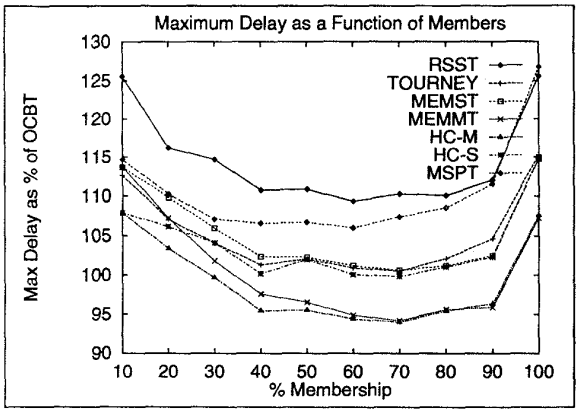


Figure 4: Effects of Group Size on Proposed Algorithms

For the relationships between classes A through D, these results only point out what was already intuitive: A and C gives 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 these graphs 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.

4.5 Analysis of Proposed Algorithms

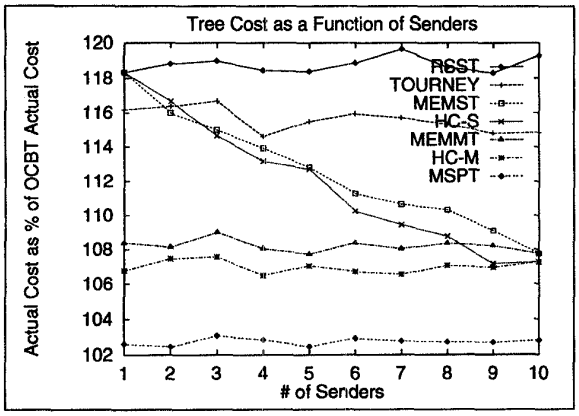
Now that we have analyzed the performance of the various algorithm classes and weight functions, we now need to compare the actual center-location methods which have been proposed, since several of them do not fall into the category of algorithms analyzed above.

Figure 4 shows the effects of varying the group size on the proposed algorithms. For simplicity, we have limited these plots to those 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 5 senders. The results for other 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 much lower. Therefore the difference in maximum delay is a higher proportion of the actual value.

It is interesting to note that near 100% member-



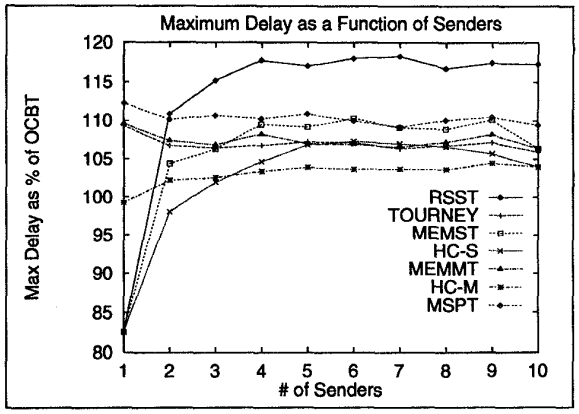


Figure 5: Effects of Number of Sources on Proposed Algorithms

ship, 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 become more random since they do not attempt to optimize for source-to-destination delay.

To investigate the effects of varying the number of sources, we simulated the performance of each algorithm on 50-node graphs with 10 members and 1 to 10 sources. Again, the results were similar for other parameters. Figure 5 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 due to their use of 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 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.

Now that we have seen the effects of varying the network conditions on the performance of the algorithms, it is interesting to see where each lies on the cost vs. delay axes. Figure 6 shows the results from

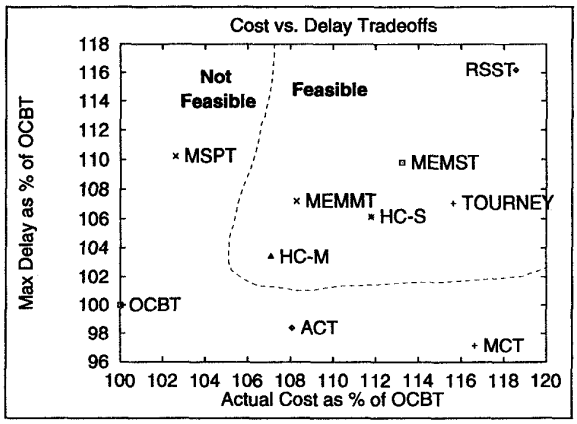


Figure 6: Performance of Proposed Algorithms

the same simulation used for figure 4 for 50 nodes, 5 senders, and 10 members in the multicast group (20% membership).

In this plot, we also include the performance of Wall's ACT and MCT algorithms for comparison as well. We remind the reader that each of OCBT, MSPT, ACT, and MCT are infeasible for general usage in the MBone today, either because they require the ability to compute actual tree costs, or because they require every node in the network to 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 3, once the center location has been determined for small groups with dynamic membership, the cost and delay will degrade towards random placement until the center-location

algorithm is run again. This effect may be less significant for large groups at steady state.

5 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 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 which we feel to be more applicable to real-world networks than existing heuristics which require knowledge of the complete network topology.

Simulation results of 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 both cost and delay. Of those using a list of sources, HC-S provides the best results.

Our simulations used random graphs. We have also run simulations of the performance of these algorithms over hierarchical graphs which more closely resemble real networks, and of the rate at which tree cost degrades as members join and leave a group. These results will be available in [12].

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.

References

[1] Michael R. Macedonia and Donald P. Brutzman. MBone provides audio and video across the Internet. *IEEE Computer*, April 1994.

[2] Michael R. Garey and David S. Johnson. Computers and intractability: A guide to the theory of NP-completeness, June 1988.

[3] Liming Wei and Deborah Estrin. A comparison of multicast trees and algorithms, September 1993. Technical Report USC-CS-93-560, Computer Science Department, University of Southern California.

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

[5] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An architecture for wide-area multicast routing. In *Proceedings of the ACM SIGCOMM*, August 1994.

[6] Tony Ballardie, Paul Francis, and Jon Crowcroft. An architecture for scalable inter-domain multicast routing. In *Proceedings of the ACM SIGCOMM*, September 1993.

[7] Matthew Doar and Ian Leslie. How bad is naive multicast routing. In *Proceedings of the IEEE Infocom '93*, 1993.

[8] Shridhar B. Shukla, Eric B. Boyer, and J. Eric Klinker. Multicast tree construction in network topologies with asymmetric link loads, September 1994. Technical Report NPS-EC-94-012, Naval Postgraduate School.

[9] Robert Voigt, Robert Barton, and Shridhar Shukla. An tool for configuring multicast data distribution over global networks. In *Proceedings of INET*, 1995.

[10] John Moy. OSPF version 2, October 1991. RFC-1247.

[11] Bernard M. Waxman. Routing of multipoint connections. *IEEE J. Selected Areas in Communications*, 6(9), December 1988.

[12] David G. Thaler and Chinya V. Ravishankar. An evaluation of distributed algorithms for center-location. Submitted to *IEEE J. Selected Areas in Communications*.