Destination-Driven Routing for Low-Cost Multicast

Anees Shaikh, *Student Member, IEEE*, and Kang Shin, *Fellow, IEEE*

***Abstract*—We present a destination-driven algorithm that optimizes for applications, such as group video or teleconferencing, that require multicast trees with low total cost. The destination-driven algorithm uses a greedy strategy based on shortest-path trees and minimal spanning trees but biases routes *through* destinations. The performance of the algorithm is analyzed through extensive simulation and compared with several Steiner tree heuristics and the popular shortest-path tree (SPT) method. The algorithm is found to produce trees with significantly lower overall cost than the SPT while maintaining reasonable per-destination performance. Its performance also compares well with other known Steiner heuristics. Moreover, the algorithm does not suffer from high complexity common to most Steiner tree heuristics and builds a route by querying only incident links for cost information.**

***Index Terms*—Communication system routing, computer networks, trees (graphs).**

I. Introduction

THE recent emergence of multimedia and collaborative computing in distributed environments provides an incentive to system designers to include communication support for these applications. A prevalent pattern in such environments is multicast (one-to-many or many-to-many) communication. In this mode, a single source node (or group of source nodes) sends identical messages simultaneously to multiple destination nodes in a point-to-point network. Single-destination, or unicast, and broadcast messaging to the entire network are both special cases of multicast. Multimedia applications are envisioned to benefit most from multicast communication capabilities. Some frequently cited examples include video or teleconferencing, distributed databases, distributed games, mass mailings, and video-on-demand services [1]–[3]. In a video-on-demand application, for instance, a single server provides a one-to-many transmission for customers who join the same movie at approximately the same time. A video conference, on the other hand, is likely to be a many-to-many transmission with multiple parties wishing to transmit and receive.

A fundamental issue in multicast communication is how to determine an efficient message route (multicast routing). Tree construction is a commonly used approach in solving the multicast routing problem. Popularity of the tree-based approach arises from the ability to potentially share many links in transmitting the message to the destination set. Also, multicast trees minimize data replication; messages need only be replicated at forking nodes. This differs from multicast achieved through multiple unicasts where every unicast requires a copy of the message. Using multiple unicasts may result in many copies of the same message traversing the same network links.

Generally, multicast trees fall into two categories, namely shortest-path trees (SPT's) and group shared trees [4]. SPT's for a single multicast may consist of many distinct trees, one for each source. That is, the shortest path to each member of the multicast is different depending on who is originating messages. Hence, SPT's are often referred to as source-specific SPT's. This type of tree is currently used in the distance-vector multicast routing protocol (DVMRP) for Internet multicast traffic on the virtual multicast backbone (MBone) network [5]–[7]. Another example of multicast SPT's is the multicast extensions for open shortest path first OSPF (MOSPF) which uses distributed link state and Dijkstra's algorithm [8] to calculate shortest paths [9]. The primary advantage of SPT's is, of course, the minimal delay to each destination. This feature makes SPT's desirable for interactive applications such as video conferencing which are characterized by very high data rates a need for timely delivery. SPT's drawback is that MOSPF and DVMRP require broadcasting of membership information and data packets, respectively. As an internetwork grows in size, the broadcast behavior of these routing methods can result in poor scalability. On the other hand, some multicast applications may have very dense groups for which the broadcast may not represent a substantial penalty [10]. An intuitive definition of a dense multicast group may be one where the number of internetwork LAN's that have group members is a high proportion of the total number of LAN's in the Internet.

To combat the scalability problems, a relatively new shared tree method, classified as the center-based tree, has been proposed. A center-based tree has one (or more) central router node with branches that emanate outward to destinations. The tree is shared per group rather than per (source, group) pair, and its formation is initiated by receivers once the central node is known. Examples of center-based trees are the core-based tree (CBT) [11] and the shared-tree mode of the protocol independent multicast (PIM)[1] architecture [10]. The drawback of center-based and other shared trees is that the same tree is used for all members of the multicast. This means that multiple sources in the group will be transmitting over the same set of links, leading to high traffic concentrations on those links.

Manuscript received January 23, 1996; revised August 6, 1996. This work was supported in part by the Office of Naval Research (ONR) under Grant N00014-1-0229 and by the National Science Foundation under Grant MIP-9203895.

The authors are with the Real-Time Computing Laboratory, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: ashaikh@eecs.umich.edu; kgshin@eecs.umich.edu).

Publisher Item Identifier S 0733-8716(97)02273-7.

[1] PIM is notable in that it supports shared center-based trees for low data rate sources or sparse multicast groups and also provides a mechanism to switch over to a SPT mode when low delay is important or the multicast group is densely distributed.

Traffic concentration may not be an issue, however, for groups which are sparsely distributed or applications with low data rates. The advantage of center-based trees is that they do not exhibit the broadcast behavior of SPT's and thus do not tax routers uninterested in the multicast with data or group information.

The goals addressed by various multicast tree types fall into two general categories [12], [13]. The first is to minimize the delay to *each* destination, which is effectively what the SPT achieves. Another optimization goal is to minimize the overall cost of the multicast tree.[2]

Low delay trees (SPT's) are important in providing good service to destinations which represent individual customers who have paid for some multicast service. Optimizing for cost of the entire multicast, however, requires a more global view. A company using a multicast video conferencing service, for example, may wish instead to minimize the overall cost of the transmission rather than the delay (or cost) to each individual site. Moreover, from a network management point of view it may be more important to keep the total cost of a multicast session as low as possible rather than minimizing cost to each individual destination. These two conflicting goals have been studied together in other graph theoretic and networking research [12], [14].

The tree cost itself may be a measure of a number of possible parameters. Our approach avoids explicitly tying cost to a particular network parameter. Cost may be inversely proportional to link capacity or proportional to distance. Link cost may also reflect the actual or anticipated congestion or error rate. It could also measure buffer space or channel bandwidth requirements. Still other networks may use an estimate of packet transmission delay as the link cost [3] or assume that cost reflects both delay and installation cost, for example [14]. Recent work has split cost and delay, in some cases trying to provide low cost while meeting some delay constraint to each destination [13], [15]. In these cases, cost is most often associated with bandwidth consumption although the relationship between delay and cost is rarely explained in the context of simulation-based studies. The focus of the comparisons here is more general.

Trees that try to reduce overall cost generally achieve their objective through sharing of links and minimal forking. SPT's, on the other hand, pay no heed to the amount of forking; their primary goal is to find the least costly path to each individual destination, even if every path is disjoint.

Cost of the tree is generally represented as the sum over the costs of each link in the multicast tree. The problem of finding the least total cost multicast tree is the Steiner tree problem in networks, and is formulated as follows [16].

GIVEN: An undirected network $G = (V, E, w)$ and a nonempty set of destinations D, where V and E are the set of vertices and edges of the network, respectively, and w is a cost function associated with each edge in G.

FIND: A subnetwork $T_G(S)$ of G such that:

- there is a path between every pair of nodes in $D \subset V$;
- the total cost, $\sum_{e \in T_G(D)} w(e)$, is minimized over all possible trees.

The vertices included in the final solution that are not members of D are called Steiner nodes. Finding a Steiner tree is known to be NP-complete in the most general case [17].

Steiner tree heuristics, though providing near-optimal results, are sometimes dismissed as too difficult to implement in practical protocols or too computationally intensive [4], [7]. In fact, most of the proposed heuristics for the Steiner tree require global network information and can be quite complex relative to SPT schemes [15], [18], [19]. Fortunately, there are several versions of Steiner tree heuristics that may be executed in a distributed fashion or using information only from nearby nodes [12], [20], [21], though these are not without their limitations [22].

In this paper we present a Steiner heuristic that is comparable in complexity to SPT's and suitable for similar dense-mode multicast groups. It uses a greedy strategy combining Dijkstra's shortest path and Prim's minimal spanning tree [23] algorithms. It reduces tree cost by biasing toward paths that pass *through* destination nodes. We show through simulation that the heuristic offers competitive performance with some other known Steiner tree heuristics but at a much lower computational cost.

We explore the ability of the algorithm to minimize an overall tree metric (cost) as opposed to a destination-specific metric (as done in SPT's). We find that it demonstrates significant improvement over the popular SPT's in terms of overall cost while at the same time providing reasonable relative performance in average per-destination cost and maximum cost to each destination (for which SPT is optimized).

In the next section we present an expanded description of the algorithm. Sections III and IV present simulation results comparing our algorithm with other Steiner heuristics and SPT's, respectively. In Section V, we summarize our results and briefly describe the future directions of this research.

II. Destination-Driven Multicast Routing

Our multicast routing algorithm arises from the observation that Prim's spanning tree algorithm to optimally broadcast to all nodes in a graph and Dijkstra's shortest path algorithm for unicast both use essentially the same greedy strategy. We modify these to distinguish between destination and nondestination nodes [24]. This distinguishing information is used to give "preference" to destination nodes.

A. Algorithm Overview

The shortest path algorithm chooses routes by measuring cost to each destination in terms of individual accumulated cost. That is, SPT chooses links from the source to each individual destination so that they add the smallest amount to the current accumulated cost for the destination. In our approach, however, we reset cost to zero at destination nodes so that they appear as new "sources." This causes the cost to a node to appear small when measured from a destination. The reason for this is that any nodes reached from a destination node incur only an incremental additional cost since we must

[2]The introduction of CBT's suggests a third possible optimization goal, namely to reduce broadcast behavior and router state memory requirements.

```
DDMC (G, s, D)
1  for each vertex v ∈ V do
2       d[v] = ∞
3       π[v] = NIL
4  d[s] = 0
5  S = ∅
6  Q = V
7  while Q ≠ ∅ do
8       u=Pop-Min(Q)
9       S = S ∪ {u}
10      for each vertex v ∈ Adj[u]
11           if d[v] > I_D(u)d[u] + w(u, v)
12                if v ∉ S
13                     d[v] = I_D(u)d[u] + w(u, v)
14                     π[v] = u
```

Fig. 1. DDMC routing algorithm.

absorb the cost for reaching the destination anyway. In this way, paths are biased toward those that run through destination nodes. As a result we refer to the algorithm as *destination-driven multicast* (DDMC).

For our purposes, the communication network is modeled as usual, using an undirected graph $G = (V, E)$ where V is a set of host or router nodes and E is the set of communication links. We assume that the cost $w(u, v)$ is nonnegative for each link $(u, v) \in E$. Given a source $s \in V$ and a set of destinations $D \subset V$ such that $s \notin D$, a multicast route is a rooted subtree of the graph G whose root is s, which contains all nodes from D, and whose leaves consist of nodes from D. Note that s need not be the only sender; since this algorithm produces a shared multicast tree, any member of $s \cup D$ will transmit using this same tree. Consequently, the tree has the same benefits and drawbacks common to shared trees as discussed in Section I.

To distinguish nodes as being in the destination set, we define an indicator function, I_D as follows.

Definition: Given a set $D \subset V$, the indicator function I_D: $V \mapsto \{0, 1\}$ of D is defined as $I_D(u) = 0$ if $u \in D$, and 1 if $u \notin D$.

B. Algorithm Operation

The algorithm executes in a manner very similar to Prim's minimum spanning tree algorithm [23]. We follow the basic framework for tree construction shown in [25] and proceed in three basic steps as shown below. A more detailed description of the algorithm may be found in [24]. Pseudocode for DDMC is given in Fig. 1.

1) *Initialization* involves setting parent pointers indeterminate and cost estimates for each node to infinity, except the source which receives cost estimate zero. In addition, we create a priority queue, Q, and fill it with all of the nodes. The queue is keyed on cost estimates where the lowest cost has the highest priority.
2) Next we repeatedly *select* the highest priority vertex from Q, add it to the tree, and
3) *relax* all of its outgoing edges. Relaxation uses the incident link cost, the current cost estimate of the neighboring node and the indicator function, to reset the cost and parent pointers, if necessary.

The indicator function is used in line 11 so that the incremental distance is zeroed if u is a destination. That is, from a destination, the only cost incurred is the additional link cost, not the cost accumulated along the path from the source. This causes each destination node to behave like a new "source." As future work, it seems reasonable to use the indicator function as a tunable bias, depending on the goals of a particular multicast application. Currently, the indicator zeros the cost estimate for a destination node; it may be useful to provide finer control over revisions to destination cost estimates to change the characteristics of the tree.

The condition in line 12, and the fact that we consider each node only once, ensure that cycles in the tree are avoided by preventing resetting of parent pointers for nodes whose routes have already been determined. In keeping with its goal of simplicity, DDMC executes only one pass through the network using the greedy strategy, rather than iteratively trying to reduce the cost of the tree as in some other approaches [13]. Note that we construct a spanning tree rooted at s for the graph G. A multicast routing tree from s to D is obtained by trimming this tree so that all leaves are destination nodes.

The correctness[3] of the algorithm follows almost directly from Prim's algorithm. The primary difference is the distinction given to destination nodes which causes the distance estimate of a node, u, to reflect whether or not its parent is a destination (line 11). Thus the order in which edges are added to the tree will be different than in Prim's algorithm. The asymptotic running time of the algorithm is $O(e \log v)$, where $e = |E|$ and $v = |V|$. This assumes that the priority queue Q is implemented as a binary heap and that the test for membership in S (line 12) can be made in constant time. Some improvement is possible if Fibonacci heaps are used for the priority queue [25].

Fig. 2 shows an example of the type of tree constructed by the DDMC algorithm along with the solution formed by SPT. The square designates the source node while destinations are double circles. Note that the DDMC paths all pass through destination node 2 en route to the other destinations. The SPT route, however, finds a shorter path by forking from the source. As a result the overall tree cost of the DDMC route is lower although the average per-destination cost and the maximum cost are identical in this case.

C. Extensions for Distributed DDMC

Efficient construction of a multicast tree requires a distributed version of the DDMC algorithm. Related work in distributed minimum spanning trees and distributed multicast path setup may be used to address this issue. The classic spanning tree algorithm of Gallager *et al.* [26], for example, proceeds as fragments of the spanning tree (initially single nodes) join together asynchronously via their least cost outgoing links to form larger fragments. Nodes within each fragment determine their local minimum outgoing edge independently and report the result to a fragment leader who initiates combination with another fragment.

[3] By correctness, we mean that the algorithm terminates with a tree connecting all nodes in $D \cup s$.

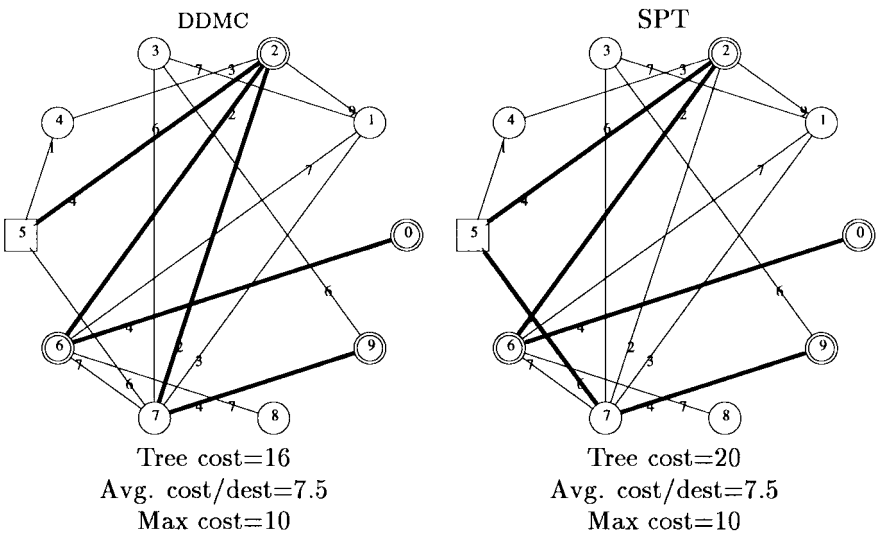

Fig. 2. Example multicast trees constructed by DDMC and SPT. Edge weights represent link costs. Links used in the final multicast tree are shown in boldface.

More recently, Bauer and Varma described a similar procedure for distributed versions of two well-known Steiner tree heuristics [21]. The distributed version of the shortest path Steiner heuristic (SPH) proceeds very similarly to the way DDMC would proceed. A single source fragment expands into the multicast tree except that in the case of DDMC, the fragment is grown by adding a single node over the least cost link (similar to Gallager's) but biased by destination nodes. The destination information may be multicast to the new fragment nodes similar to the discovery phase described in [21] in which the fragment nodes update their notions of cheapest outgoing links. Unlike distributed SPH, however, a distributed version of DDMC would not have to decide what the closest fragment is based on shortest paths to all other destinations. The "greedy" nature of the algorithm results in a need to query only incident links to pick the way in which the fragment should be expanded.

III. DDMC as a Steiner Tree Heuristic

Though we are interested in the ability of DDMC to meet the conflicting goals of lowering overall cost while considering per-destination costs, it is also important to consider its performance as a Steiner tree heuristic. Recall that heuristics for the Steiner tree have the primary goal of minimizing overall tree cost only. In this section we present a simulation-based study comparing DDMC's performance to three other heuristics that vary in performance and computational complexity.

A. Heuristics for the Steiner Tree

Given the absence of a polynomial time algorithm for the general Steiner minimum tree problem, investigation of efficient, quality heuristics remains an active research area. The problem has been addressed both in a multicast communication context [12], [27], and as a purely graph theoretic problem [16], [19]. Most of the well-known heuristics fall into the class known as path-distance heuristics since they approach the problem by iteratively enlarging partial solutions using shortest paths. The three primary path-distance approaches are the shortest-path heuristic (SPH) [28], average distance heuristic (ADH) [29], and distance network heuristic (DNH)[4] [18]. In our study we consider only ADH and SPH, however, since previous studies have shown that they generally outperform DNH (although their dominance is not strict) [16], [29]. We also include, for more practical reasons, comparison with the simple pruned minimum spanning tree heuristic (MST) [28]. MST executes a minimum spanning tree algorithm (e.g., Prim [23]) and then prunes nondestination nodes with degree 1, resulting in complexity $O(e \log v)$[5] [16]. While MST performs relatively poorly on average, it has the advantage of having low computational complexity and requires cost information only from neighboring nodes as it proceeds. DDMC shares these same advantages.

The general operation of ADH proceeds by joining together two subtrees at a time through a central node via shortest paths. Finding a central node amounts to finding a node with the minimum average distance to the set of current subtrees. The algorithm begins with $|D|+1$ subtrees consisting of a single destination each (plus the source) and ends with a solution consisting of a single tree. Computation is dominated by the calculation of shortest paths between all nodes in the network leading to its worst-case time complexity $O(v^3)$.

The SPH heuristic begins with an arbitrary node in the multicast destination set, for example the source, and repeatedly adds the remaining members to the tree via shortest paths in order of closest first. Most of the computation time in SPH is spent calculating shortest paths from each multicast member to all other nodes. Thus its time complexity is $O(|D|v^2)$. Note that SPT is not equivalent to the SPH heuristic. SPT is an algorithm for finding a tree of lowest cost paths to all individual destinations in the network. SPH, on the other hand, uses shortest paths to approximate a Steiner minimal tree connecting a subset of nodes in the network with low total tree cost.

[4]DNH is often referred to as the KMB algorithm after (one set of) its developers.

[5]This assumes that Fibonacci heaps, as mentioned in Section II-B, are not used.

TABLE I
COMPUTATIONAL COMPLEXITY FOR STEINER HEURISTICS

ADH	SPH	DDMC	MST
$O(v^3)$	$O(\|D\|v^2)$	$O(e \log v)$	$O(e \log v)$

Both ADH and SPH require shortest path calculations through the entire network, which in turn requires global link cost information. Time complexity for each of the algorithms considered is summarized in Table I. Recall that in general $|D| < v$ and $e < v^2$.

The use of limited link cost information does not come without a price, however. Due to its low complexity and use of limited knowledge MST, for example, has the drawback that its theoretical worst-case performance is up to $v - |D|$ times worse than the optimal solution. In contrast, ADH and SPH produce trees within a factor of two times optimal in the worst case [16]. Moreover, it was shown in [22] that greedy heuristics, such as MST and DDMC which use only local information, can be no better than $2n/3$ times worse than optimal in the worst case. We do not present a derivation of the worst-case error ratio of DDMC here but we suspect that it is similar to (although likely better than) MST. Although we are studying this problem in a networking context, we are more interested in DDMC's performance in practice, where the worst-case rarely occurs. We show below that it compares well on average with the other heuristics.

B. Simulation Setup

The studies were done using a modular, extensible simulator, *STsim* programmed in C and run on Sun Sparc 20 workstations. *STsim*'s interface, developed using Tcl and the Tk toolkit, is shown in Fig. 3. It allows rapid generation of an experiment set with a variety of network parameters, data collection options, and multicast routing algorithms.

We ran a set of 1000 randomly generated experiment instances for each data point and report the average over all runs. A predetermined set of random seeds assured that all algorithms received identical networks, destination sets, and multicast group sizes as input. Networks are generated in the usual way, by considering all pairs of nodes and randomly deciding whether a link exists between them. A parameter was also introduced to control the average node degree when generating networks (where unspecified we used a default average node degree of three). Integer costs within a small given range were assigned randomly to each link based on a uniform distribution. For the experiments discussed here, destinations were distributed uniformly through the network. The simulations consisted of three basic phases: network and destination set generation, algorithm execution, and multicast tree analysis.

C. Simulation Results

We find that DDMC performs surprisingly well against other, more complex, Steiner heuristics. We measured the ability of each algorithm to minimize the total tree cost as destination set size increases and as network size increases. Observing performance while changing group size provides a notion for how well the algorithms perform relatively for both sparse and dense multicast groups. We found that increasing network size exhibits trends similar to those when increasing destination set size. Fig. 4 summarizes the results for simulations in a fixed size (64-node) network.

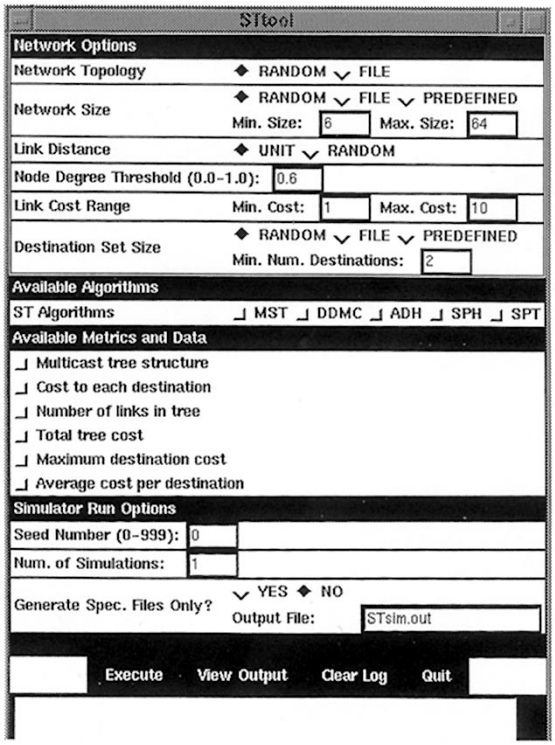

Fig. 3. STsim user interface for generating routing experiments.

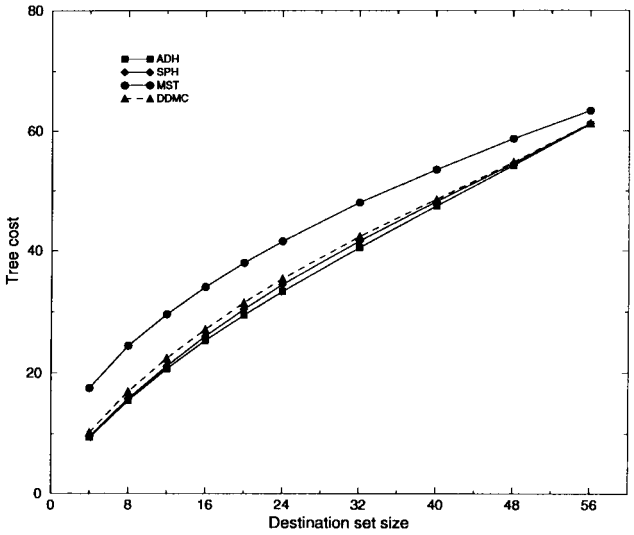

Fig. 4. Total tree cost as group size increases in a 64-node network with a small link cost range.

In the interest of exploring performance in more heterogeneous networks, where link costs may vary widely, we also ran simulations on more variable networks. In Fig. 5, the link costs ranged over two orders of magnitude. We show the maximum size of the 95% confidence interval for each of the algorithms in Table II. The interval is narrow and within the same range for each of the algorithms considered.

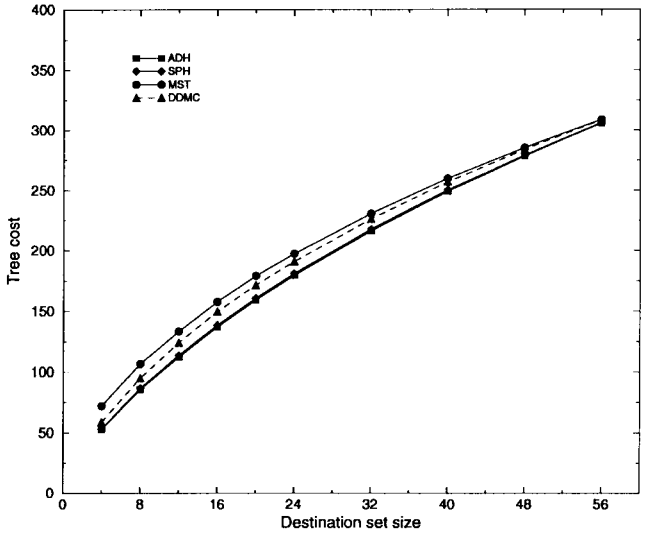

Fig. 5. Total tree cost as group size increases in a 64-node network with widely variable link costs.

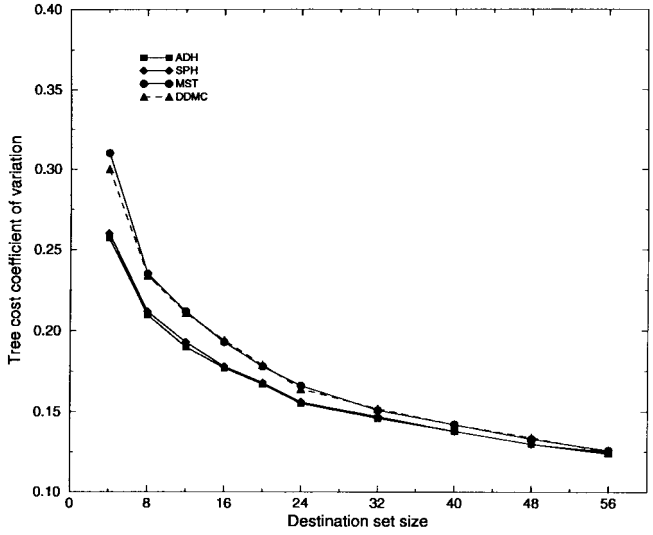

Fig. 6. Coefficient of variation for total tree cost as group size increases in a 64-node network.

TABLE II
CONFIDENCE INTERVAL SIZES FOR MEAN TREE COST WITH 95% CONFIDENCE LEVEL

link cost range	ADH	SPH	DDMC	MST
1–50	2.33	2.33	2.37	2.38
1–100	4.72	4.74	4.77	4.83

As shown in Fig. 4, ADH consistently has the lowest cost of the algorithms studied. Not surprisingly, it also is the most complex of the heuristics. Note, however, that both SPH and DDMC perform only slightly worse. While DDMC gives up a slight performance margin, it is characterized by very low complexity and does not rely on global shortest path calculations. MST, as expected, fares worse than the other algorithms. Note that when the multicast group is very dense, however, it performs well since MST computes a minimum broadcast tree spanning all nodes. Trends are similar for the wide link cost range (1–100) shown in Fig. 5; the difference in performance between DDMC and other algorithms remains approximately equal to the smaller link costs case. MST, however, does perform slightly better in the more variable networks.

Finally, we investigate the variability of the solutions provided by DDMC relative to the other heuristics. Fig. 6 shows the relative coefficients of variation[6] for the results presented in Fig. 5. Although the variability of the solutions found by DDMC is higher than ADH or SPH, the difference is not large. In general, the variability displayed by all the algorithms is quite low.

IV. COMPARISON WITH SHORTEST PATH TREES

In this section we compare the DDMC algorithm against SPT's across several performance metrics. The results of our simulation-based study show that DDMC constructs low-cost trees that in general use fewer links than SPT's. In addition, performance is reasonably close to SPT's in local metrics such as average per-destination cost and maximum destination cost.

The following graphs show how the performance of DDMC compares to SPT as the multicast group size changes and as average node degree in the network increases. As mentioned earlier, changing group size gives an indication of how well algorithm performance scales as groups grow denser. We also study the effect of increasing node degree to gain some insight into how the algorithms perform when more routing options are available.

Fig. 7 shows the overall tree cost and average cost to each destination as the group size grows in a fixed size (64-node) network. The total tree cost is simply the sum of all edge costs in the multicast tree. Average cost per destination is calculated as the sum of the costs to each individual destination divided by the total number of destinations. It is apparent from the graph that DDMC's overall tree cost is much less than that of SPT. In addition, DDMC's tree cost grows at a significantly lower rate than SPT's, indicating that tree cost scales much better with DDMC. Also, DDMC grows only slightly faster than SPT in terms of cost to each destination and while DDMC does not match SPT performance, the difference is much smaller than with total tree cost.

In Fig. 8, we show the coefficients of variation for total tree cost and average cost per destination. As expected, DDMC's variability is higher than SPT for cost per destination but smaller for total tree cost. When link costs vary over a smaller range, the variability is only slightly less for each of the curves.

Another metric of interest is the maximum of all per-destination costs; that is, the maximum cost suffered by any single destination in the multicast tree. It may be important for some environments or applications to know which destination is incurring the highest cost. We show the relative performance of DDMC to SPT by plotting the cost ratios for maximum cost and average destination cost as group size grows in Fig. 9. As expected, the relative maximum destination cost of DDMC is higher than SPT's. The relative average destination cost is also higher, as mentioned above, but the relative difference does

[6] Coefficient of variation is defined as the ratio of standard deviation to the mean.

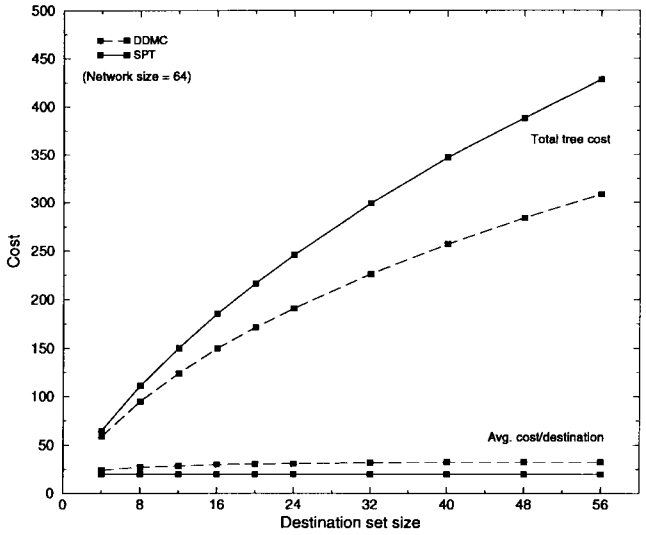

Fig. 7. Total tree cost and average cost per destination as group size increases in a 64-node network.

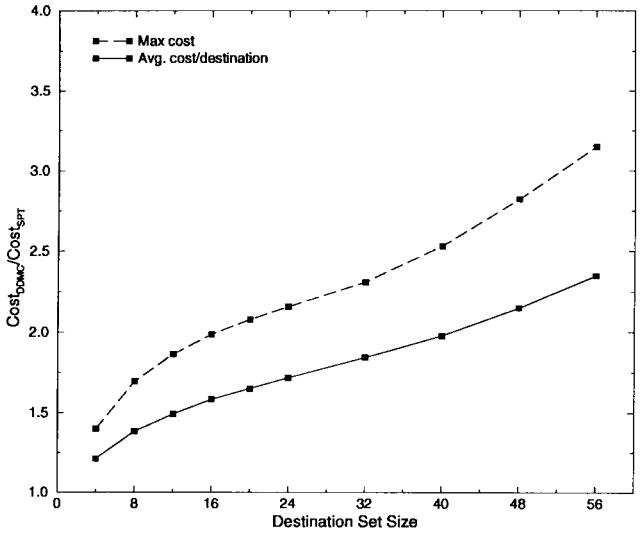

Fig. 9. Ratio of maximum cost and average cost per destination for DDMC to SPT as group size increases in a 64-node network.

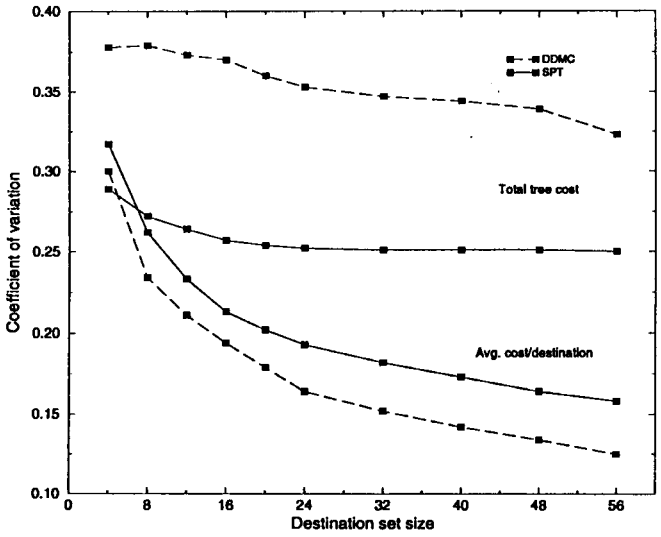

Fig. 8. Coefficient of variation for total tree cost and average cost per destination as group size increases.

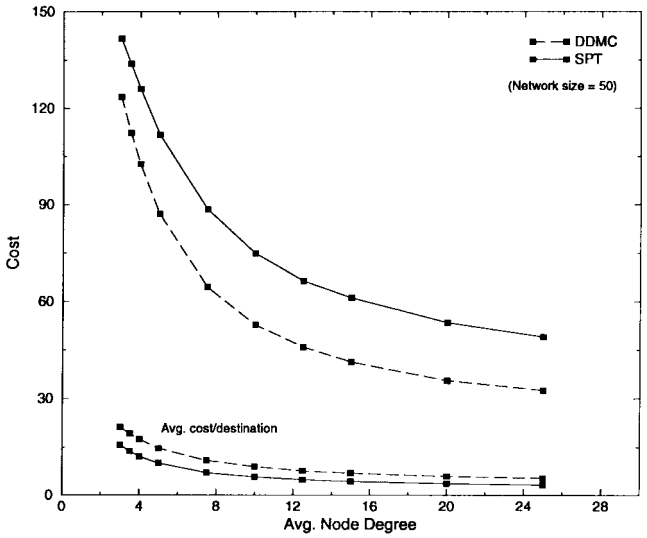

Fig. 10. Total tree cost and average cost per destination as average node degree increases in a 64-node network.

not grow as quickly as with maximum cost. This is because DDMC is geared toward combining paths through destinations to maintain an overall low cost. This may occasionally require some tree paths to follow a somewhat more circuitous route to a destination than the absolute shortest route. If the tolerable cost to a destination is greater than the maximum achieved by DDMC, however, then DDMC will provide a very low-cost tree that still meets per-destination cost constraints.

The effect of increasing node degree is shown in Fig. 10. The node degree might reflect the complexity and cost of network switches and it is interesting to examine how SPT and DDMC perform with a range of node degree constraints. In these experiments we use a random number of destinations for each of the simulation runs and present the average cost at each data point. The total tree cost is quite high for both algorithms when the node degree is low since there are far fewer alternative links available for forming the multicast tree. As the number of alternatives grows, it becomes easier for both algorithms to find cheaper routes and the cost decreases. Again, DDMC's total tree cost is significantly lower than SPT's. This is due to the fact that given more link options, SPT will fork more frequently as it identifies cheap individual paths. SPT does outperform DDMC in average cost per destination but, as with increasing set size, only slightly. Moreover, as the node degree increases, the difference between the two algorithms in terms of average destination cost grows smaller.

As a final comparison, Fig. 11 shows the trend in number of links used to construct the multicast tree as the group size grows. Here, as expected, the DDMC algorithm fares better than SPT's because it is less likely to take disjoint paths to destinations, thus conserving link resources. This has the effect of decreasing network congestion as well as cost. The difference is not significant for very large group sizes, however, because DDMC uses virtually the same paths as

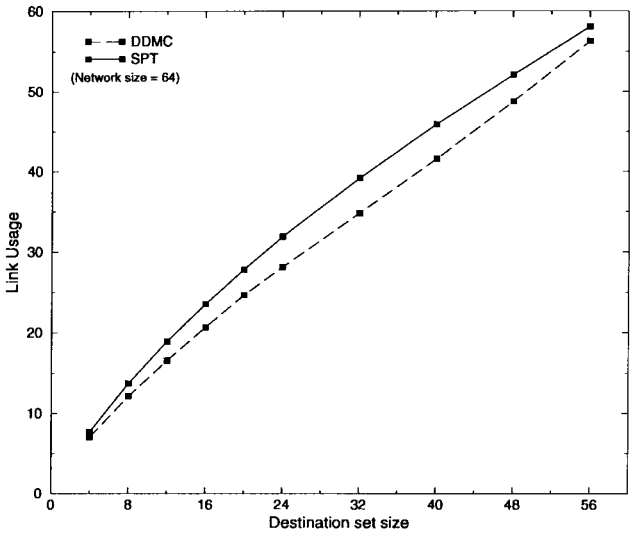

Fig. 11. Link usage as multicast group size grows.

SPT's since the destination density is high. DDMC diverges from the shortest-path only where there is an intervening destination that receives path "preference." When the group is very dense, most shortest paths will pass through destination nodes anyway, thus making the difference in link usage small.

V. Conclusion

In this paper we presented an efficient algorithm for multicast routing that biases toward routes through destination nodes. The underlying philosophy is that cost for reaching a destination must be absorbed. Thus it makes sense to use paths through destinations since cost is only incremental after reaching a destination node. This DDMC algorithm is optimized for low overall tree cost to benefit applications such as corporate video conferencing where a primary concern is to keep costs down over the entire transmission.

Currently, the most widely used multicast tree type is the shortest-path tree due to its low per-destination cost or delay. We have shown through simulation that DDMC significantly outperforms SPT's in terms of controlling overall tree cost. At the same time, DDMC does not suffer much in per-destination metrics. For applications in which low tree cost is important, DDMC excels. In addition, since DDMC's greedy strategy is based on minimal spanning tree and shortest path algorithms, it is not as complex as many Steiner tree heuristics. Despite this, simulation studies show that on average its performance is comparable with more complex heuristics. Moreover, the algorithm requires each node to query only its incident links for cost information, thus avoiding the drawback of requiring global cost information.

We are planning additional simulations to gauge DDMC's scalability. Since DDMC is geared toward controlling overall network resources, it would be interesting to investigate the effect of multiple trees in a network, for example. Another particular feature is the incorporation of dual metric handling, for example, cost and delay. Our study treated cost as a general parameter; it is not yet clear how cost should be represented and in what way it may be related to delay in simulation studies. Also, we have not addressed the issue of dynamic groups where members may join and leave. It is possible, however, to use either the naive approach [30] adding shortest paths to new nodes or the recently proposed ARIES algorithm [31] which selectively repairs parts of the tree after a given number of membership changes. Finally, further work is necessary to analytically derive the worst-case error ratio of the solutions provided by DDMC to the optimal solution.

Additional goals include the development of an actual protocol using DDMC. We gave a sketch of how DDMC may be used in a distributed fashion following the approaches described in [21] and [26] of expanding tree fragments and having nodes independently calculate their least costly outgoing links. More work is needed to provide a notion of message complexity and convergence time for such a scheme. It may be more beneficial, however, to incorporate DDMC into a protocol architecture such as PIM as an alternative for applications which require low overall tree costs.

It is clear from the wide variety of anticipated multicast applications that no single tree type can satisfy requirements of all of them. DDMC is a practical, low-complexity algorithm for applications that require reasonable per-destination cost and low overall cost.

Acknowledgment

The authors wish to thank J. Rexford for her valuable suggestions on an earlier draft of this paper and the referees for their insightful and helpful comments.

References

[1] B. Rajagopalan, "Reliability and scaling issues in multicast communication," in *Proc. ACM SIGCOMM*, Baltimore, MD, Oct. 1992, pp. 188–198.
[2] W. D. Sincoskie, "System architecture for a large scale video-on-demand service," *Computer Networks ISDN Syst.*, vol. 22, no. 2, pp. 155–162, Sept. 1991.
[3] Y. Tanaka and P. C. Huang, "Multiple destination routing algorithms," *IEICE Trans. Comm.*, vol. E76-B, no. 5, pp. 544–552, May 1993.
[4] L. Wei and D. Estrin, "The tradeoffs of multicast trees and algorithms," in *Proc. Int. Conf. Comp. Comm. Networks*, San Francisco, CA, Sept. 1994.
[5] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LAN's," *ACM Trans. Comp. Syst.*, vol. 8, no. 2, pp. 85–110, May 1990.
[6] H. Eriksson, "MBone: The multicast backbone," *Comm. ACM*, vol. 37, no. 8, pp. 54–60, Aug. 1994.
[7] C. Huitema, *Routing in the Internet.* Englewood Cliffs, NJ: Prentice-Hall, 1995, ch. 11, pp. 235–260.
[8] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathemtik*, vol. 1, pp. 269–271, 1959.
[9] J. Moy, "Multicast routing extensions for OSPF," *Comm. ACM*, vol. 37, no. 8, pp. 61–66, Aug. 1994.
[10] 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*, London, UK, Aug. 1994, pp. 126–135.
[11] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees: An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, Ithaca, NY, Sept. 1993, pp. 85–95.
[12] K. Bharath–Kumar and J. M. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Trans. Comm.*, vol. 31, no. 2, pp. 343–353, Mar. 1983.
[13] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A source-based algorithm for delay-constrained minimum-cost multicasting," in *Proc. IEEE INFOCOM*, Boston, MA, Apr. 1995, pp. 377–385.
[14] S. Khuller, B. Raghavachari, and N. Young, "Balancing minimum spanning trees and shortest-path trees," *Algorithmica*, vol. 14, no. 4, pp. 305–321, 1995.

[15] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 286–292, June 1993.
[16] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem.* New York: North-Holland, Number 53 in Annals of Discrete Mathematics, 1992.
[17] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–103.
[18] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees in graphs," *Acta Informatica*, vol. 15, pp. 141–145, 1981.
[19] P. Winter, "Steiner problem in networks: A survey," *Networks*, vol. 17, pp. 129–167, 1987.
[20] D. W. Wall, "Mechanisms for broadcast and selective broadcast," Ph.D. dissertation, CSL-TR-80-190, Stanford Univ., June 1980.
[21] F. Bauer and A. Varma, "Distributed algorithms for multicast path setup in data networks," *IEEE/ACM Trans. Networking*, vol. 4, no. 2, pp. 181–191, Apr. 1996.
[22] J. M. Jaffe, "Distributed multi-destination routing: The constraints of local information," *SIAM J. Comp.*, vol. 14, no. 4, pp. 875–888, Nov. 1985.
[23] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Sys. Tech. J.*, vol. 36, pp. 1389–1401, 1957.
[24] A. Shaikh, S. Lu, and K. Shin, "Localized multicast routing," in *Proc. IEEE GLOBECOM*, Singapore, Nov. 1995, pp. 1352–1356.
[25] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Cambridge, MA: MIT Press (New York: McGraw-Hill), 1990.
[26] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. Programming Languages and Syst.*, vol. 5, no. 1, pp. 66–77, Jan. 1983.
[27] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Selected Areas Comm.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.
[28] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner tree problem in graphs," *Mathematica Japonica*, vol. 24, no. 6, pp. 573–577, 1980.
[29] V. J. Rayward–Smith and A. Clare, "On finding Steiner vertices," *Networks*, vol. 16, pp. 283–294, 1986.
[30] M. Doar and I. Leslie, "How bad is naïve multicast routing?," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1993, pp. 82–89.
[31] F. Bauer and A. Varma, "ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm," Tech. Rep. UCSC-CRL-95-36, Dept. Computer Engineering, Univ. Calif., Santa Cruz, July 1995.

Anees Shaikh (S'93) received the B.S. and M.S. degrees in electrical engineering from the University of Virginia, Charlottesville, in 1994.

He is currently working toward the Ph.D. degree in computer science and engineering at the University of Michigan, Ann Arbor. His research interests include multicast communication, quality-of-service networking, and fault-tolerant computing.

Mr. Shaikh is a member of Eta Kappa Nu and Tau Beta Pi.

Kang Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is Professor and Director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are investigating various issues related to real-time and fault-tolerant computing. From 1978 to 1982, he was a faculty member of Rensselaer Polytechnic Institute, Troy, NY. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, the Computer Science Division within the Department of Electrical Engineering and Computer Science at University of California at Berkeley, and International Computer Science Institute, Berkeley, CA, IBM T. J. Watson Research Center, and Software Engineering Institute at Carnegie Mellon University. He also chaired the Computer Science and Engineering Division, EECS Department, The University of Michigan, from 1991 to 1994. He has authored or co-authored more than 400 technical papers and numerous book chapters on distributed real-time computing and control, fault-tolerant computing, computer architecture, robotics and automation, and intelligent manufacturing. He co-authored (with C. M. Krishna) *Real-Time Systems* (McGraw-Hill, 1997). He was an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, an Area Editor of the *International Journal of Time-Critical Computing Systems*, and Guest Editor of August 1987 IEEE TRANSACTIONS ON COMPUTERS.

Dr. Shin was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, a Program Co-Chair for the 1992 *International Conference on Parallel Processing*, and served numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991–1993, was a Distinguished Visitor of the Computer Society of the IEEE, In 1987, he received the Outstanding IEEE TRANSACTIONS ON AUTOMATIC CONTROL Paper Award. In 1989, he received the Research Excellence Award from The University of Michigan.