Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks*

H.F. Salama D.S. Reeves Y. Viniotis
Center for Advanced Computing and Communication
ECE Department and CSC Department
North Carolina State University, Raleigh, NC 27695-7911, USA.
emails: {hfsalama,reeves,candice}@eos.ncsu.edu

T.L. Sheu
IBM, Hardware Networking Systems, RTP, NC 27709, USA.
email: sheu@ralvm29.vnet.ibm.com

Abstract

Multicast (MC) routing algorithms capable of satisfying the QoS requirements of real-time applications will be essential for future high-speed networks. We compare the performance of all of the important MC routing algorithms when applied to networks with asymmetric link loads. Each algorithm is judged based on the quality of the MC tree it generates and its efficiency in managing the network resources. Simulation results over random networks show that unconstrained algorithms are not capable of fulfilling the QoS requirements of real-time applications in wide-area networks. One algorithm, reverse path multicasting, is not suitable for asymmetric networks irrespective of the requirements of the application. The three constrained Steiner tree (CST) heuristics reported to date are also studied. Simulations show that all three heuristics behave similarly and that they can manage the network efficiently and construct low cost MC trees that satisfy the QoS requirements of real-time traffic. The execution times of the CST heuristics depend on the MC group size, but they are always larger than those of the unconstrained algorithms.

Keywords

Multicast routing algorithms, quality of service, high-speed networks, asynchronous transfer mode, distributed real-time applications

*This work was supported in part by the Center for Advanced Computing and Communication at North Carolina State University, by Air Force Office of Scientific Research grant F49620-92-J-0441DEF, and by IBM SUR project #1429.

1 INTRODUCTION

Recent advances in optical fiber and switch technologies have resulted in a new generation of high-speed networks that can achieve speeds of up to a few gigabits per second, along with very low bit error rates. In addition, the progress in audio, video, and data storage technologies has given rise to new distributed real-time applications. These applications may involve multimedia, e.g. videoconferencing which requires low end-to-end delays, or they may be distributed control applications requiring high transmission reliability. Quality of service (QoS) parameters are used to express the applications' requirements which must be guaranteed by the underlying network. Distributed applications will utilize future networks, and in many cases they will involve multiple users. Hence the increasing importance of multicast (MC) routing algorithms which are able to manage the network resources efficiently and to satisfy the QoS requirements of each individual application.

In the past, very few networks applications involved multiple users and none of them had QoS requirements. In addition, the bandwidth requirements of most applications were very modest. Thus simple MC routing algorithms were sufficient to manage the network bandwidth. In many cases MC trees were simply constructed by the superposition of multiple unicast paths. The situation is different, however, for the emerging real-time applications discussed above. For example, videoconferencing is now available over the Internet (Macedonia and Brutzman 1994). So the question now is: can the existing best-effort networks and the simple MC algorithms they implement provide the performance guarantees essential for real-time applications such as videoconferencing?

We will study the performance of existing simple MC routing algorithms, which are used in current wide area networks, when applied to high-speed networks and their ability to satisfy the requirements of real-time applications. In addition, we will compare the performance of a number of new algorithms which were designed specifically to meet the QoS requirements of high-speed network applications.

Previous work on MC routing assumes networks with symmetric link loads, i.e. given two links interconnecting two nodes, one link in each direction, the loading factors and delays of these two links are assumed to be equal. This is a special case that does not hold for actual networks, and thus it is desirable to study the general case where a network has asymmetric link loads[†].

This paper investigates the problem of setting up MC trees. The networks studied resemble realistic asymmetric high-speed wide area networks. The QoS requirements are based on the requirements of actual real-time traffic, e.g. voice and video. MC routing algorithms are evaluated on their ability to provide performance guarantees, the quality of the MC trees they construct, and their effectiveness in managing the network resources.

1.1 Definitions

A communication network is represented as a directed connected simple graph $G = (V, E)$, where V is a set of nodes and E is a set of directed links. The existence of a link $e = (u, v)$ from u to v implies the existence of a link $e' = (v, u)$ for any $u, v \in V$. Any link $e \in E$ has a cost $C(e)$ and a delay $D(e)$ associated with it. A link's cost is a measure of the utilization

[†]The terms asymmetric networks, directed networks, and networks with a asymmetric link loads all have the same meaning.

of that link's resources. Thus $C(e)$ should be a function of the amount of traffic traversing the link e and the expected buffer space needed for that traffic. A link's delay is the delay a data packet experiences on that link (the sum of the switching, queueing, transmission, and propagation components). $C(e)$ and $D(e)$ may take any positive real values. Because of the asymmetric nature of computer networks, it is often the case that $C(e) \neq C(e')$ and $D(e) \neq D(e')$.

A MC group $G_i = g_1, ..., g_n \subseteq V, n = |G_i| \leq |V|$ is a set of nodes participating in the same network activity, and is identified by a unique group address i. $S_i = s_1, ..., s_m \subseteq V, m = |S_i| \leq |V|$ is a set of source nodes for the MC group G_i. A MC source $s \in S_i$ may or may not be itself a member of the group G_i. A MC tree $T(s, G_i) \subseteq E$ is a tree rooted at the source $s \in S_i$ and spanning all members of the group G_i. The total cost of a tree $T(s, G_i)$ is simply $\sum_{t \in T(s,G_i)} C(t)$. An algorithm that minimizes the total cost of a MC tree will encourage the sharing of links[‡]. A path $P(s, g) \subseteq T(s, G_i)$ is a set of links connecting $s \in S_i$ to $g \in G_i$. The cost of the path $P(s, g)$ is $\sum_{p \in P(s,g))} C(p)$ and end-to-end delay along that path is $\sum_{p \in P(s,g))} D(p)$.

1.2 Classification of MC Routing Algorithms

MC routing algorithms can be classified into one of two categories. Shortest paths algorithms attempt to minimize the cost of each path from the source node to a multicast group member node. Bertsekas and Gallager (1992) describe several shortest paths algorithms. The other category is the minimum Steiner algorithms. Their objective is to minimize the total cost of the MC tree. This problem is known to be *NP*-complete. Winter (1987) provides a survey of both exact and heuristic minimum Steiner tree algorithms. If the destination set of a minimum Steiner tree includes all nodes in the network, it is called a minimum spanning tree (Prim 1957).

The upper bound on acceptable end-to-end delay, Δ, is part of the QoS requirements of distributed multimedia applications, and it is necessary and sufficient for the network to satisfy the given bound, i.e. there is no need to minimize the end-to-end delay. MC routing algorithms proposed specifically for high-speed networks construct a minimum cost MC tree without violating the constraint implied by the upper bound on delay. These are called constrained Steiner tree (CST) algorithms to distinguish them from other algorithms which don't have this delay constraint.

Dynamic MC routing algorithms find the path from the source to one destination at a time. Thus they permit destination nodes to join and leave a MC group and the corresponding MC trees at any moment. In static algorithms, however, the MC group is fixed and paths from the source to all destinations are computed at the same time when establishing a MC session. Attempting to satisfy the application QoS requirements when destinations are allowed to join and leave a MC session dynamically is an interesting but difficult problem. We study only static algorithms or static versions of dynamic algorithms.

Some applications involve multiple sources transmitting to the same MC group, e.g. videoconferencing. One Approach is to use source-specific trees with each source constructing its own MC tree. An alternative approach is to have a shared tree rooted at a central node, and to let all sources transmit their packets to the central node which

[‡]A shared link is a link on the paths from the source to more than one destination

in turn forwards them over the shared tree to the destination nodes. This requires the construction of only one MC tree, but the links of this tree will be heavily loaded with traffic from all the sources transmitting to that MC group. In our work, each MC group has only one source transmitting to it and thus it suffices to consider source-specific trees only.

Some MC routing algorithms can be implemented in a distributed fashion, while others permit only centralized implementation. Some algorithms need to keep global information about the state of the network at each node, while for others local information about nearby neighbors suffices. All algorithms discussed in this paper, except one, are centralized with global information at each node. We also assume that information kept at the nodes, whether global or local, is always up-to-date.

This paper is organized as follows: Section 2 presents the unconstrained and constrained MC routing algorithms which we consider in our work. Section 3 describes the characteristics of the networks we study. The performance metrics used are discussed in section 4 and followed by simulation results. Section 5 concludes the paper.

2 MULTICAST ROUTING ALGORITHMS

In this section we present brief discussion of the distinguishing features of each of the algorithms we consider in our work. We study three unconstrained MC routing algorithms, one semi-constrained heuristic, and three constrained MC routing heuristics that have been designed specifically for high-speed networks carrying real-time applications. In addition, we use the following three optimal algorithms as a basis for evaluating the performance of the different heuristics. The unconstrained optimal minimum Steiner tree, **OPT**, algorithm always finds the minimum cost solution for the MC routing problem. The constrained optimal minimum Steiner tree, **COPT**, algorithm finds the minimum cost solution for the same problem subject to a given delay constraint. Our implementation of these two algorithms uses a branch and bound technique. Their execution times are very large, because the problem is *NP*-complete, so we could only apply them to small networks. The third optimal algorithm is the least-delay, **LD**, MC routing algorithm. We implemented it as a Dijkstra's shortest paths algorithm (Bertsekas and Gallager 1992) in which $C(e) = D(e)$, i.e. it guarantees minimum end-to-end delay from the source to each MC group member. The worst case time complexity of Dijkstra's algorithm is $O(|V|^2)$.

2.1 Unconstrained Algorithms

The algorithms described below attempt to optimize a given cost function without taking into consideration the QoS requirements of the application.

Very few algorithms have been proposed for the minimum Steiner tree problem in directed networks (Winter 1987), and all of them operate under special conditions, e.g. acyclic networks, and thus they can not be applied to the networks we work on. In the case of undirected networks, however, there are several heuristics of reasonable complexity. Doar and Leslie (1993) show that the total cost of trees generated using Kou, Markowsky, and Berman (1981), **KMB**, heuristic for the minimum Steiner tree is on the average only 5% worse than the cost of the optimal undirected minimum Steiner tree while its time complexity is $O(|G_i||V|^2)$. Thus KMB is an efficient unconstrained minimum Steiner tree heuristic for undirected networks. We will study how efficient it is when applied to directed networks with delay constraints.

Dijkstra's shortest paths algorithm (with link cost being a function of the link utilization) is widely used in communication protocols, e.g. MOSPF (Moy 1992), and it yields satisfactory performance. It is a least-cost, **LC**, algorithm which minimizes the cost of the path from the source node to each MC group member individually. We study LC's performance in networks with delay constraints to determine its applicability to networks that are heavily loaded with multimedia applications.

The reverse path multicasting, **RPM**, algorithm (Deering and Cheriton 1990) creates MC trees for which the cost of the reverse path, i.e. the path from the destination to the source, is minimal. The cost of the forward path is minimal only if the network has symmetric link costs. RPM is an attractive choice for network developers, because it can be implemented in a distributed fashion. Protocol Independent Multicasting (PIM) (Deering et.al. 1995) is a candidate protocol for multicasting over the Internet. PIM uses a dynamic MC routing algorithm based on RPM. Thus it is important to study RPM to determine how asymmetric link loads affect its performance. It is important to note that due to the limited local information a node v doesn't know the cost of the link $e = (u, v)$. It only knows the cost of the reverse link $e' = (v, u)$. Even if link e is saturated and can not accept any additional traffic, node v will not be aware of that and may still try to add e to future MC trees.

The MC routing heuristic for ATM networks proposed by Waters (1994) is semi-constrained. It uses the maximum end-to-end delay from the source to any node in the network as the delay constraint. Note that this constraint is not related directly to the application QoS constraints, and that, depending on the network delays, this internally computed constraint may be too strict or too lenient as compared to the QoS requirements of the application. The heuristic then constructs a broadcast tree that does not violate the internal delay constraint. Finally the broadcast tree is pruned beyond the MC nodes. In (Salama et.al. 1994) we implemented the original algorithm proposed in (Waters 1994) which resembles a semi-constrained minimum spanning tree, and we also implemented a modified version which is closer to a semi-constrained shortest paths broadcast tree. The modified version always performs better with respect to tree costs, end-to-end delays, and network balancing. In this paper we will consider only the modified semi-constrained, **MSC**, heuristic. MSC is dominated by the computation of the internal delay bound, which uses a modified Dijkstra algorithm and runs in $O(|V|^2)$.

2.2 Constrained Algorithms

The first heuristic for the CST problem was given by Kompella, Pasquale, and Polyzos (1993). We label this **KPP** heuristic. KPP assumes that the link delays and the delay bound, Δ, are integers. The heuristic is dominated by computing a constrained closure graph which takes time $O(\Delta|V|^3)$. Thus KPP takes polynomial time only if Δ is bounded. When the link delays and Δ take non integer values it is necessary to limit the granularity of the computation in order to achieve reasonable running times. The side effects of this are discussed in (Widyono 1994). When the granularity is comparable to the average link delays, KPP's accuracy is compromised and in many cases it can not even construct a constrained MC tree. We use a granularity of $\Delta/10$.

Both KMB and KPP heuristics use Prim's algorithm (PRIM 1957) to obtain a minimum spanning tree when given a closure graph. Prim's algorithms is only optimal for undirected networks. This might affect the performance of the two heuristics when applied to directed networks.

Widyono (1994) proposed four unconstrained MC heuristics and four CST heuristics. The four constrained heuristics are based on a constrained Bellman-Ford algorithm presented in the same report. Constrained Bellman-Ford uses a breadth first search to find the constrained least-cost paths from the source to all other nodes in the network. We will consider only the constrained adaptive ordering, **CAO**, heuristic as it yields the best performance of the heuristics Widyono proposed. In CAO, the constrained Bellman-Ford algorithm is used to connect one group member at a time to the source. After each run of constrained Bellman-Ford, the unconnected member with the cheapest constrained path to the source is chosen and is added to the existing subtree. The costs of links in the already existing subtree are set to zero. The author has not conducted a conclusive analysis of constrained Bellman-Ford's time complexity, but he found that there are cases in which its running time grows exponentially. CAO is always capable of constructing a constrained MC tree, if one exists, because of the nature of the breadth first search it conducts.

The bounded shortest multicast algorithm, **BSMA**, is another CST heuristic (Zhu, Parsa, and Garcia-Luna-Aceves 1995). BSMA starts by computing a LD tree for a given source s and MC group G_i. Then it iteratively replaces superedges[§] in $T(s, G_i)$ with cheaper superedges not in the tree without violating the delay bound until the total cost of the tree can not be reduced any further. BSMA uses a kth-shortest path algorithm to find cheaper paths for path switching. It runs in $O(k|V|^3 \log |V|)$. k may be very large in case of large densely connected networks, and it may be difficult to achieve acceptable running times. We use a path switching algorithm that is slightly different than the one used by the authors of BSMA in order to account for the effect of directed networks. BSMA always finds a constrained MC tree, if one exists, because it starts with a LD tree.

The CST heuristics described above are of higher complexity than the unconstrained algorithms. We studied the performance of both classes of algorithms to determine which algorithms are capable of fulfilling the QoS requirements of real-time applications, and to find out if there is an actual need for the more complex CST heuristics. In addition, we investigated the effect of the delay constraint on the efficiency of the CST heuristics, and on their ability to minimize the total cost of a MC tree.

3 THE EXPERIMENTAL SETUP

We used simulation for our experimental investigations to avoid the limiting assumptions of analytical modeling. Asynchronous transfer mode (ATM) networks permit the applications to specify their own QoS requirements, and they allow cell multicasting in the physical layer. Thus, it was appropriate for us to comply with the ATM standards.

20-node, 50-node, and 100-node full duplex ATM networks with homogeneous link capacities of 155 Mbps (OC3) were used in the experiments. The positions of the nodes are fixed in a rectangle of size 3000 $*$ 2400 Km^2, roughly the area of the USA. A random generator (Waxman 1988) was used to create links interconnecting pairs of nodes The

[§]A superedge is a path in the tree between two branching nodes or two MC group members or a branching node and a MC group member.

probability of a link to exist between any two nodes is a function of the distance between these two nodes. We adjusted the parameters of the random generator to yield networks with average node degrees of 4. Any node has a node degree ≥ 2.

Each node represented an output-buffered non-blocking ATM switch. Each link had its own FCFS output buffer of size 8 cells. The propagation speed in the link was taken to be two thirds the speed of light. The propagation delay was dominant under these conditions, and the queueing component was neglected when calculating the link delay, $D(e)$.

For the MC sources we used variable bit rate (VBR) video sources with an average rate of 0.5 Mbps and peak to average ratio of 3:1. These represent realistic bursty multimedia traffic sources. Any session traversing a link e, reserves a fraction of e's bandwidth equal to the average rate of the traffic it generates. The link cost, $C(e)$, is equal to the reserved bandwidth on that link, because it is a suitable measure of the utilization of both the link's bandwidth and its buffer space. $C(e)$ is dynamic, and varies as new sessions are established or existing sessions are torn down.

A link can accept sessions and reserve bandwidth for them until its cost, i.e. the sum of the average rates of the sessions traversing that link, exceeds 85% of the link's capacity, then it gets saturated. This simple admission control policy allows statistical multiplexing and efficient utilization of the available resources.

Interactive voice and video sessions have tight delay requirements. We used a value of 0.03 seconds for Δ which represents only an upper bound on the end-to-end propagation time across the network. This relatively small value was chosen in order to allow the higher level end-to-end protocols enough time to process the transmitted information without affecting the quality of interaction.

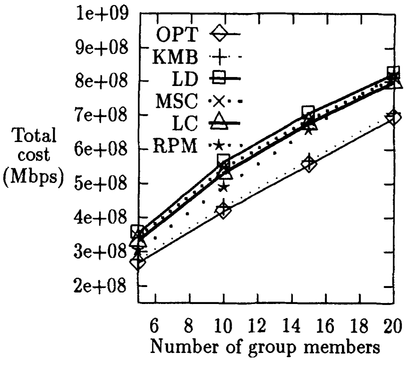
4 PERFORMANCE METRICS AND EXPERIMENTAL RESULTS

The performance of a MC routing algorithm was evaluated based on the quality of the MC trees it creates and the algorithm's efficiency in managing the network. The quality of a MC tree can be defined in the following ways:

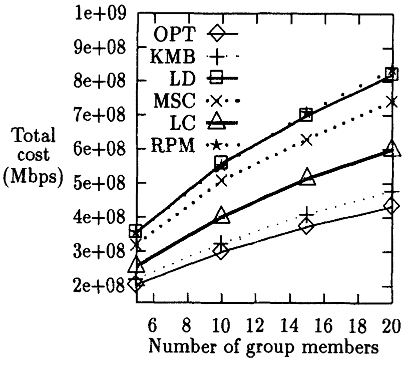
- The total cost of the tree. This reflects the algorithm's ability to construct a MC tree using cheap, lightly loaded links.
- The maximum end-to-end delay from the source to any MC group member. This indicates the algorithm's ability to achieve the delays required by the application.

An algorithm's effectiveness in managing the network resources was judged by monitoring how frequently that algorithm fails to construct an acceptable MC tree for a given network with given link loads. There are two causes of failure: either the created tree does not satisfy the delay bound or the algorithm fails to find unsaturated links, and thus it can not create a tree that spans all MC group members. Another measure of an algorithm's effectiveness is the number of MC trees that the algorithm can create before the cumulative failure rate exceeds a certain limit.

Two experiments were conducted on the algorithms discussed in section 2. We present the simulation results for the unconstrained algorithms first to determine the conditions, if any, under which the unconstrained algorithms do not perform well. Then we show the results obtained for the CST heuristics, and discuss their advantages and disadvantages.



(a) B_{min} = 45 Mbps B_{max} = 85 Mbps.



(b) B_{min} = 5 Mbps B_{max} = 125 Mbps.

Figure 1 Total cost of a MC tree, unconstrained algorithms, 20 nodes, average degree 4.

4.1 Simulation Results for the Unconstrained Algorithms

The first experiment compares the different algorithms when each of them is applied to create a MC tree for a given source node generating video traffic with an average rate of 0.5 Mbps, and a given MC group.

For each run of the experiment we generate a random set of links to interconnect the fixed nodes, we generate random background traffic for each link, we select a random source node and a MC group of randomly chosen destination nodes. The average rate of each link's background traffic is a random variable uniformly distributed between B_{min} and B_{max}. As the range of load variation, i.e. the difference between B_{max} and B_{min}, increases the asymmetry of the link loads also increases, because the load on link $e = (u, v)$ is independent of the load on the link $e' = (v, u)$. The experiment is repeated with different MC group sizes. We measured the total cost of the MC tree, the maximum end-to-end delay, and the failure rate of the algorithm. Note that an unconstrained algorithm may construct a MC tree with a maximum delay that violates the imposed delay bound. Such a tree is rejected and removed by the admission control process, but not before we measure its characteristics. The experiment was run repeatedly until confidence intervals of less than 5%, using 95% confidence level, were achieved for all measured quantities.

Figure 1 shows the total cost of a MC tree versus the MC group size for two different link loading conditions for 20-node networks. KMB heuristic yields very low tree costs. Note that in the more asymmetric case(figure 1(b)) KMB's costs are 10% worse than OPT, which is not as good as its performance when applied to symmetric or slightly asymmetric networks. LC does not perform as good as KMB, because it attempts to minimize the cost per path from source to destination, not the total cost of the entire tree. MSC creates trees that are less expensive than LD but more expensive then both KMB and LC, because the internally generated delay bound is so strict that it limits the algorithm's ability to minimize the costs. KMB, LC, and MSC create cheaper trees when the range of link load variation increases, because they are capable of locating the low-cost links and adding them to the tree. LD yields the most expensive trees, and its

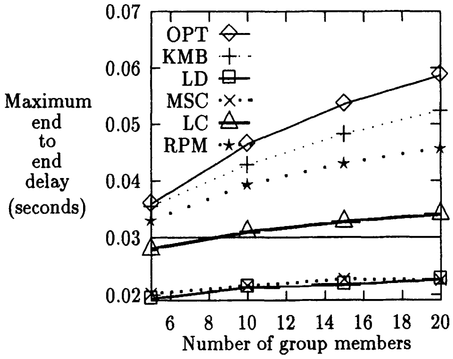


Figure 2 Maximum end-to-end delay, unconstrained algorithms, 20 nodes, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps.

performance is independent of the range of link load variation. The reason is that LD optimizes the end-to-end delay which is independent of the network loading conditions. RPM generates more expensive trees as the asymmetry of the network increases. This is due to the fact that the more asymmetric the network is, the less related the cost of the reverse link, $C(e')$, and the cost of the forward link, $C(e)$, are. When RPM selects a cheap reverse link, e', it is very probable that the forward link, e, is more expensive than it. In the extreme case, the forward link, e, may be already saturated. When this happens the entire tree is rejected by the enforced admission control policy. For extremely asymmetric networks, as is the case in figure 1(b), RPM performs as bad as LD.

Figure 2 shows the maximum end-to-end delay[||], and RPM does not perform well here either. OPT and KMB also perform very poorly with respect to maximum delay, because they do not attempt to minimize the end-to-end delay to the individual destinations. LC results in maximum delays that are in some cases less than 0.03 seconds which is within the QoS requirement. It finds the least-cost path to each group member. This indirectly minimizes the number of hops for such a path and hence indirectly reduces the length of the path. Figure 2 also shows that maximum end-to-end delays resulting from MSC are almost as good as the optimal LD. This is again due to MSC's strict internally generated upper bound on delay. As the number of group members increases, the maximum delays increase, because the MC trees span more nodes, hence the probability of a remote node being a member in the MC group is larger.

Delay bound violation is one of the reasons to reject a MC tree. An algorithm's failure to construct a MC tree due to delay bound violation is strongly related to the maximum delays discussed above. Therefore it is not surprising for KMB, RPM, and even LC to have very high failure rates, $> 30\%$. MSC's failure rate, however, is almost as low as LD's failure rate, $< 2\%$.

Figure 3 shows the results of the second experiment in which we start with a completely unloaded network and keep adding MC sessions and constructing the corresponding MC

[||]It is sufficient to show one case of network loading, because we found that the performance of the different algorithms relative to each other with respect to the maximum end-to-end delay is independent of the range of link load variation.

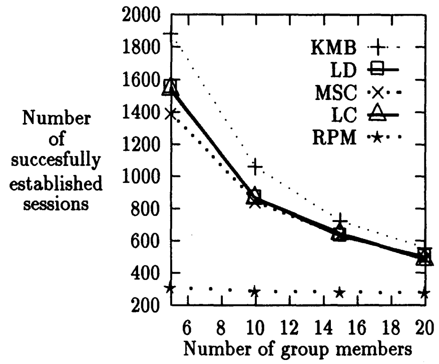


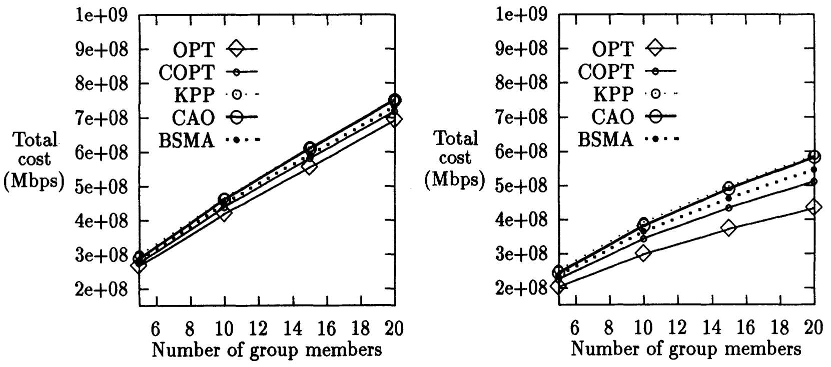
Figure 3 Number of successful sessions, unconstrained algorithms, 20 nodes, average degree 4, no delay constraint.

trees until the cumulative tree failure rate exceeds 15%. A MC session consists of a random source node generating VBR video traffic with an average rate of 0.5 Mbps, and a MC group of randomly chosen destination nodes. The experiment was repeated with MC groups of different sizes. Failure due to delay bound violation was disabled in this experiment, because the results of the first experiment have shown that the unconstrained algorithms can not satisfy a delay bound of 0.03 seconds. Here we will determine how efficiently these unconstrained algorithms can manage the network in the absence of a delay bound. The experiment is repeated, until the confidence interval for the number of successfully established MC sessions is $< 5\%$ using 95% confidence level. Similar to the first experiment, in this experiment a random network topology is generated before each run. This experiment could not be applied to the optimal minimum Steiner tree algorithm, because of its large execution time.

It is obvious from figure 3 that as the size of the MC group increases, the number of MC trees that an algorithm can construct before the network saturates decreases. This is because the size of a MC tree increases as the group size increases. KMB yields the best performance, because it has the ability to locate the cheapest links in the network and include them in the MC tree. This results in approximately uniform link load distribution across the network throughout the experiment. LD, LC and MSC can also manage the network resources efficiently, although not as efficiently as KMB. This is surprising for LD, which does not attempt to manage the link bandwidth at all. RPM is very inefficient even for small group sizes, because it attempts to add saturated links to the MC trees as has been discussed earlier.

Experiment 2 shows that, in the absence of a delay constraint, all algorithms, except RPM, can manage the network resources efficiently. RPM's approach, to estimate the cost of the forward link to be equal to the cost of the reverse link, is futile. It results in very asymmetric networks with a few very heavily loaded, saturated links and many lightly loaded, underutilized links.

The two experiments discussed above show that none of the unconstrained algorithms is satisfactory for applications having delay constraints. The semi-constrained heuristic, MSC, has a delay bound that is too tight which reduces its performance with respect to



(a) B_{min} = 45 Mbps B_{max} = 85 Mbps (b) B_{min} = 5 Mbps B_{max} = 125 Mbps

Figure 4 Total cost of a MC tree, CST heuristics, 20 nodes, average degree 4, $\Delta = 0.03$ seconds.

cost. Similarly, LD is optimal with respect to minimizing delays but it doesn't attempt to optimize the tree cost at all. We will study the CST heuristics to determine if they can achieve a compromise between the unconstrained cost-oriented algorithms (OPT and KMB) and the delay-oriented algorithms (LD and MSC).

4.2 Simulation Results for the Constrained Algorithms

We re-ran the same first experiment of section 4.1 on COPT and the three CST heuristics to determine the characteristics of the constrained MC trees these algorithms construct.

Figure 4 shows that the three constrained CST heuristics yield similar total tree costs. BSMA always gives better costs than CAO and KPP, but never by more than 5%. The costs of the heuristics are within 5–15% from the cost of COPT. We also show the cost of the unconstrained OPT algorithm. Comparing the costs of COPT and the three CST heuristic to the cost of OPT gives an indication of how much cost must be sacrificed in order to satisfy the delay constraint. COPT is never more than 10% more expensive than OPT. The average cost inefficiency of one algorithm with respect to optimal depends on the random distribution of the link costs, i.e. the network loading conditions.

It can be seen from figure 5 that the maximum end-to-end delays for the constrained algorithms are below the 0.03 seconds delay bound. Again all constrained algorithms yield similar delay performance, but BSMA is slightly better. This is because BSMA starts with a least-delay tree and then improves its cost. The small delays of the initial tree persist in the final tree. Figure 5 also shows the maximum delays of LD for comparison. LD's maximum delays are considerably less than maximum delay the CST algorithms can achieve. However, this is not a big advantage, because it is sufficient to satisfy the delay constraint. We found that the granularity we chose for KPP ($\Delta/10 = 3$ msec) is sufficient to yield close to optimal performance. KPP succeeds in more than 97% of the runs in constructing a constrained MC tree, and is always within 0.5% from COPT's success rate in the case of 20-node networks.

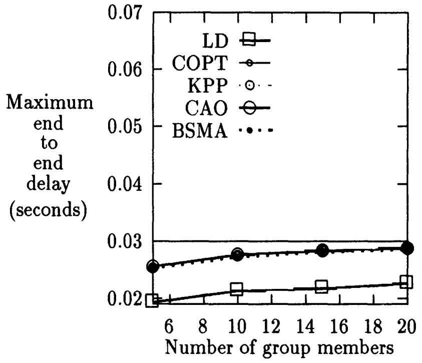


Figure 5 Maximum end-to-end delay, CST heuristics, 20 nodes, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds.

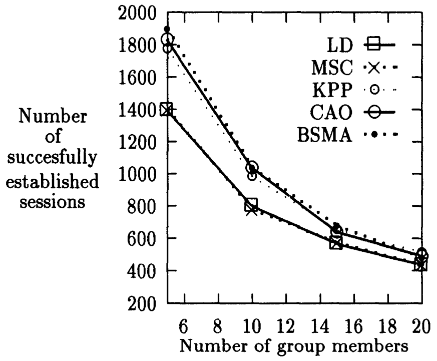
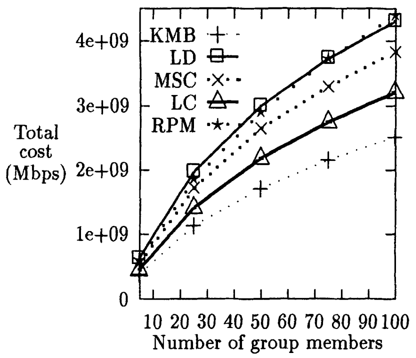


Figure 6 Number of successful sessions, 20 nodes, average degree 4, Δ = 0.03 seconds.

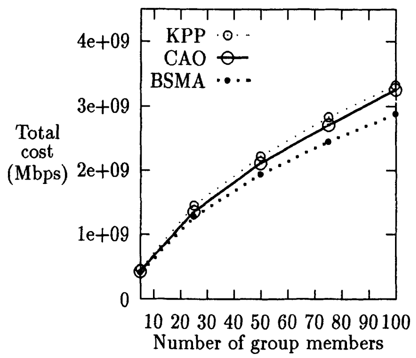
We conducted the second experiment of section 4.1 on the CST heuristics to evaluate their efficiency in managing the network bandwidth. The experiment was first modified, however, to permit failure due to delay bound violation. An algorithm can thus fail to construct a MC tree due to either violating the delay bound or due to link saturation. We also conducted this modified experiment on the LD and MSC algorithms. We could not run it on COPT, however, because of its large execution time.

Figure 6 shows that again the three CST heuristics yield almost identical performance and that they can manage the network bandwidth better than LD and MSC.

We repeated the first experiment using 100-node networks with very asymmetric link loads. The 100 nodes were randomly placed in an area of the same size as the 20 nodes were previously placed. We could not apply OPT and COPT to the 100-node networks due to their excessive running times. We show figure 7 as an example of the results we obtained for the 100-node networks. Comparing the results of the first experiment for both the 20-node networks and the 100-node networks we conclude that the performance of the algorithms, both unconstrained and constrained, relative to each other remains the



(a) Unconstrained algorithms.



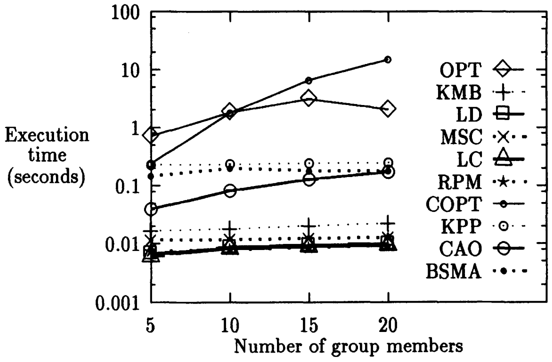
(b) CST heuristics.

Figure 7 Total cost of a MC tree, 100 nodes, average degree 4, B_{min} = 5 Mbps B_{max} = 125 Mbps, Δ = 0.03 seconds.

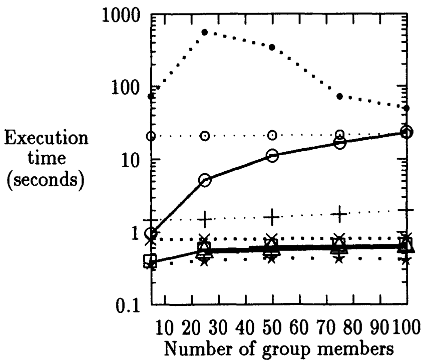
same regardless of the size of the networks. We noticed however that KPP's success rate in constructing a constrained tree is up to 5% worse than optimal (LD) for the 100-node networks. This is because as the number of nodes increases within the same area, the average link delay decreases. For 100-node networks the average link delay is small and comparable to KPP's granularity, which affects the heuristic's success rate.

4.3 Execution Times

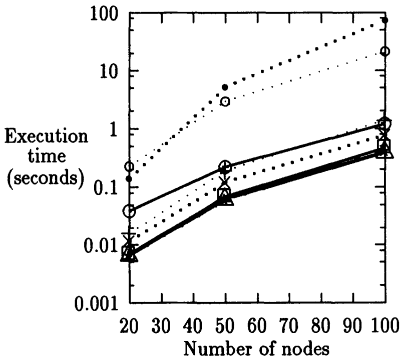
Figure 8 shows the average execution times of all algorithms studied in this paper. Note, however, that the code used for the algorithms was not optimized for speed. These results are therefore not conclusive. Figures 8(a) and 8(b) show the execution times for different MC group sizes. Figure 8(c) shows the growth of the execution times with the network size for a fixed MC group of 5 members. The running times of OPT and COPT are very large as can be seen from figure 8(a). RPM, in spite of its poor performance, is the fastest algorithm. The running times of the CST heuristics are large, except CAO's running time for small group sizes. CAO's running time increases as the group size increases, because it runs the constrained Bellman-Ford algorithm once for each group member. BSMA's running time is very large for 100-node network as shown in figure 8(b). It is particularly slow for MC group of medium size. For small MC groups, the number of superedges in a MC tree is small, and BSMA does not have to apply the time consuming kth-shortest path algorithm many times. For large MC groups, the kth-shortest path algorithm is fast because the number of nodes outside the initial tree is small, and thus the number of possible alternate paths to replace a superedge is small. For medium size MC groups, however, the number superedges is large and at the same time the number of nodes outside the tree is also large which leads to very long running times of BSMA. The three CST heuristics may be too slow, without modification, to use on networks with thousands of nodes.



(a) 20 nodes, average degree 4, variable MC group size.



(b) 100 nodes, average degree 4, variable MC group size.



(c) Variable network size, average degree 4, 5 MC group members.

Figure 8 Execution times , B_{min} = 5 Mbps B_{max} = 125 Mbps, Δ = 0.03 seconds. Subfigures (b) and (c) use the same legend as (a).

5 CONCLUSIONS

Distributed real-time applications have QoS requirements that must be guaranteed by the underlying network. In many cases these applications will involve multiple users and hence the increasing importance of multicasting. MC routing can be an effective tool to manage the network resources and fulfill the applications' requirements. Several MC routing algorithms are proposed for high-speed networks carrying real-time traffic. Our work is the first detailed, quantitative evaluation of these algorithms under realistic conditions.

We have studied the performance of unconstrained MC routing algorithms when applied to wide area networks with asymmetric link loads. KMB heuristic constructs low cost trees with large end-to-end delays that exceed the upper bound on delay imposed by the application. The other two unconstrained algorithms studied, Dijkstra's LC and the RPM heuristic, are also unable to satisfy the required delay bound. KMB is most efficient in managing the network bandwidth followed by LC. RPM performs poorly when applied networks with asymmetric link loads. Note, however, that it has the fastest execution time, and it is a distributed dynamic algorithm. We conclude that unconstrained MC routing algorithms can not be applied to real-time applications on networks spanning large areas due to their inadequate delay performance.

The semi-constrained heuristic uses an internally computed delay bound that is too strict and thus limits its ability to construct low cost trees and to manage the network resources.

We then studied three CST heuristics. All three heuristics yield similar performance, but their execution times differ considerably. BSMA is faster than KPP in case of small networks only. As the networks size increases BSMA's running time grows much faster than KPP's. CAO is the fastest CST heuristic, but it is slower than the unconstrained and semi-constrained algorithms. The CST heuristics construct trees that are not considerably more expensive than COPT's trees. The maximum end-to-end delays obtained from the CST heuristics are larger than those obtained using LD, but they are still within the given delay bound. In short, all three heuristics construct low cost trees, which satisfy the given delay bound, and can manage the network resources efficiently. To prefer one algorithm over the two others some implementation issues must be taken into consideration: the amount of network state information the algorithm needs, is a distributed implementation of the algorithm possible, and can the running time of the algorithm be reduced?

6 ACKNOWLEDGMENT

The authors would like to thank P. Manyem for his assistance with the optimal minimum Steiner tree algorithms.

7 REFERENCES

Bertsekas, D. and Gallager, R.G. (1992) *Data Networks.* Second Edition, Prentice-Hall.

Deering, S. and Cheriton, D. (1990) Multicast Routing in Datagram Networks and Extended LANs. *ACM Transactions on Computer Systems*, **8**, 85–110.

Deering, S. et. al. (1995) Protocol Independent Multicast (PIM): Protocol Specification. Internet Draft.

Doar, M. and Leslie, I. (1993) How Bad is Naive Multicast Routing. *Proceedings of INFOCOM*, 82–9.

Kompella, V.P., Pasquale, J.C. and Polyzos, G.C. (1993) Multicast Routing for Multimedia Communication. *IEEE/ACM Transactions on Networking*, **1**, 286–92.

Kou, L., Markowsky, G. and Berman, L. (1981) A Fast Algorithm for Steiner Trees. *Acta Informatica*, **15**, 141–5.

Macedonia, M.R. and Brutzman, D.P. (1994) MBone Provides Audio and Video Across the Internet. *IEEE Computer*, **27**, April, 30–6.

Moy, J. (1992) Multicast Extension to OSPF. Internet Draft.

Prim, R.C. (1957) Shortest Connection Networks and Some Generalizations. *The Bell Systems Technical Journal*, **36**, 1389–401.

Salama, H.F., Reeves, D.S., Viniotis, I. and Sheu, T.L. (1994) Comparison of Multicast Routing Algorithms for High-Speed Networks. IBM Technical Report IBM-TR29.1930.

Waters, A.G. (1994) A New heuristic for ATM Multicast Routing. *Proceedings of the 2nd IFIP Workshop on Performance Modeling and Evaluation of ATM Networks*, 8.1–9.

Waxman, B.M. (1988) Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, **6**, 1617–22.

Widyono, R. (1994) The Design and Evaluation of Routing Algorithms for Real-Time Channels. Technical Report TR-94-024, Tenet Group, Department of EECS, University of California at Berkeley.

Winter, P. (1987) Steiner Problem in Networks: A Survey. *Networks*, **17**, 129–67.

Zhu, Q., Parsa, M. and Garcia-Luna-Aceves, J.J. (1995) A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting. *Proceedings of INFOCOM*, 377–85.

8 BIOGRAPHY

Hussein F. Salama received his B.Sc. degree in Communication and Electronics Engineering with honors from Cairo University, Egypt, in 1990. He received his M.Sc. degree in Electrical Engineering in 1993 from N. C. State University where he is currently pursuing his Ph.D. degree. His research interests involve communication networks and multimedia.

Douglas S. Reeves received his Ph.D. in Computer Science from Pennsylvania State University in 1987. Since that time he has been a faculty member at N. C. State University, where he is now an Associate Professor of Computer Science and Engineering. His research interests include real-time communication, parallel computing, and multimedia.

Yannis Viniotis received the B.Sc. degree from the University of Patras, Greece, and M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park. He is currently an Associate Professor of Electrical and Computer Engineering at North Carolina State University, Raleigh, NC. His research interests are in computer communication system design and analysis, with particular emphasis on Quality of Service, multimedia, multicasting and adaptive network control algorithms. He has published over 30 articles and lectured extensively on these topics. He was a guest editor of the *Performance Evaluation* Journal, in 1995, on High Speed Networks. He has chaired two international conferences on networking, in 1992 and 1993.

Tsang-Ling Sheu received the B.S. degree in Electrical Engineering from National Cheng-Kung University, Taiwan, in 1981, the M.S. degree in Electrical Engineering from Virginia Polytechnic Institute & State University, Blacksburg, in 1985, and the Ph.D. degree in Computer Engineering from Pennsylvania State University, University Park, in 1989. In September 1989, he joined IBM Corp. at Research Triangle Park, North Carolina, as a technical staff member. His primary works with IBM include the design and development of ATM switches, ATM adapters, and a multimedia distribution node for video on demand. His research areas include ATM Switching, Multimedia Networking, LAN/WAN Internetworking, and Parallel/Distributed Processing. He was the recipient of 1990 IBM Outstanding Paper Award. Dr. Sheu is a member of the IEEE.