Evaluation of Multicast Routing Algorithms for Distributed Real-Time Applications

H. F. Husseim, D. S. Reeves,
I. Viniotis, T. Sheu

Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University

TR-95/6
March 1995

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

Hussein F. Salama Douglas S. Reeves Ioannis Viniotis
Department of Electrical and Computer Engineering
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7911
Phone: (919) 515-5348
Fax: (919) 515-2285
Emails: {hfsalama,reeves,candice}@eos.ncsu.edu

Tsang-Ling Sheu
IBM, Hardware Networking Systems
Research Triangle Park, NC 27709
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 large networks. One algorithm, reverse path multicasting, is not suitable for asymmetric networks irrespective of the requirements of the application. All 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 are larger than those of the unconstrained algorithms.

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 efficient 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 real-time QoS requirements. In addition, the bandwidth requirements of most applications were very modest. Thus simple

*This work was supported in part by an IBM SUR project and in part by the Center for Advanced Computing and Communication at North Carolina State University.

MC routing 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 [1]. So the question now is: can existing best-effort networks and the simple MC algorithms they implement provide the performance guarantees required by such real-time applications like 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[1].

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

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 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[2], because the cost of a link is only added once to the total cost of the tree regardless of the number of destinations which receive their packets via that link. Link sharing is always desirable, because only one copy of a packet will traverse a given link, thus maximizing link sharing in a MC tree results in reducing the total amount of network bandwidth reserved for that MC session. A path $P(s, g) \subseteq T(s, G_i) \subseteq E$ is a set of links connecting $s \in S_i$ to $g \in G_i$. The cost of $P(s, g)$ is $\sum_{p \in P(s,g))} C(p)$.

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 describe several shortest paths algorithms in [2]. The other category is the minimum Steiner algorithms. The objective of minimum Steiner tree algorithms is to minimize the total cost of the MC tree. However, this problem is known to be *NP*-complete. Winter [3] 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 [4].

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.

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

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

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. 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 alternative is to use source-specific trees with each source constructing its own MC tree. The other alternative is to have a shared tree rooted at a central node, and to let all source transmit their packets to the central node which 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 each of the algorithms we consider in our work. It is not possible, due to space limitations, to describe the operation of each algorithm in detail, so we will only discuss the distinguishing features of each algorithm. We study one unconstrained minimum Steiner tree algorithm, one unconstrained shortest paths algorithm, and three constrained MC routing heuristics that have been designed specifically for high-speed networks carrying real-time applications. We consider also a semi constrained heuristic and an unconstrained MC routing heuristic that is part of the MC protocol being designed for the Internet.

The least-delay, **LD**, MC routing algorithm is a version of Dijkstra's shortest paths algorithm [2] in which $C(e) = D(e)$, i.e. it guarantees minimum the end-to-end delay from the source to each MC group member. LD is thus optimal with respect to end-to-end delays. If LD can not satisfy the imposed delay constraint, then no other algorithm can. We will use it as a reference when evaluating the delays obtained from other MC routing algorithms. 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 [3], 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 [5] show that the total cost of trees generated using Kou, Markowsky, and Berman's, **KMB**, heuristic [6] 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 a function of the link utilization) is widely used in communication protocols, e.g. MOSPF [7], and it yields satisfactory performance. This 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 networks that are heavily loaded with multimedia applications.

The reverse path multicasting, **RPM**, algorithm [8] 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 distributedly. Protocol Independent Multicasting (PIM) [9] 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 [10] 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 this constraint is not related directly to the application QoS constraints, and that this internally computed constraint may be too strict or too lenient as compared to the QoS requirements of the application. The heuristic then proceeds to construct a broadcast tree that does not violate the internal delay constraint. Finally the broadcast tree is pruned beyond the MC nodes. In [11] we implemented the original algorithm proposed in [10] 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

Noronha and Tobagi [12] present an optimal algorithm for constructing multiple CST trees simultaneously in a directed network. Their algorithm is based on integer programming. It manages the network resources efficiently, but its running time is two orders of magnitude slower than KMB and Dijkstra's algorithms. Such an algorithm can not be used in real networks, and thus we did not consider it in our experiments.

The first heuristic for the CST problem was given by Kompella, Pasquale, and Polyzos [13]. 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 [14]. We use a granularity of $\Delta/10$.

Both KMB and KPP heuristics use Prim's algorithm [4] 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 [14] proposes 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 compared the other 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 has been performed, but he found that there are cases in which its running time grows exponentially.

The bounded shortest multicast algorithm, **BSMA**, is another CST heuristic [15]. BSMA starts by computing a LD tree for the given source s and MC group G_i. Then it iteratively replaces paths in $T(s, G_i)$ with cheaper paths 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 $O(k|V|^3 \log|V|)$ and thus for large densely connected networks k may be very large and it may be difficult to achieve acceptable running times. We use 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.

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 inaccuracies of analytical modeling. 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.

A 20-node full duplex ATM network with homogeneous link capacities of 155 Mbps (OC3) was used in the experiments. The positions of the 20 nodes are fixed in a rectangle of size 2400 * 3000 Km^2, roughly the area of the USA. A random generator [16] was used to create links interconnecting pairs of nodes with probability:

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

where $d(u,v)$ is the distance from node u to v, and L is the maximum distance between two nodes. The parameter α controls the ratio of short links to long links, while β controls the average node degree of the network. By assigning the values 0.15 and 2.2 to α and β respectively, we got networks with average node degrees of 4. Most current networks have average node degrees ranging from 3 to 3.5, but we expect them to get denser in the future to be able to carry the increasing loads. 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 switching, queueing, and transmission components of the delay were 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..

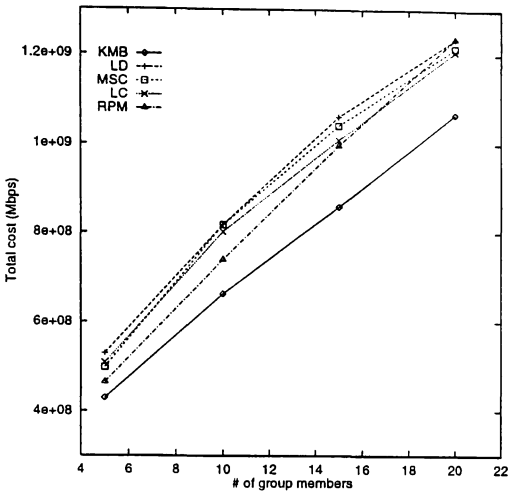
A link can accept sessions and reserve bandwidth for them until its cost, i.e. the sum of the average rates of the sessions traversing that link, exceeds 85% of the link's capacity, then it gets saturated. This simple admission control policy allows statistical multiplexing and efficient utilization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

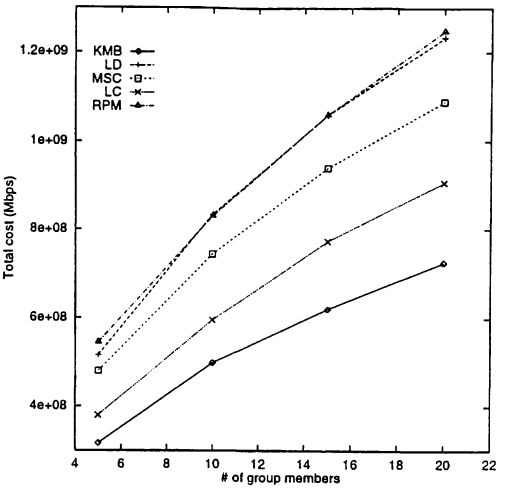
4 Performance Metrics and Experimental Results

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 number of nodes in the tree. Call establishment is a time consuming process that involves setting up virtual circuits. This temporarily interrupts other processes running at any node in the tree being established. In addition, each node in the tree will have more traffic traversing it. It is therefore desirable to reduce the number of nodes in a MC tree.



(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-node network, average degree 4.

- 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 will show the results obtained for the CST heuristics, and discuss their advantages and disadvantages.

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 random source node generating video traffic with an average rate of 0.5 Mbps, and a MC group of randomly chosen destination nodes under different network loading conditions. The experiment was repeated with MC group sizes of 5, 10, 15, and 20[3] members. Background traffic is generated for each link in the network. 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 $e' = (v, u)$. We will measure the total cost of the MC tree, the maximum end-to-end delay, the number of nodes in the tree, 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 keeps creating MC trees until the confidence intervals for the measured quantities are $< 5\%$ using 95% confidence level. The experiment also keeps changing the network topology by generating random links as has been discussed in section 3.

Figure 1 shows the total cost of a MC tree for two different link loading conditions. KMB heuristic yields the least tree costs, because it is the only true minimum Steiner tree heuristic. LC does not perform as good

[3]A MC group with 20 members represents the special case of broadcasting

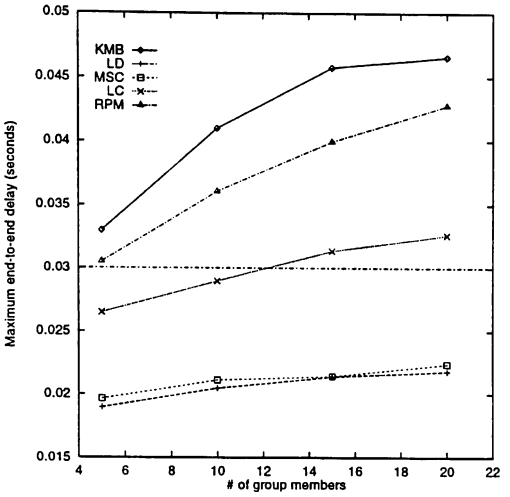


Figure 2: Maximum end-to-end delay, unconstrained algorithms, 20-node network, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps.

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 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. If RPM selects a cheap reverse link, e', it is very probable that the forward link, e, is more expensive than it. For extremely asymmetric networks, as is the case in figure 1(b), RPM performs as bad as LD.

For the performance metrics discussed below, it is sufficient to show only one case of network loading (figures 2 and 3), because we found that the performance of the different algorithms relative to each other with respect to these metrics is independent of the range of link load variation.

Figure 2 shows the maximum end-to-end delay, and RPM does not perform well here either. KMB also performs very poorly with respect to maximum delay, because it does 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 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. LD is optimal with respect to end-to-end delays. Figure 2 also shows that maximum end-to-end delays resulting from MSC are almost as good as 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%. LD achieves the least possible failure rate, because it minimizes the end-to-end delay. MSC's failure rate is also very low, < 2%.

All algorithms yield similar performance with respect to the number of nodes in the tree as can be seen from figure 3.

Figure 4 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 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

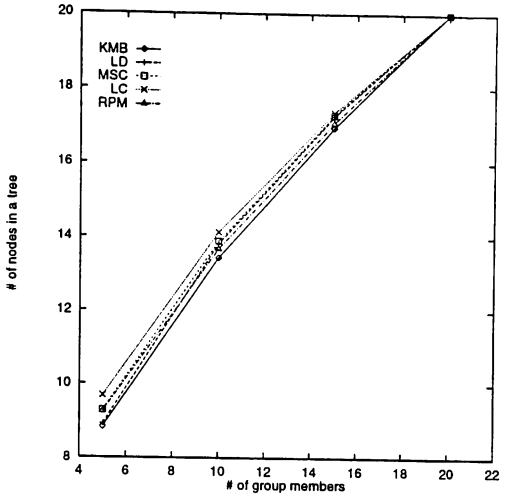


Figure 3: Number of nodes in a MC tree, unconstrained algorithms, 20-node network, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps.

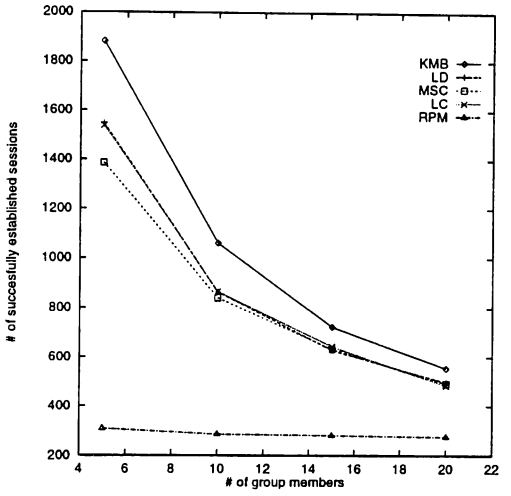
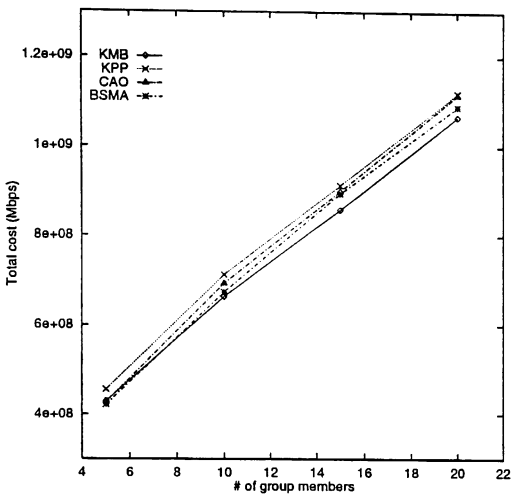


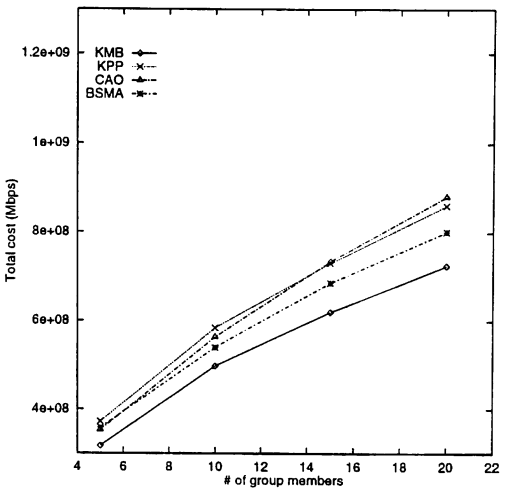
Figure 4: Number of successful sessions, unconstrained algorithms, 20-node network, average degree 4, no delay constraint.

repeated with MC group sizes of 5, 10, 15, and 20 members. 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 runs until the confidence interval for the number of successfully established MC sessions is $< 5\%$ using 95% confidence level. Similar to the first experiment, this experiment keeps changing the random network topology.

It is obvious from figure 4 that as the size of the MC group increases, the number of MC trees that an algorithm can construct before the network saturates decreases, because the number of links in a 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 efficient 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. As has been mentioned before, RPM adds a link e to an MC tree based on the cost of the reverse link e'. If e' is lightly loaded and remains lightly loaded, RPM will keep adding sessions to e, and it will not receive any notice that e is heavily loaded. Even when e saturates, e' will not be notified. Applying RPM results in



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



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

Figure 5: Total cost of a MC tree, CST heuristics, 20-node network, average degree 4, $\Delta = 0.03$ seconds.

extremely asymmetric, inefficient networks.

The two experiments discussed above show that none of the three unconstrained algorithms (KMB, LC, and RPM) can be used with applications having delay constraints. The semi constrained heuristic, MSC, has a delay bound that is too tight which reduces its performance with respect to 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 algorithms in general, and KMB's heuristic for the minimum Steiner tree in particular, and the delay-oriented algorithms, LD and MSC.

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.

4.2 Simulation Results for the CST Heuristics

We re-ran the same first experiment of section 4.1 on KPP, CAO, and BSMA heuristics, to determine the characteristics of the constrained MC trees these heuristics construct.

Figure 5 shows that the three constrained CST yield similar total tree costs. We also show the cost of the unconstrained KMB heuristic for comparison. The costs of the constrained heuristics are on the average only 5–10% more expensive than the unconstrained KMB heuristic, in spite of the delay constraint which limits their flexibility in choosing cheap links. Note that as the range of link load variation increases, all three CST heuristics can create cheaper trees, because, similar to KMB, they can select low-cost links and add them to the MC tree.

It can be seen from figure 6 that the maximum end-to-end delays for the three CST heuristics are below the 0.03 seconds delay bound. Again all three heuristics 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 low delays of the initial tree persist in the final tree. Figure 6 also shows the maximum delays of LD for comparison. LD's maximum delays are optimal and they are considerably less than maximum delay the CST heuristics can achieve. However, this is not a big advantage, because it is sufficient to satisfy the delay constraint. We found that all three CST heuristic are capable of satisfying the delay bound. Their rate of violating the delay bound is as low as LD, i.e. $< 2\%$.

Figure 7 confirms the observation made above that all three CST heuristics create MC trees that have similar performance. We conclude from figures 3 and 7 that the number of nodes in a MC tree is independent

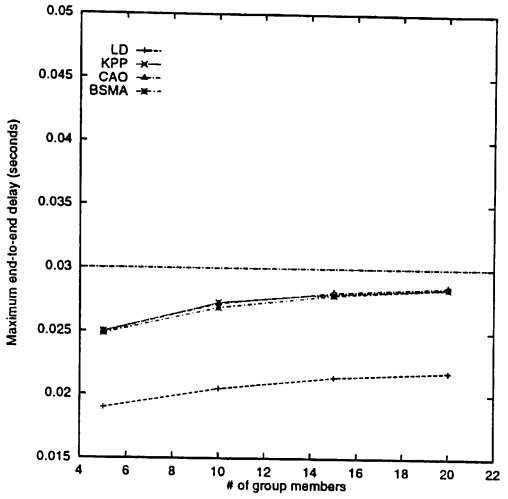


Figure 6: Maximum end-to-end delay, CST heuristics, 20-node network, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds.

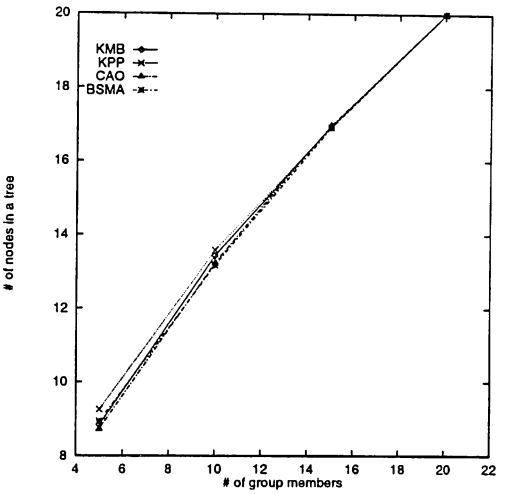


Figure 7: Number of nodes in a MC tree, CST heuristics, 20-node network, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds.

of which MC routing algorithms is used. This conclusion holds only for the link cost function we defined in section 3.

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 couldn't run it, however, on KMB, because it violates the delay bound too frequently.

Figure 8 shows that again the three CST heuristics yield almost identical performance and that they can manage the network bandwidth better than LD and MSC.

Figure 9 shows the average execution times of all algorithms studied in this paper. Note, however, that the code used was not optimized for speed. These results are therefore not conclusive. The figure shows that the running times of the CST heuristics are much larger than the running times of the unconstrained algorithms. KPP is a particularly time consuming algorithm. CAO is the fastest CST heuristic, but its execution time depends on the number of group members. The figure also shows that KMB is the slowest among the unconstrained and semi constrained algorithms.

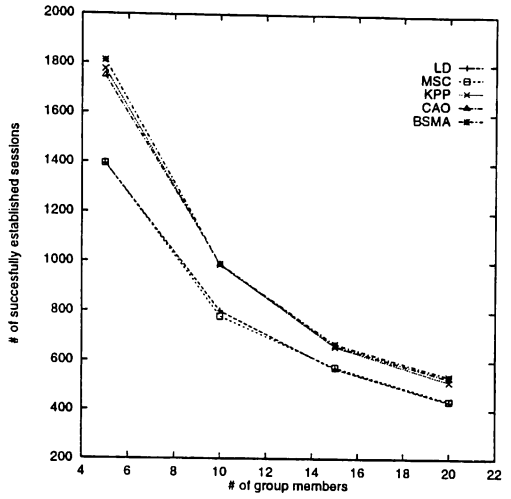


Figure 8: Number of successful sessions, 20-node network, average degree 4, $\Delta = 0.03$ seconds.

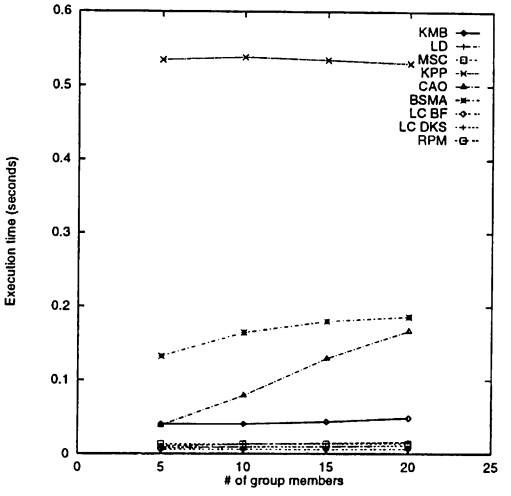


Figure 9: Average execution time, 20-node network, average degree 4, $\Delta = 0.03$ seconds.

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, while RPM performs very poorly. This indicates that the unconstrained algorithms can not be applied to real-time applications on networks spanning large areas due to inadequate delay performance. RPM is not suitable for any networks with asymmetric link loads.

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. CAO is faster than BSMA, and KPP is the slowest. CAO is, however, slower than the unconstrained and semi constrained algorithms. The CST heuristics construct trees that are not considerably more expensive than KMB'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, how scalable is the algorithm to larger networks, is a distributed implementation of the algorithm possible, and what is the average running time of the algorithm.

References

[1] M.R. Macedonia and D.P. Brutzman, "MBone Provides Audio and Video Across the Internet," IEEE Computer, Vol. 27, pages 30–36, April 1994.

[2] D. Bertsekas and R.G. Gallager, "Data Networks," pages 393–404, Second Edition, Prentice-Hall, 1992.

[3] P. Winter, "Steiner Problem in Networks: A Survey," Networks, Vol. 17, pages 129–167, 1987.

[4] R.C. Prim, "Shortest Connection Networks and Some Generalizations," The Bell Systems Technical Journal, Vol. 36, pages 1389–1401, November 1957.

[5] M. Doar and I Leslie, "How Bad is Naive Multicast Routing," Proceedings of INFOCOM '93, pages 82–89, 1993.

[6] D. Wall, "Selective Broadcast in Packet-Switched Networks," Proceedings of the Sixth Berkeley Workshop on Distributed Data Management and Computer Networks, pages 239–258, February 1982.

[7] J. Moy, "Multicast Extension to OSPF," Internet Draft, September 1992.

[8] S. Deering and D. Cheriton, "Multicast Routing in Datagram Networks and Extended LANs," ACM Trans. on Computer Systems, Vol. 8, No. 2, pages 85–110, May 1990.

[9] S. Deering et. al., "Protocol Independent Multicast (PIM): Protocol Specification," Internet Draft, January 1995.

[10] A.G. Waters, "A New heuristic for ATM Multicast Routing," Proceedings of the 2nd IFIP Workshop on Performance Modelling and Evaluation of ATM Networks, pages 8.1–8.9, July 1994.

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

[12] C.A. Noronha Jr. and F.A. Tobagi, "Optimum Routing of Multicast Streams," Proceedings of INFOCOM '94, pages 865–873, 1994.

[13] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos, "Multicast Routing for Multimedia Communication," IEEE/ACM Trans. on Networking, Vol. 1, No. 3, pages 286–292, June 1993.

[14] R. Widyono, "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, June 1994.

[15] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting," Submitted to INFOCOM '95, April 1995.

[16] B.M. Waxman, "Routing of Multipoint Connections," IEEE Journal on Selected Areas in Communications, Vol. 6, No. 9, pages 1617–1622, December 1988.