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

Hussein F. Salama Douglas S. Reeves Yannis Viniotis
North Carolina State University
Raleigh, NC 27695

Abstract

Multicast (MC) routing algorithms capable of satisfying the quality of service (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 trees 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. Simulations also reveal that one of the unconstrained algorithms, reverse path multicasting (RPM), is quite inefficient when applied to asymmetric networks. We study how combining routing with resource reservation and admission control improves RPM's efficiency in managing the network resources. The performance of one semiconstrained heuristic, MSC, three constrained Steiner tree (CST) heuristics, KPP, CAO, and BSMA, and one constrained shortest path tree (CSPT) heuristic, CDKS are also studied. Simulations show that the semiconstrained and constrained heuristics are capable of successfully constructing MC trees which satisfy the QoS requirements of real-time traffic. However, the cost performance of the heuristics varies. BSMA's MC trees are lower in cost than all other constrained heuristics. Finally, we compare the execution times of all algorithms, unconstrained, semiconstrained, and constrained.

*This work was supported in part by IBM SUR project #1429, in part by the Center for Advanced Computing and Communication at North Carolina State University, and in part by AFOSR grants F49620-92-J-0441DEF and F49620-96-1-0061. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFOSR or the U.S. Government.

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. Bertsekas and Gallager [1] define the routing function as consisting of two parts. The first part selects a route for the session during the connection establishment phase, and the second part ensures that each packet of that session is forwarded along the assigned route. In this paper, we consider only the route selection algorithms.

In the past, very few network 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 [2]. So the question now is: are the simple MC routing algorithms, which were originally designed for best-effort networks, suitable for networks carrying real-time applications?

A number of new MC routing algorithms designed specifically for real-time applications were proposed during the past few years. However, there has not been a study yet that applies all of these algorithms in a realistic networks environment and provides a fair quantitative comparison of all algorithms under identical networking conditions. Such a study is necessary to determine whether or not these algorithms are capable of constructing MC trees with characteristics suitable for real-time applications, e.g., low end-to-end delays, and how efficient they are in managing the network resources. A MC routing algorithm that constructs trees, with characteristics suitable for real-time applications, together with the appropriate scheduling, forwarding, and policing mechanisms can provide QoS guarantees for

real-time applications.

In this paper, we evaluate the ability of the simple MC routing algorithms, which are used in current wide-area networks, to satisfy the requirements of real-time applications. In addition, we compare the performance of the new algorithms which were designed specifically for real-time 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 costs 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].

1.1 Definitions

A communication network is represented as a directed connected simple graph $N = (V, E)$, where V is a set of nodes and E is a set of directed links. The existence of a link $e = (u, v)$ from node u to node v implies the existence of a link $e' = (v, u)$ for any $u, v \in V$, i.e., full duplex in networking terms. 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 nonnegative 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 $G = \{g_1, ..., g_n\} \subseteq V$, where $n = |G| \leq |V|$, is a set of nodes participating in the same network activity, and is identified by a unique group address i. A node $s \in V$ is a MC source for the MC group G. A MC source s may or may not be itself a member of the group G. A MC tree $T(s, G) \subseteq E$ is a tree rooted at the source s and spanning all members of the group G. The total cost of a tree $T(s, G)$ is simply $\sum_{t \in T(s,G)} C(t)$. A path $P(s, g) \subseteq T(s, G)$ is a set of links connecting s to $g \in G$. The cost of the path $P(s, g)$ is $\sum_{p \in P(s,g)} C(p)$ and the end-to-end delay along that path is $\sum_{p \in P(s,g)} D(p)$. Thus the maximum end-to-end delay of a MC tree is $\max_{g \in G}(\sum_{p \in P(s,g)} D(p))$.

[1]The terms asymmetric and directed are used interchangeably in this paper, and similarly the terms symmetric and undirected

1.2 Classification of MC Routing Algorithms

In this section, we briefly discuss the different classes of MC routing algorithms which are considered in this paper, and provide examples of algorithms representing each class[2]. MC routing algorithms can be classified into one of two categories. The first category is the shortest path algorithms which minimize the cost of each path from the source node to a multicast group member node. Bellman-Ford's algorithm and Dijkstra's algorithm are two well known shortest path algorithms [1]. They are the basis of the distance vector and link state routing protocols respectively. The other category is the minimum Steiner tree algorithms. Their objective is to minimize the total cost of the MC tree. This problem is known to be *NP*-complete [4]. Hwang [5] provides a survey of both exact and heuristic minimum Steiner tree algorithms. Efficient minimum Steiner tree heuristics are given in [6, 7, 8, 9]. If the destination set of a minimum Steiner tree includes all nodes in the network, the problem reduces to the minimum spanning tree problem which can be solved in polynomial time [10]. An analytical study of the tradeoffs between shortest path trees and minimum Steiner trees can be found in [11].

In order to support real-time applications, network protocols must be able to provide QoS guarantees. For example, a guaranteed upper bound on end-to-end delay, Δ, must be provided to certain distributed multimedia applications. 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 an efficient MC tree without violating the constraint implied by the upper bound on delay. These are called delay-constrained algorithms to distinguish them from other algorithms. Optimal algorithms for constructing delay-constrained minimum Steiner trees exist [12][3], but their average execution times are prohibitively large, because the problem is *NP*-complete [13]. Several delay-constrained Steiner tree heuristics have been proposed during the past few years [13, 14, 15]. The heuristics proposed in [14] use a delay-constrained Bellman-Ford shortest path algorithm during the computation of the delay-constrained Steiner tree. Sun and Langendoerfer [16] present a delay-constrained heuristic based on Dijkstra shortest path algorithm. A heuristic that constructs a MC tree subject to both an upper bound on end-to-end delay and an upper bound on delay variation[4] is given in [17]. In the remainder of this paper, we refer to delay-constrained MC

[2]The reader is referred to [3] for a complete classification and a complete survey of MC routing algorithms for communication networks.

[3]The algorithm in [12] constructs multiple optimal delay-constrained minimum Steiner trees simultaneously.

[4]Delay variation is the difference between the minimum and the maximum end-to-end delays on the same tree.

routing algorithms simply as constrained algorithms.

In shortest reverse path MC trees, each path from the source node to a destination node in the MC tree is the shortest path from that destination to the source. Such a MC tree is an optimal shortest path tree only if the link costs are symmetric. Several MC routing protocols [20, 21, 22] use algorithms which construct shortest reverse path trees, because: they require limited state information at each node and they can be implemented distributedly.

Other classifications of MC routing algorithms are based on the implementation of the algorithms: centralized or distributed, and whether or not an algorithm permits nodes to join and leave a MC group dynamically. 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, both unconstrained and constrained, are evaluated based on their ability to provide performance guarantees, the quality of the MC trees they construct, their effectiveness in managing the network resources, and their time complexity. Only static MC groups are considered. Implementation issues such as distributing the algorithms and the amount of state information needed at each node are not addressed in this work. The remainder of 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 a 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 four constrained MC routing heuristics. 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, 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 path algorithm [1] 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 $O(|V|^2)$.

Some of the algorithms studied define parameters that can be adjusted to trade off tree cost for fast execution times or accuracy for fast execution times. Due to the space and time limitations, we only study the most accurate, most cost efficient version of an algorithm, such that its execution times are not prohibitively large.

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 [5], 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 [23] show that the total cost of trees generated using the Kou, Markowsky, and Berman, **KMB**, heuristic [7] for the minimum Steiner tree is on the average only 5% worse than the cost of the optimal solution while its time complexity is $O(|G||V|^2)$. 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 path algorithm is used in communication protocols, e.g., MOSPF [24], 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 to determine its applicability to networks that are heavily loaded with multimedia applications.

As has been mentioned in the previous section, several MC routing protocols adopt algorithms that construct shortest reverse path MC trees. We study only one of these algorithms, namely reverse path multicasting, **RPM** [18, 19]. RPM is a distributed, dynamic algorithm which requires only limited state information to be stored at each node.

2.2 Constrained Algorithms

The first three algorithms described in this section are constrained Steiner tree (CST) heuristics, the fourth algorithm is a constrained shortest path tree (CSPT) heuristic, and the last one is a semiconstrained Steiner tree heuristic.

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 noninteger values, Kompella et al. propose to multiply out fractional values to get integers. Following this approach, KPP is guaranteed to construct a constrained tree if one exists. However, in some cases the granularity of the delay bound becomes very small, and hence the number of bits required to represent it increases considerably. As a result the order of complexity, $O(\Delta|V|^3)$, may become too high. To avoid prohibitively large computation times, we use a fixed granularity of $\Delta/10$ throughout our experiments. However, fixing the granularity has side effects. When the granularity is comparable to the average link delays, KPP's accuracy is compromised and in many cases it can not find a solution for the CST problem when one exists.

Both KMB and KPP heuristics use Prim's algorithm [10] to obtain a minimum spanning tree of a closure graph. However, Prim's algorithm is only optimal for undirected networks, and this might affect the performance of the two heuristics when applied to directed networks.

Widyono [14] proposed four unconstrained MC heuristics and four CST heuristics. The four CST 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 the constrained Bellman-Ford algorithm, the unconnected member with the minimum-cost 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 [15], **BSMA**, is the third CST heuristic we study in this work. BSMA starts by computing a LD tree for a given source s and MC group G. Then it iteratively replaces superedges[5] in $T(s, G)$ with lower cost 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 lower cost superedges. It runs in

[5] 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.

$O(k|V|^3 \log|V|)$ time. 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 superedge replacement 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. It is possible to trade off MC tree cost for fast execution speed when using BSMA by either limiting the value of k in the kth-shortest path algorithm or by limiting the number of superedge replacements. In our experiments, we neither limit the value of k nor limit the number of superedge replacements, because the objective of our work is to achieve the best possible cost performance while satisfying the delay bound.

A CSPT heuristic is proposed in [16]. This heuristic computes an unconstrained LC tree. If the end-to-end delay to any group member violates the delay bound, the path from the source to that group member is replaced with the least-delay path. Thus if the LC tree violates the delay bound, a LD tree must be computed and the two trees are merged. Again, this algorithm will always find a constrained MC tree if one exists. This CSPT heuristic is $O(|V|^2)$, because it uses Dijkstra's algorithm for computing both the LC and LD trees. We call it the constrained Dijkstra heuristic, **CDKS**.

We consider the minimum Steiner tree heuristic proposed by Waters [25] to be semi-constrained, because 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's 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 multicast nodes. We call this the semiconstrained (SC) heuristic. In [26], we implemented the original algorithm proposed in [25] which resembles a semi-constrained minimum spanning tree, and we also implemented a modified version which is closer to a semi-constrained shortest path tree. Simulation results given in [26] showed that the modified version, denoted as the modified semiconstrained, **MSC**, heuristic always performs better than the original heuristic with respect to tree costs, end-to-end delays, and network balancing. Therefore, it suffices to study MSC. SC and MSC are dominated by the computation of the internal delay bound. This computation uses an extension to Dijkstra's algorithm, and therefore it takes $O(|V|^2)$ time in the worst case.

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.

Full duplex ATM networks with homogeneous link capacities of 155 Mbps (OC3) were used in the experiments. The positions of the nodes were fixed in a rectangle of size $3000 * 2400$ Km^2, roughly the area of the USA. A random generator [3] (based on Waxman's generator [27] with some modifications) was used to create links interconnecting the nodes. The output of this random generator is always a connected network in which each node's degree is ≥ 2. Therefore the output is always a two-connected network[6]. Noronha and Tobagi [28] have shown, using simulation, that the performance of a MC routing algorithm when applied to a real network is almost identical to its performance when applied to a random two-connected network. We adjusted the parameters of the random generator such that, similar to real networks, the probability of existence of a short link is larger than the probability of existence of a longer link. We also adjusted the parameters of the random generator to yield networks with an average node degree of 4 which is approximately the average node degree of current networks. Figure 1 shows an example of a randomly generated 20-node network.

Each node represented a non-blocking ATM switch, and each link had a small output buffer. The propagation speed through the links 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. Any session traversing a link e, reserved a fraction of e's bandwidth equal to the equivalent bandwidth [29] of the traffic it generated. The link cost, $C(e)$, was taken 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. Therefore, the cost of a heavily utilized link was larger than the cost of a lightly utilized link. $C(e)$ was dynamic, and varied as new sessions were established or existing sessions were torn down.

A link could accept sessions and reserve bandwidth for them until its cost, i.e., the sum of the equivalent bandwidths of the sessions traversing that link, exceeded 85% of the link's capacity, then it got saturated. This admission control policy allowed statistical multiplexing and efficient utilization of the available resources. More sophisticated admission

[6] A two-connected network has at least two paths between any pair of nodes.

control policies for real-time traffic exist, but the simple policy just described was sufficient for the purposes of our study. A detailed study of admission control algorithms for real-time traffic can be found in [30].

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 the 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 low-cost, lightly utilized links.
- The maximum end-to-end delay from the source to any MC group member. It indicates the algorithm's ability to satisfy the delay bound imposed 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 each of the algorithms discussed in section 2. We present the simulation results for the unconstrained algorithms first, in section 4.1, in order to determine the conditions, if any, under which the unconstrained algorithms do not perform well. Then in section 4.2, we study the performance of RPM. Finally, in section 4.3, we show the results obtained for the constrained algorithms, 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 given source node generating video traffic with an equivalent bandwidth of 0.5 Mbps, and a given MC group. For each run of

the experiment we generated a random set of links to interconnect the fixed nodes, we generated random background traffic for each link, we selected a random source node and a MC group of randomly chosen destination nodes. The equivalent bandwidth of each link's background traffic was a random variable uniformly distributed between B_{min} and B_{max}. As the range of the link loads, i.e., the difference between B_{max} and B_{min}, increased, the asymmetry of the link loads also increased, because the load on link $e = (u, v)$ is independent of the load on the link $e' = (v, u)$. The experiment was 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 was considered a failure and was rejected and removed, but not before we measured its characteristics. Thus the total cost and maximum end-to-end delay of MC trees which fail to satisfy the delay bound were included in the cost and delay measurements. The experiment was run repeatedly until confidence intervals of less than 5%, using 95% confidence level, were achieved for all measured quantities. On the average, 300 different networks were simulated in each experiment in order to reach such confidence levels. At least 250 networks were simulated in each case.

Figure 2 shows the percentage increase in total cost of an unconstrained MC tree relative to optimal versus the MC group size for two different link loading conditions for 20-node networks. When the range of link loads is larger, an efficient algorithm should be able to locate lower cost links and use them to construct lower cost MC trees. KMB heuristic yields very low tree costs. Note, however, that in the more asymmetric case (figure 2(b)) KMB's costs are approximately 10% worse than OPT, which is not as good as its performance when applied to symmetric networks [23]. LC does not perform as well as KMB, because it attempts to minimize the cost per path from source to destination, not the total cost of the entire tree. LC's costs are up to 40% worse than OPT. LD yields the most expensive trees, and the cost of the least-delay trees is independent of the range of the link loads. That is why its performance relative to optimal deteriorates as the range of link loads increases. We repeated the same experiment using larger networks. However, OPT could not be applied to networks with more than 20 nodes due to its excessive running times, so we had to measure the percentage increase in the total cost of a MC tree relative to the total cost of the second best unconstrained algorithm after OPT, namely KMB. Figure 3 shows the cost performance of the algorithms when applied to 200-node networks. Comparing this figure with figure 2 indicates that the cost performance of the algorithms relative to each other remains approximately unchanged, as the network size increases.

Figure 4 shows the maximum end-to-end delay for 20-node, and 200-node networks[7]. OPT and KMB perform 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 and the delay along that path. 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. The maximum end-to-end delays increase as the network size increases, because the paths connecting two randomly chosen nodes consist of fewer, but longer links (smaller end-to-end delay) in case of small networks, while in case of large networks, these paths consist of more shorter links (larger end-to-end delay).

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 OPT, KMB, and even LC to have very high failure rates, $> 30\%$ in case of 20-node networks. LD's failure rate, however, is $< 2\%$ in case of 20-node networks.

In the second experiment, we started with a completely unloaded network and kept adding MC sessions and constructing the corresponding MC trees until the cumulative tree failure rate exceeded 15%. A MC session consisted of a random source node generating VBR video traffic with an equivalent bandwidth 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. Our objective here was to determine how efficiently these unconstrained algorithms manage the network in the absence of a delay bound. The experiment was repeated, until the confidence interval for the number of successfully established MC sessions was $< 5\%$ using the 95% confidence level. Similar to the first experiment, in this experiment a random network topology was generated before each run. All algorithms discussed in this section perform routing, admission control, and resource reservation simultaneously. Therefore, these algorithms do not add saturated links to the MC trees being constructed. Instead, they search for alternate links or paths that are not yet saturated. This experiment could not be applied to the optimal

[7]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 the link loads.

minimum Steiner tree algorithm, OPT, because of its large execution time.

Figure 5 shows the results of the second experiment. 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 size of a MC tree increases as the group size increases. KMB yields the best performance, because it has the ability to locate the lowest cost 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 and LC can also manage the network resources efficiently, although not as efficiently as KMB. LD's performance is no more than 20% worse than KMB's performance. Combining admission control and resource reservation with routing allows LD to find alternate routes when links on the absolute least-delay paths are saturated. That is why LD performs as good as LC although it is not as efficient as LC in constructing low-cost trees.

The first experiment shows that the unconstrained algorithms (OPT, KMB, and LC) are not satisfactory for applications having delay constraints. Therefore, such algorithms will not be suitable for future high-speed networks. LD is optimal with respect to minimizing delays but it does not attempt to optimize the tree cost at all. The second experiment shows that all algorithms discussed so far are efficient network managers with KMB being the best. We will study the constrained algorithms in section 4.3 to determine if they can achieve a compromise between the unconstrained cost-oriented algorithms (OPT, KMB, and LC) and the delay-oriented algorithms (LD), but first we will study the performance of one more unconstrained MC routing algorithm, RPM.

4.2 RPM's Performance

We pointed out in section 1.2 that RPM generates reverse shortest path trees, i.e., trees in which the reverse paths from each destination back to the source are least-cost. If the link costs are symmetric, the costs of the resulting forward paths from the source to the destinations will also be least-cost, and RPM will construct exactly the same trees as the LC shortest path algorithm. In this section, we compare RPM's reverse shortest path trees to LC's forward shortest path trees. First we show the results of the first experiment described in the previous section for both algorithms.

Figure 6 shows the percentage increase in total cost of RPM and LC generated MC trees relative to KMB for 200-node networks. When the asymmetry of the network is small, RPM's costs are only slightly more than LC's costs. As the range of the link loads increases and hence the asymmetry of the network increases, however, the costs

of RPM's trees do not change while LC is capable of finding much lower cost trees. The figure shows that RPM is less than 30% worse than KMB when the network asymmetry is small. When the network asymmetry is large, however, RPM is up to 80% worse than KMB as can be seen from figure 6(b). Thus it is obvious that RPM's approach of using the reverse link's cost as an estimate of the forward link's cost is ineffective for asymmetric networks.

The maximum end-to-end delays along RPM's MC trees are the same as those along LC's MC trees as can be seen from figure 7. This is because the average lengths of the reverse shortest paths and the forward shortest paths are equal, and thus propagation delays are equal in both directions.

RPM is used in practice [21], because it requires only limited information to be available at each node in order to construct a reverse shortest path MC tree. Current implementations of RPM do not perform routing, resource reservation and admission control at the same stage. Currently, separate resource reservation protocols [31] are applied to perform admission control tests and reserve resources on the MC trees constructed by the routing protocols. If resource reservation fails due to the existence of saturated links in a MC tree, then the MC session can not be established, because none of the existing RPM-based routing protocols is capable of finding alternate links or paths to replace the saturated links. We implemented a version of RPM that separates routing from resource reservation and admission control to imitate the situation just described. We also implemented another version that performs routing, resource reservation, and admission control simultaneously. We call these two algorithms RPM_SEP and RPM_COMB respectively[8]. Similarly, we implemented two versions of the shortest path algorithm, LC. The first one, LC_SEP, separates routing from resource reservation and admission control while the second version, LC_COMB, combines routing with resource reservation and admission control. We ran the second experiment of the previous section on these four algorithms to determine RPM's efficiency in managing the available link bandwidth, and to examine the effect of separating routing from resource reservation and admission control.

Figure 8 shows the number of successfully established sessions versus the MC group size for 20-node networks. Both versions of LC yield good performance, because the routing part of LC always uses low-cost links to construct the MC trees. Therefore, the load on the links increases gradually, and the difference between the minimum link load and the maximum link load at any time is small. Thus, the algorithm is capable of constructing a large number of trees before any links saturate and admission control comes into play. When links start to saturate, LC_COMB is capable

[8]Both algorithms give identical results when applied to experiment 1, because in that experiment resources are always available.

of using alternate paths when it fails to add a saturated link to a tree. That's why LC_COMB performance is better than LC_SEP's performance. RPM_SEP is very inefficient even for small group sizes. As has been mentioned before, RPM_SEP adds a link $e = (u, v)$ to an MC tree based on the cost of the reverse link $e' = (v, u)$. If e' is lightly utilized and remains lightly utilized, RPM will keep adding sessions to e, and it will not receive any indication that e is heavily utilized. This leads to extremely asymmetric link loads. Even when the forward link e saturates, RPM_SEP will not be notified, and it will still attempt to construct MC trees containing e. Such trees will be later rejected by admission control. Applying RPM_SEP results in extremely inefficient management of the network bandwidth. RPM_COMB causes very asymmetric link loads similar to RPM_SEP, but the close interaction between routing and admission control enables it to find alternate paths to replace saturated links. This improves RPM_COMB's efficiency to the extent that it performs as good as LC_COMB. Thus combining resource reservation and admission control with routing leads to more efficient management of the available resources.

4.3 Simulation Results for the Constrained Algorithms

The results given in section 4.1 and 4.2 show that the unconstrained MC routing algorithms are not capable of providing satisfactory service to applications with delay constraints. In this section, we study the performance of the constrained algorithms to determine the characteristics of the MC trees they construct.

We re-ran the same first experiment of section 4.1 on COPT, CDKS, the three CST heuristics, and MSC. Figure 9 shows the percentage increase in the total cost of a constrained (or semiconstrained) MC tree relative to the total cost of COPT in case of 20-node networks. When the range of the link loads is larger, the difference in performance between the algorithms is larger. BSMA yields better cost performance than KPP and CAO. For the scenarios we simulated, BSMA's trees costs are less than 7% more expensive than optimal, while CAO and KPP's trees are up to 15% more expensive than optimal. CDKS and MSC generate trees that are always more expensive than the trees of the CST heuristics. CDKS's and MSC's costs are up to 25% and 45% more than COPT when the range of link loads is large. MSC's internally generated delay bound is so strict for the cases we studied that it restricts the algorithm's ability to minimize the tree costs. We repeated the same experiment using networks with more than 20 nodes, but we could not apply COPT to these networks due to its excessive execution times. The percentage increase in the total costs of the constrained algorithms relative to BSMA (the second best constrained algorithm after COPT) when applied to

100-node networks[9] are shown in figure 10. It is evident from figures 9 and 10 that the performance of the different algorithms relative to each other remains unchanged as the size of the network increases.

It can be seen from figure 11 that the maximum end-to-end delays for the constrained algorithms are below the 0.03 seconds delay bound. All constrained algorithms yield similar delay performance, but CDKS is slightly better. This is because CDKS constructs a LC tree and then replaces paths which violate the delay bound with paths from the LD tree. Figure 11 also shows the maximum delays of LD for comparison. LD's maximum delays are considerably less than the maximum delays the constrained algorithms can achieve. MSC's maximum delays are comparable to LD's maximum delays. However, this is not a big advantage, because it is sufficient to simply satisfy the delay constraint.

We found that the granularity we chose for KPP ($\Delta/10 = 0.003$ seconds) is sufficient in the case of 20-node networks. In that case, KPP's success rate in constructing a constrained MC tree is almost as high as the optimal success rate achieved by COPT and LD. We noticed however that KPP's success rate in constructing a constrained tree is up to 5% worse than optimal for 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 accuracy, and hence its success rate.

We conducted the second experiment of section 4.1 on the constrained 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 LD. We could not run it on COPT, however, because of its large execution time. Figure 12 shows that the three CST heuristics yield similar performance and that they can manage the network bandwidth efficiently, with BSMA being the best. CDKS, LD and MSC are not as efficient as the CST heuristics.

In summary, all constrained algorithms, and the semiconstrained algorithm, can meet the delay requirements of real-time applications, and thus they would all be suitable for high-speed networks. What differentiates them, of course, is their costs and execution times, which we study next.

[9] We could not apply BSMA to networks with more than 100 nodes, because its execution times become too large as will be shown in the next section.

4.4 Execution Times

Figures 13, 14, and 15 show the average execution times of all algorithms studied in this paper. These execution times were measured when running the first experiment discussed in the previous sections. Note, however, that the code used for the algorithms was not optimized for speed. Figures 13 and 14 show the execution times for 20-node and 100-node networks with variable MC group sizes, while figure 15 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 13. LD, LC, and RPM[10] are the fastest algorithms. The running times of the CST heuristics are large, except CAO's running time for small group sizes. CAO's running time increases almost linearly as the group size increases, because it runs the constrained Bellman-Ford algorithm once for each group member not already in the tree. Note from figure 15(b) that CAO's growth rate is slower than MSC's and CDKS's growth rates. This holds for small MC group sizes only. BSMA's running time is very large for 100-node networks as shown in figure 14(b). It is particularly slow for MC groups 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 of superedges is large and at the same time the number of nodes outside the tree is also large, which leads to the long running times of BSMA. CDKS and MSC are as fast as the unconstrained algorithms. Therefore, the three CST heuristics may be too slow, in spite of their high efficiency, to use on networks with thousands of nodes, while the less efficient but much faster CDKS and MSC may be better suited for large networks.

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

[10] We implemented a centralized version of RPM.

evaluation of all these algorithms under realistic high-speed networking environments.

We 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. LC is also unable to satisfy the required delay bound. KMB is more efficient in managing the network bandwidth, than LC.

RPM performs poorly when applied to networks with asymmetric link loads. It creates expensive MC trees and is very inefficient in managing the network bandwidth, because it results in very asymmetric link loads. Current implementations of RPM do not contain a resource reservation and admission control module. Resource reservation is a separate protocol that has minimal interaction with routing, and thus it is currently not possible to select alternate paths to replace any saturated links in the MC tree when they get rejected by admission control. We have shown that incorporating RPM together with admission control and resource reservation in a single module dramatically improves RPM's efficiency in managing the available network bandwidth. Simulation results have also shown that, similar to the other unconstrained algorithms, RPM is not capable of satisfying the delay bounds imposed by real-time applications. Note, however that RPM is a fast distributed dynamic algorithm, and therefore the simplest to implement and maintain among all algorithms studied in this paper.

We concluded that the unconstrained MC routing algorithms, KMB, LC, and RPM, can not be applied to real-time applications on networks spanning large areas. Then we studied a semiconstrained algorithm and four constrained algorithms: three CST heuristics and one CSPT heuristic. All three CST heuristics construct low-cost trees, which satisfy the given delay bound, and manage the network resources efficiently, but BSMA is the best. The execution times of the CST heuristics differ considerably. As the networks size increases BSMA's average execution time grows much faster than KPP's and CAO's average execution times. CAO has fast execution times in case of small group sizes. Note however, that worst case execution times of both KPP and CAO grow exponentially with the network size, while the worst case execution times of BSMA are polynomial in the network size. Overall, we conclude that all three CST heuristics, though efficient, may be too complex to apply to large-scale networks.

The CSPT heuristic, CDKS, does not perform as good as the the three CST heuristics, but its tree costs are always within 25% from optimal. The semiconstrained algorithm, MSC, is always capable of constructing delay-constrained multicast tree in the scenarios we studied. However, using a strict internally computed delay bound limits

the algorithm's ability to construct low-cost MC trees. However, both CDKS and MSC have fast execution times and scale well to large network sizes. Overall, CDKS achieves a good compromise between reasonable tree costs and fast execution times.

The need for delay-constrained algorithms is evident from the experiments presented in this paper. We suggest that any future work on delay-constrained MC routing should focus on simple, fast algorithms. Distributed, scalable implementations of such algorithms must be proposed in order for them to have the potential of being adopted by future MC routing protocols. In this paper, we considered a static set of MC group members. Allowing nodes to join and leave an existing MC group dynamically is another feature that should be considered in future work on delay-constrained MC routing.

6 Acknowledgment

We would like to thank Prabhu Manyem for his assistance with the optimal minimum Steiner tree algorithms: OPT and COPT. We would also like to thank the reviewers of this paper for their insightful comments.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall, 2nd ed., 1992.

[2] M. Macedonia and D. Brutzman, "MBone Provides Audio and Video Across the Internet," *IEEE Computer*, vol. 27, no. 4, pp. 30–36, April 1994.

[3] H. Salama, *Multicast Routing for Real-time Communication on High-Speed Networks*. PhD thesis, North Carolina State University, Department of Electrical and Computer Engineering, 1996. In Preparation.

[4] R. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations* (R. Miller and J. Thatcher, eds.), pp. 85–103, Plenum Press, 1972.

[5] F. Hwang and D. Richards, "Steiner Tree Problems," *Networks*, vol. 22, no. 1, pp. 55–89, January 1992.

[6] S. Ramanathan, "An Algorithm for Multicast Tree Generation in Networks with Asymmetric Links," in *Proceedings of IEEE INFOCOM '96*, pp. 337–344, 1996.

[7] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[8] V. Rayward-Smith, "The Computation of Nearly Minimal Steiner Trees in Graphs," *International Journal of Mathematical Education in Science and Technology*, vol. 14, no. 1, pp. 15–23, January/February 1983.

[9] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Mathematica Japonica*, vol. 24, no. 6, pp. 573–577, February 1980.

[10] R. Prim, "Shortest Connection Networks and Some Generalizations," *The Bell Systems Technical Journal*, vol. 36, no. 6, pp. 1389–1401, November 1957.

[11] K. Barath-Kumar and J. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Transactions on Communications*, vol. COM-31, no. 3, pp. 343–351, March 1983.

[12] C. Noronha and F. Tobagi, "Optimum Routing of Multicast Streams," in *Proceedings of IEEE INFOCOM '94*, pp. 865–873, 1994.

[13] V. Kompella, J. Pasquale, , and G. Polyzos, "Multicasting for Multimedia Applications," in *Proceedings of IEEE INFOCOM '92*, pp. 2078–2085, 1992.

[14] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real-Time Channels," Tech. Rep. ICSI TR-94-024, University of California at Berkeley, International Computer Science Institute, June 1994.

[15] Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting," in *Proceedings of IEEE INFOCOM '95*, pp. 377–385, 1995.

[16] Q. Sun and H. Langendoerfer, "Efficient Multicast Routing for Delay-Sensitive Applications," in *Proceedings of the Second Workshop on Protocols for Multimedia Systems (PROMS)*, pp. 452–458, October 1995.

[17] G. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," in *Proceedings of IEEE INFOCOM '96*, pp. 353–360, 1996.

[18] Y. Dalal and R. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, December 1978.

[19] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 85–110, May 1990.

[20] A. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain

Multicast Routing," in *Proceedings of ACM SIGCOMM '93*, pp. 85–95, September 1993.

[21] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol." Internet RFC 1075, `http://ds.internic.net/rfc/rfc1075.txt`, November 1988.

[22] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 153–162, April 1996.

[23] M. Doar and I. Leslie, "How Bad is Naive Multicast Routing," in *Proceedings of IEEE INFOCOM '93*, pp. 82–89, 1993.

[24] J. Moy, "MOSPF, Analysis and Experience." Internet RFC 1585, `http://ds.internic.net/rfc/rfc1585.txt`, May 1994.

[25] A. Waters, "A New Heuristic for ATM Multicast Routing," in *Proceedings of the Second IFIP Workshop on Performance Modeling and Evaluation of ATM Networks*, pp. 8.1–8.9, July 1994.

[26] H. Salama, D. Reeves, Y. Viniotis, and T.-L. Sheu, "Comparison of Multicast Routing Algorithms for High-Speed Networks," Tech. Rep. TR 29.1930, IBM, September 1994.

[27] B. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, December 1988.

[28] C. Noronha and F. Tobagi, "Evaluation of Multicast Routing Algorithms for Multimedia Streams," in *Proceedings of IEEE International Telecommunications Symposium*, August 1994.

[29] R. Guerin, h. Ahmadi, and M. Naghshineh, "Equivalent Capacity and its Application to Bandwidth Allocation in High-speed Networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 7, pp. 968–981, September 1991.

[30] S. Rampal and D. Reeves, "An Evaluation of Routing and Admission Control Algorithms for Multimedia Traffic," *Computer Communications*, vol. 18, no. 10, pp. 755–768, October 1995.

[31] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang, "An Architectural Comparison of ST-II and RSVP," in *Proceedings of IEEE INFOCOM '94*, 1994.

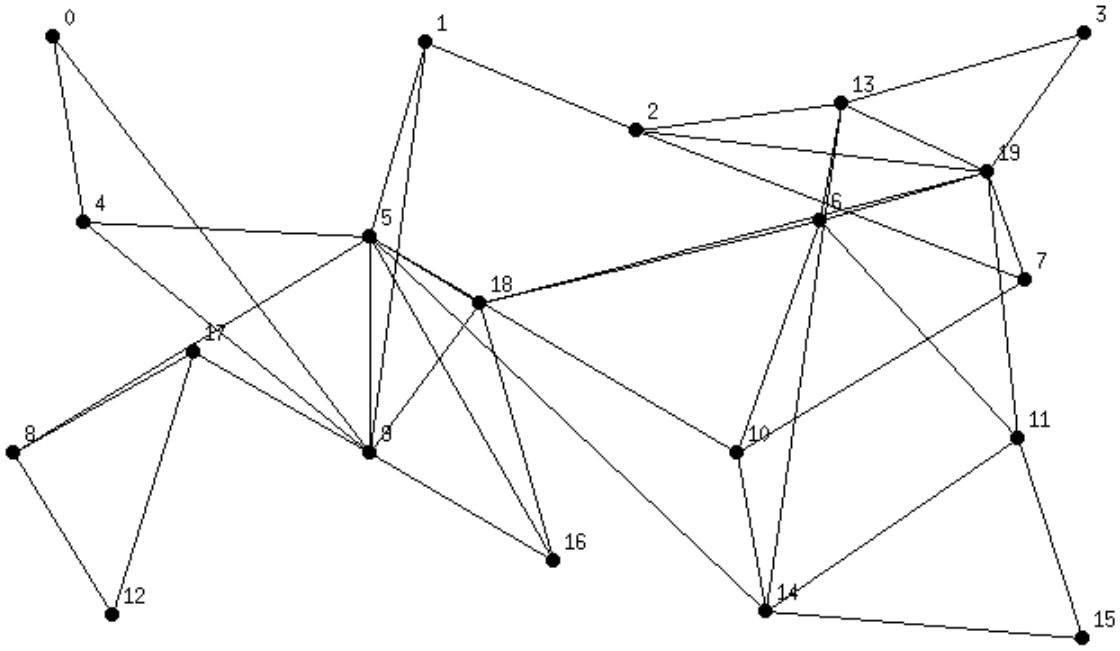
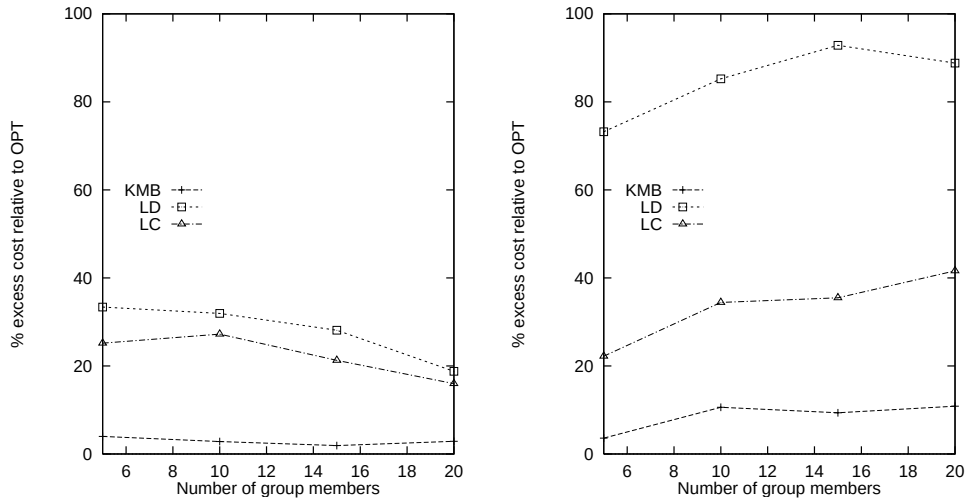


Figure 1: A randomly generated network, 20 nodes, average degree 4.



(a) (b)

Figure 2: Total cost of a MC tree relative to optimal, unconstrained algorithms, 20 nodes, average degree 4. (a) B_{min} = 45 Mbps, B_{max} = 85 Mbps. (b) B_{min} = 5 Mbps, B_{max} = 125 Mbps.

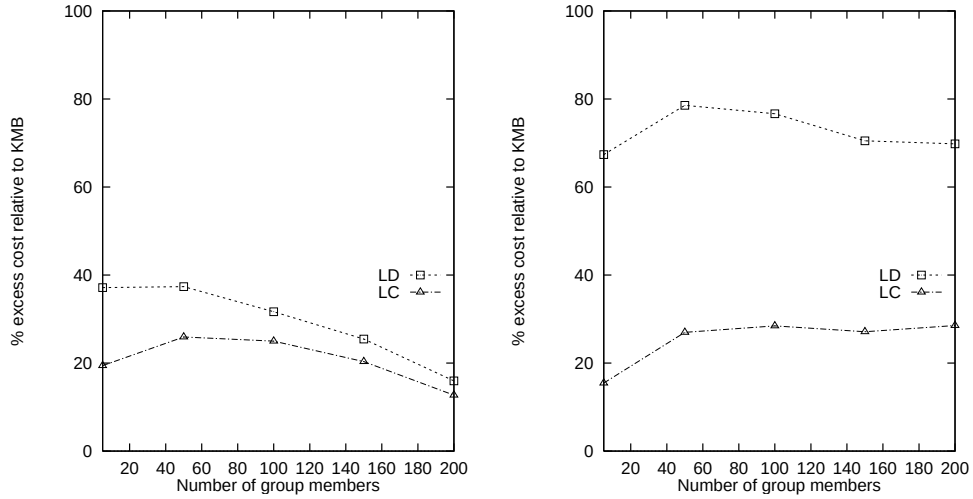


Figure 3: Total cost of a MC tree relative to KMB, unconstrained algorithms, 200 nodes, average degree 4. (a) B_{min} = 45 Mbps, B_{max} = 85 Mbps. (b) B_{min} = 5 Mbps, B_{max} = 125 Mbps.

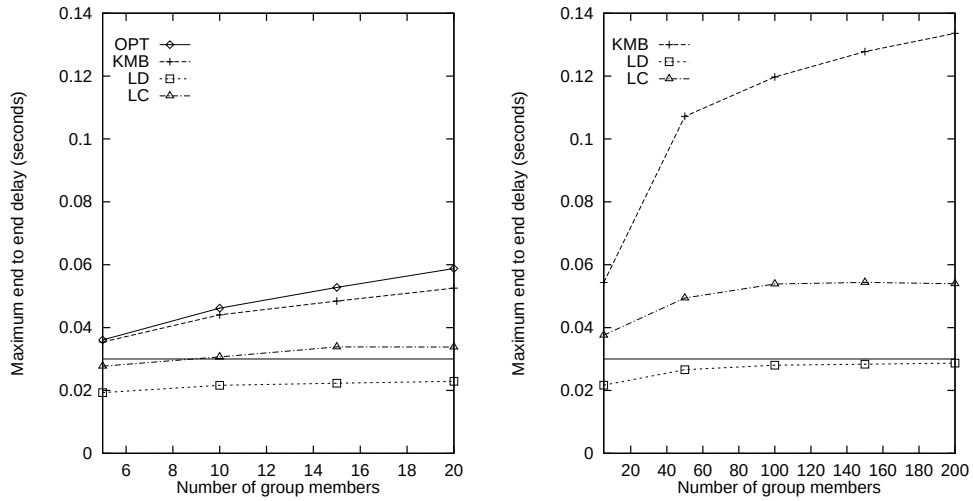


Figure 4: Maximum end-to-end delay, unconstrained algorithms, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps. (a) 20 nodes. (b) 200 nodes.

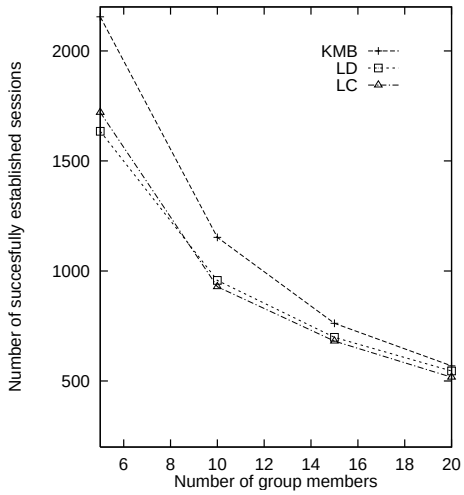


Figure 5: Number of successful sessions, unconstrained algorithms, 20 nodes, average degree 4, no delay constraint.

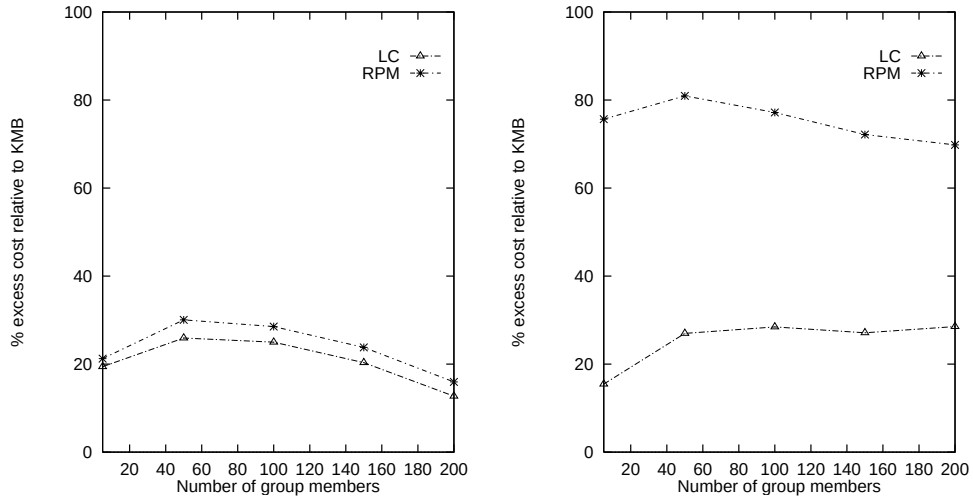


Figure 6: Total cost of a MC tree relative to KMB, LC and RPM, 200 nodes, average degree 4. (a) B_{min} = 45 Mbps, B_{max} = 85 Mbps. (b) B_{min} = 5 Mbps, B_{max} = 125 Mbps.

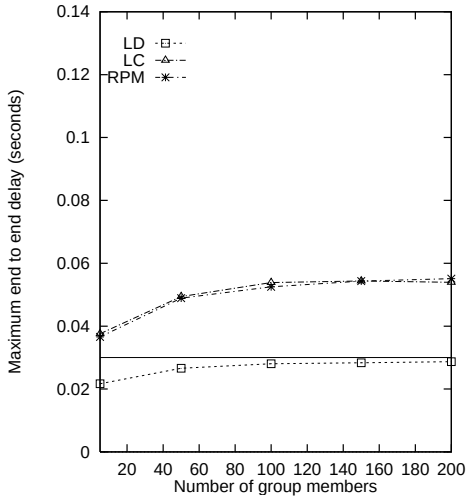


Figure 7: Maximum end-to-end delay, LC and RPM, 200 nodes, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps.

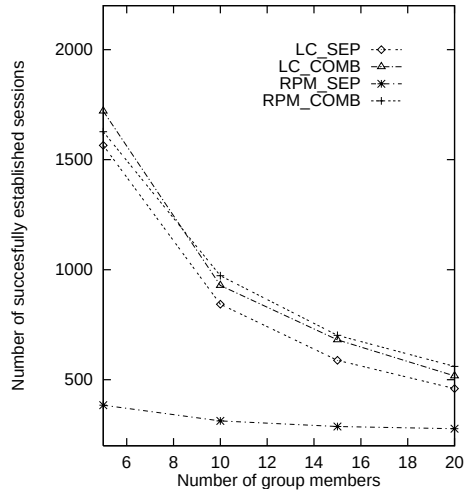


Figure 8: Number of successful sessions, LC and RPM, 20 nodes, average degree 4, no delay constraint.

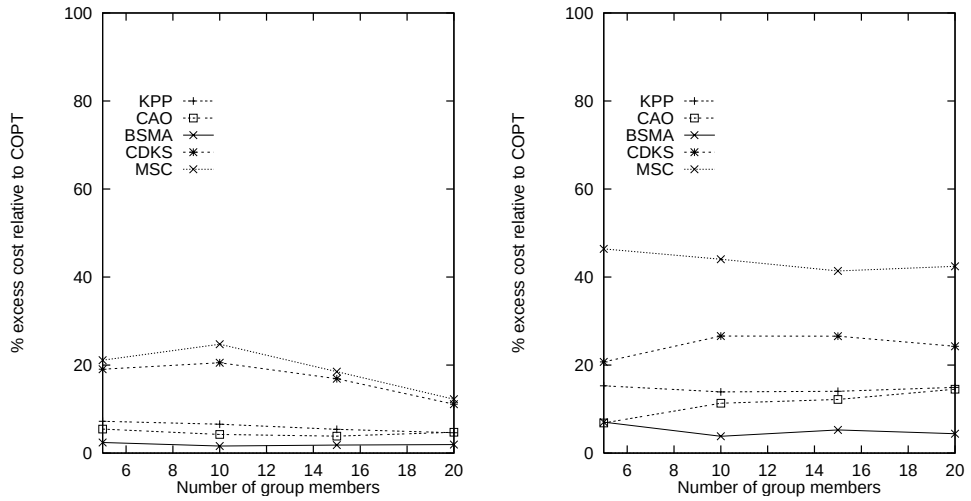


Figure 9: Total cost of a MC tree relative to COPT, constrained algorithms, 20 nodes, average degree 4, Δ = 0.03 seconds. (a) B_{min} = 45 Mbps, B_{max} = 85 Mbps. (b) B_{min} = 5 Mbps, B_{max} = 125 Mbps.

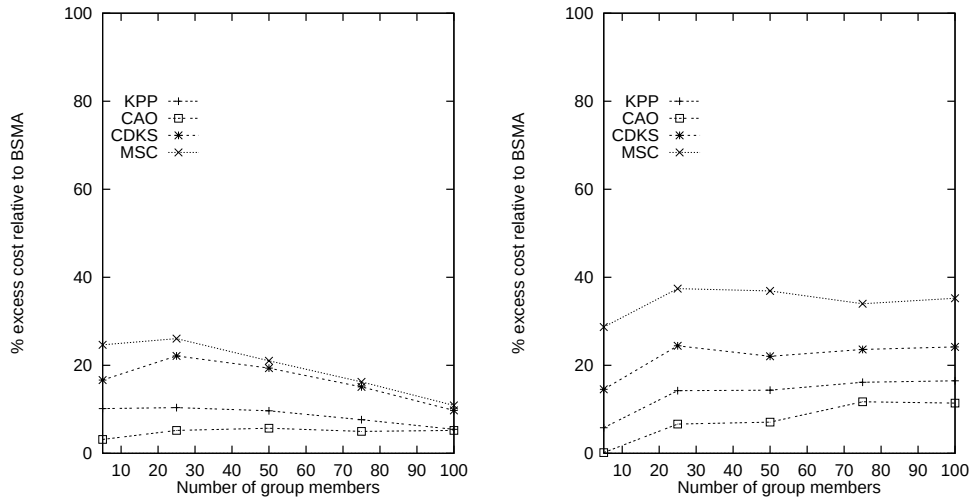


Figure 10: Total cost of a MC tree relative to BSMA, constrained algorithms, 100 nodes, average degree 4, Δ = 0.03 seconds. (a) B_{min} = 45 Mbps, B_{max} = 85 Mbps. (b) B_{min} = 5 Mbps, B_{max} = 125 Mbps.

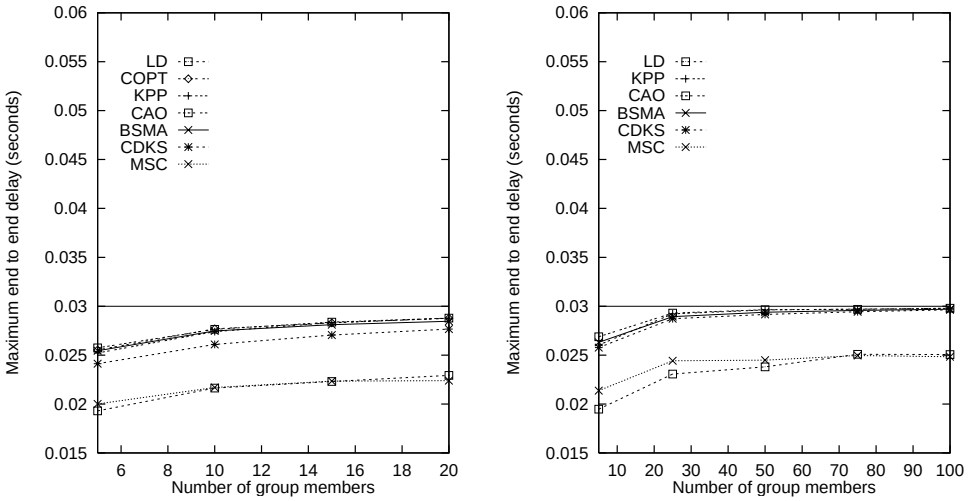


Figure 11: Maximum end-to-end delay, constrained algorithms, average degree 4, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds. (a) 20 nodes. (b) 100 nodes.

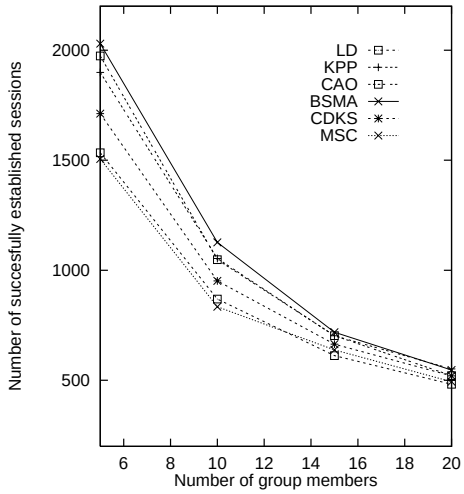
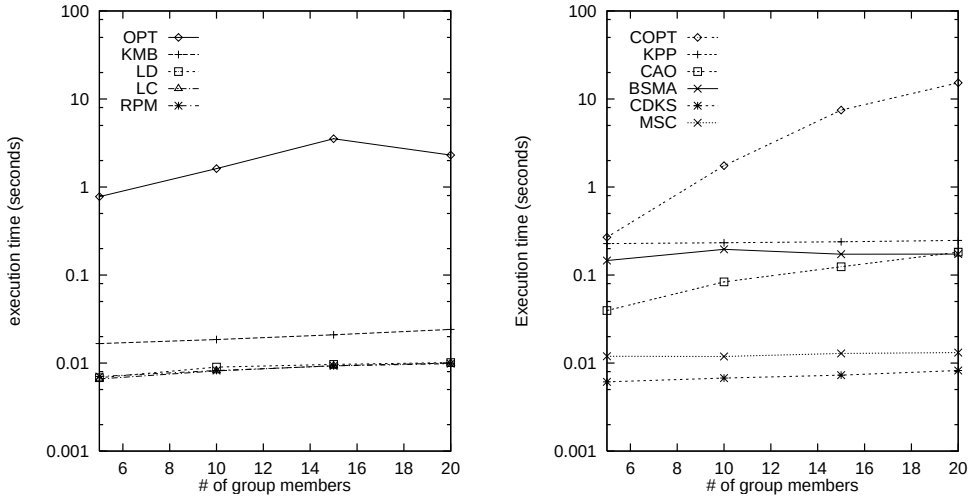
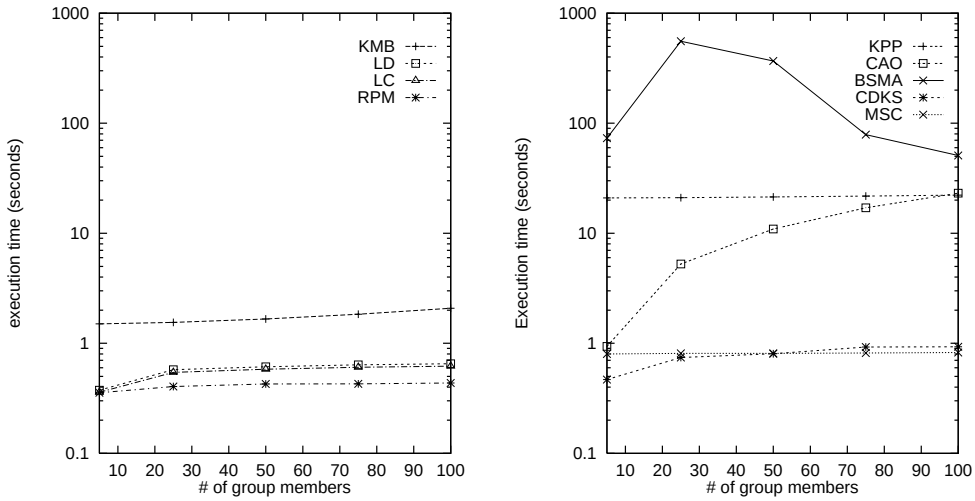


Figure 12: Number of successful sessions, constrained algorithms, 20 nodes, average degree 4, Δ = 0.03 seconds.



(a) (b)

Figure 13: Execution times, 20 nodes, average degree 4, variable MC group size, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds. (a) Unconstrained algorithms. (b) Constrained algorithms.



(a) (b)

Figure 14: Execution times, 100 nodes, average degree 4, variable MC group size, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds. (a) Unconstrained algorithms. (b) Constrained algorithms.

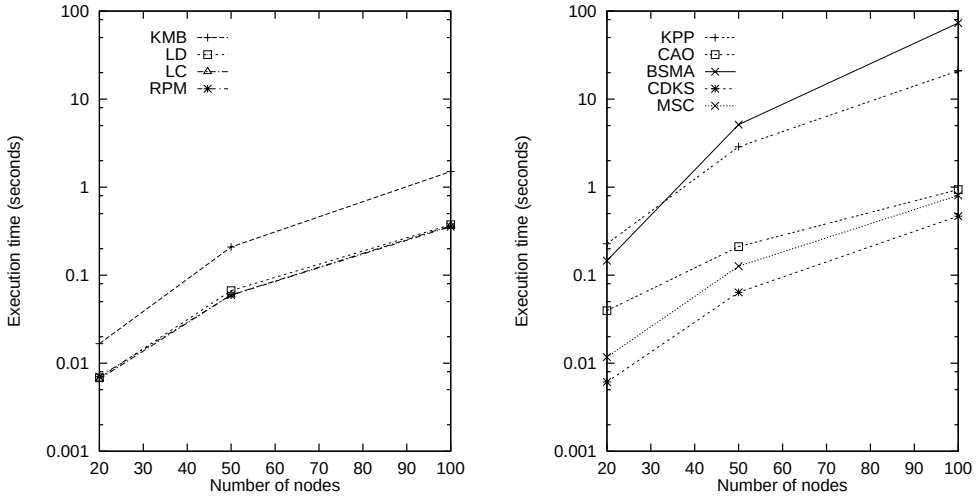


Figure 15: Execution times, Variable network size, average degree 4, 5 MC group members, B_{min} = 5 Mbps, B_{max} = 125 Mbps, Δ = 0.03 seconds. (a) Unconstrained algorithms. (b) Constrained algorithms.