EVALUATION OF MULTICAST ROUTING ALGORITHMS FOR MULTIMEDIA STREAMS

Ciro A. Noronha Jr. and Fouad A. Tobagi

Technical Report No. CSL-TR-94-619

April 1994

This work was in part supported by NASA under grant NAGW-419, by NSF under grant NCR-9016032 and by Pacific Bell. Ciro Noronha was supported by a graduate scholarship from FAPESP from Sept/89 to Aug/93 under grant 89/1658.

Evaluation of Multicast Routing Algorithms for Multimedia Streams

Ciro A. Noronha Jr and Fouad A. Tobagi

Technical Report: CSL-TR-94-619

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Abstract

Multimedia applications place new requirements on networks as compared to traditional data applications: (i) they require relatively high bandwidths on a continuous basis for long periods of time; (ii) involve multipoint communications and thus are expected to make heavy use of multicasting; and (iii) tend to be interactive and thus require low latency. These requirements must be taken into account when routing multimedia traffic in a network. This report presents a performance evaluation of routing algorithms in the multimedia environment, where the requirements of multipoint communications, bandwidth and latency must be satisfied. We present an exact solution to the optimum multicast routing problem, based on integer programming, and use this solution as a benchmark to evaluate existing heuristic algorithms, considering both performance and cost of implementation (as measured by the average run time), under realistic network and traffic scenarios.

Key Words and Phrases: multicast routing, audio and video streams, multimedia, linear programming, shortest path routing, minimum cost routing.

1 Introduction

Multimedia represents the integration of a variety of media, such as data, video, audio and still images. The traffic underlying networked multimedia applications has different requirements than that underlying traditional data applications. These differences pertain to three aspects; namely:

Bandwidth - multimedia streams use relatively high bandwidth on a continuous basis for long periods of time, while the average bandwidth used by data applications is low. For example, a high-quality compressed video stream can use anywhere from 1.5 to 8 Mb/s for extended periods of time, while the average bandwidth used by typical data applications can be well below 1 Mb/s.

Multipoint Communications - it is expected that a significant fraction of the multimedia traffic will be multipoint. Examples are videoconferencing, one-way video distribution and collaborative computing. Data applications, on the other hand, typically make only occasional use of multicasting.

Low Latency (on the order of 100-200 ms end-to-end), required for some applications (such as videoconferencing or collaborative computing) that provide interactive communications. Data applications typically do not have latency constraints.

A *stream* is a continuous flow of information (i.e., video frames or audio samples) that has to be delivered in a timely fashion. Some video/audio encoders produce constant bit-rate streams; others produce variable bit rate streams. However, even variable bit-rate streams are not as bursty as data traffic; both for constant bit rate and for variable bit rate streams one can define a certain *bandwidth requirement* to transport the stream. In a computer network, streams are divided into packets for transmission. Several streams can be multiplexed in time and sent through a channel, each stream using a fraction of the channel's bandwidth. A *multicast stream* is a stream with one source and multiple destinations.

Applications generate streams in *sessions*. A session is composed of one or more streams which are logically related. For example, a videoconference with P participants can be seen as a session composed of P multicast streams, from each of the participants to the other $P-1$ conferees. The transport layer can treat a multicast stream either as a set of unicasts or as a single multicast communication. In the former case, network resources will be wasted because multiple copies of the same information are sent over some links; this waste can be especially severe for high-bandwidth streams. However, if the network is able to replicate the information at appropriate locations, and the transport layer at the source node takes advantage of this feature, at most one copy of the stream is transmitted over any given link.

The *routing algorithm* is responsible for computing the routes for a session, and, in the case of multicast streams, for deciding where the stream should be replicated to reach all destinations. For traditional data applications, there are no bandwidth or latency requirements (in fact, the bandwidth is not even known); routing is done from a topological point of view, with little regard for the usage of the path selected. In addition, multicasts happen only occasionally; it is not very important to route them efficiently. On the other hand, multimedia applications require a multicast routing algorithm that can take into account the bandwidth and latency requirements when routing the stream. Moreover, due to the relatively high bandwidths involved, the algorithm has to be efficient. Existing routing algorithms do not directly take into account the bandwidth and latency requirements when computing the routes, and can route only one stream at a time; an algorithm that can simultaneously route a number of streams can potentially optimize better the usage of network resources.

In this report, we present an evaluation of routing algorithms for multicast streams, taking into account their bandwidth and latency requirements, under realistic traffic and network scenarios. In section 2, we formally define the problem, discuss the existing routing algorithms, and present an integer programming formulation for the optimum multicast stream routing problem, which will be used as a benchmark for the existing algorithms. In section 3, we discuss the previous work in evaluating multicast routing algorithms, which has been mostly limited to considering a single multicast in an empty network, where the bandwidth constraints do not come into play. In section 4, we present an evaluation of the optimum multicast routing algorithm and of the existing algorithms, both from a performance point of view and an implementation cost point of view, under realistic network and traffic scenarios. Finally, in section 5, we present our conclusions.

2 Multicast Routing Algorithms

When an application generates a multicast, the multicast routing algorithm is responsible for finding routes for each of its component streams; each route should have enough free bandwidth to support the stream, and should not exceed its latency constraint. If there are multiple routes that satisfy the requirements for a given stream, the routing algorithm will choose one so as to optimize a certain objective function. In this section, we formulate the routing problem, discuss the existing routing algorithms, and present an optimum routing algorithm that is able to address all the requirements of a multicast session.

2.1 Problem Formulation

The network is described by a set of N nodes and K links, interconnected according to a certain topology. Each of the links has an available bandwidth of V bits/second, and a cost of C \$/bit. Additionally, each bit transmitted through the link will suffer a delay of D milliseconds; since streams have priority over data, we consider that D is not a function of the traffic in the link. For a given multicast stream, we define the *Multicast Path* to be the tree formed by merging the paths from the source to each of the destinations. The *Cost* of a multicast path is defined to be the sum of the costs of the links it uses, weighted by the stream bandwidth; the *Delay* of a multicast path is defined to be the maximum over the delays between the source and each of the destinations.

Formally, the problem can be stated as: "Given the network and a session composed of T multicast streams, with multicast i, $i = 1, \ldots, T$, being characterized by its source s_i, its set of n_i destinations $\{d_{i1}, \dots, d_{in_i}\}$, its maximum delay constraint L_i and its bandwidth request r_i, find the set of multicast paths that satisfies the bandwidth and delay constraints for each stream and minimizes an arbitrary linear combination of the costs and delays of each multicast stream."

2.2 Existing Routing Algorithms

Current multicast routing algorithms have the following limitations: (i) they are able to route only a single multicast; (ii) most do not directly take into account the bandwidth and latency constraints of the streams; and (iii) the structure of the algorithm defines the optimization criterion, which is either cost or delay. To compute the routes for a multiple-multicast session, these algorithms have to be applied sequentially to each multicast in the session, in a given order (and the routes found will be function of the order used). For each multicast in the session, in order to take the bandwidth requirements into account, the network topology must be temporarily pruned of the links not having enough free bandwidth to support the stream, prior to routing it. Finally, after the route has been computed, its delay has to be checked against the latency constraint; if the constraint is not satisfied, the algorithm fails.

As far as the objective function is concerned, existing algorithms can be classified into: (i) Shortest-Path Algorithms; and (ii) Minimum-Cost Algorithms:

Shortest Path Algorithms: The routes are computed independently from the source to each destination, using the shortest path; the paths are then merged in a single tree. There are several exact algorithms available to compute the shortest path [1]. If the link labels used in the routing are the link delays, this approach will yield the minimum delay routing from the source to each of the destinations. Other measures, such as cost, can also be used for link labels. This algorithm is used in [2] and in the Multicast OSPF routing protocol [3] (where the link labels can be set arbitrarily by the network administrator). In this report, we consider the shortest path algorithm with the link delays as labels, which we denote by **SP/delay**, and with the link costs as labels, which we denote by **SP/cost**.

Minimum Cost Algorithms: The routing is done so as to minimize the sum of the labels of the links used; if the link labels are set to the link costs, this approach will lead to the minimum cost multicast routing. This is the well-known problem of finding minimum-cost Steiner trees in graphs, which is known to be NP-complete [4]. Many exact solutions (with exponential worst-case run times) and heuristics have been proposed to address this problem; see [5] for a comprehensive survey of the field. The most important heuristic solutions are the algorithms by Kou, Markowsky and Berman [6] (which we will refer to as KMB), Rayward-Smith [7] (which we will refer to as RS) and Takahashi and Matsuyama [8] (TM). It has been shown that the cost of the trees found by these heuristics is at most twice the cost of the minimum cost tree. These heuristics, however, were designed for <u>undirected</u> graphs; but since multicasts are essentially directed, when the links with not enough free bandwidth are pruned from the network, the resulting graph is in general directed. We have modified the KMB heuristic to take into account the direction of the links [9]; in the evaluation, we will denote this algorithm by **KMB**. Kompella et al [10] proposed a variation of the KMB heuristic to compute minimum-cost routes with a delay constraint in an undirected graph; the link delays are assumed to belong to a discrete set.

2.3 The Optimum Multicast Routing Algorithm

The problem defined in section 2.1 can be formulated as an integer programming problem, which can be solved exactly. The problem is NP-complete, and the worst case run time is exponential, but the solution can serve as a benchmark for the heuristic algorithms. Defining:

N : Number of nodes in the network.

K : Number of (directed) links in the network.

$\boldsymbol{A}$: $N \times K$ network topology matrix; $A_{ij} = 1$ if node i is the source of link j, $A_{ij} = -1$ if node i is the destination of link j, and $A_{ij} = 0$ if link j is not connected to node i.

$\boldsymbol{C}$: $1 \times K$ cost vector.

$\boldsymbol{D}$: $1 \times K$ delay vector.

$\boldsymbol{V}$: $K \times 1$ available capacity vector.

T : Number of multicast streams.

s_i : Source node for multicast i

n_i : Number of destinations for multicast i

$\{d_{ik}\}$: Set of destinations for multicast i, $k = 1, \ldots, n_i$

r_i : Bandwidth requirement for multicast i

$\boldsymbol{X}^i$: $K \times n_i$ multicast routing matrix for multicast stream i. $X^i_{jk} = 1$ if link j is used in the multicast path for stream i to reach destination d_{ik}, otherwise $X^i_{jk} = 0$, $k = 1, \ldots, n_i$.

Y^i : $K \times 1$ multicast path vector for stream i. $Y^i_j = 1$ if link j is in the multicast path for stream i, otherwise $Y^i_j = 0$.

M_i : Delay for multicast request i.

L_i : Latency constraint for multicast request i.

$\boldsymbol{B}^i$: $N \times n_i$ source-destination matrix for multicast stream i; $B^i_{jk} = 1$ if $j = s_i$, $B^i_{jk} = -1$ if $j = d_{ik}$, and $B^i_{jk} = 0$ otherwise, $k = 1, \ldots, n_i$.

β_c : Weight of the cost in the optimization.

β_d : Weight of the delay in the optimization.

The optimum routing problem can be formulated as follows:

GIVEN: $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{V}, N, K, T, \boldsymbol{B}^i, r_i, \boldsymbol{L}, \beta_c, \beta_d$

MINIMIZE:

$$\sum_{i=1}^{T} r_i \left(\beta_c \boldsymbol{C} \boldsymbol{Y}^i + \beta_d M_i\right) \tag{1}$$

WITH RESPECT TO: $\boldsymbol{X}^i, \boldsymbol{Y}^i, M_i, \quad i = 1, \ldots, T$

UNDER CONSTRAINTS:

$(k = 1, \ldots, n_i, j = 1, \ldots, K, i = 1, \ldots, T)$

1. There must be a path from the source to each of the destinations for all the streams:

$$\boldsymbol{A}\boldsymbol{X}^i = \boldsymbol{B}^i \tag{2}$$

2. The multicast path is formed by merging the unicast routes:

$$X^i_{jk} \leq Y^i_j \leq 1 \tag{3}$$

3. The delay for a multicast is the delay to the farthest destination:

$$M_i - \sum_{j=1}^{K} D_j X^i_{jk} \geq 0 \tag{4}$$

4. There is a maximum delay constraint for each of the multicast streams:

$$M_i \leq L_i \tag{5}$$

5. The total flow through a link cannot exceed its bandwidth:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i \leq \boldsymbol{V}; \tag{6}$$

6. No bifurcation of flow:

$$\boldsymbol{X}, \boldsymbol{Y} \quad \text{are binary.} \tag{7}$$

Constraints (1) to (6) define a linear programming problem, which can be solved by the simplex method. When constraint (7) is included, the problem becomes an integer programming problem[1]. In [9, 11] we present an efficient solution method for this integer problem, based on the well-known branch-and-bound method [12], which has two parts:

- For the linear relaxation of the problem (excluding constraint 7), we proposed two successive decompositions: the first one decomposes the problem of routing a multicast session with T multicast streams into T single-multicast routing problems, and the second one further decomposes each of these single-multicast routing problems into n_i unicast routing problems.
- Enhanced value-fixing rules based on the structure of the problem to prune the search space for the integer problem.

Different choices of β_c and β_d in equation (1) lead to different optimizations. In this paper, we consider the following cases:

- $\beta_d = 0, \beta_c > 0$: Cost minimization. We denote routing solutions using these values by **Optimum/cost**.
- $\beta_c \gg \beta_d > 0$: Cost minimization, with delay as a secondary objective; denoted by **Optimum/cost/delay**.
- $\beta_d \gg \beta_c > 0$: Delay minimization, with cost as a secondary objective; denoted by **Optimum/delay/cost**.

The case $\beta_c = 0, \beta_d > 0$, which would correspond to delay minimization without regard to cost, will not be considered because: (i) for a single multicast, this problem can be solved exactly using the shortest path algorithm, and the optimum will not make any difference; and (ii) constraints (2) to (7) do not guarantee that there will be no loops in the route[2]; when $\beta_c = 0$, additional equations would have to be added to the constraint set to avoid routing

[1] In a packet-switched network, it is conceivable that bifurcation of flow could be allowed. In this case, constraint (7) does not apply, and the problem is no longer NP-complete.

[2] This is not needed when $\beta_c > 0$ because the cost minimization will guarantee that loops do not exist.

loops (or the final solution would have to be pruned). Therefore, there is no advantage in using this objective function. On the other hand, for single-multicast sessions, the multicast path found by setting $\beta_d \gg \beta_c > 0$ (minimize delay, with cost as a secondary objective) will have the same delay as a multicast path found using the shortest path algorithm. However, this only means that the routes used in both cases to reach the farthest destination will be the same (or have the same delay). The shortest path algorithm will also minimize the delay to the other destinations; the integer programming solution, on the other hand, is able to make use of the fact that the only constraint on the routes to the other destinations is that their delay should not exceed the delay to the farthest destination to further minimize the cost of the multicast, while still achieving the minimum delay.

It should be noted that this integer programming formulation solves the optimum *static* routing problem; in other words, given the network and the session, it will find the best routes (as defined by the objective function) without any regard for the future sessions; the success of the routing computation (i.e., the feasibility of the problem) is independent both of the particular objective function used and of the costs and delays of the links. In a dynamic scenario, where sessions arrive, are routed (or blocked), exist in the network for a certain time (if accepted), and terminate, the routing algorithm is executed for each session arriving, and decides whether it can be accepted or not, and if accepted, which routes to use. The optimum routing algorithm described here takes the best decision for each session, but there is **no** guarantee that the sequence of decisions is optimal in any sense. A true optimum routing algorithm for this scenario should try to optimize also the sequence of decisions, based on the network topology and in the traffic statistics, which is a much more complex problem.

3 Previous Work in Evaluation of Multicast Routing Algorithms

The algorithms described in section 2.2 have been studied in the context of multicast routing by a number of researchers. In most cases, they studied the routing of a single multicast in an empty network, and the performance measures were the multicast costs and delays.

Kumar and Jaffe [13] compared a number of minimum delay and minimum cost algorithms when the link cost and delay weights are the same and derived analytical bounds in cost/delay under that assumption. They also proposed a general algorithm that is able to "trade off" cost and delay, by initially performing a minimum cost routing, and then replacing individual paths with excessive delay with the shortest path; the trade-off is controlled by the number of paths that are replaced. They evaluated numerically costs, delays and run times for routing a single multicast on an empty network, using the algorithms discussed in their paper. For the evaluation, they used both an earlier version of the NSFNet topology, and randomly-generated topologies of different sizes and node degrees[3]. The random topologies were generated by starting with a ring and adding links (uniformly) at random, until the desired degree is reached. The link labels (used both for cost and delay) were set to unity in some cases, and chosen at random between $\{1, 2, 3, 4\}$ in others. Their main conclusions were:

- In general, algorithms that minimize cost take about one order of magnitude more time to run than algorithms that minimize delay.
- The differences in cost/delay between the algorithms evaluated are in the order of 30-40%.
- Results for the NSFNet and for random topologies of the same size are similar.

Waxman [14] studied the problem of dynamically adding and removing destinations on a single multicast that had been already routed in an empty network. Waxman studied the RS

[3] Ratio between the number of links and the number of nodes.

and KMB heuristics, together with a dynamic algorithm he proposed to add the routes to the new destinations joining the multicast. He used randomly-generated network topologies that are supposed to "resemble" actual networks. The algorithm to generate random topologies resembling actual networks Waxman proposed is based on the observation that, in actual networks, links are more likely to exist between nodes that are located closely together than between nodes that are far apart. To generate the topologies, the nodes are first distributed at random over a rectangular grid. For every pair of nodes (u, v) introduce a link with probability:

$$P(\{u,v\}) = \beta \exp\left[\frac{-d(u,v)}{L\alpha}\right] \tag{8}$$

where α and β are parameters in the range $(0, 1]$, $d(u, v)$ is the Euclidean distance between u and v and L is the maximum distance between two nodes. The parameter β controls the degree of the graph, and the parameter α the density of "short" links in relation to "long" links. Note that this method does not guarantee that the network generated is connected. In all cases, the link labels (costs) were set to the distance between the nodes. Waxman found that, in the average, the RS and KMB heuristics find solutions which are very close to the optimum minimum cost, and the worst case cost (observed) is about 35% higher than the minimum. When the destination set changes, if one is not allowed to re-route the existing multicast, the cost of the new multicast (using his routing algorithm) can be up to 2-4 times the minimum in the worst case.

Doar and Leslie [15] also studied the problem of adding and removing destinations to an existing multicast. They used Waxman's method for generating the network topology, but scaling the link existence probability in equation (8) by $k\overline{d}/N$, where N is the number of nodes, $\overline{d}$ is the average degree of a node, and k is an experimental factor found to be about 0.25, to keep the node degree constant as the network size is changed. They evaluated the use of the shortest path algorithm to add the routes to the nodes that have joined the multicast, and compared the resulting cost to the minimum cost for that multicast (obtained by applying the KMB algorithm). They found that the difference in cost can be as large as

2-3 times, for networks of 50 nodes.

Leung and Yum [16] proposed a number of minimum-cost heuristic algorithms, and compared their performance with the RS algorithm for routing a single multicast over an empty network. Their algorithms are applicable to directed graphs as well, although this aspect is not explored in their paper. One of their algorithms is actually a variation on the basic TM algorithm (and was also given by Chow [17]). They evaluate the algorithms using two networks: one is an early version of the ARPANET and the other is random; the main conclusion is that there is very little difference between the cost achieved by the heuristics and the optimum.

Chow [17] explicitly studies the case of single multicasts in directed graphs, and proposes a minimum cost heuristic algorithm for the case where not all nodes are capable of multicasting. He uses the exact minimum cost algorithm proposed by Dreyfus and Wagner [18] to compute the optimum solution, and compares the cost of this solution with the cost obtained by his heuristics. He finds that there is very little difference between the costs of the optimum solution and the heuristic solution, but the optimum takes much more time to run.

Ammar et al [19] studied the minimum cost routing under a variety of additional constraints using a non-linear integer programming formulation. They used a 16-node irregular network, and reported the costs for the multicasts under various scenarios.

Kompella et al [10] studied the case of minimum-cost routing under maximum delay constraints. They assumed both unit and random link costs, random (integer) link delays, undirected links, and gave a variation of the KMB heuristic to compute a sub-optimal solution in the case where the latency constraints belong to a discrete set of values. The worst-case run time of their algorithm is exponential. They evaluated their algorithm using random networks; the only constraint imposed in the network topology was that at least one solution to the routing problem under consideration must exist. Their traffic model was a single multicast being routed over an empty network. The main results were: (i) the cost of their heuristic is about 10% higher than the cost of the optimum solution (obtained by exhaustive search); (ii) the cost of using the shortest path algorithm in the same scenarios is about 70-80% higher than the optimum; these results have little dependency with the values of the link costs.

Table 1: Summary of previous work in evaluation of multicast routing algorithms

Paper	**Objective Function**	**Direction of Flow**	**Number of Requests**	**Algorithms**	**Network Topologies**
Kumar and Jaffe [22, 13]	Cost	Full-Duplex	Single	Steiner tree heuristics; Shortest Path	ARPANET; random
Waxman [14]	Cost	Full-Duplex	Single	Minimum Steiner tree (KMB, RS)	Random
Ammar et al [19]	Cost	Full-Duplex	Single	Heuristic based on simplex	16-node irregular topology
McKinley and Liu [20]	Cost	Full-Duplex	Single	Extension to KMB; set-covering	Mesh of buses; grid of buses
Jiang [21]	Cost	Full-Duplex	Single Conferences	Extensions to KMB and RS	Random
Leung and Yum [16]	Cost	Full-Duplex	Single	Heuristics for the minimum Steiner tree	ARPANET; single random network
Chow [17]	Cost	Half-Duplex	Single	Optimal; Takahashi Matsuyama heuristic	Specific regular topologies
Kompella et al [10]	Cost with Delay Limit	Full-Duplex	Single	Extension to the KMB heuristic	Random
Doar and Leslie [15]	Cost	Full-Duplex	Single	Shortest-Path, KMB	Random

McKinley and Liu [20] considered the problem of routing a single multicast in a network composed of a mesh of buses; each bus is a shared channel providing broadcast to all nodes connected to it. They propose a number of algorithms for routing in this scenario, and compare their costs (measured in number of buses used to establish a multicast). They also give an exact algorithm when the topology is a regular grid.

Jiang [21] considered the problem of establishing video-conferences; a video-conference with P participants is established as P multicasts, from each conferee to all other members of the conference. He proposed variations to the KMB and RS algorithms to take into account the link bandwidths. Links are undirected, with the bandwidth available in any direction. For the evaluation, he used Waxman's algorithm to generate the network graph, and assigned the capacities of 10, 30 and 100 Mb/s to the links at random, with probabilities 0.6, 0.3 and 0.1 respectively. He assumed that 75% of the capacity of each link was reserved for stream traffic, and that each stream used 3 Mb/s. He reported on the session blocking probability for single video-conference sessions over empty networks, for several different variations of the RS and KMB heuristics.

In summary, most of the previous work in this field has focused on proposing and evaluating minimum-cost heuristic algorithms for routing a single multicast, usually on undirected graphs generated at random (see table 1). None of them has evaluated the performance of the routing algorithm in a realistic scenario, where streams are established, exist for a period of time, and terminate. In such a scenario, the blocking probability (i.e., the probability that a session arrives and cannot be routed) is the basic performance measure, instead of just cost or delay.

4 Evaluation of the algorithms

In this section, we present an evaluation of the various multicast routing algorithms. The evaluation has two parts: (i) a *performance* evaluation, where the results of the different algorithms are compared, and (ii) an *implementation cost* evaluation, where the average run times of the algorithms are compared.

The first step is to compare the cost and delay results when routing a single-multicast session in an empty network, as it was done in much of the previous work in the area. The purpose of this step is to compare the cost/delay characteristics of the heuristic algorithms with the optimum, when the cost and delay of each link are independent measures.

The next step in the evaluation is determining the effect of the network topology; it is our objective to evaluate the algorithms under *realistic* conditions. Therefore, we determine what properties make a network "realistic". We then characterize the algorithms in a baseline case, using the "realistic" topologies, and show how variations in this baseline case affect the results. We also investigate how the performance changes as the bandwidth of the network is upgraded, and what is the best way to increase the capacity of a network. Finally, we characterize the cost of the different algorithms, as measured by their average run times.

4.1 The Evaluation Scenarios

The evaluation scenarios have two parts: (i) the traffic model, and (ii) the network scenario. In this section, we describe both parts of the scenario.

The Traffic Model

We consider that all multicasts in a session arrive and depart simultaneously. The session arrivals form a Poisson process, with rate λ, and the session duration is exponentially distributed, with rate μ. We assume that sources and destinations are uniformly distributed over the network, and that the set of destinations is fixed for the duration of the session (i.e., no destinations join or leave the multicast while it is in progress). In some cases, we consider the problem of routing a single multicast session in an empty network; this would correspond to a very small λ/μ. The following kinds of sessions are considered:

- *Single-Multicast Sessions:* Each session is composed of a single multicast, whose number of destinations is chosen at random, uniformly between 1 and n_{max}; the value of n_{max} is chosen depending on the number of nodes in the network under evaluation.

- *Videoconference Sessions:* Each session has P multicasts and corresponds to a videoconference with P participants. P is chosen at random between 2 and 4.

We consider that all streams in a session have the same bandwidth requirement; the exact value depends on the particular evaluation scenario being considered. We also assume that blocked sessions are lost, and our primary performance measure is the overall session blocking probability. Given the traffic characteristics, we define the **network capacity** for a certain blocking probability as the load (λ/μ) for which that blocking probability is reached.

The Network Model

The network model is characterized by the following parameters:

Size: Number of nodes (N) and links (K) in the network.

Topology: Interconnection pattern between the nodes and links.

Link Parameters: Link costs, delays and capacities.

For this evaluation, we consider that the capacities of all links in the network are equal, and therefore can be normalized to 1. Moreover, we set all the link costs to 1; this way, the cost of a multicast is proportional to its usage of network resources.

For the network topology, we consider: (i) topologies drawn from existing networks, to evaluate the performance of the algorithms under realistic scenarios; and (ii) topologies generated at random, to test the algorithms over a broad range of topologies. For topologies drawn from existing networks, the link delays are set to the propagation delays in the nodes. For random topologies, the nodes are distributed at random over a rectangle, and the link delays are set to the cartesian distance between the endpoints of the link. For this evaluation, we consider the nodes placed in a rectangle with dimensions 15 ms by 10 ms, roughly equivalent to the dimensions of the United States. Additionally, we restrict our attention to random topologies which are strongly connected[4].

We consider the following kinds of random topologies:

Completely Random Topologies: Nodes are interconnected at random.

Random Topologies, Short Links: In "actual" networks, links are more likely to exist between nodes that are "closely located" than between nodes that are "far apart". In this kind of topology, links are more likely to connect nodes that are close.

[4]A strongly connected network has at least one path between any pair of nodes.

Two-Connected Topologies: Existing networks are usually two-connected[5]; this kind of random topologies is restricted to two-connected topologies.

The algorithms used to generate these random topologies are described in the Appendix.

4.2 Evaluation of Costs and Delays

In this section, we evaluate the costs and delays of the various algorithms for a single-multicast session being routed over an empty network. This is what was done in much of the previous work in this field; therefore, we limit ourselves to a single network scenario, and perform the evaluation only to compare the cost/delay results of the optimum multicast routing algorithm with the heuristics.

For the network scenario, we chose a simplified version of the NSFNet T3 backbone, shown in figure 1; the numbers next to the links represent the propagation delays in milliseconds over the link. We set all the link costs to 1.

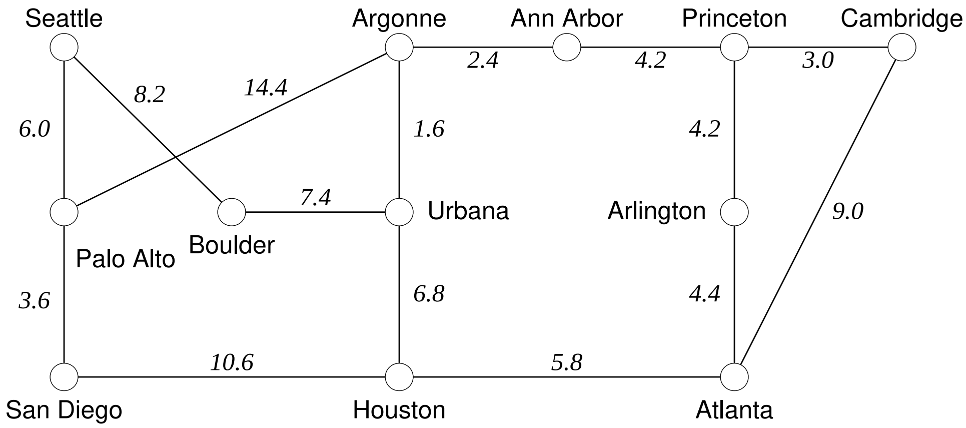

Figure 1: The NSFNet T3 backbone (simplified)

Figure 2 shows the average cost for a single multicast, in hops, as a function of the number of destinations, for the various multicast routing algorithms. We observe that, as expected (and reported by others), the cost obtained by the KMB algorithm is very close to the optimum. The costs for the routes computed by the algorithms which minimize delay

[5]Have at least *two* paths between any pair of nodes.

are 0.5 to 1 hop higher than those of the optimum, and the difference increases with increasing number of destinations. Figure 3 shows the average delays as a function of the number of destinations in the same scenario; we see that, when one compares the different solutions, the small improvement in cost of the minimum-cost algorithms over the minimum-delay algorithms is "paid for" with a larger (in relative terms) difference in delay. For example, for a 9-destination multicast, the difference in cost between the shortest-path and the KMB algorithms is about 1 hop for a total cost of 9 hops, or about 11%, while the difference in delay is 9 ms for a total delay of 23 ms, or 39%.

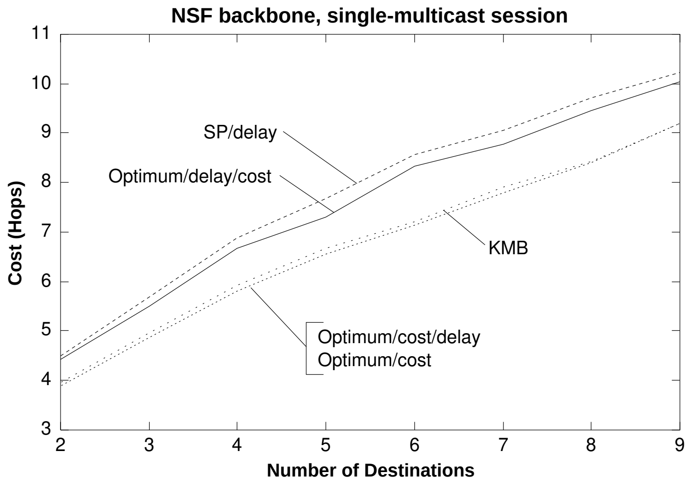

Figure 2: Cost of the multicast as a function of the number of destinations in the NSFNet T3 backbone

It should be stressed that the cost/delay results cannot be directly used to predict the network performance on a dynamic environment, where sessions compete for the resources. In general, all one can say is that "low cost" is a desirable property, because lower-cost routes will use less network resources (if the costs are set correctly) and thus reduce the probability that an incoming session is blocked, but at the price of a higher delay. It is still necessary to numerically assess the effect of the routing algorithms in such environments, in terms of session blocking probability and network capacity, which are really the measures of interest.

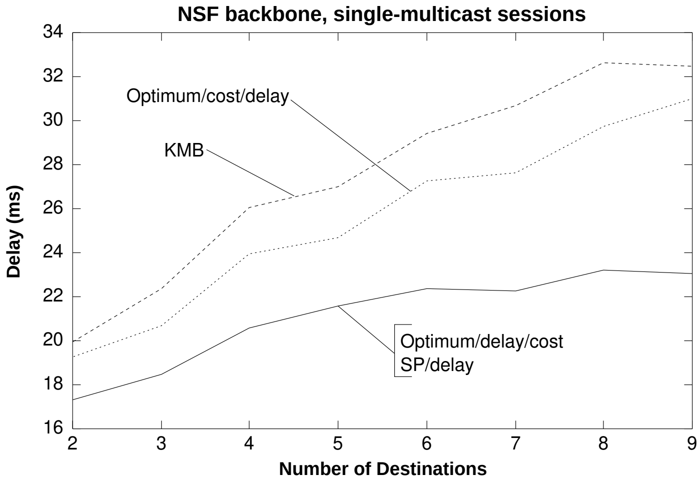

Figure 3: Delay of the multicast as a function of the number of destinations in the NSFNet T3 backbone

4.3 Effect of the Network Topology

One of our goals is to evaluate the algorithms under *realistic* network scenarios. Existing networks are usually two-connected, and, as observed by Waxman, links are more likely to exist between "close" nodes than between nodes that are "far apart" (i.e., they are "biased" towards short links). In order to evaluate the effect of the kind of topology in the results, we considered first the problem of routing a single multicast session in an empty network. The evaluation scenarios were:

Network: (i) a simplified version of the NSFNet T3 backbone, shown in figure 1; (ii) two-connected topologies, generated at random; (iii) random topologies, biased towards short links; and (iv) completely random topologies. The random topologies are of the same size as the NSFNet (12 nodes, 15 full-duplex links); all the links have the same capacity, the node positions are generated at random in an area similar to that of the United States, and the link delays are set proportional to the distances. All topologies are at least strongly-connected.

Traffic: One multicast session, composed of 5 multicasts, each with 5 destinations; the bandwidth of each stream was generated at random, using a bimodal distribution, with varying average.

Experiment: Start with an empty network; for a given average stream bandwidth, try to route the session and record the number of cases where routing was successful, as a function of the average stream bandwidth.

Routing Algorithm: Optimum routing algorithm (the objective function is irrelevant, as we are dealing with a single session and recording only the number of successful routes).

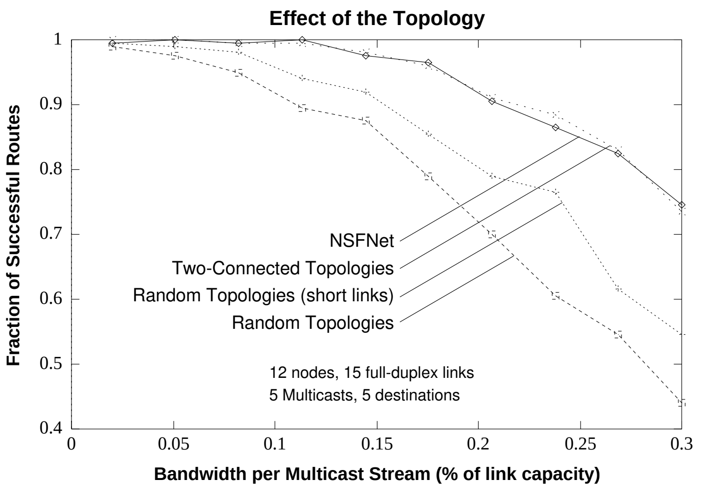

Figure 4: Fraction of successful routes for different kinds of topologies, using the optimum routing algorithm

The results of this experiment are shown in figure 4, and indicate that the performance is much better for the two-connected networks (both existing - NSFNet - and generated at random). The plot also confirms that what is important when generating a random topology that "resembles" an actual topology is to make it two-connected, and **not** bias it towards short links; the performance of the routing algorithm in the NSFNet and in the random two-connected networks is essentially the same. Two-connected networks have higher performance due to the larger number of independent paths; networks that do not have this property will have a link that, if congested, "divides" the network into two disconnected subnetworks, causing all subsequent sessions with members in both subnetworks to be blocked.

In this kind of network, the performance is essentially dictated by the network itself, not by the particular routing algorithm used; our simulations have confirmed this observation. It is interesting to note that Kumar and Jaffe [13] evaluated the cost and delay of several multicast routing algorithms, using both an early version of the ARPANET (a precursor of the NSFNet) and random topologies with the same number of nodes, and seem surprised to find that, for a given routing algorithm, the performance was essentially the same in both scenarios. The ARPANET was designed to be two-connected for reliability purposes; the algorithm used by Kumar and Jaffe to generate the random topologies, described in [22], also generates two connected networks by construction, since it starts with a ring. Our result explains their observation; random topologies and existing topologies of the same size will yield similar results *provided that they are two-connected*; otherwise, the results will be very different, as indicated by figure 4.

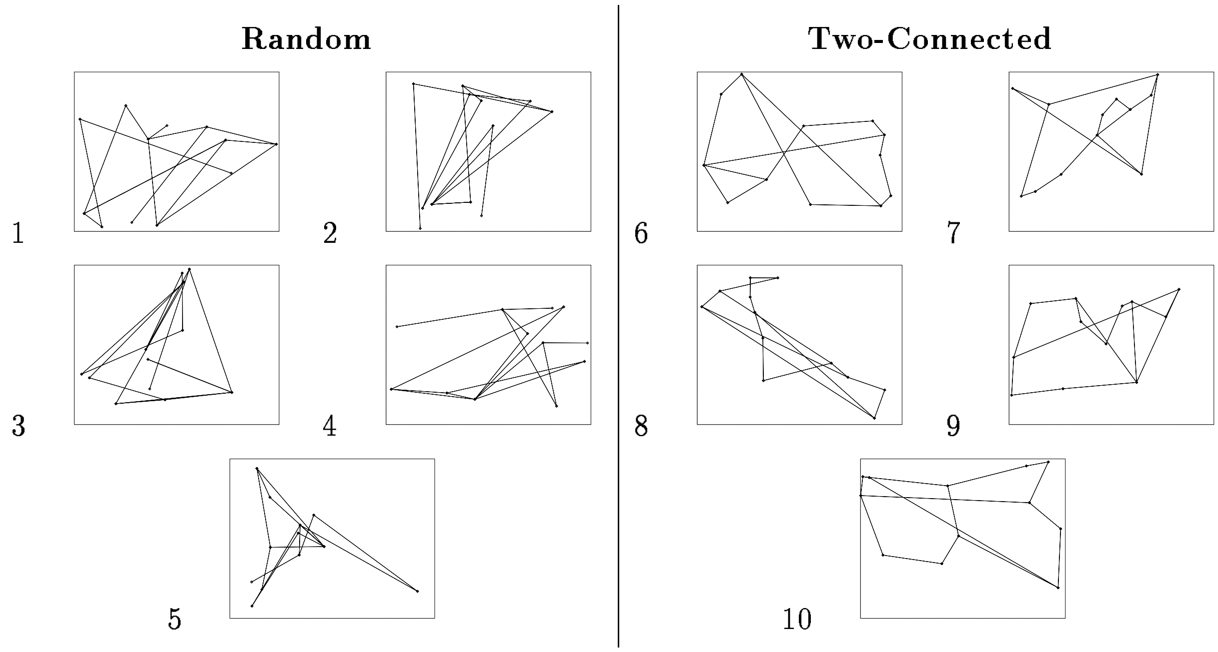

Figure 5: Random network topologies used in the evaluation

In order to confirm these results in a more general dynamic environment, where sessions come and go, we generated at random ten topologies, all with 12 nodes and 15 full-duplex links, which are shown in figure 5. Half of the topologies were completely random (topologies 1 to 5), and the other half (6 to 10) was composed of two-connected topologies. We obtained the blocking probability in each of the networks, for each of the algorithms. We also repeated the same process for the NSFNet T3 backbone (also with 12 nodes and 15 links), shown in figure 1. The traffic was composed of single-multicast sessions, with a random number of destinations between 1 and 10 and exponential duration and interarrival times. The session blocking probability results for each of the topologies, using the Optimum/cost routing algorithm, are shown in figure 6; similar results were observed for the other algorithms. The main observation is that the blocking probability is much higher in the completely random topologies, and confirms our conclusions from the single-session evaluation.

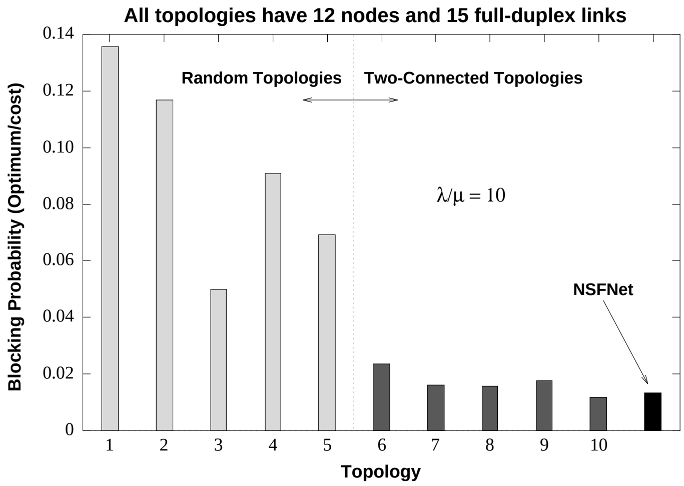

Figure 6: Blocking probability in random topologies and two-connected topologies

For the remainder of this report, we consider only two-connected networks.

4.4 The Baseline Case

In this section, we present a baseline case, and characterize its performance. We then discuss how the results change as the traffic scenario changes from the baseline case. The baseline case corresponds to two-connected topologies, with 12 nodes and 15 links (same size as the NSFNet); traffic is composed of single-multicast sessions, with a random number of destinations between 1 and 10. The sessions arrive according to a Poisson process, and stay in the network for an exponential amount of time. Each stream requires 10% of the link bandwidth and there is no latency constraint.

In figure 7 we plot the overall blocking probability, averaged over a large number of two-connected topologies, as a function of the offered load (λ/μ), for all algorithms. The figure indicates that, as expected, the blocking probability for the cost-based algorithms (Optimum/cost, Optimum/cost/delay, KMB) is lower than for the delay-based algorithms (Optimum/delay, SP/cost, SP/delay). At 1% blocking, the network capacity for this traffic scenario is about 17 for the cost-based algorithms, and 13 for the delay-based algorithms. At 10% blocking, the values are 25 and 22 respectively. We repeated the same runs for the NSFNet T3 backbone and found similar results (with a more marked difference between the two groups of algorithms). The results for the NSFNet are shown in figure 8.

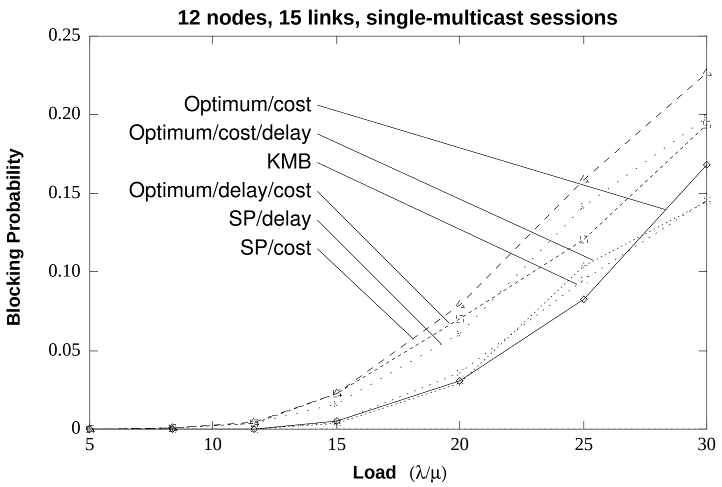

Figure 7: Blocking probability for two-connected topologies with 12 nodes and 15 links, single-multicast sessions

When multicasts with different numbers of destinations co-exist in a network, we expect that the blocking probability be higher for the multicasts with higher number of destinations. In figure 9 we show the blocking probability as a function of the number of destinations, at different load values, for the Optimum/cost algorithm. Figure 9 shows that, for low loads, the blocking probability is a weak function of the number of destinations. Even at high loads, the ratio between the blocking probability for 10-destination multicasts and unicasts (one destination) is in the range of 2-3. The plots for the other algorithms are similar.

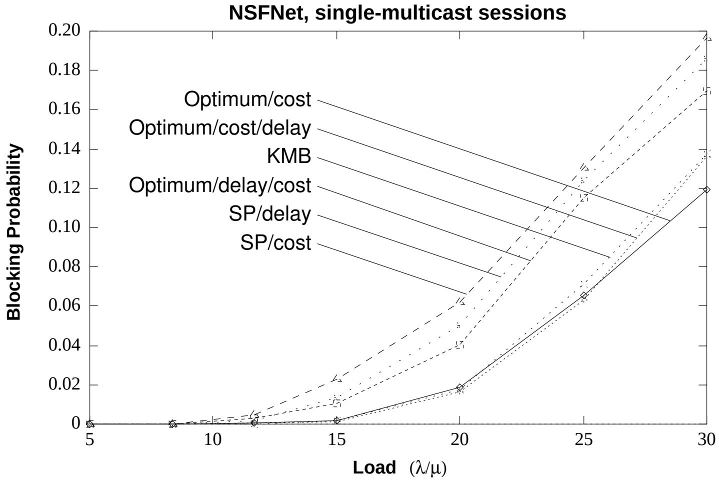

Figure 8: Blocking probability for the NSFNet, single-multicast sessions

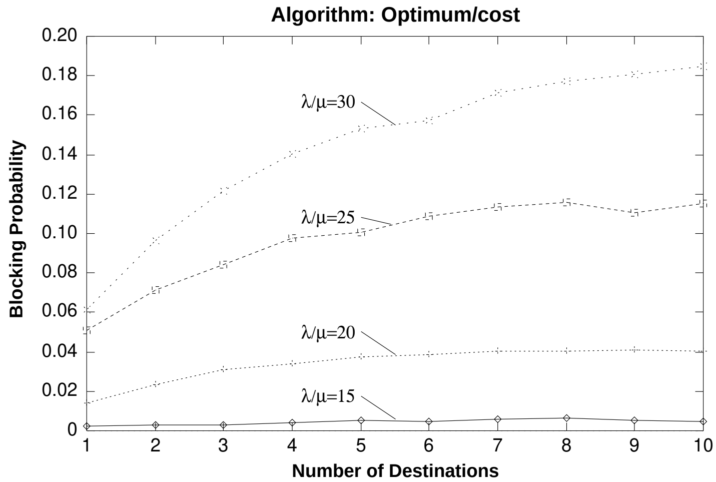

Figure 9: Blocking probability as a function of the number of destinations in the multicast, algorithm Optimum/delay/cost

4.5 Introducing a Delay Constraint

The objective of this set of runs was to determine the effect of a constraint in the blocking probability of the various algorithms. For these runs, we chose the NSFNet T3 backbone, under single multicast sessions, with the stream bandwidth fixed at 10%. The diameter[6] of this network in delay is 28 ms (shortest path from Seattle to Arlington).

[6]Maximum shortest path over all pairs of nodes.

From the delay histograms for the the runs where no delay constraint was imposed, we observed that there were no successful routes with delay higher than 80 ms. Therefore, any constraint equal or higher than 80 ms would have no effect in the results. We imposed a constraint of 40 ms, which is a reasonable value considering that what is being discussed here is the component of the delay in the wide-area network; in an audio/video communication, there are other components to be considered, such as the delays due to the encoders/decoders and the delays in the local networks where the sources and destinations are attached [23]. In figure 10 we plot the the blocking probability as a function of the load under the 40 ms constraint. Table 2 shows the fraction of the routes in the unconstrained case which would not satisfy the 40 ms latency requirement.

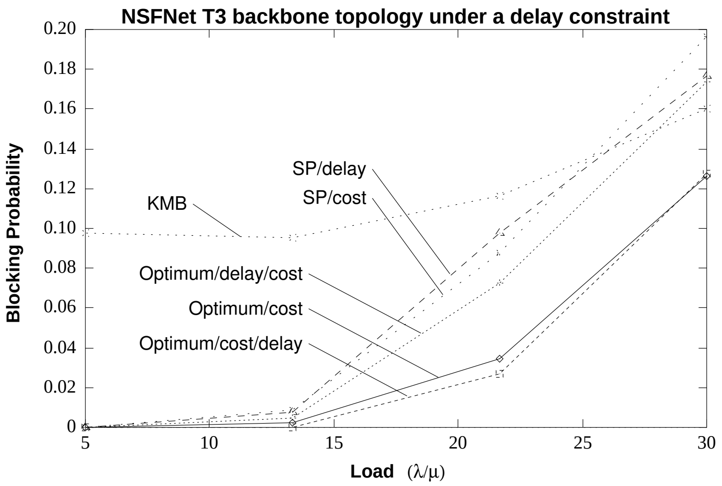

Figure 10: Simulation results for the NSFNet T3 backbone, under delay-constrained traffic

Figure 10 indicates that, as hinted by the numbers in table 2, the delay-based algorithms are essentially unaffected by this constraint. The optimum algorithms take the delay constraint into account when computing the routes; they are also not affected because they can usually identify alternate routes satisfying the constraint. In the case of the KMB algorithm, however, there is a large effect since its objective is cost, not delay. Even at low loads, the blocking is high because the algorithm is forced to reject sessions whose routes would exceed the latency constraint. The bigger the number of destinations in the multicast, the larger the effect: while at low load the KMB algorithm is capable to accommodate almost all unicasts

Table 2: Fraction of routes that do not satisfy the 40 ms latency in the unconstrained case

Algorithm	**Fraction**
Optimum/cost	16%
KMB	9%
Optimum/cost/delay	2.6%
SP/cost	$\approx$ 0%
Optimum/delay/cost	$\approx$ 0%
SP/delay	$\approx$ 0%

under the 40 ms constraint, the blocking probability is over 20% for 10-destination multicasts. This is illustrated in figure 11, where we plot the blocking probability as a function of the number of destinations when $\lambda/\mu = 5$; under that load, the blocking for all algorithms except the KMB is zero.

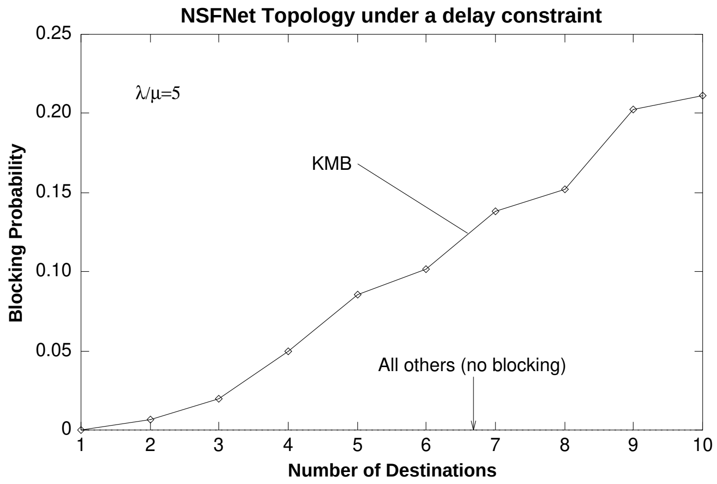

Figure 11: Blocking probability as a function of the number of destinations under low load, in the presence of a delay constraint

As the delay constraint is decreased from 40 ms, the blocking probability "plateau" observed at low loads for the KMB algorithm (see figure 10) will significantly increase (a constraint of 30 ms would move it to about 30%, and 20 ms to over 70%). As the constraint is made tighter, the optimum algorithms will tend to use the shortest paths in delay, regardless of their objective functions. Of course, if the constraint is set to a value lower than the network diameter, the blocking will be high for all algorithms.

4.6 Upgrading the Network

In this session, we consider the problem of adding bandwidth to the network. We consider networks with a fixed number of nodes and a fixed session arrival rate, and increase the network capacity by adding links to it, and observing the corresponding decrease in the blocking probability.

Figure 12 shows the blocking probability for two-connected 6-node networks, when the number of full-duplex links varies from 6 (ring topology) to 15 (fully-connected topology). The figure indicates that the blocking probabilities for the cost-based algorithms (Optimum/cost, Optimum/cost/delay and KMB) are lower than for the delay-based algorithms (SP/delay, SP/cost and Optimum/delay/cost). The curve representing the relation between the blocking probability and the number of links is concave, and has two distinct regions: (i) the high-blocking region, where an increase in the number of links results in an essentially linear decrease in blocking probability, and (ii) a low-blocking region, where the network is capable of carrying almost all the offered traffic, and addition of links has little effect.

Figures 13 and 14 show the blocking probability for 12-node networks, with varying number of links (the NSFNet corresponds to the point where $K = 15$ in the plots). The plot in figure 13 shows the blocking probability when the number of destinations is uniformly distributed between 1 and 4, and the plot in figure 14 shows the blocking probability when the number of destinations is between 1 and 10. Both curves show the high-blocking region, where an increase in the number of links produces an approximately linear decrease in the blocking probability. Additionally, figure 13 shows the low-blocking region.

Figure 15 shows the blocking probability for a large (50-node) network, when the number of full-duplex links varies from 50 to 300. In this case, we considered only the heuristic algorithms as the run time for the optimum would be extremely large. Again, we see a small advantage for the cost-based algorithm (KMB) over the delay-based ones.

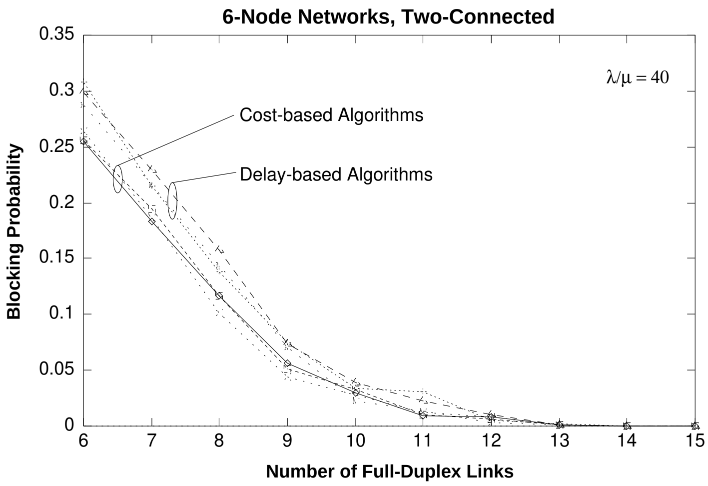

Figure 12: 6-node networks, varying number of links under constant session arrival rate; number of destinations varies from 1 to 5

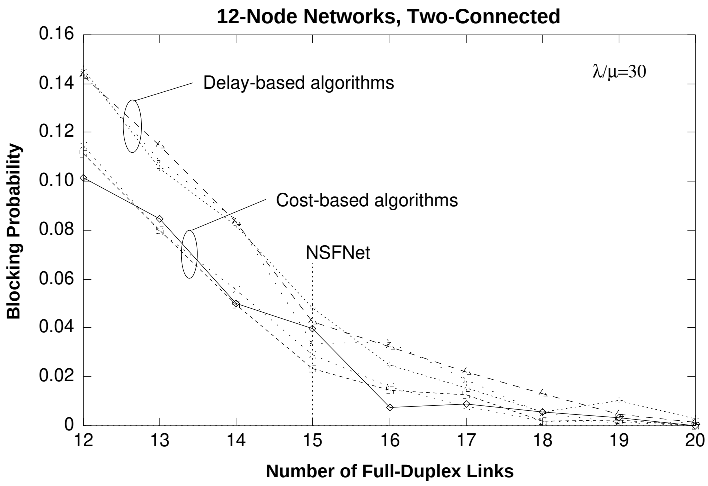

Figure 13: 12-node networks, varying number of links under constant session arrival rate; number of destinations varies from 1 to 4

Finally, we considered the following question: given a network, is it better to add bandwidth by adding links (as done in the plots in figures 12 to 15), or to increase the bandwidth of the existing links? To answer this question, we considered again a 50-node network, under single multicast sessions, with 50 full-duplex links. Since we focus on two-connected networks, the topology of this network is a ring. Using the KMB algorithm to compute the routes, we obtained the blocking probability for the case where links are added to the

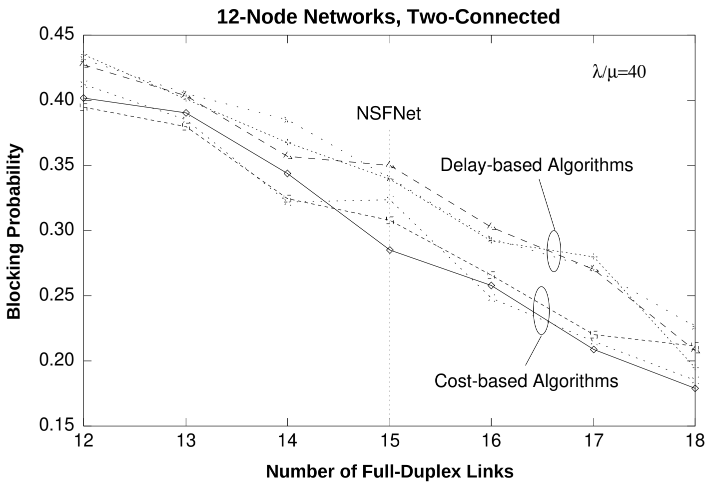

Figure 14: 12-node networks, varying number of links under constant session arrival rate; number of destinations varies from 1 to 10

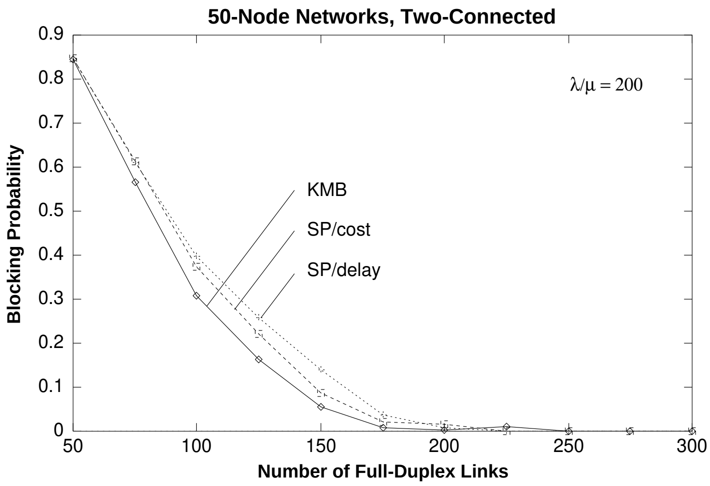

Figure 15: 50-node networks, varying number of links under constant session arrival rate; number of destinations varies from 1 to 10

network (same as figure 15), and for the case where the bandwidth of the existing links is increased, and the topology preserved. Note that in both cases we are adding the same amount of bandwidth to the network; what changes is *where* this bandwidth is added. The results can be seen in figure 16; it is clear that, from a blocking probability point of view, when upgrading the available bandwidth in the network, it is far better to do it by adding new links than by increasing the bandwidth of existing links. This happens because as links are added, not only the capacity increases, but the average path length in the network decreases, further reducing the blocking probability. In practice, however, adding additional links to a network may be more expensive than increasing the bandwidth of the existing links.

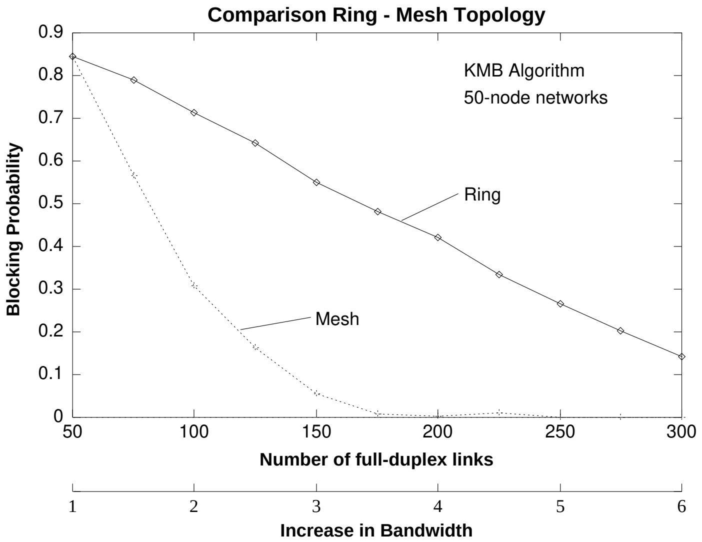

Figure 16: 50-node networks, comparison between ring and mesh topologies

4.7 Other Scenarios

In this section, we investigate other variations of the baseline case, namely videoconferencing sessions, non-unit cost and other bandwidth distributions for the traffic.

Videoconferencing Sessions

We considered multiple-multicast (videoconferencing) sessions, with the stream bandwidth fixed at 10% of the link capacity. The network scenarios were 12 nodes, 15 links; 6 nodes, 8 links; and 6 nodes, 12 links. The main observation, valid for all scenarios, is that there is very little difference in blocking probability between all the algorithms, although the cost-based algorithms still have a small advantage. This is due to the fact that a session is blocked if any of its components is blocked; therefore, the blocking probability is a much "coarser" measure of performance in multiple-multicast sessions than for single-multicast sessions. Moreover, since in the cases evaluated the number of multicasts in the session is small and each stream requests a small fraction of the link bandwidth, the problem is in most cases naturally decomposable (i.e., there is no coupling between the routes of the streams in the session) and there should be little difference between the optimum solution (which takes all streams into account simultaneously when computing routes) and the solution found by the heuristic algorithms (which considers each stream in isolation). Note that if the stream bandwidth is a significant fraction of the link bandwidth, this is not true anymore, as shown in [9], and there will be a large difference between the optimum and the heuristics. The simulation results for the NSFNet under videoconferencing traffic (number of participants uniformly distributed between 2 and 4, stream bandwidth set to 10% of the link capacity) are shown in figure 17.

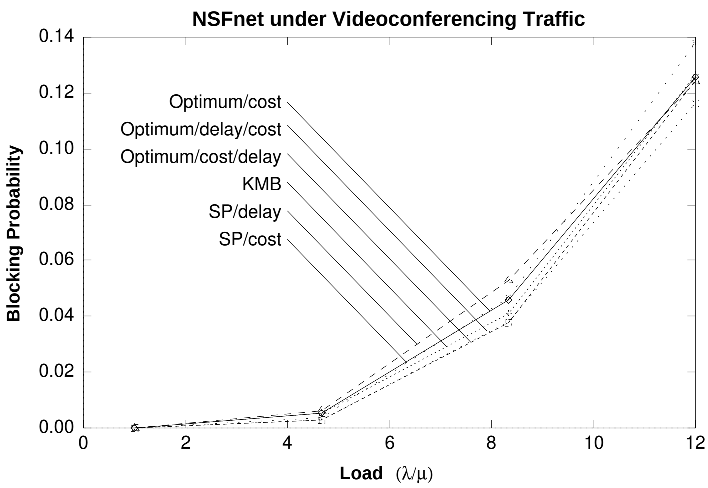

Figure 17: Blocking probability for the NSFNet under videoconference traffic (up to 4 destinations, 10% of the link bandwidth)

Another observation is that, except at very low loads, the blocking probability is now very dependent on the number of participants in the conference for all the algorithms; for example, under the Optimum/cost/delay algorithm, the blocking probability at $\lambda/\mu = 10$ is about 5% for conferences with 2 participants, while it reaches 22% for conferences with 4 participants, for networks of the size of the NSFNet.

Non-Unit Costs

In this section, we investigate the effect of non-unit costs. We repeated the simulation for the following three scenarios, using the NSFNet topology:

- Unit link costs;
- Link costs generated at random, uniformly between 0 and 1; and
- Link costs set to the link lengths (i.e., same values as the link delays).

The results are shown in figure 18 for the Optimum/cost algorithm; the figure indicates that the blocking probability is basically the same for random costs and for unit costs, while the blocking probability for the case where the costs are set equal to the link lengths is higher. The reason is that, when the costs are all equal (or uniformly distributed, which means that when one takes an average of many cases, they are, on the average, equal), minimizing the cost means minimizing the amount of network resources used to route the multicast, and that should lead to a lower blocking probability.

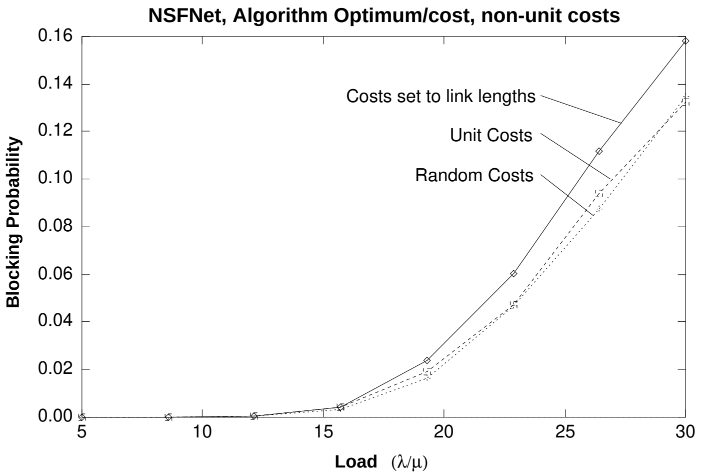

Figure 18: NSFNet, non-unit costs

Other Bandwidth Distributions

In the previous sections, we have assumed that all the streams require the same bandwidth (10% of the link capacity). In an actual network, we expect to find a mixture of bandwidths, corresponding to different qualities of video. This mixture is likely to have more streams at lower bandwidths (i.e., lower video qualities) than at higher bandwidths. Additionally, the bandwidths will belong to a discrete set of values (for example, 384 kb/s, 768 kb/s and 1.984 Mb/s for H.261; 1.5 Mb/s for MPEG I; 2 to 8 Mb/s for MPEG II).

To assess the influence (if any) of the request bandwidth distribution in the performance evaluation, we repeated the baseline case simulations, changing the stream bandwidth from its previous (deterministic) value of 10% of the link bandwidth, to a discrete random variable, assuming the values of 4.5%, 9%, 18% and 36% of the link bandwidth, with probabilities 0.3, 0.3, 0.3 and 0.1 respectively (this would correspond approximately to streams of 2 Mb/s, 4 Mb/s, 8 Mb/s and 16 Mb/s being sent over 45 Mb/s links); the average bandwidth requested is 13%. We observed the same qualitative results as when the streams request 10% of the link bandwidth. In other words, the results presented here are not sensitive to the distribution of the stream bandwidth.

4.8 Run Times for the Algorithms

In this section, we characterize the average run times for the algorithms as a function of the network size. The algorithms were implemented in a DEC 5000/240 workstation in C, and compiled with the highest level of optimization available. Figure 19 shows the average run time for each of the algorithms, for single-multicast sessions in a 6-node network, with the number of destinations chosen at random between 1 and 4. The figure indicates that the run time for the optimum algorithm is one to two orders of magnitude higher than for the heuristics; the difference increases with increasing network size.

Figure 20 shows the run times for the heuristic algorithms only, for single-multicast sessions in 50-node networks; the number of destinations for each multicast is chosen at random between 1 and 10. The figure indicates that the ratio between the run times for the KMB and Shortest-Path algorithms is essentially constant; this is to be expected, because the KMB algorithm corresponds essentially to running the shortest path algorithm a number of times.

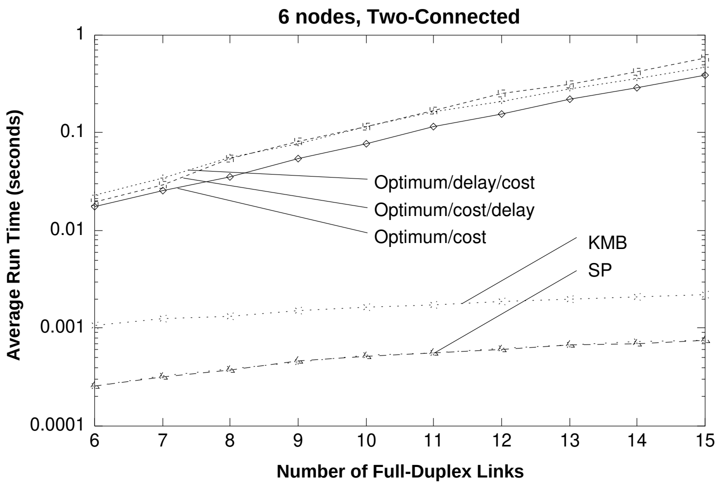

Figure 19: Run times for 6-node networks

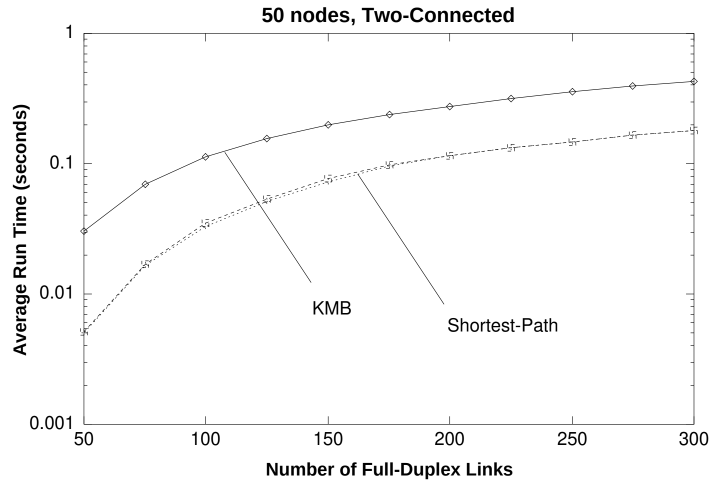

Figure 20: Run times for 50-node networks

One final observation about the run times: for the optimum routing algorithm, we observed that typically the run time for successful sessions (i.e., sessions for which there is at least one solution to the routing problem, given the network usage) is much shorter than the run time when no solution exists. In other words, if there is a solution to the routing problem as formulated in section 2.1, then the optimum routing algorithm will in most cases find it quickly; otherwise, it might take a long time to determine that no solution exists. This is not the case for the heuristics; they take approximately the same time to route a successful session and to give up and declare a session blocked; in fact, a blocked multiple-multicast session might take *less* time to process because not all routes are computed.

The difference in run times could be used to speed-up the optimum routing algorithm by imposing a time limit for finding the solution; if no feasible solution is found when the time limit is reached, the problem is declared infeasible. Such an algorithm is not "optimum" anymore, because there is always a chance that it can miss a solution, or return a sub-optimal one. We evaluated this tradeoff for our implementation of the algorithm, in the DEC 5000/240 workstation, using both the NSFNet topology and random topologies, for sessions with 4-5 multicasts, with 2-5 destinations. The results are shown in figure 21, where we plot the fraction of feasible solutions that would be missed due to the imposition of a time limit, as a function of this time limit. The figure corresponds to 2,346 feasible sessions, and indicates that a reasonable limit is 500 seconds; higher limits would bring diminishing returns. With the 500-second limit, less than 0.2% of the feasible solutions are missed. We should stress that the 500-second figure applies only to our implementation in a DEC 5000/240 workstation and only to networks with 12 nodes and 15 links; for slower CPUs (or bigger networks), higher limits should be used.

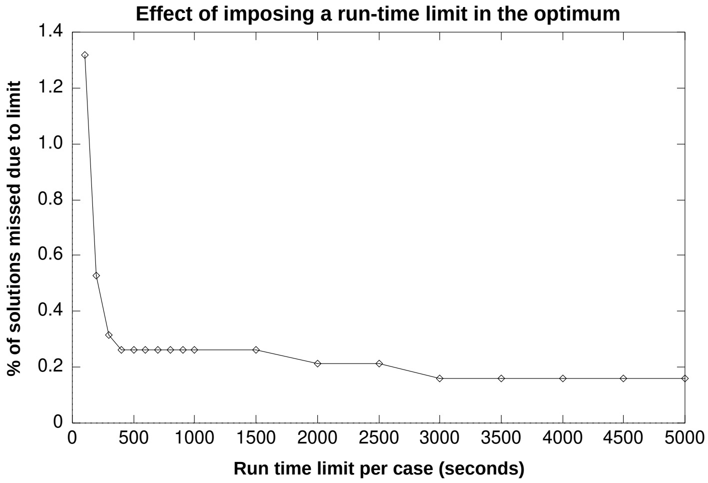

Figure 21: Effect of adding a run-time limit to the optimum

5 Conclusions

The main conclusion of this report is that cost-based algorithms yield, in general, lower blocking probabilities than delay-based algorithms, at the expense of higher delays; the network capacity (defined as the load λ/μ for a target blocking probability) can be 1.2 to 2.0 times higher when the former are used. However, traditional minimum-cost algorithms cannot cope with latency constraints.

We proposed an algorithm for optimum multicast routing; however, due to the large run times (one to two orders of magnitude higher than the heuristics), its main use is as a benchmark for heuristic algorithms (except possibly for small networks). We have found that, under realistic network and traffic conditions, the routes found by the heuristic algorithms are close to the optimum; the only exception is when there are active latency constraints. In this case, the best performance is achieved by the optimum.

Finally, we have found that the best way (from a traffic point of view) to upgrade a network under stream traffic is to add links and make it into a mesh, thus reducing the path length, instead of just increasing the bandwidth of existing links; this is true both for unicast and multicast stream traffic.

Appendix

In this Appendix, we present the algorithms used to generate the random topologies for the evaluation in section 4.

Generating a Completely Random Topology - Full Duplex Links

INPUTS: Number of nodes, N, number of full-duplex links, K, and dimensions of the rectangle (S_x, S_y) in which to place the nodes. It is assumed that $K \geq N-1$.

OUTPUT: The network topology.

Step 1: Generate N node locations at random (horizontal coordinate is generated uniformly between 0 and S_x; vertical coordinate is generated uniformly between 0 and S_y).

Step 2: Create a set of nodes denoted by T, initially empty. Choose one of the nodes at random and add this node to T.

Step 3: Choose at random a node in T and a node not in T, and place a full-duplex link between them in the network topology. Add the node that was not in T to T. Repeat this step until all N nodes are in T.

Step 4: In step 3, $N-1$ links were placed. Place the remaining $K-N+1$ full-duplex links in the topology by choosing their endpoints at random, but allowing only one link between any pair of nodes.

Generating a Random Topology, Short Links

INPUTS: Number of nodes, N, number of links, K, and dimensions of the rectangle (S_x, S_y) in which to place the nodes. It is assumed that $K \geq N-1$, and the links are full-duplex.

OUTPUT: The network topology.

Step 1: Generate N node locations at random (horizontal coordinate is generated uniformly between 0 and S_x; vertical coordinate is generated uniformly between 0 and S_y).

Step 2: Using Prim's algorithm [1], generate a minimum-distance spanning tree and add it to the topology. This is computed by considering the distance graph (i.e., an auxiliary graph where there is a link between every pair of nodes whose cost is the distance between the nodes) and finding its minimum-cost spanning tree. In this step, $N-1$ links are placed.

Step 3: Choose a node at random, say, node n. Let S_n denote the set of nodes to which node n is not connected, say, $\{s_1, s_2, \ldots, s_k\}$. If $d_1, d_2, \ldots, d_n$ are the distances from node n to nodes $s_1, s_2, \ldots, s_k$, choose a node from the set S_n at random, with probabilities proportional to $1/d_1, 1/d_2, \ldots, 1/d_k$, say, s_i, and add a full-duplex link in the topology between n and s_1. Repeat this step until the desired K links are placed, but allowing only one link between any pair of nodes.

Generating a Two-Connected Topology

INPUTS: Number of nodes, N, number of links, K, and dimensions of the rectangle (S_x, S_y) in which to place the nodes. It is assumed that $K \geq N-1$, and the links are full-duplex.

OUTPUT: The network topology.

Step 1: Generate N node locations at random (horizontal coordinate is generated uniformly between 0 and S_x; vertical coordinate is generated uniformly between 0 and S_y).

Step 2: Generate a ring as follows: first, find the two nodes closest to each other and add a full-duplex link between them in the topology. This forms a full-duplex path. Then proceed to form a ring, adding the nodes closest to each extremity of the path, until the ring closes. In this step, N links are placed.

Step 3: Choose a node at random, say, node n. Let S_n denote the set of nodes to which node n is not connected, say, $\{s_1, s_2, \ldots, s_k\}$. If $d_1, d_2, \ldots, d_n$ are the distances from node n to nodes $s_1, s_2, \ldots, s_k$, choose a node from the set S_n at random, with probabilities proportional to $1/d_1, 1/d_2, \ldots, 1/d_k$, say, s_i, and add a full-duplex link in the topology between n and s_1. Repeat this step until the desired K links are placed, but allowing only one link between any pair of nodes.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, New Jersey, 1987.

[2] S.E. Deering and D.R. Cheriton, "Multicast Routing in datagram internetworks and extended LANs," *ACM Trans. on Computer Systems*, May 1990, vol.8, no.2, p. 85-110.

[3] J. Moy, "Multicast Extensions to OSPF," *RFC 1584*, Proteon, Inc., March 1994.

[4] R.M. Karp, "Reducibility among combinatorial problems", in *Complexity of Computer Communications*, R.E. Miller and J.W. Thatcher (editors), pp.85-103, Plenum Press, New York, 1972.

[5] F.K. Hwang and D.S. Richards, "Steiner Tree Problems," *Networks*, Jan. 1992, vol.22, no.1, p. 55-89.

[6] L. Kou, G. Markowsky and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica* 15, pp 141-5, 1981.

[7] V.J. Rayward-Smith and A. Clare, "On Finding Steiner Vertices," *Networks*, Fall 1986, vol. 16, no.3, pp. 283-94.

[8] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Math. Japonica* 6, 1980, pp. 573-7.

[9] C. Noronha and F. Tobagi, "Optimum Routing of Multicast Audio and Video Streams in Communications Networks," *Technical Report CSL-TR-94-618*, Computer Systems Laboratory, Stanford University, April 1994.

[10] V.P. Kompella et al. "Multicast Routing for Multimedia Communication," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, June 1993, pp 286-92.

[11] C. Noronha and F. Tobagi, "Optimum Routing of Multicast Streams," to appear at *IEEE INFOCOM 94*, Toronto, Canada, June 1994.

[12] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research*, 5th edition, New York, McGraw-Hill, 1990.

[13] K. Bharath-Kumar and J.M. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Trans. on Comm.*, vol. COM-31, no. 3, March 1983, pp 343-51.

[14] B.M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, (Dec. 1988) vol.6, no.9, p. 1617-22.

[15] M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing?" *IEEE INFOCOM '93*, San Francisco, CA, USA, pp 82-9.

[16] Yiu-Wing Leung et al. "Efficient algorithms for multiple destinations routing," *ICC 91*, Denver, CO, USA, 23-26 June 1991, p. 1311-17 vol.3.

[17] C.-H. Chow, "On multicast path finding algorithms," *IEEE INFOCOM '91*, Bal Harbour, FL, USA, 7-11 April 1991, pp. 1274-83.

[18] S.E. Dreyfus and R.A. Wagner, "The Steiner Problem in Graphs," *Networks*, vol. 1, pp 195-207, 1972.

[19] M.H. Ammar et al, "Routing Multipoint Connections Using Virtual Paths in an ATM Network," *IEEE INFOCOM '93*, San Francisco, CA, USA, pp 98-105.

[20] McKinley, P.K., and Liu, J.W.S., "Multicast tree construction in bus-based networks," *COMMUNICATIONS OF THE ACM*, (Jan. 1990) vol.33, no.1, p. 29-42.

[21] X. Jiang, "Routing broadband multicast streams," *Computer Communications* (Jan.-Feb. 1992) vol.15, no.1, p. 45-51.

[22] K. Bharath-Kumar and J.M. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IBM Research Report RC9098*, Oct. 1981.

[23] İ. Dalgıç and F. Tobagi, "Evaluation of 10Base-T and 100Base-T Ethernets Carrying Video, Audio and Data Traffic," to appear at *IEEE INFOCOM 94*, Toronto, Canada, June 1994.