OPTIMUM ROUTING OF MULTICAST AUDIO AND VIDEO STREAMS IN COMMUNICATIONS NETWORKS

Ciro A. Noronha Jr. and Fouad A. Tobagi

Technical Report No. CSL-TR-94-618

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 also supported by a graduate scholarship from FAPESP under grant 89/1658.

Optimum Routing of Multicast Audio and Video Streams in Communications Networks

Ciro A. Noronha Jr. and Fouad A. Tobagi

Technical Report: CSL-TR-94-618

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

Abstract

In this report, we consider the problem of routing multicast audio and video streams in a communications network. After describing the previous work in the area and identifying its shortcomings, we show that the problem of optimally routing multicast streams can be formulated as an integer programming problem. We propose an efficient solution technique, composed of two parts: (i) an extension to the decomposition principle, to speed up the linear relaxation of the problem, and (ii) enhanced value-fixing rules, to prune the search space for the integer problem. We characterize the reduction in run time gained using these techniques. Finally, we compare the run times for the optimum multicast routing algorithm and for existing heuristic algorithms.

Key Words and Phrases: multicast routing, audio and video streams, multimedia, linear programming, integer programming, decomposition.

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. Due to their bandwidth and latency requirements, streams are given priority over data; in other words, in a given link, a certain amount of bandwidth is reserved for stream traffic. 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 bandwidth of the path used. Moreover, multicasts happen only occasionally; it is not very important to route them efficiently (in fact, bridged networks implement multicast by broadcasting the information). 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 section 2, we describe the two elements of the routing problem, namely the network and the traffic, and give a formal problem definition for the multicast stream routing problem. In section 3, we describe the previous work in the area of multicast routing algorithms, indicating the shortcomings of existing algorithms. In section 4 we present an integer programming formulation for the optimum multicast stream routing problem, and in section 5 we present an efficient solution technique, based on the branch-and-bound method, which has two parts: (i) an extension of the decomposition procedure, to speed-up the linear relaxation of the problem, and (ii) enhanced value-fixing rules, to prune the search space for the integer solution. This speed-up is characterized in section 6, were we also compare the run times for the optimum solution and for the heuristic solutions. Finally, in section 7, we summarize our conclusions. The extension to the decomposition procedure is given in the Appendix.

2 The Problem Formulation

When an application requests a multicast session, 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 characterize the elements of the routing problem, namely the *network* and the *traffic model*, and formulate the routing problem.

2.1 The Network

The network is a collection of *nodes*, interconnected by *links* subject to a certain *topology*. In this report, we consider all links to be point-to-point and directed (i.e., information can flow in only one direction); latter we show how other kinds of topologies (such as shared-medium and WDM networks) can be accommodated. Full-duplex connections between a pair of nodes correspond to two independent links, one in each direction; this is needed because multicasts are unidirectional, and the bandwidth is managed separately in each direction. Each link is characterized by the following parameters:

Capacity: Link bandwidth, in bits/second.

Cost: Monetary cost of using the link, in \$/bit. In other words, the cost of routing a stream of r bits/second over a link costing C \$/bit is rC \$/second.

Delay: Each link has associated with it a certain delay D, which is the time between the instant a bit of information becomes ready to be transmitted at the source of the link, and the instant it is received in the other end. The delay D has three components, as illustrated in figure 1:

- The node processing delay, D_N. For a given node, this component of the delay is a constant. Since routing is usually hardware-assisted, we neglect this component of the delay.
- The queueing delay, D_Q. This component of the delay is a function of all the traffic in the link, and thus is a function of the routing. It includes the time waiting in the queue and the service time. For data applications, it is common to use a $M/M/1$ model for this queue; in this case, the delay D_Q is a function of the flow in the link. However, since we consider that stream traffic has priority over other kinds of traffic, and has a certain amount of bandwidth pre-allocated to it, D_Q is bounded and typically small, and can be considered independent of the load.
- The propagation delay, D_P. This component of the delay is a function only of the physical length of the link.

In this report, we consider that the link delay D is constant and independent of the flow in the link.

Formally, a network topology with N nodes and K links is described by:

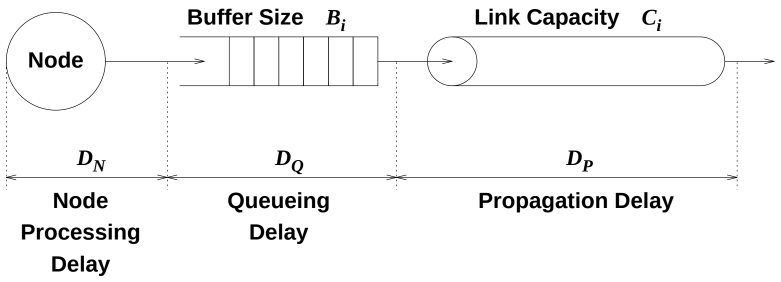


Figure 1: The components of the link delay

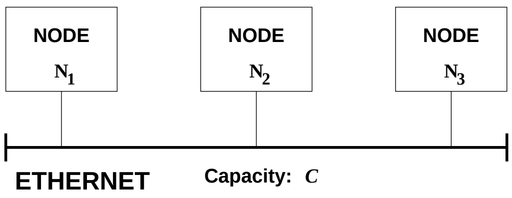
Topology Matrix: Denoted by $\boldsymbol{A}$, is an $N \times K$ matrix where element (i, j) is 1 if node i is the origin of link j, -1 if node i is the destination of node j, and 0 if link j is not connected to node i.

Link Parameters Vector: Denoted by $\boldsymbol{W}$, it is a vector with K triplets (V, C, D), where (V_i, C_i, D_i) are the capacity, cost and delay of link i.

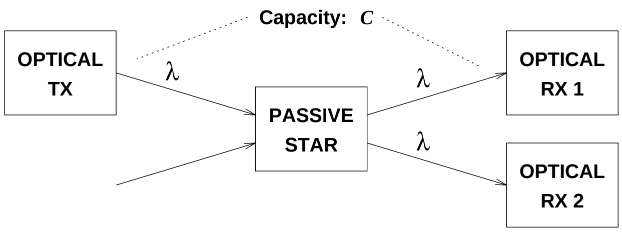
We will denote the network described by $\boldsymbol{A}$ and $\boldsymbol{W}$ by $G(\boldsymbol{A}, \boldsymbol{W})$.

The formulation presented here can also be used to describe network topologies which are capable of physical broadcast or multicast. Two examples of such networks are shown in figures 2(a) (a shared-medium network, such as an Ethernet, where a transmission from a node is heard by all other nodes, providing physical broadcast) and 2(b) (an optical network where two or more receivers can be tuned to the same wavelength λ, providing physical multicast).

The models for the networks of figures 2 (a) and (b) are shown in figure 3. For the bidirectional (broadcast) case (figure 3(a)), two "virtual nodes" V_1 and V_2 are created. The access from the nodes to V_1 and from V_2 to the nodes is performed at zero cost/delay. The interconnection between V_1 and V_2 represents the shared channel, which imposes a limit on the capacity of the actual system. For the unidirectional (multicast) case (figure 3(b)), only one "virtual node" (V) is needed, and the actual channel is represented by the link between the source and the virtual node. The virtual node is reached with zero cost/delay, and the actual physical delay and the costs for each of the outgoing links are assigned to the branches leading to the destination nodes.



(a) Bidirectional physical multicast



(b) Unidirectional physical multicast

Figure 2: Physical multicast

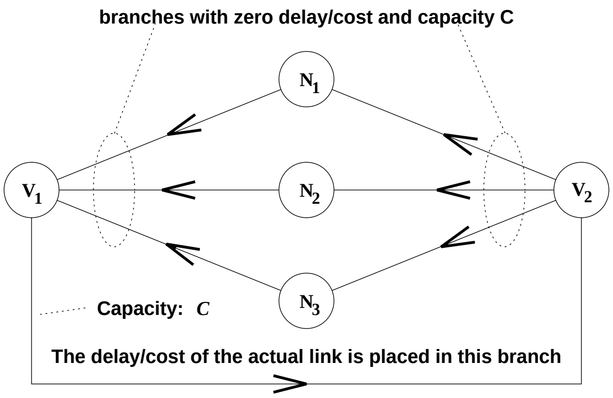
2.2 The Traffic Model

Multicast streams are offered to the network in *sessions*. A session is a group of multicasts which are logically related. One example is a video-conference, where each of the participants can see all other conferees; if there are P participants, a video-conference session would be composed of P multicasts, one from each of the participants to the other $P-1$ conferees. We will denote by T the number of multicasts in the session; each of these multicasts is characterized by:

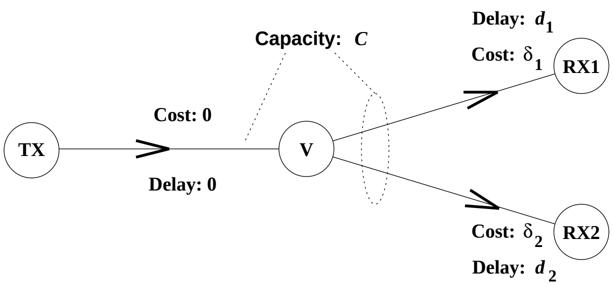
Addressing Parameters: Source s_i and n_i destinations, denoted by $\{d_{i1}, d_{i2}, \ldots, d_{in_i}\}$, $i = 1, \ldots, T$.

Bandwidth Requirement: The amount of bandwidth needed to carry the stream, in bits/second; it will be denoted by r_i, $i = 1, \ldots, T$.

Latency Constraint: The maximum delay acceptable between the source and any of the destinations in this multicast; it will be denoted by L_i, $i = 1, \ldots, T$.



(a) Model for the bidirectional physical multicast



(b) Model for the unidirectional physical multicast

Figure 3: Model for the physical multicast

2.3 Problem Definition

In this section, we give a formal problem definition for the multicast stream routing problem. We define:

Path from node s to node d: Sequence of k links $\{l_1, l_2, \ldots, l_k\}$, where the source of link l_1 is node s, the destination of link l_k is node d, and the node which is the destination of link l_i is also the source of link l_{i+1}, $i = 1, \ldots, k-1$.

Multicast Path from node s to nodes $\{d_1, d_2, \ldots, d_n\}$: Tree formed by merging the paths from s to $d_1, d_2, \ldots, d_n$.

Cost of a Multicast Path: Sum of the costs of the links belonging to the multicast path.

Delay of a Path: Sum of the delays of the links in the path.

Delay of a Multicast Path: Maximum of the delays of the paths that compose the multicast path.

Formally, the multicast stream routing 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 $\{d_{i1}, \ldots, d_{in_i}\}$, its maximum delay constraint L_i and its bandwidth requirement r_i, find a multicast path for each stream that satisfies its bandwidth and delay constraints, while minimizing a given linear combination of the costs and delays of the multicast paths.

3 Previous Work in Multicast Routing

In this section, we describe the previous work in multicast routing algorithms. The description is divided into two parts: (i) algorithms used for multicast routing, and (ii) work in evaluation of multicast routing algorithms.

3.1 Algorithms for Multicast Routing

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:

- Shortest-Path Algorithms; and
- 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). Note that this algorithm computes the routes from a topological point of view; to accommodate for the stream bandwidth requirement, the links which do not have enough free bandwidth to support the stream must be pruned from the network graph prior to applying the algorithm. Moreover, it can only route one stream at a time; for sessions composed of multiple multicasts, it has to be applied sequentially in some arbitrary order to each of the multicasts in the session (and the routes found are order-dependent). Finally, the routes computed must always be checked to ensure that the latency constraint is satisfied, as it is not explicitly taken into account by the algorithm.

Minimum-Cost Algorithms

As with the shortest path algorithm, existing minimum cost algorithms can only compute one route at a time. 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. For a single multicast, 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. For exact solutions, mention should be made to the elegant algorithm proposed by Dreyfus and Wagner [6], the branch-and-bound solution by Shore et al [7], and the linear programming formulation by Beasley [8]. The most important heuristic solutions are the algorithms by Kou, Markowsky and Berman [9] (which we will refer to as KMB), Rayward-Smith [10] (which we will refer to as RS) and Takahashi and Matsuyama [11] (TM). As with the shortest-path algorithms, the network topology must be pruned of the links that do not have enough free capacity. Additionally, the routes have to be checked to ensure that the delay constraint is satisfied. Kompella et al [12] proposed a variation of the KMB algorithm which is able to take into account the latency constraint.

There is another difficulty in applying minimum-cost Steiner tree algorithms to multicast routing: these algorithms are intended to solve *connectivity* problems; in other words, given a graph composed of <u>undirected</u> links and a subset of its nodes, the algorithm will find the subgraph with lowest cost that still provides connectivity (i.e., there is at least one path in the subgraph between any pair of nodes in the subset under consideration). Because the links are undirected, this minimum-cost subgraph will be a tree[1]. However, when routing multicast streams, there is a very well-defined direction of flow, from the source to each of the destinations. Moreover, even if a link is physically full-duplex, the bandwidth in both directions is managed independently, so it has to be treated as a pair of directed links.

There are heuristics specifically designed for directed graphs (for example, see [5, 13, 14]); they are, however, much more complex than their undirected counterparts. An alternative to using these heuristics for multicast routing is to modify one of the well-known undirected heuristics to take into account the directionality of the links. While it is not clear how this would be done to the RS heuristic, it is relatively simple to modify the KMB

[1]If the links are directed, the solution to the minimum-cost connectivity problem is in general a set of trees.

and TM heuristics to operate in directed graphs. In the sequel, we introduce a modification to the KMB algorithm, which makes it capable of operating in directed graphs; this algorithm will be used latter in section 6, where solutions from different algorithms are compared.

The Modified KMB Algorithm

We first introduce a trivial modification to Prim's algorithm [1] to find a minimum-weight directed spanning tree, and then use it to obtain the modified KMB algorithm.

The Minimum Weight Directed Spanning Tree

PROBLEM: Given a directed graph $G(\boldsymbol{A}, \boldsymbol{W})$, where the link weights $\boldsymbol{W}$ are composed only of the link costs C, and a source node s_0, find the minimum-cost spanning tree T_M rooted at s_0, composed of paths from s_0 to all other nodes.

ALGORITHM:

Step 1: Initially, the tree T_M is empty. Add the node s_0 to T_M.

Step 2: Between all the links whose *source* is a node in T_M and whose *destination* is a node that is not in T_M, select the one with the least cost. Add this link and its destination node to T_M.

Step 3: If all nodes in the graph have been added to T_M, stop; otherwise, return to step 2.

The Modified KMB Algorithm

PROBLEM: Given a directed graph $G(\boldsymbol{A}, \boldsymbol{W})$, where the link weights $\boldsymbol{W}$ are composed only of the link costs C, a source node s and a set of n destination nodes $d_1, d_2, \ldots, d_n$, find the minimum cost directed subtree T_H, rooted at s, and composed of paths from s to $d_1, d_2, \ldots, d_n$.

ALGORITHM:

Step 1: Build an auxiliary directed graph G_1 as follows: the nodes in G_1 are s and $d_1, d_2, \ldots, d_n$. For every pair of nodes (n_i, n_j) in G_1, add a directed link in G_1 from n_i to n_j whose cost is equal to the cost of the shortest path (in cost) in the original graph G. Note that the links from $d_1, \ldots, d_n$ to s can be skipped.

Step 2: Using the Minimum Weight Directed Spanning Tree algorithm described above, find the minimum spanning tree T_1 of G_1 with root in s.

Step 3: Construct a subgraph G_2 of G by replacing each link in T_1 by its corresponding shortest path in G.

Step 4: Find the minimum weight directed spanning tree T_2 of G_2 with root in s.

Step 5: Prune from T_2 any leaf nodes that are not in the $d_1, \ldots, d_n$ set; the resulting tree is the minimum cost T_H.

Using Existing Algorithms for Multicast Stream Routing

As indicated before, both the Shortest-Path and the Minimum-Cost algorithms only find the routes in a topological sense, and for one multicast at a time. The issues of bandwidth, latency, and multiple-multicast sessions have to be handled externally to the algorithm. In other words, both the Shortest-Path and the Minimum-Cost algorithms are just a part of the more general multicast stream routing algorithm. This general multicast routing algorithm is illustrated in figure 4 and described below:

PROBLEM: Given a multicast session composed of T multicast streams, each stream having its source, set of destinations and bandwidth and latency requirements, and a network described by $G(\boldsymbol{A}, \boldsymbol{W})$, find multicast paths for each of the streams satisfying their requirements, optimizing a certain objective function.

ALGORITHM:

Step 1: Create a vector $\boldsymbol{U}$ to hold the current usage of each link. Initially set $U_j = 0, \quad j = 1, \ldots, K$. Make a list of not-yet-routed streams, initially containing all requests in the session.

Step 2: Take one request from the list of not-yet-routed streams; let us denote it by request i. For routing this stream, temporarily remove from the network topology all links in which $V_j < r_i + U_j, \quad j = 1, \ldots, K$ (i.e., all links that do not have enough free bandwidth to support this multicast).

Step 3: Route this request using the topology created in step 2, using the single-request routing algorithm (i.e., shortest path, minimum cost, etc).

Step 4: If the routing in step 3 was successful[2], and if the delay of the multicast satisfies the latency constraint, update the $\boldsymbol{U}$ vector as follows:

$$U_j \leftarrow U_j - r_i \qquad j \in \text{multicast path for stream } i$$

Otherwise, if no route was found or the route found does not satisfy this multicast's latency constraint, terminate; there is no solution to the routing problem.

Step 5: Remove request i from the list of not-yet-routed streams and record its route. If all streams have been processed, terminate; routes for all the components of the session have been found. Otherwise, return to step 2.

3.2 Previous Work in Evaluation of Multicast Routing Algorithms

The shortest path and minimum cost algorithms have been studied in the context of multicast routing by a number of researchers. Kumar and Jaffe [15] compared a number of

[2]The routing may fail if the network becomes disconnected when removing links in step 2.

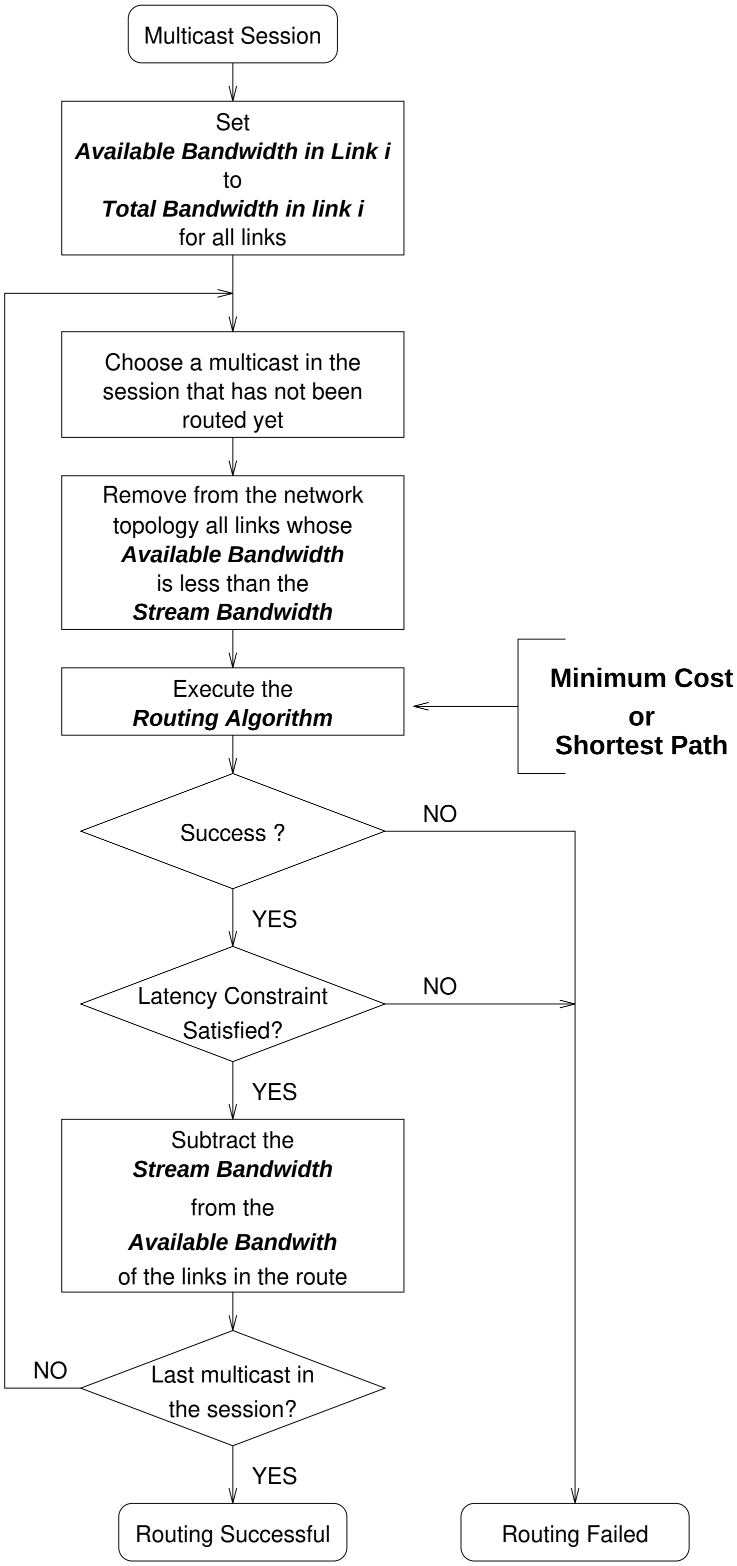


Figure 4: General Multicast Stream Routing Algorithm

minimum delay and minimum cost algorithms when the link cost and delay weights are the same (i.e., $C_i = D_i$) and derived some analytical bounds for this scenario. Waxman [16] and latter Doar and Leslie [17] studied the problem of dynamically adding and removing destinations on a multicast that had been already routed. Waxman studied the RS and KMB heuristics, while Doar and Leslie compared shortest path with KMB for re-routing the multicast. Leung and Yum [18] proposed a number of minimum-cost heuristics, and compared their performance with the RS heuristic. Their heuristics are applicable to directed graphs as well, although this aspect is not explored in their paper. One of their heuristics is actually a variation on the basic Takahashi-Matsuyama heuristic, and was also given by Chow [19]. Chow explicitly studies the case of directed graphs, and proposes a minimum cost heuristic for the case where not all nodes are capable of multicasting. Ammar et al [20] studied the minimum cost routing under a variety of additional constraints using a non-linear integer programming formulation.

Kompella et al [12] studied the case of minimum-cost routing under maximum delay constraints. They assume independent weights for costs and delays ($C_i \neq D_i$), undirected links, and give a variation of the KMB heuristic (with exponential worst-case run time) to compute a sub-optimal solution in the case where the link delays D_i belong to a discrete set of values.

Jiang [21] studies the problem of establishing video-conferences; a video-conference with N participants is established as N multicasts, from each conferee to all other members of the conference. He proposes variations to the KMB and RS heuristics to take into account the link bandwidths. Links are still considered to be undirected, with the bandwidth available in any direction.

In summary, most of the previous work in this field has focused on proposing and evaluating minimum-cost heuristics for routing a single multicast, usually on undirected graphs.

4 An Integer Programming Formulation for the Optimum Multicast Routing

In this section, we show that the multicast routing problem defined in section 2.3 can be written as an integer programming problem, or a linear programming problem if flow bifurcation is allowed.

Definitions:

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:

1. For every stream, there must be a path from its source to each of its destinations. This is equivalent to writing a set of flow conservation equations for routing one unit of flow from the source to each of the destinations:

$$\boldsymbol{A}\boldsymbol{X}^i = \boldsymbol{B}^i \quad i = 1, \ldots, T; \tag{2}$$

2. If a link is in the path from the source to any of the destinations, then it must be included in the multicast path.

$$X^i_{jk} \leq Y^i_j, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K, \quad i = 1, \ldots, T; \tag{3}$$

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

$$M_i - \sum_{j=1}^{K} D_j X^i_{jk} \geq 0, \quad k = 1, \ldots, n_i, \quad i = 1, \ldots, T; \tag{4}$$

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

$$M_i \leq L_i, \quad i = 1, \ldots, T; \tag{5}$$

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; a single path is taken from the source to each of the destinations.

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

Equations (1) to (6) define a linear programming problem, which can be solved by the simplex method; the solution for $\boldsymbol{X}$ and $\boldsymbol{Y}$ is generally non-integer. When constraint (7) is included, the problem becomes an integer programming problem[3].

5 Solution of the Optimum Multicast Routing Problem

In this section, we present an efficient solution technique for the integer programming problem presented in section 4. The technique is based on the well-known branch-and-bound method [22], which has two phases: (i) the linear relaxation (where the integer constraints are relaxed, and the problem is solved as a linear problem) and (ii) the branch and bound phase, where the values of variables with integer constraints are fixed either to zero or to one. In this section, we present enhancements to speed-up both phases.

5.1 Solution to the Linear Relaxation

In this section, we provide a solution to the linear relaxation of the integer programming problem presented in section 4. In this phase we disregard the integer constraints and solve the problem defined by equations (1) to (6), which generally yield non-integer X^i_{jk}.

[3] 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.

Although the linear relaxation could be solved using the traditional simplex method, it is possible to apply an extension of the decomposition procedure [23] to obtain a solution more efficiently. In other words, it is possible to decompose the problem of routing T multicast streams into T single-multicast routing problems. Moreover, each of the single-multicast routing problems can be further decomposed into n_i unicast routing problems, which can be efficiently solved by methods such as network simplex [23]. In the Appendix we show the extension to the decomposition equations presented in [23] used in the solution of the optimum multicast routing problem.

5.1.1 First Decomposition

In this section, we show the decomposition of the problem of routing T multicast streams into T single-multicast routing subproblems. To accomplish this, we observe that equations (2), (3), (4) and (5) apply to each multicast in isolation, while equation (6) is the only connection between flows belonging to different multicasts. Rewriting those equations, and using positive slack variables to turn inequalities into equalities, we find:

GENERAL CONSTRAINTS (valid for $i = 1, \ldots, T$):

$$\begin{aligned}
\boldsymbol{A}\boldsymbol{X}^i &= \boldsymbol{B}^i \\
X^i_{jk} - Y^i_j + Z^i_{jk} &= 0, \quad k = 1, \ldots, n_i, \quad j = 1, \ldots, K \\
Y^i_j + S_{Yj} &= 1, \quad j = 1, \ldots, K \\
\sum_{j=1}^{K} D_j X^i_{jk} - M_i + S^i_{ak} &= 0, \quad k = 1, \ldots, n_i \\
M_i + S_{Li} &= L_i
\end{aligned}$$

(Z^i_{jk}, S_{Y_j}, S^i_{ak} and S_{L_i} are positive slack variables)

COMPLICATING CONSTRAINTS:

$$\sum_{i=1}^{T} r_i \boldsymbol{Y}^i + \boldsymbol{S} = \boldsymbol{V}$$

Let $\boldsymbol{I}_K$ denote the $K \times K$ identity matrix; we define:

$$
\mathcal{X}_i = \left[\begin{array}{c} \boldsymbol{X}^i_1 \\ \boldsymbol{X}^i_2 \\ \vdots \\ \boldsymbol{X}^i_{n_i} \\ \hline \boldsymbol{Y}^i \\ \hline M_i \\ \hline \boldsymbol{Z}^i_1 \\ \vdots \\ \boldsymbol{Z}^i_{ni} \\ \hline \boldsymbol{S}_Y \\ \hline \boldsymbol{S}^i_a \\ \hline S_{Li} \end{array} \right] \tag{8}
$$

$$
\mathcal{A}_i = \left[\begin{array}{cccc|c|c|c|c|c|c}
\boldsymbol{A} & 0 & \cdots & 0 & & & & & & \\
0 & \boldsymbol{A} & \cdots & 0 & & & & & & \\
\vdots & \vdots & & \vdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \cdots & \boldsymbol{A} & & & & & & \\
\hline
\boldsymbol{I}_K & 0 & \cdots & 0 & -\boldsymbol{I}_K & & & & & \\
0 & \boldsymbol{I}_K & \cdots & 0 & -\boldsymbol{I}_K & & & & & \\
\vdots & \vdots & & \vdots & \vdots & 0 & \boldsymbol{I}_{Kn_i} & 0 & 0 & 0 \\
0 & 0 & \cdots & \boldsymbol{I}_K & -\boldsymbol{I}_K & & & & & \\
\hline
\multicolumn{4}{c|}{0} & \boldsymbol{I}_K & 0 & 0 & \boldsymbol{I}_K & 0 & 0 \\
\hline
\boldsymbol{D} & 0 & \cdots & 0 & & -1 & & & & \\
0 & \boldsymbol{D} & \cdots & 0 & & -1 & & & & \\
\vdots & \vdots & & \vdots & 0 & \vdots & 0 & 0 & \boldsymbol{I}_{n_i} & 0 \\
0 & 0 & \cdots & \boldsymbol{D} & & -1 & & & & \\
\hline
\multicolumn{4}{c|}{0} & 0 & 1 & 0 & 0 & 0 & 1
\end{array} \right] \tag{9}
$$

$$\mathcal{B}_i = \left[\begin{array}{c} \boldsymbol{B}_1^i \\ \boldsymbol{B}_2^i \\ \vdots \\ \boldsymbol{B}_{n_i}^i \\ \hline 0 \\ \hline \mathbf{1} \\ \hline 0 \\ \hline L_i \end{array} \right] \tag{10}$$

$$\mathcal{C}_i = \left[\begin{array}{c|c|c|c|c|c|c} 0 & r_i \beta_c \boldsymbol{C} & r_i \beta_d & 0 & 0 & 0 & 0 \end{array} \right] \tag{11}$$

$$\mathcal{D}_i = \left[\begin{array}{c|c|c|c|c|c|c} 0 & r_i \boldsymbol{I}_K & 0 & 0 & 0 & 0 & 0 \end{array} \right] \tag{12}$$

We can re-write the multicast optimization problem as:

MINIMIZE:

$$\sum_{i=1}^{T} \mathcal{C}_i \mathcal{X}_i$$

WITH RESPECT TO:

$$\mathcal{X}_i$$

SUBJECT TO:

$$\begin{aligned} \mathcal{A}_i \mathcal{X}_i &= \mathcal{B}_i \\ \sum_{i=1}^{T} \mathcal{D}_i \mathcal{X}_i + \boldsymbol{S} &= \boldsymbol{V} \end{aligned}$$

which is the same formulation as the one presented in equations (19), (20) and (21) in the Appendix, with $\boldsymbol{E} = 0$ and $\boldsymbol{F} = 0$. The condition $0 \leq \boldsymbol{\mathcal{X}}_i \leq \boldsymbol{U}_i$ is satisfied because $\boldsymbol{X}^i$, $\boldsymbol{Y}^i$ and $\boldsymbol{Z}^i$ are binary, and M_i and $\boldsymbol{S}_a$ are limited by the maximum delay in the network. Each of the subproblems corresponds exactly to the problem of routing a single multicast request over an empty network, with arbitrary link weights.

5.1.2 Second Decomposition

In the previous section, we showed how the general multicast problem can be decomposed into T subproblems, each one corresponding to one multicast request, and a master problem. In this section, we will look into these multicast subproblems in detail, and show how they further decompose into n_i unicast routing problems.

The subproblem to be solved in the first decomposition is:

MAXIMIZE:

$$(\omega \mathcal{D}_i - \mathcal{C}_i)\mathcal{X}_i + \alpha_i \tag{13}$$

WITH RESPECT TO:

$$\omega$$

SUBJECT TO:

$$\mathcal{A}_i \mathcal{X}_i = \mathcal{B}_i \tag{14}$$

Introducing (8), (11) and (12) in the objective function (13), it becomes:

$$(\omega \mathcal{D}_i - \mathcal{C}_i)\mathcal{X}_i = r_i \left[(\boldsymbol{\omega} - \beta_c \boldsymbol{C}) \boldsymbol{Y}^i - \beta_d \boldsymbol{M} \right] \tag{15}$$

We can now rewrite the problem defined by equations (14) and (15) into the format presented in the Appendix; most of the matrices involved can be identified by inspection from equations (8) to (12):

$$\overline{\mathcal{X}}_j = \boldsymbol{X}_j^i$$

$$\overline{\mathcal{A}}_j = \boldsymbol{A}$$

$$\overline{\mathcal{B}}_j = \boldsymbol{B}_j^i$$

$$\overline{\mathcal{C}}_j = 0$$

$$\overline{\mathcal{Y}} = \left[\begin{array}{c} \boldsymbol{Y}^i \\ \hline M_i \end{array} \right]$$

$$\overline{\mathcal{E}} = \left[\begin{array}{c|c} -r_i(\omega - \beta_c C) & r_i\beta_d \end{array} \right]$$

$$\overline{\mathcal{D}}_j = \left. \left[\begin{array}{l} 0 \\ \vdots \\ \boldsymbol{I}_K \quad \text{(position } Kj\text{)} \\ \vdots \\ 0 \\ \hline 0 \\ \hline 0 \\ \vdots \\ D_i \quad \text{(row } n_iK+K+j\text{)} \\ \vdots \\ 0 \\ \hline 0 \end{array} \right] \right\} n_iK + K + n_i + 1$$

$$\overline{\mathcal{F}} = \left. \left[\begin{array}{c|c} -\boldsymbol{I}_K & \\ \vdots & 0 \\ -\boldsymbol{I}_K & \\ \hline \boldsymbol{I}_K & 0 \\ \hline & -1 \\ & -1 \\ 0 & \vdots \\ & -1 \\ \hline 0 & 1 \end{array} \right] \right\} n_iK + K + n_i + 1$$

$$\overline{\mathcal{S}} = \begin{bmatrix} \boldsymbol{Z}_1^i \\ \vdots \\ \boldsymbol{Z}_{n_i}^i \\ \boldsymbol{S}_Y \\ \boldsymbol{S}_a^i \\ S_{Li} \end{bmatrix}$$

$$\overline{\mathcal{V}} = \left[\begin{array}{c} 0_{n_i K \times 1} \\ \hline \mathbf{1}_{K \times 1} \\ \hline 0_{n_i \times 1} \\ \hline L_i \end{array}\right]$$

The problem can be expressed as follows:

MINIMIZE:

$$\sum_{j=1}^{n_i} \overline{\mathcal{C}}_j \overline{\mathcal{X}}_j + \overline{\mathcal{E}} \overline{\mathcal{Y}} = \overline{\mathcal{E}} \overline{\mathcal{Y}} \tag{16}$$

WITH RESPECT TO:

$$\overline{\mathcal{X}}_j, \overline{\mathcal{Y}}$$

SUBJECT TO:

$$\begin{aligned} \overline{\mathcal{A}}_j \overline{\mathcal{X}}_j &= \overline{\mathcal{B}}_j \qquad j = 1, \cdots, n_i & (17) \\ \sum_{j=1}^{n_i} \overline{\mathcal{D}}_j \overline{\mathcal{X}}_j + \overline{\mathcal{F}} \overline{\mathcal{Y}} + \overline{\mathcal{S}} &= \overline{\mathcal{V}} & (18) \end{aligned}$$

5.1.3 Summary of the Decomposition Procedure

In sections 5.1.1 and 5.1.2 we have shown that the problem of routing T multicast streams can be first divided into T subproblems, each one representing the routing of a single multicast request to its n_i destinations. This problem can be further decomposed into n_i unicast routing problems, from the source to each of the destinations of that stream. In other words, the original problem of routing the T multicast streams is decomposed into

$\sum_{i=1}^{T} n_i$ unicast routing problems, as illustrated in figure 5. In this section we summarize the full algorithm.

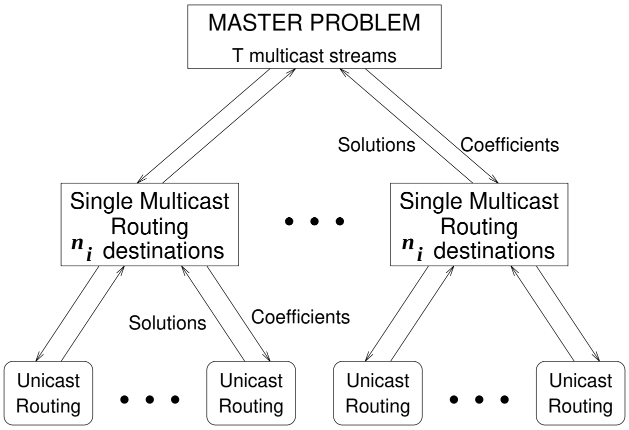


Figure 5: Illustration of the decomposition algorithm

The complete solution algorithm is:

INITIALIZATION STEP:

Start with a feasible initial solution. The unicast subproblems can be initialized by just computing the shortest path (or any path) between each source and its destinations; the initial solutions to the multicast subproblems are directly computed from the solution to the unicast subproblems by using equation (18). The initial solution to the master problem can be found by the two-phase method [23, 22]. The revised simplex array (see equation 32) is then built.

MAIN STEP

- Select the entering variable according to one of the rules indicated below. If no variable can enter, then stop - the optimum has been reached. The rules to identify the entering variable are:

  - If $\omega_k > 0$, then S_k can enter the basis, $k = 1, \cdots, K$

- If $r_i\left[(\boldsymbol{\omega} - \beta_c \boldsymbol{C})\boldsymbol{Y}^i - \beta_d M_i\right] + \alpha_i > 0$, then the λ_{ij} corresponding to this solution can enter the basis. The solution itself is found by applying the decomposition principle again, as follows:

  * Select the entering variable according to the rules below. If no entering variable can be found, then the solution is at hand. The entering variable is identified as follows ($[\overline{\boldsymbol{\omega}} \quad \overline{\boldsymbol{\alpha}}]$ are the dual variables for this subproblem):

    · If $\overline{\omega}_k > 0$, then $\overline{S}_k$ can enter the basis, $k = 1, \ldots, (n_i+1)(K+1)$.

    · If $\overline{\omega}_{n_i * K + k} - \sum_{l=0}^{n_i - 1} \overline{\omega}_{l*K+k} + r_i(\omega_k - \beta_c C_k) > 0$ then $\boldsymbol{Y}_k^i$ can enter the basis, $k = 1, \ldots, K$.

    · If $\overline{\omega}_{n_i(K+1)+1} - \sum_{l=1}^{n_i} \overline{\omega}_{l+n_iK} - r_i\beta_d > 0$, then M_i can enter the basis.

    · If $\left[\begin{array}{ccc} \overline{\omega}_{1+(j-1)K} + D_1\overline{\omega}_{n_iK+K+j} & \cdots & \overline{\omega}_{jK} + D_K\overline{\omega}_{n_iK+K+j} \end{array}\right] \boldsymbol{X}_j^i + \overline{\alpha}_j > 0$ then the $\overline{\lambda_{ij}}$ corresponding to this solution can enter the basis, $j = 1, \ldots, n_i$. The solution can be found by using network simplex (see [23]), by maximizing this objective function subject to $\boldsymbol{A}\boldsymbol{X}_j^i = \boldsymbol{B}_j^i$.

  * For each entering variable, pivot as follows:

    · If $\overline{S}_k$ will enter the basis, the column to pivot is $\boldsymbol{B}^{-1}\begin{bmatrix} \boldsymbol{e}_k \\ 0 \end{bmatrix}$

    · If $\boldsymbol{Y}_k^i$ will enter the basis, the column to pivot is $\boldsymbol{B}^{-1}\left[\begin{array}{c} -\boldsymbol{e}_k \\ \vdots \\ -\boldsymbol{e}_k \\ \hline \boldsymbol{e}_k \\ \hline 0 \end{array}\right]$

    · If M_i will enter the basis, the column to pivot is $\boldsymbol{B}^{-1}\begin{bmatrix} 0 & {\scriptstyle [n_iK \times 1]} \\ 0 & {\scriptstyle [K \times 1]} \\ -1 & {\scriptstyle [n_i \times 1]} \\ 1 & {\scriptstyle [1 \times 1]} \\ 0 & {\scriptstyle [n_i \times 1]} \end{bmatrix}$

· If $\overline{\lambda_{ij}}$ will enter, the column to pivot is

$$\boldsymbol{B}^{-1}\left[\begin{array}{c} 0\\ \vdots\\ \boldsymbol{X}_j^i\\ \vdots\\ 0\\ \hline 0\\ \hline \left.\begin{array}{c}0\\ \vdots\\ \sum_{k=1}^{K} D_k X_{kj}^i\\ \vdots\\ 0\end{array}\right\} n_i\times 1 \\ \hline 0\\ \hline \boldsymbol{e}_j\end{array}\right]$$

(the summation appears in row $n_iK + K + j$)

- For each entering variable, pivot as follows:
  - If S_k will enter the basis, the column to pivot is $\boldsymbol{B}^{-1}\begin{bmatrix} \boldsymbol{e}_k \\ 0 \end{bmatrix}$
  - If λ_{ij} will enter the basis, then the column to pivot is $\boldsymbol{B}^{-1}\begin{bmatrix} r_i\boldsymbol{Y}^i \\ \boldsymbol{e}_i \end{bmatrix}$, where $\boldsymbol{Y}^i$ corresponds to the extreme point associated with λ_{ij}

5.2 Solution to the Integer Programming Problem

The the general integer programming problem can be solved by the "branch-and-bound" method. The solution to the linear relaxation, as described in section 5.1, is a necessary step to the final solution. In this section, we will describe only the additions to the branch-and-bound method to prune the search space, thus making it run faster.

A phase in the branch-and-bound method consists on setting binary variables either to 0 or to 1. We can use our knowledge of the structure of the problem to further reduce the number of free variables in this phase. The variables with integer constraints are $\boldsymbol{X}$ and $\boldsymbol{Y}$; however, if $\boldsymbol{X}$ is integer, the constraint set and the objective function will automatically ensure that $\boldsymbol{Y}$ is also integer. Therefore, when fixing values, we can consider only the components of $\boldsymbol{X}$. The following rules can be applied to reduce the number of free variables:

At initialization: Since the links leading to the source of a multicast will never carry any flow for that multicast, one can set:

$$X^i_{jk} = 0 \qquad i = 1, \ldots, T; \quad k = 1, \ldots, n_i; \quad j : A_{s_i j} = -1$$

where s_i is the source of multicast i. By the same token, links originating from multicast destinations cannot be in the path to that destination:

$$X^i_{jk} = 0 \qquad i = 1, \ldots, T; \quad j, k : A_{d_{ik} j} = 1$$

When X^i_{kj} is set to 0: This means that request i, on its way to its k^{th} destination, will not use link j. Denoting by $\mathcal{N}$ the node from which link j originates, and assuming that l links originate from $\mathcal{N}$, if $\mathcal{N}$ is the origin of request i (or if one of the incoming links to $\mathcal{N}$ has been set to 1 as a result of previous value-fixing) and if the X^i_{kj} corresponding to $l-2$ branches out of the remaining $l-1$ branches have been fixed to 0, then the last remaining X^i_{kj} out of $\mathcal{N}$ can be set to 1.

When X^i_{kj} is set to 1: This means that request i, on its way to its k^{th} destination, will use link j. Denoting by $\mathcal{N}$ the node from which link j originates, and assuming that l links originate from $\mathcal{N}$, the X^i_{kj} corresponding to the remaining $l-1$ links must be all set to 0. Set also $Y^i_j = 1$, as per equation (3).

Capacity Constraints: At any given step in the value-fixing process, there are some X^i_{jk} that have been set to 1, setting the corresponding Y^i_j also to 1. This means that a certain amount of bandwidth (given by r_i) has been reserved on that link already. Therefore, streams requiring more than the free bandwidth on that link should not use it. Formally:

$$\text{Define:} \quad U_j = V_j - \sum_{i:Y^i_j \text{ is set to 1}} r_i \qquad j = 1, \ldots, K$$

$$\text{If } r_i > U_i, \text{ set } X^i_{jk} = 0, \quad k = 1, \ldots, n_i, \quad i : Y^i_j \text{ is not set}$$

Note that these rules can be applied recursively, i.e., if setting a variable to 1 implies setting another variable to 0 (or to 1), the corresponding rule can also be invoked. Moreover, if a conflict arises (e.g., the rule being used calls for fixing a certain variable to 0 when it is already fixed to 1, or vice-versa), then it is not necessary to solve the linear relaxation for that particular case; it can be immediately marked as infeasible.

6 Run Time Evaluation

In this section, we give a numerical evaluation of the average run times of the algorithm presented in this report, and compare them with run times for the heuristic solutions as described in section 3.1. The algorithms were implemented in a DEC 5000/240 workstation in C, and compiled with the highest level of optimization available. The evaluation consisted of routing a single multicast session in an empty network.

The numerical evaluation is organized as follows:

Step 1: Obtain the run times for the linear relaxation of the integer programming problem presented in section 4, both for the traditional simplex method and for the decomposition shown in section 5.1, to characterize the improvement gained when the original problem is decomposed into subproblems.

Step 2: Obtain the run times for the integer programming problem, with and without the pruning rules of section 5.2, to characterize the speed-up when they are employed.

Step 3: Obtain the run times for the heuristic algorithms of section 3.1 (shortest path and minimum cost) and compare with the linear programming approach.

6.1 Evaluation Scenarios

The evaluation scenarios have two components: (i) the *network* scenario, and (ii) the *traffic* scenario. This section describes the choices made for these two components in the evaluation.

For the network scenario, we chose the following:

Topology: A simplified version of the NSFNet T3 backbone, with 12 nodes and 15 full-duplex links, shown in figure 6. We also considered topologies with 6 nodes and 8 full-duplex links (half of the size of the NSFNet), and 6 nodes and 15 full-duplex links (a completely-connected topology), generated at random.

Link Costs: Link costs have been set to 1 for all topologies.

Link Delays: For the NSFNet, the link delays have been set equal to the propagation delays, as indicated in figure 6, where the link delays (in milliseconds) are shown next to the links. For the 6-node topologies, the link delays were generated at random between 0 and 18 ms.

Link Capacities: All links are assumed to have the same capacity.

For the traffic scenario, we chose the following:

Number of Multicasts per Session: between 1 and 4.

Number of Destinations: variable, although all the multicasts in a given session have the same number of destinations.

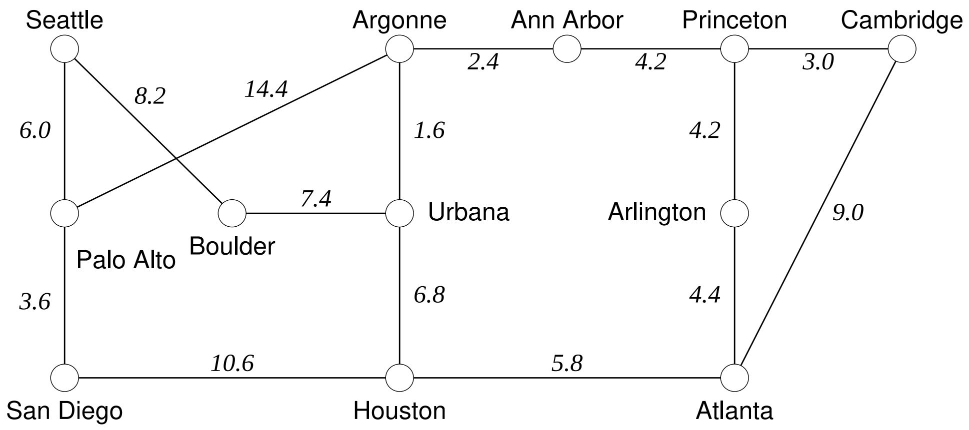


Figure 6: The NSFNet T3 backbone (simplified)

Addressing: sources and destinations are chosen at random, uniformly between all the nodes in the network.

Stream bandwidth: the bandwidth requirement for all streams in a given session is constant. In this evaluation, for single-multicast sessions, the bandwidth is irrelevant as long as it is less than the link capacity. For 2-multicast sessions, we set the bandwidth requirement to 60% of the link bandwidth, and for 4-multicast sessions, to 30%.

Latency Constraint: no latency constraint was imposed.

We considered the following objective functions for the optimum multicast routing algorithm:

- minimum cost ($\beta_c = 1$, $\beta_d = 0$ in equation (1)); this solution will be referred to as "Cost" in the discussion;
- minimum cost, with delay as a secondary objective ($\beta_c \gg \beta_d > 0$); this solution will be referred to as "Cost/delay"; and
- minimum delay, with cost as a secondary objective ($\beta_d \gg \beta_c > 0$); this will be referred to as "Delay/cost".

Note that, for single-multicast sessions, the multicast path found by the "Delay/Cost" solution 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 equivalent. 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.

6.2 Run Time Evaluation of The Optimum Multicast Routing Algorithm

In this section, we evaluate the speed-up gained by the use of decomposition (as compared to the traditional simplex) and by the enhanced value-fixing rules, when computing the optimum solution for the multicast stream routing problem.

Speed-Up due to Decomposition

The run time for the linear relaxation of the routing problem is a function of is size; the important parameters are: (i) number of links in the topology; (ii) number of destinations in the multicast; and (iii) number of multicasts in the session.

We applied both the traditional simplex method and the decomposition shown in section 5.1 to the scenarios described in section 6.1; the general conclusion is that decomposition significantly reduces the time needed to compute the optimum. The difference in performance is a function of the size of the problem and the objective function, going all the way from *increase* in run time by a factor of 2 (observed at 6 nodes, 15 links, single multicast, minimum cost) when using decomposition, to an improvement by factor of 100 or more; for example, to compute the routes for a 4-multicast session in the NSFNet, each multicast having 5 destinations, the traditional simplex method took 542 seconds, while using decomposition the time was reduced to 4.8 seconds.

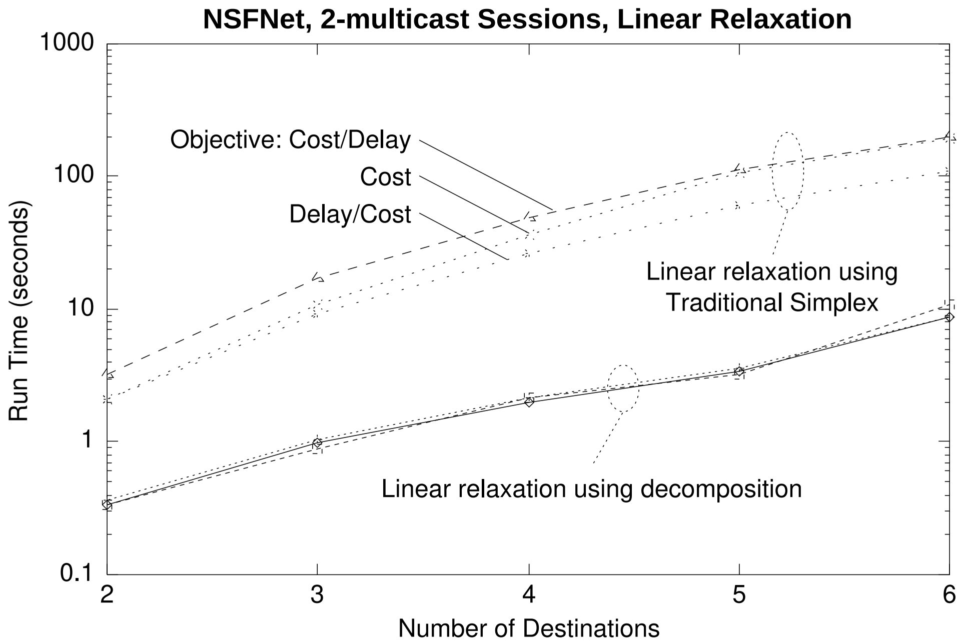


Figure 7: Observed run times for the NSFNet topology, 2-multicast sessions, linear relaxation only

In figure 7, we plot the run time as a function of the number of destinations in the multicast, for 2-multicast sessions in the NSFNet, using the various objective functions. For this scenario, an improvement of 10 times is observed. Figure 7 also shows that the run time for the linear relaxation is largely independent of the objective function when using decomposition. Figure 8 shows a case where decomposition actually *increases* the computation time, namely minimizing cost in a 6-node, 15-link network for a single-session multicast. The reason for the increase is that this is a small, strongly-connected network, where the paths are simple to find; therefore, the overhead of the decomposition procedure is not compensated by the reduction in execution time. The figure also shows that, if the objective is minimizing delay, then the run time is reduced by using decomposition. For sessions of more than one multicast, decomposition always reduces the run time.

Speed-Up due to the Pruning of the Search Space

In this section, we evaluate the reduction in the search space of the integer problem resulting from the value-fixing rules described in section 5.2. In all cases, the linear relax-

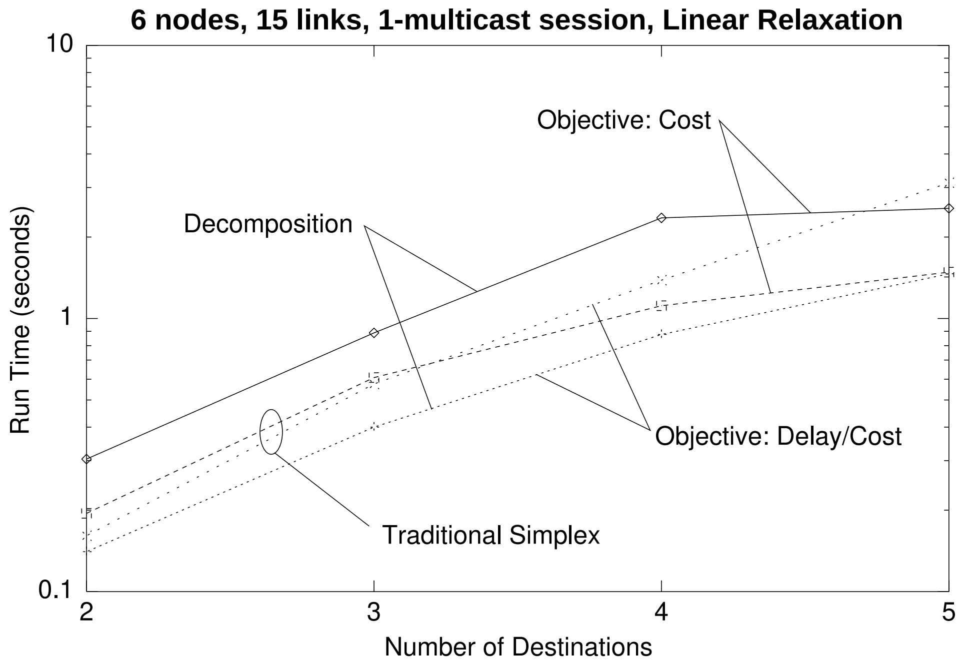


Figure 8: Observed run times for a 6-node, 15-link topology, 1-multicast sessions, linear relaxation only

ations of the original problem were solved using decomposition. The main result is that, in general, the impact of the value-fixing (pruning) rules is much less dramatic than the effect of decomposition. Figure 9 shows the run times for 2-multicast sessions in the NSFNet, as a function of the number of destinations. The figure shows that, while there is little advantage if the objective is to minimize cost, there is a large advantage (2 to 4 times) if the objective is to minimize delay. Similar comments can be made for the other scenarios.

6.3 Run Times for the Heuristic Algorithms

In this section, we evaluate the run times for the minimum-cost and shortest-path algorithms, as described in section 3.1. The general conclusion is that these algorithms run much faster than the optimum presented in this report; however, for multiple multicast sessions, they might not find any solution to the multicast routing problem, even though a solution exists. The run times for the KMB heuristic and the shortest-path algorithm are shown in figure 10, for single-multicast sessions over the NSFNet; the heuristics are one to

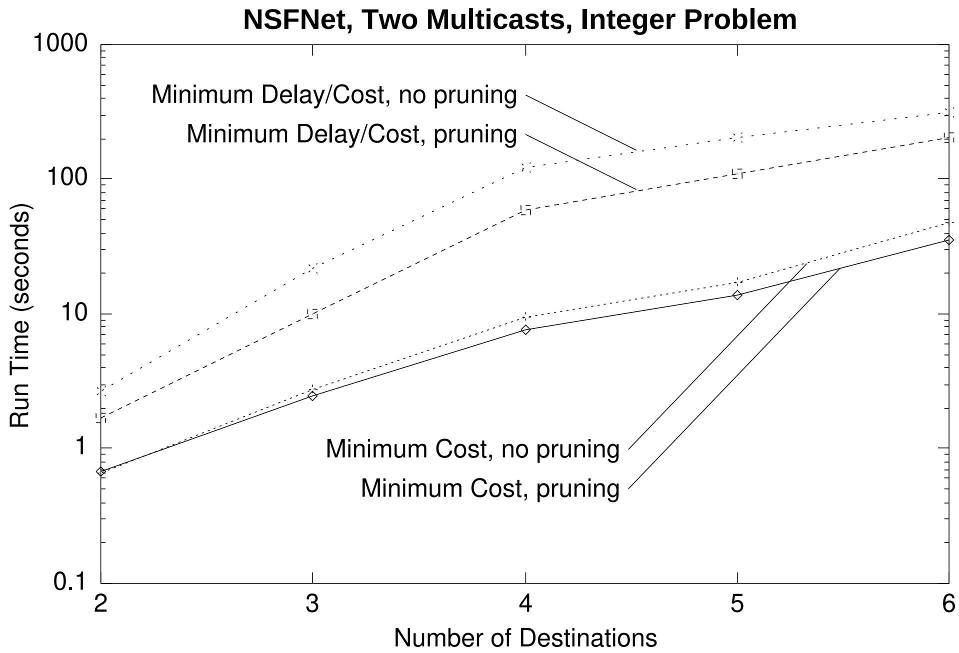


Figure 9: Observed run times for the NSFNet topology, 2-multicast sessions, integer problem

two orders of magnitude faster than the optimum.

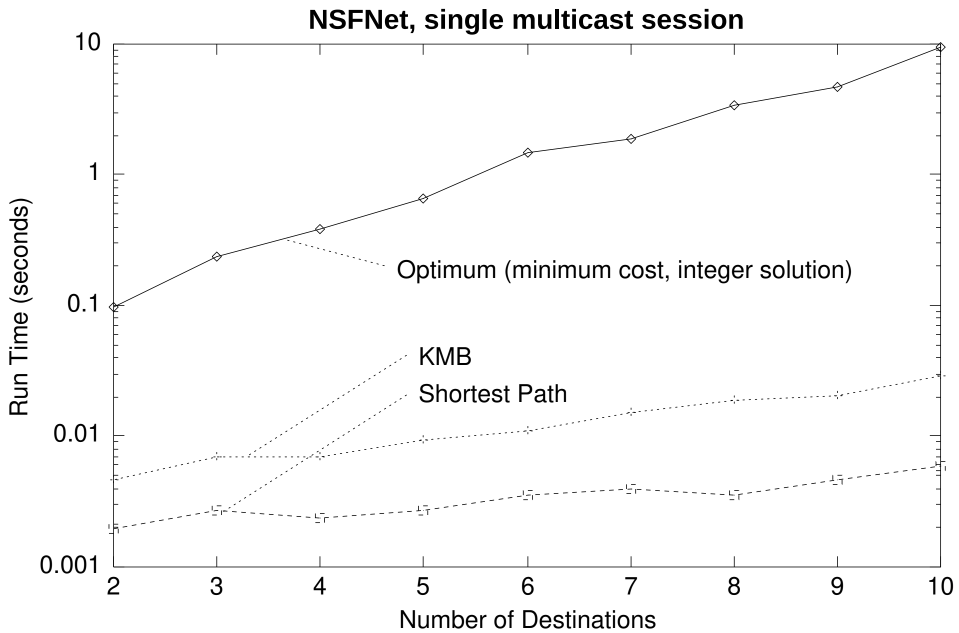


Figure 10: Observed run times for the NSFNet topology, single-multicast sessions

Table 1: Success Probability for a 6-node, 8-link network under 2-multicast sessions

# Destinations	KMB	Shortest Path	Optimum
2	0.95	0.95	1.0
3	0.90	0.80	
4	0.75	0.70	
5	0.60	0.50	

Table 1 shows the fraction of successful routes for 2-multicast sessions, routed through a 6-node, 8-link network, for the different methods. In all cases, there was at least one solution, and the optimum algorithm always found it; the heuristics not always were able to find the routes. For example, for 5-destination multicasts, the shortest path failed in 50% of the cases, and the KMB failed in 40%. However, it should be stressed that these figures are typical of scenarios where the stream bandwidth requirements are a large fraction of the link bandwidth; if they are not, then the fraction of successful routes for the heuristic algorithms is close to the optimum.

7 Conclusions

In this report, we showed that the optimum multicast routing problem for multimedia streams can be formulated as an integer programming problem, and proposed an efficient solution technique. We have shown that the proposed techniques vastly reduce the run times, when compared with traditional methods. We also compared the run time of the optimum solution with well-known heuristic algorithms, with the appropriate modifications to operate in the multicast stream environment. The run time for the heuristics is still one to two orders of magnitude less than that for the optimum, but they might fail to find a solution even if one exists, when routing multiple-multicast sessions. In a companion report [24], we present a complete evaluation of the optimum multicast routing algorithm under realistic conditions, and compare its performance to the heuristic algorithms.

For large networks, the optimum solution presented here is not practical; its main use is as benchmark for other multicast routing algorithms.

Appendix

In this appendix we show an extension of the decomposition problem shown in [23], used in the solution of the linear relaxation of the optimum multicast routing problem.

A Formulation

Consider the following optimization problem (bold letters represent matrices or vectors):

GIVEN: $\boldsymbol{A}_i, \boldsymbol{b}_i, \boldsymbol{C}_i, \boldsymbol{D}_i, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{V}, \boldsymbol{U}_i$

MINIMIZE:

$$\sum_{i=1}^{T} \boldsymbol{C}_i \boldsymbol{x}_i + \boldsymbol{E}\boldsymbol{y} \tag{19}$$

WITH RESPECT TO: $\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{s}$

SUBJECT TO:

$$\begin{aligned}
&\boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}_i, \quad i = 1, \ldots, T && (20)\\
&\sum_{i=1}^{T} \boldsymbol{D}_i \boldsymbol{x}_i + \boldsymbol{F}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{V} && (21)\\
&0 \le \boldsymbol{x}_i \le \boldsymbol{U}_i\\
&\boldsymbol{y} \ge 0\\
&\boldsymbol{s} \ge 0
\end{aligned}$$

Where $\boldsymbol{A}_i$ is $N \times K$, $\boldsymbol{b}_i$ is $N \times 1$, $\boldsymbol{C}_i$ is $1 \times K$, $\boldsymbol{D}_i$ is $M \times K$, $\boldsymbol{E}$ is $1 \times L$, $\boldsymbol{F}$ is $N \times L$, $\boldsymbol{y}$ is $L \times 1$, $\boldsymbol{x}_i$ is $K \times 1$, and $\boldsymbol{s}, \boldsymbol{V}$ are $M \times 1$. Additionally, we will assume that all elements in vector $\boldsymbol{V}$ are non-negative.

B The Decomposition Procedure

Define $\boldsymbol{X}_i = \{\boldsymbol{x}_i : \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}_i, \quad 0 \le \boldsymbol{x}_i \le \boldsymbol{U}_i\}$. Since $\boldsymbol{x}_i$ is limited, the set $\boldsymbol{X}_i$ will have a finite number k_i of extreme points. Therefore, any $\boldsymbol{x}_i \in \boldsymbol{X}_i$ can be written as a convex combination of those extreme points, as follows:

$$\boldsymbol{x}_i = \sum_{j=1}^{k_i} \lambda_{ij} \boldsymbol{x}_{ij} \tag{22}$$

$$\lambda_{ij} \geq 0 \tag{23}$$

$$\sum_{j=1}^{k_i} \lambda_{ij} = 1 \tag{24}$$

The $\boldsymbol{x}_{ij}$ in equation (22) are the extreme points of $\boldsymbol{X}_i$.

Introducing equations (22)-(24) into equations (19) and (21)[4], the optimization problem becomes:

GIVEN: $\boldsymbol{x}_{ij}, \boldsymbol{C}_i, \boldsymbol{D}_i, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{V}$

MINIMIZE:

$$\sum_{i=1}^{T} \sum_{j=1}^{k_i} \boldsymbol{C}_i(\lambda_{ij} \boldsymbol{x}_{ij}) + \boldsymbol{E}\boldsymbol{y} \tag{25}$$

WITH RESPECT TO: λ_{ij}, $\boldsymbol{y}, \boldsymbol{s}$

SUBJECT TO:

$$\sum_{i=1}^{T} \sum_{j=1}^{k_i} \boldsymbol{D}_i(\lambda_{ij} \boldsymbol{x}_{ij}) + \boldsymbol{F}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{V} \tag{26}$$

$$\sum_{j=1}^{k_i} \lambda_{ij} = 1, \quad i = 1, \ldots, T \tag{27}$$

$$\lambda_{ij} \geq 0$$

$$\boldsymbol{y} \geq 0$$

$$\boldsymbol{s} \geq 0$$

Equations (25), (26) and (27) can be written in matrix format as:

MINIMIZE:

$$\begin{bmatrix} \boldsymbol{C}_i \boldsymbol{x}_{ij} & \boldsymbol{E} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{y} \\ \boldsymbol{s} \end{bmatrix} \tag{28}$$

SUBJECT TO:

[4]Equation (20) is automatically satisfied by the $\boldsymbol{x}_{ij}$.

$$
\begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M \\ \overline{\boldsymbol{D}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{y} \\ \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{1} \end{bmatrix} \tag{29}
$$

where:

$\overline{\boldsymbol{A}} = [\boldsymbol{D}_i \boldsymbol{x}_{ij}]$

$\boldsymbol{I}_M$ is the $M \times M$ identity matrix

$$
\overline{\boldsymbol{D}} = \begin{bmatrix} 1\cdots1 & 0 & \cdots & 0 \\ 0 & 1\cdots1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \underbrace{0}_{k_1} & \underbrace{0}_{k_2} & \cdots & \underbrace{1\cdots1}_{k_T} \end{bmatrix} = \begin{bmatrix} \underbrace{\boldsymbol{e}_1\cdots\boldsymbol{e}_1}_{k_1} & \cdots & \underbrace{\boldsymbol{e}_T\cdots\boldsymbol{e}_T}_{k_T} \end{bmatrix}
$$

$\boldsymbol{e}_i$ is the i^{th} unit vector (i.e., a vector that has a "1" in row i and "0" elsewhere)

Let us assume that a basic feasible solution in terms of the λ_{ij}'s and $\boldsymbol{y}, \boldsymbol{s}$ is available[5], and let $[\ \boldsymbol{\omega}\ \ \boldsymbol{\alpha}\ ]$ be the vector of dual variables corresponding to this basic solution ($\boldsymbol{\omega}$ is $1 \times M$ and $\boldsymbol{\alpha}$ is $1 \times T$). To determine the entering variable, one has to compute the following vector:

$$
\begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M \\ \overline{\boldsymbol{D}} & 0 & 0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_i \boldsymbol{x}_{ij} & \boldsymbol{E} & 0 \end{bmatrix} \tag{30}
$$

The positive elements in the vector defined by equation (30) correspond to variables that can enter the basis and improve the objective function. Substituting the values for $\overline{\boldsymbol{A}}$ and $\overline{\boldsymbol{D}}$ in equation (30), we find:

$$
\begin{bmatrix} (\boldsymbol{\omega}\boldsymbol{D}_i - \boldsymbol{C}_i)\boldsymbol{x}_{ij} + \alpha_i & \boldsymbol{\omega}\boldsymbol{F} - \boldsymbol{E} & \boldsymbol{\omega} \end{bmatrix} \tag{31}
$$

From (31) we learn:

1. If $(\boldsymbol{\omega}\boldsymbol{D}_i - \boldsymbol{C}_i)\boldsymbol{x}_{ij} + \alpha_i > 0$, then λ_{ij} can enter the basis;

[5] This solution can be obtained using the well-know two-phase method, for example

2. If $(\boldsymbol{\omega F} - \boldsymbol{E})_k > 0$, then y_k can enter the basis;

3. If $\omega_l > 0$, then s_l can enter the basis.

Given $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$, conditions 2 and 3 above are immediate to compute. Since $\boldsymbol{x}_{ij}$ is an extreme point of $\boldsymbol{X}_i$, condition 1 above can be rewritten as:

a. Solve the following problem:

Maximize $(\boldsymbol{\omega D}_i - C_i)\boldsymbol{x}_i + \alpha_i$

Subject to $\boldsymbol{A}_i\boldsymbol{x}_i = \boldsymbol{b}_i, \quad 0 \leq \boldsymbol{x}_i \leq \boldsymbol{U}_i$

b. If the objective value of the problem solved in (a.) is positive, then the λ_{ij} corresponding to the optimum $\boldsymbol{x}_i$ in that problem can enter the basis. Otherwise, no λ_{ij} from subproblem i can enter the basis.

C Solution Algorithm

In the following description, we will denote by *Master Problem* the problem described by equations (28) and (29).

INITIALIZATION STEP:

Begin with a basic feasible solution to the master problem. Store the basis inverse $\boldsymbol{B}^{-1}$, $\overline{\boldsymbol{b}} = \boldsymbol{B}^{-1}\begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{1} \end{bmatrix}$ and $[\ \boldsymbol{\omega} \quad \boldsymbol{\alpha}\ ] = \hat{\boldsymbol{c}_B}\boldsymbol{B}^{-1}$, where $\hat{c}_{ij} = \boldsymbol{C}_i\boldsymbol{x}_{ij}$ for the basic λ_{ij} variables, and $\hat{c}_k = E_k$ for the basic y_k variables, in the revised simplex array shown below:

$$\begin{array}{|c|c|} \hline \begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix} & \hat{\boldsymbol{c}_B}\boldsymbol{B}^{-1}\overline{\boldsymbol{b}} \\ \hline \boldsymbol{B}^{-1} & \boldsymbol{B}^{-1}\overline{\boldsymbol{b}} \\ \hline \end{array} \tag{32}$$

MAIN STEP:

- Select the the entering variable according to the rules given in section B. If there is no candidate to enter the basis, stop – the optimum solution has been reached.

- If a variable has been selected to enter the basis, compute its column, adjoin it to the revised tableau, and pivot. This will update all the variables in the tableau. The columns are computed as follows:

  - If λ_{ij} will enter the basis, the column is $\boldsymbol{B}^{-1}\begin{bmatrix}\boldsymbol{D}_i\boldsymbol{x}_{ij}\\ \boldsymbol{e}_i\end{bmatrix}$

  - If y_k will enter the basis, the column is $\boldsymbol{B}^{-1}\begin{bmatrix}\boldsymbol{F}_k\\ 0\end{bmatrix}$

  - If s_l will enter the basis, the column is $\boldsymbol{B}^{-1}\begin{bmatrix}\boldsymbol{e}_l\\ 0\end{bmatrix}$

- Repeat the previous steps until the optimum has been reached.

D Finding an Initial Solution

The procedure described in the previous section assumes that an initial basic feasible solution is available. In this section, we make use of the two-phase method to identify this initial solution. Let us consider the problem in the format described by equation (29). We add a number of artificial variables that enable us to immediately identify an initial feasible basis for this extended problem. We then optimize to drive the artificial variables out of the basis. We add T artificial variables to the problem, denoted by the $T \times 1$ vector $\boldsymbol{a}$, as follows:

MINIMIZE:

$$\begin{bmatrix} 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \begin{bmatrix} \lambda_{ij} \\ \boldsymbol{y} \\ \boldsymbol{s} \\ \boldsymbol{a} \end{bmatrix} \tag{33}$$

SUBJECT TO:

$$\begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M & 0 \\ \overline{\boldsymbol{D}} & 0 & 0 & \boldsymbol{I}_T \end{bmatrix} \begin{bmatrix} \lambda_{ij} \\ \boldsymbol{y} \\ \boldsymbol{s} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} \tag{34}$$

Since all elements of $\boldsymbol{V}$ are non-negative by hypothesis, one can immediately identify a basis in equation (34): it will be composed by the $\boldsymbol{s}$ variables (with $s_i = V_i$) and by the artificial variables $\boldsymbol{a}$ (with $a_i = 1$). Therefore,

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I}_M & 0 \\ 0 & \boldsymbol{I}_T \end{bmatrix} = \boldsymbol{I}_{M+T} = \boldsymbol{B}^{-1} \tag{35}$$

$$\hat{\boldsymbol{c}_B} = \begin{bmatrix} \underbrace{0 \cdots 0}_{M} & \underbrace{1 \cdots 1}_{T} \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix} = \hat{\boldsymbol{c}_B} \boldsymbol{B}^{-1} = \hat{\boldsymbol{c}_B} \tag{37}$$

$$\overline{\boldsymbol{b}} = \boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} \tag{38}$$

$$\hat{\boldsymbol{c}_B} \boldsymbol{B}^{-1} \overline{\boldsymbol{b}} = \begin{bmatrix} \underbrace{0 \cdots 0}_{M} & \underbrace{1 \cdots 1}_{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{V} \\ \mathbf{1} \end{bmatrix} = T \tag{39}$$

Equations (35) to (39) can be represented by the following revised simplex array:

$0 \cdots 0 \quad 1 \cdots 1$	T
$\boldsymbol{I}_{M+T}$	$\boldsymbol{V}$ $\mathbf{1}$

To determine the entering variable, we must compute:

$$\begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{A}} & \boldsymbol{F} & \boldsymbol{I}_M & 0 \\ \overline{\boldsymbol{D}} & 0 & 0 & \boldsymbol{I}_T \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 1 \cdots 1 \end{bmatrix} \tag{40}$$

which resolves into:

$$\begin{bmatrix} \boldsymbol{\omega} \boldsymbol{D}_i \boldsymbol{x}_{ij} + \alpha_i & \boldsymbol{\omega} \boldsymbol{F} & \boldsymbol{\omega} & \boldsymbol{\alpha} - \mathbf{1} \end{bmatrix} \tag{41}$$

The positive entries in the vector on (41) correspond to variables that can enter the basis. Therefore,

- If $\boldsymbol{\omega} \boldsymbol{D}_i \boldsymbol{x}_{ij} + \alpha_i > 0$, then λ_{ij} can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{D}_i \boldsymbol{x}_{ij} \\ \boldsymbol{e}_i \end{bmatrix}$.
- If $(\boldsymbol{\omega} \boldsymbol{F})_k > 0$, then y_k can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{F}_k \\ 0 \end{bmatrix}$.
- If $\omega_l > 0$, then s_l can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{e}_l \\ 0 \end{bmatrix}$.
- If $\alpha_m - 1 > 0$, then the artificial a_m can enter the basis, and the column to pivot is $\boldsymbol{B}^{-1} \begin{bmatrix} 0 \\ \boldsymbol{e}_m \end{bmatrix}$.

The solution procedure indicated earlier in this appendix used here. At optimality, one of the following three situations will happen:

1. All the artificials are out of the basis. An initial feasible solution has been found. One just has to compute $[\ \boldsymbol{\omega} \quad \boldsymbol{\alpha}\ ] = \hat{\boldsymbol{c}_B} \boldsymbol{B}^{-1}$ and $\hat{\boldsymbol{c}_B}(\boldsymbol{B}^{-1} \overline{\boldsymbol{b}})$, and proceed according to what was indicated in section C.

2. There is at least one artificial in the basis at non-zero level. This means that the original problem is infeasible, i.e., has no solution.

3. There is at least one artificial in the basis at zero level. We can proceed to the main optimization as described in item 1, but during the pivoting process, we always select one of the remaining artificials as the leaving variable, if possible.

References

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

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

[3] Moy, J., "Multicast Extensions to OSPF," *IETF Internet Draft*, July 1993.

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

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

[7] M.L. Shore et al, "An Algorithm for the Steiner Problem in Graphs," *Networks*, vol.12, pp 323-33, 1982.

[8] J.E. Beasley, "An Algorithm for the Steiner Problem in Graphs," *Networks*, vol.14, pp 147-59. 1984.

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

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

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

[12] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, June 1993, pp 286-92.

[13] M. Dror, B. Gavish and J. Choquette, "Directed Steiner Tree Problem on a Graph: Models, Relaxations and Algorithms," *INFOR*, vol.28, no. 3, Aug. 1990, pp. 266-81.

[14] N. Maculan and P. Souza, "An Approach for the Steiner Problem in Directed Graphs," *Annals of Operations Research*, vol. 33, 1991, pp. 471-80.

[15] 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.

[16] Waxman, B.M. "Routing of multipoint connections," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, (Dec. 1988) vol.6, no.9, p. 1617-22.

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

[18] 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.

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

[20] 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.

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

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

[23] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows, 2nd ed.*, John Wiley & Sons, New York, 1990.

[24] C. Noronha and F. Tobagi, "Evaluation of Multicast Routing Algorithms for Multimedia Streams," *Technical Report CSL-TR-94-619*, Computer Systems Laboratory, Stanford University, April 1994.