ABSTRACT

ILIA BALDINE. Multicast Routing With End-to-End Delay and Delay Variation Constraints (Under the direction of Professor George Rouskas.)

We study the problem of constructing multicast trees to meet the quality of service requirements of real-time, interactive applications operating in high-speed packet-switched environments. In particular, we assume that multicast communication depends on (a) bounded delay along the paths from the source to each destination, and (b) bounded variation among the delays along these paths. We first establish that the problem of determining such a constrained tree is \mathcal{NP} -complete. We then derive heuristics that demonstrate good average case behavior in terms of the maximum inter-destination delay variation of the final tree. In addition, our heuristics achieve their best performance under conditions typical of multicast scenarios in high-speed networks. We also show that it is possible to dynamically reorganize the initial tree in response to changes in the destination set, in a way that is minimally disruptive to the multicast session.

MULTICAST ROUTING WITH END-TO-END DELAY AND DELAY VARIATION CONSTRAINTS

by

ILIA BALDINE

A thesis submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Master of Science

COMPUTER SCIENCE

Raleigh

1995

APPROVED BY:

BIOGRAPHY

Ilia Baldine was born August 7, 1972 in Dubna, Moscow Region, Russia. He received the Bachelor of Science in Computer Science degree from the Illinois Institute if Technology, Chicago, in 1993. Prior to the Illinois Institute of Technology he attended Moscow State University, Department of Mathematics and Cybernetics (Moscow, Russia) in 1989-1991 and Hope College (Holland, MI) in 1991-1992.

He was admitted in the Master of Science program at the Department of Computer Science of North Carolina State University in 1993.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to everyone who has helped me in the course of this work. I especially would like to thank

- Professor George Rouskas for his continuous guidance and advice
- Members of my graduate committee Professor Douglas Reeves and Professor Robert Fornaro for their valuable comments
- My parents, without whose love and support this work would have been impossible

Contents

Li	ist of Figures	vi
1	Introduction1.1 Issues in Multicast Communications	1 1
2	Overview of Related Work	4
3	Network Model for Multicasting	7
4	Multicast Trees with Bounded Delay and Delay Variation4.1 Problem Description	10 10 12
5	Multicast Tree Algorithms for DVBMT 5.1 Algorithm Description 5.2 Delay Variation Multicast Algorithm (DVMA) 5.3 Correctness and Complexity Analysis of DVMA 5.4 A Family of Algorithms for DVBMT 5.5 Summary	$egin{array}{c} 15 \\ 15 \\ 18 \\ 24 \\ 25 \\ 26 \end{array}$
6	Dynamic Reorganization of the Multicast Tree 6.1 Handling leave Requests	27 29 29
7	Numerical Results 7.1 Performance of $DVMA$ Family of Algorithms	32 32 39 40

		v
8	8 Summary and Future Work	45
	8.1 Summary	45
	8.2 Future Work	45
В	Bibliography	47

List of Figures

3.1	Graph $G = (V, A)$, source node s , destination set M and the link-delay function $(\mathcal{D}(s, v), \mathcal{D}(s, w))$ and $\mathcal{D}(w, u)$	8
4.1	Instance of $DVBMT$ corresponding to an instance of $PARTITION$ with $S = \{1, 2, 3\}$	14
5.1	Flowchart of our approach to obtaining a multicast tree for the <i>DVBMT</i> problem	17
5.2	The initial tree $T_{initial}$ consisting of a path from s to w and the set $U = M - V_{T_{initial}}$ of unconnected destinations	21
5.3	The partially-built tree T , graph G and node $u \in U$, $U = M - (M \cap V_T)$ (Step 1)	21
5.4	The path to $v \to u$ is being selected among the l shortest paths between v and u in $G'(\operatorname{Step} 2) \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	22
5.5	The new tree T' after the addition of node u , new set U of still unconnected destinations (Step 3)	22
5.6	Heuristic algorithm for the $DVBMT$ problem	23
6.1 6.2	Vertices r and v have left the multicast group Paths to destinations which are in W will change	28 30
7.1	Algorithm comparison for networks with average node degree equal to 2.5 , and multicast group size equal to 5% of the number of nodes	36
7.2	Algorithm comparison for networks with average node degree equal to 2.5, and multicast group size equal to 10% of the number of nodes	36
7.3	Algorithm comparison for networks with average node degree equal to	90
7 4	2.5, and multicast group size equal to 15% of the number of nodes	37
7.4	Algorithm comparison for networks with average node degree equal to 2.5, and multicast group size equal to 20% of the number of nodes	37
7.5	Algorithm comparison for networks with average node degree equal to	
	1.5 and multicast group size equal to 5% of the number of nodes	38

7.6	Algorithm comparison for networks with average node degree equal to	
	4, and multicast group size equal to 5% of the number of nodes	38
7.7	DVMA performance comparison for different values of parameter k (k	
	is a multiple of n), average node degree equal 2.5, and multicast group	
	size equal 10%	39
7.8	Performance comparison between DVMA and the higher complexity	
	heuristic DVMAv2	40
7.9	Performance of the dynamic algorithm compared to the optimal with	
	50/50 join/leave mix	43
7.10	Performance of the dynamic algorithm compared to the optimal with	
	75/25 join/leave mix	43
7.11	Performance of the dynamic algorithm compared to the optimal with	
	100/0 join/leave mix	44

Chapter 1

Introduction

1.1 Issues in Multicast Communications

In multicast communication messages are concurrently sent to multiple destinations, all members of the same multicast group. Mechanisms to support such a form of communication are becoming an increasingly important component of the design and implementation of distributed systems [1]. One of the core issues that needs to be addressed as part of providing such mechanisms is the issue of routing, which primarily refers to the determination of a set of paths to be used for carrying the messages from the source to the destinations nodes. For reasons related to the efficient use of network resources involved in a multicast session, typical approaches to multicast routing require the transmission of packets along the branches of a tree spanning the source and destination nodes.

The problem of computing multicast trees has received considerable attention in the past, and several algorithms have been proposed based on a number of optimization goals. One frequently considered optimization objective is to minimize the total cost of the tree, which is taken as the sum of the costs on the links of the multicast tree. The minimum cost tree is known as the Steiner tree [2], and finding such a tree is a well-known \mathcal{NP} -complete problem [3].

While total tree cost as a measure of bandwidth efficiency is certainly an important parameter, it is not sufficient to characterize the quality of the tree as perceived by the interactive multimedia and real-time applications which are expected to utilize emerging high-speed networks. More specifically, networks supporting real-time traffic will be required to provide certain quality of service guarantees in terms of the end-to-end delay along the individual paths from the source to each of the destination nodes. The problem of routing multicast traffic with real-time constraints has been studied in [4, 5] and heuristics to compute low-cost trees which guarantee an upper bound on the end-to-end delay have been developed.

In this work we consider an additional criterion that can be used to characterize the quality of the multicast tree for interactive, real-time applications. In particular, we assume that, in addition to end-to-end delay bounds, the multicast tree must also guarantee bounds on the *variation* among the delays along the individual source-destination paths. One can think of such a bound as providing synchronization among the various receivers, in order to ensure that no receiver is "left behind" and none is "far ahead" during the lifetime of the session. Although delay variation has not, to the best of our knowledge, been considered in the design of multicast tree algorithms, the maximum delay variation among the paths of the final tree was one of the performance metrics included in the comparative study in [6, 7].

There are several situations in which the need for bounded variation among the end-to-end delays arises. During a teleconference, for instance, it is important that the current speaker be heard by all participants at the same time, or else the communication may lack the feeling of an interactive face-to-face discussion. As another example, consider the use of multicast messages to update multiple copies of a replicated data item (or file) in a distributed database system. Minimizing the delay variation in this case would minimize the length of time during which the database is in an inconsistent state. Furthermore, being able to look at the information carried by the multicast message long before others can do the same, might, for certain applications, translate into gaining a competitive edge. A distributed game scenario in which a number of players are connected over the network to a game server, and compete against each other using information sent by the server to their screens, would be one such example.

1.2 Thesis Organization

This thesis is organized as follows. In Chapter 2 we will present a survey of existing algorithms dealing with the problems described above. Chapter 3 presents a model that captures the salient features of multicast communication over packet-switched networks. In Chapter 4 we show that the problem of constructing trees to guarantee a bounded variation of the end-to-end delays along the source-destination paths is \mathcal{NP} -complete. In Chapter 5 we develop heuristic algorithms for this problem. Following this, in Chapter 6 we also describe an algorithm for dynamically reorganizing the initial tree as nodes are added to, or deleted from the multicast group. We present some numerical results in Chapter 7, and conclude the paper in Chapter 8.

Chapter 2

Overview of Related Work

The problem of multicast routing has been studied quite extensively. In this chapter we will attempt to present a brief overview of the work done in this direction. A comprehensive study of multicast routing algorithms for high-speed networks, along with performance comparisons of different approaches can be found in [6], [7], [8].

The previous chapter has already described some of the issues in multicasting. The most important problem associated with multicasting is routing. To be more exact, the problem involves building a multicast tree which satisfies certain parameters, specified by the user. In addition, we may also wish to do it in most efficient manner from the point of view of the network. This involves using up as few of network resources as possible.

Multicast routing algorithms can be roughly classified into three categories depending on the optimization goals. The first one is governed exclusively by the user constraints on the multicast tree. The only constraint that has been used in the past is the end-to-end delay in the routing tree. The constraint stems from the user's desire to deliver the information to the clients as fast as possible. This problem can be easily solved in polynomial time by any of the existing shortest path algorithms (e.g. Dijkstra's, see [9]), which create a tree of shortest paths, spanning the source node and the destination nodes. The cost of an edge in a graph representing a network in this case is proportional to the propagation delay on the corresponding link of the network. It is obvious that if a tree of shortest paths cannot satisfy the user-imposed

delay constraint, no other tree can.

The second variety looks at the problem purely from the standpoint of the network and attempts to minimize the usage of network resources by the multicast connection. This routing problem has a corresponding well-defined graph problem, known as the Steiner tree problem. The Steiner tree problem involves constructing a tree in a weighted graph, which spans some subset of nodes and possesses minimal cost among all such trees. The weight of an edge in such a graph, representing network topology, could be some measure of the utilization of the corresponding link. Unfortunately, this problem has been proven to be \mathcal{NP} -complete (see [2]), which makes it unlikely that it can be solved by an algorithm in polynomial time. However, there exist several polynomial heuristics, which can successfully approximate the optimal solution to within a factor of 2 or better, like the KMB heuristic (see [10]) or the Rayward-Smith heuristic (see [11]).

The KMB heuristic operates by first creating an induced complete graph on the subset of nodes comprising the destination nodes and the source node. It uses shortest path distances between nodes as costs of the edges of this induced graph. It then uses Prim/Dijkstra's minimum spanning tree algorithm to find such a tree in the induced graph. The last step of the heuristic is the mapping of this tree back into the original graph. In contrast, the Rayward-Smith heuristic proceeds iteratively, starting with the set of subtrees of the size of one node each (the nodes being members of the multicast group and the source node). It uses an empirical function to unite these subtrees into the final multicast tree.

An attempt to reconcile the requirements of the user with the efficient use of network resources produced a third variety of multicast routing problems, known as constrained Steiner tree problem. This is the area where most of the work in multicast routing has been concentrated in the past few years. The goal is to minimize the cost of the multicast tree while satisfying some user-defined constraint or constraints. These requirements involve assigning two different costs to each edge of the graph representing a network. One measure usually reflects the delay on the corresponding link, while the other describes the utilization of that link. Again, the only user constraint, that has been considered so far, is the maximum end-to-end delay

in a multicast tree. Due to the fact that the underlying unconstrained Steiner problem is \mathcal{NP} -complete, only heuristical approaches can be used in the solution of the constrained problem. Two heuristics, developed in the past several years, which are worth noting, are BSMA and KPP.

Both BSMA and KPP try to minimize the overall routing tree cost, while satisfying the end-to-end delay constraint, however, they use two very different approaches. KPP was developed using the KMB heuristic as a basis. It uses an empirical function to construct the required multicast tree. BSMA, on the other hand, attempts to partially enumerate all delay-constrained routing trees and choose the one with the smallest cost. It begins with a tree of shortest paths (since if the tree of shortest paths can't satisfy the delay constraint, no other tree can), and proceeds to optimize it using a technique called *path switching* - taking out "expensive" paths and replacing them with paths which have smaller cost and which satisfy the delay constraint. Both these heuristics possess high time complexity.

In our study, in addition to the delay constraint, we introduce a second userimposed limitation on the routing tree - the delay-variation constraint. We do not consider network-side optimization as our goal. However, we plan to introduce cost optimization as one of the requirements in our future work.

Chapter 3

Network Model for Multicasting

We consider the routing of multicast connections in a packet-switched communication network. The network is represented by a weighted directed graph G = (V, A), where V denotes the set of nodes, and A, the set of arcs, corresponds to the set of communication links connecting the various nodes. We will use n = |V| to refer to the number of nodes in the network. Without loss of generality, we only consider graphs with at most one arc between an ordered pair of nodes.

We define a link-delay function $\mathcal{D}: A \to \mathcal{R}^+$ which assigns a non-negative weight to each link in the network. More specifically, the value $\mathcal{D}(\ell)$ associated with link $\ell \in A$ is a measure of the total delay that data packets experience on that link, including the queueing, transmission, and propagation components. Suppose now that both links $\ell = (v, u)$ and $\ell' = (u, v)$ exist for some $v, u \in V$. Since, in practice, communication networks can be asymmetric in nature, we allow link-delay functions that assign different values to links ℓ and ℓ' , i.e., it is possible that $\mathcal{D}(\ell) \neq \mathcal{D}(\ell')$.

Under the multicast routing scenario we are considering, packets originating at some source node $s \in V$ in the network have to be delivered to a number of destinations. We will call this set $M \subseteq V - \{s\}$ of destination nodes the destination set or multicast group, and will use m = |M| to denote its size (See Figure 3.1 for an illustration). In general, several multicast sessions may proceed concurrently within the network, each characterized by a source node and a destination set.

We assume that communication in the network is connection-oriented, and that

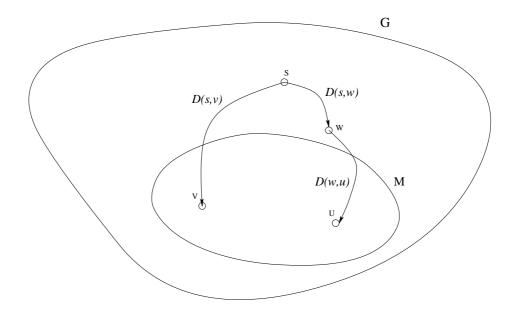


Figure 3.1: Graph G = (V, A), source node s, destination set M and the link-delay function $(\mathcal{D}(s, v), \mathcal{D}(s, w))$ and $\mathcal{D}(w, u)$

multicast connections are established by issuing a *connect request*; similarly, at the conclusion of a session a *disconnect request* is issued. In response to a connect request, and prior to any data been transferred from the source to the destinations, a connection establishment process is initiated. Central to the connection establishment is the determination of routes between the source and the destinations, over which data packets and acknowledgments will be carried for the duration of the multicast session.

Let s and M be the source and multicast group, respectively, of a certain multicast session. We assume that multicast packets for this session are routed from s to the destinations in M via the links of a multicast tree $T = (V_T, A_T)$ rooted at s. This tree is constructed during the route determination phase of the connection establishment process, based on information supplied as part of the connect request (more on this shortly). The multicast tree is a subgraph of G (i.e., $V_T \subseteq V$ and $A_T \subseteq A$) spanning s and the nodes in M (that is, $M \cup \{s\} \subseteq V_T$). In addition, V_T may contain relay nodes, that is, nodes intermediate to the path from the source to a destination. Relay nodes are not consumers of multicast packets; rather, they simply forward these packets

along the downstream links of the multicast tree, and also forward acknowledgments from the destination nodes towards the source, along the upstream links.

Let T be a multicast tree for the source-multicast group pair (s, M), and let $P_T(s, v)$ denote the unique path from source s to destination $v \in M$ in the tree T. Then, multicast packets from s to v experience a total delay of $\sum_{\ell \in P_T(s,v)} \mathcal{D}(\ell)$ along this path. We now introduce two parameters that can be used to characterize the quality of the multicast tree as perceived by the application performing the multicast. These parameters relate end-to-end delays along individual source-destination paths to the desired level of quality of service, as follows.

- Source-destination delay tolerance, Δ. Parameter Δ represents an upper bound on the acceptable end-to-end delay along any path from the source to a destination node. This parameter reflects the fact that the information carried by multicast packets becomes stale Δ time units after its transmission at the source, and as such, it is of no value to the receivers.
- Inter-destination delay variation tolerance, δ . Parameter δ is the maximum difference between the end-to-end delays along the paths from the source to any two destination nodes that can be tolerated by the application. In essence, this parameter defines a synchronization window for the various receivers.

By supplying values for parameters Δ and δ , the application in effect imposes a set of constraints on the paths of the multicast tree. The application will proceed only if a tree satisfying these constraints can be found; otherwise, the application will abort. In the following chapter we take a closer look at the problem of determining multicast trees that guarantee a desired level of performance in terms of the quality of service criteria discussed above.

Chapter 4

Multicast Trees with Bounded Delay and Delay Variation

In the previous chapter we have described the network model, which will be used in the course of this paper. We now come to the part where we formally state the problem, which is in the center of our study, and prove that it is \mathcal{NP} -complete.

4.1 Problem Description

Let Δ and δ be the delay and delay variation tolerances, respectively, as specified by a higher level application that wishes to initiate a multicast session. Our objective is to determine a multicast tree such that delays along all source-destination paths in the tree are within the two tolerances. This problem, which we will call the *Delay*and *Delay Variation-Bounded Multicast Tree (DVBMT)* problem, arises naturally as a decision problem, and can be formally expressed as follows.

Problem 4.1.1 (DVBMT) Given a network G = (V, A), a source node $s \in V$, a multicast group $M \subseteq V - \{s\}$, a link-delay function $\mathcal{D} : A \to \mathcal{R}^+$, a delay tolerance Δ , and a delay variation tolerance δ , does there exist a tree $T = (V_T, A_T)$ spanning s and the nodes in M, such that:

$$\sum_{\ell \in P_T(s,v)} \mathcal{D}(\ell) \leq \Delta \qquad \forall \ v \in M$$
(4.1)

$$\left| \sum_{\ell \in P_T(s,v)} \mathcal{D}(\ell) - \sum_{\ell \in P_T(s,u)} \mathcal{D}(\ell) \right| \leq \delta \quad \forall \ v, u \in M$$

$$(4.2)$$

We will refer to (4.1) as the source-destination delay constraint, while (4.2) will be called the inter-destination delay variation constraint. We will also say that tree T is a feasible tree for a multicast session with source s and destination set M, if and only if T satisfies both (4.1) and (4.2). Note that, in order for the multicast session to proceed, it is necessary and sufficient that a single feasible tree be constructed, as any feasible tree can meet the quality of service requirements as expressed by parameters Δ and δ .

An interesting observation regarding constraints (4.1) and (4.2) is that they represent two conflicting objectives. Indeed, the delay constraint (4.1) dictates that short paths be used. But choosing the shortest paths may lead to a violation of the delay variation constraint among nodes that are close to the source and nodes that are far away from it. Consequently, it may be necessary to select longer paths for some nodes in order to satisfy the latter constraint. Then, the problem of finding a feasible tree for DVBMT becomes one of selecting paths in a way that strikes a balance between these two objectives.

The source-destination constraint (4.1) has been previously considered in the context of designing constrained Steiner trees for real-time, interactive applications [4, 5]. To the best of our knowledge, however, our work is the first to explicitly consider the inter-destination delay variation constraint (4.2) in the construction of multicast trees. This should not, however, be taken to mean that the importance of providing guarantees on the maximum value of the end-to-end delay variation has not been recognized. In fact, as part of a recent study [6, 7] to evaluate the relative performance of a large number of multicast algorithms and their suitability to high-speed real-time applications, the following quantity was measured and used as a criterion in the evaluation:

$$\delta_T = \max_{u,v \in M} \left\{ \left| \sum_{\ell \in P_T(s,u)} \mathcal{D}(\ell) - \sum_{\ell \in P_T(s,v)} \mathcal{D}(\ell) \right| \right\}$$
(4.3)

Quantity δ_T is the maximum inter-destination delay variation in tree T, and, given a value for δ , it can be used to determine whether tree T can meet the quality of

service requirements of the application. According to the study, none of the existing algorithms provides good performance in terms of δ_T ; this is not surprising, as none of the algorithms considered in [6, 7] takes the delay variation constraint (4.2) into account. Our work addresses the problem of designing multicast algorithms that overcome this inefficiency.

4.2 Proof that DVBMT is \mathcal{NP} -complete

Before proceeding, we would like to resolve the open question regarding the existence of efficient algorithms for DVBMT. Unfortunately, the following theorem establishes that DVBMT is \mathcal{NP} -complete, implying that a polynomial-time algorithm that determines a feasible tree for any arbitrary instance of this problem is unlikely to be found. The proof uses a polynomial transformation from PARTITION problem, which is known to be \mathcal{NP} -complete. PARTITION problem is restated here for the sake of completeness (see [12] for more).

Problem 4.2.1 (PARTITION) Given a set of k elements $S = \{1, 2, ..., k\}$ with a_i the weight of element i, and $A = \sum_{i=1}^k a_i$, does there exist a partition of S into two sets, S_1 and S_2 , such that $\sum_{i \in S_1} a_i = \sum_{j \in S_2} a_j = \frac{A}{2}$? (The a_i 's may be assumed integer.)

We are now ready to prove that DVBMT is indeed \mathcal{NP} -complete.

Theorem 4.2.1 DVBMT is \mathcal{NP} -complete whenever the size of the multicast group $|M| \geq 2$.

Proof. It is easy to see that DVBMT is in the class \mathcal{NP} , since a nondeterministic algorithm need only guess a tree spanning s and the nodes in the destination set M, and verify in polynomial time that the tree is a feasible one (i.e., that it satisfies both (4.1) and (4.2)).

We now transform PARTITION to DVBMT; note that it is sufficient to find a transformation for the case |M| = 2. Let $S = \{1, 2, ..., k\}$ be the set of elements of weights $a_i, i = 1, ..., k$, making up an arbitrary instance of PARTITION, and let

 $A = \sum_{i=1}^{k} a_i$. We construct an instance of DVBMT as follows (see Figure 4.1). The network G = (V, A) has n = k + 3 nodes, with $V = \{s, v, u, r_1, r_2, \dots, r_k\}$, where s is the source node and $M = \{v, u\}$ is the destination set. The set A of links is:

$$A = \{(s, v), (s, r_1), \dots, (s, r_k), (r_1, u), \dots, (r_k, u), (r_1, r_2), \dots, (r_1, r_k), (r_2, r_1), (r_2, r_3), \dots, (r_2, r_k), \dots, (r_k, r_1), \dots, (r_k, r_{k-1})\}$$

$$(4.4)$$

In other words, there is a directed link from s to v, one link from s to each node r_i , one link from each node r_i to u, and one link from r_i to $r_j, i, j = 1, ..., k, i \neq j$ (i.e., the subgraph of G containing only nodes $r_i, i = 1, ..., k$, is a complete graph on these nodes). As we can see, there is only one path from s to destination node v consisting of the single link (s, v); however, a path from s to the other destination u may contain any number of the nodes $r_i, i = 1, ..., k$, and in any order (refer also to Figure 4.1).

We now define the link-delay function \mathcal{D} for this instance of DVBMT as:

$$\mathcal{D}(\ell) = \begin{cases} \frac{A}{2}, & \text{if } \ell = (s, v) \\ 0, & \text{if } \ell = (x, u), x \in V \\ a_i, & \text{if } \ell = (x, r_i), x \in V \end{cases}$$

$$(4.5)$$

As a result of this definition, if the path from s to u passes through node r_i for some i, then a delay equal to a_i is incurred along the link that leads to r_i . Finally, the delay and delay variation tolerances are $\Delta = \frac{A}{2}$, and $\delta = 0$, respectively.

It is obvious that this transformation can be performed in polynomial time. We now show that a feasible tree exists for the above instance of DVBMT if and only if set S has a partition. If S has a partition S_1 , S_2 , then $S_1 = \{a_{\pi_1}, \ldots, a_{\pi_l}\}$ for some l < k. The tree consisting of path (s, v) and path $(s, r_{\pi_1}), (r_{\pi_1}, r_{\pi_2}), \ldots, (r_{\pi_{l-1}}, r_{\pi_l}), (r_{\pi_l}, u)$, is then a feasible tree for DVBMT, as the delay along both paths is equal to $\frac{A}{2}$.

Conversely, let T be a feasible tree for DVBMT. Then T must include the path (s, v) of delay $\frac{A}{2}$, as this is the only path from the source to v. Let

$$(s, r_{\pi_1}), (r_{\pi_1}, r_{\pi_2}), \dots, (r_{\pi_{l-1}}, r_{\pi_l}), (r_{\pi_l}, u)$$
 (4.6)

be the path from s to u on tree T. Since T is a feasible tree and $\delta = 0$, the delay along the latter path is equal to $\frac{A}{2}$, and l < k (for if l = k, the path from s to

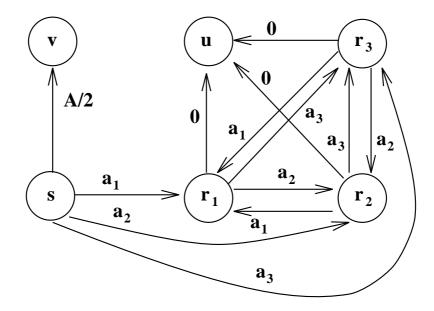


Figure 4.1: Instance of DVBMT corresponding to an instance of PARTITION with $S = \{1, 2, 3\}$

u would include all $r_i, i = 1, \ldots, k$, and the delay along the path would equal A, contradicting our hypothesis that T is a feasible tree). Then, $\sum_{i=1}^{l} a_{\pi_i} = \frac{A}{2}$, implying that $S_1 = \{a_{\pi_1}, \ldots, a_{\pi_l}\}, S_2 = S - S_1 \neq \emptyset$, is a partition of S.

The next chapter introduces a heuristic approach to determining feasible trees for any arbitrary instance of DVBMT.

Chapter 5

Multicast Tree Algorithms for DVBMT

In the previous chapter we have formulated the problem, which we will will attempt to solve. As mentioned before, due to the fact that the problem in question is \mathcal{NP} -complete, no polynomial algorithmic solution for it is likely to exist. Instead, we use a heuristical approach. In this chapter we will describe several heuristics which attempt to cope with the DVBMT problem. We will also prove their correctness and analyze their time complexity.

5.1 Algorithm Description

Consider an application running at node s, and suppose that the application issues a request for establishing a multicast connection with destination set M. Along with the request, the application also supplies values for the path delay tolerance Δ , and inter-destination delay variation tolerance δ ; these values reflect certain quality of service requirements which the network must guarantee in order for the multicast session to proceed. As part of the connection establishment process, upon receiving the request, a multicast tree satisfying constraints (4.1) and (4.2) needs to be determined. In this chapter we present algorithms that can be used to construct such a tree. Our algorithms operate under the assumption that complete information regarding the

network topology is stored locally at node s, making it possible to determine the multicast tree at the source itself. This information may be collected and updated using one of several existing topology-broadcast algorithms [13].

The sequence of actions taken by node s during the course of constructing a multicast tree is illustrated in the flowchart of Figure 5.1, where we have assumed that the values of the delay and delay variation tolerances Δ and δ , respectively, provided by the application are negotiable ¹. As a first step, the tree of shortest paths from s to all nodes in M is constructed; this can be achieved using Dijkstra's algorithm [14], which, for dense graphs with n nodes, takes time $\mathcal{O}(n^2)$. Let T_0 be this tree of shortest paths. If T_0 does not satisfy the path delay constraint (4.1) no tree may satisfy it, implying that the delay tolerance Δ supplied by the application is too tight; negotiation may then be necessary to determine a looser value of Δ .

Suppose now that the (original or negotiated) value of Δ is such that the delay requirement (4.1) is met for tree T_0 . If T_0 also meets the delay variation requirement (4.2) then T_0 is a feasible tree for this instance of the DVBMT problem, and the multicast session may take place over the tree of shortest paths. As a result, the route determination phase completes successfully, and the connection establishment process may then proceed to a subsequent phase (such as bandwidth reservation along the paths of the tree, etc.).

On the other hand, it is possible that tree T_0 fail to satisfy constraint (4.2). In that case, our approach is to have the source execute a search algorithm in an attempt to construct a new tree in which the delays along all source-destination paths satisfy both (4.1) and (4.2). Based on the results of the previous chapter, however, an algorithm that efficiently solves any arbitrary instance of DVBMT may not exist, meaning that the search algorithm has to employ a heuristic approach. Nevertheless, suppose that a heuristic algorithm is available, and that it returns a tree which constitutes a solution

¹Using values for parameters Δ and δ other than the ones recommended by the application may result in a degradation of the quality of service as perceived by some or all of the receivers in the multicast group. However, as long as the negotiated values do not differ significantly from the original ones (in which case a basic grade of service can be guaranteed), it may be possible for the multicast session to proceed. It is also conceivable that incentives to join the multicast session be offered to receivers expected to experience a lower quality of service as a result of the network's inability to guarantee the initial requirements.

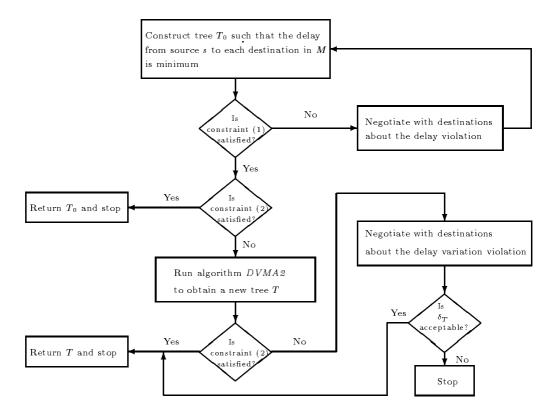


Figure 5.1: Flowchart of our approach to obtaining a multicast tree for the *DVBMT* problem

to the given instance of the DVBMT problem; then a tree for the multicast session has been found.

However, a heuristic algorithm may fail to discover a feasible tree, either because no such tree exists or because of the ineffectiveness of the search strategy employed. Other than abandoning the connection altogether, the only course of action available to the application at that point would be to initiate another round of negotiations in hope of determining a new value for the delay tolerance δ , one that would be acceptable to all parties involved in the multicast session. If such a value can be agreed upon the source would go through another iteration in the flowchart of Figure 5.1, otherwise the multicast session would have to be abandoned.

An alternative that would result in a considerable speed-up of the negotiation process would be to design the search algorithm so that it always returns, among the trees considered, the one with the smallest value of δ_T in (4.3). Indeed, regardless of whether a solution to the given instance of DVBMT problem exists or not, the tree corresponding to the smallest value of δ_T is the best tree that can be obtained with the search algorithm at hand. If this tree is available at the termination of the algorithm, all that has to be determined during the negotiation process is whether an acceptable level of quality of service can be sustained for the given value of δ_T and there is no need to repeat the route determination process; this is shown in Figure 5.1.

The following section presents DVMA, a new multicast tree heuristic designed to solve the DVBMT problem. Following that, we show how the basic idea behind DVMA can be used to develop a solution to the corresponding dynamic problem, i.e., the problem of updating the multicast tree in response to receiver requests for joining or leaving an ongoing multicast session.

5.2 Delay Variation Multicast Algorithm (DVMA)

Let T_0 be the tree of shortest paths from source s to the nodes in the destination set M for the multicast connection under consideration. Let us also assume that T_0 meets the delay requirement (4.1), but that it does not meet the delay variation requirement (4.2). The Delay Variation Multicast Algorithm (DVMA), described in detail in Figure 5.6, can then be used to search through the space of candidate trees (i.e., trees spanning s and the nodes in M) for a feasible solution to the DVBMT problem. DVMA either returns a feasible tree, or, having failed to discover such a tree, it returns one which (a) satisfies the delay constraint (4.1) and (b) has the least value of δ_T among the trees considered by the algorithm. The basic idea behind the operation of DVMA is now described.

Let M be the destination set, and assume for the moment that a feasible tree $T = (V_T, A_T)$ spanning s and a subset of M has already been determined. Let $U = M - (M \cap V_T)$ be the set of destination nodes not in the tree T; in other words, no paths from the source s to the nodes in U have been determined yet. DVMA operates by appropriately augmenting tree T to eventually include all nodes in U; to

this end, it repeats the following three steps as long as $U \neq \phi$:

- 1. Select a destination node $u \in U$.
- 2. Find a "good" path from a node $v \in V_T$ to u that uses no nodes in V_T other than v, and no links in A_T ; note that v could be the source node s.
- Construct a new tree T' by including all nodes and links of this path to the initial tree T, and update U to exclude u and any other destination nodes along this path.

The second step is crucial to the operation of DVMA, and warrants further explanation. Recall that our objective is to construct a feasible tree that includes all nodes in M, therefore a "good" path in Step 2 above is one which, if connected to T in Step 3, the resulting tree T' would be a feasible tree for the subset of the set of destination nodes it contains. In order to find such a path, we construct the l shortest paths from a node v of T to u. The graph used to find these paths is created by excluding all nodes of T other than v, and all links of T from the original graph G. The exclusion of these nodes and links from G guarantees that connecting any of the l paths so constructed to T will not create a cycle.

It is possible, though, that none of the l paths from v to u will yield a feasible tree. For this reason, we repeat the process for all nodes $v \in V_T$ in an attempt to find a "good" path between any $v \in V_T$ and u. Even so, the algorithm may still not be able to find such a path; for instance, a feasible tree for this destination set may not exist in the first place. Recall, however, that we would like the algorithm to return the best tree (in terms of maximum inter-destination delay variation) it can find. We now modify our definition of a "good" path so that, if a path yielding a feasible tree T' can not be found, a "good" path is one which

- (a) the total delay from s to u (i.e., the delay from s to v in T, plus the delay from v to u over the path) is at most Δ , and
- (b) the tree T' created by connecting this path to T has the least value of maximum delay variation among the trees constructed by connecting the other paths to T.

In essence, the purpose of the greedy rule (b) above is to prune the search space, i.e., to prevent certain candidate trees from receiving further consideration.

The only question that remains to be answered then, is how an initial tree $T_{initial}$ is constructed. To answer this question consider T_0 , the tree of shortest paths, which, by hypothesis, does not satisfy the delay variation constraint (4.2). Let w be the destination node with the longest path in this tree. Since it is not possible to make the delay from s to w any smaller than the delay incurred over the path from s to w in T_0 , the only alternative to constructing a feasible tree is to find longer paths from s to some or all of the other destination nodes. Hence, our approach is to start with an initial tree T consisting only of the shortest path from s to w, and repeat the three steps described above to create a feasible tree that will include all other destination nodes.

To complete the description of the search strategy employed by DVMA, note that it is possible that no feasible tree for the given destination set includes the shortest path from s to w. However, if a feasible tree exists, it will contain some path from s to w. Therefore, if the process of constructing a feasible tree starting from the shortest path from s to w fails, the second shortest path from s to w is considered as the initial tree and the process is repeated. Figure 5.2 shows the initial tree $T_{initial}$ which consists of one path, from s to w - the farthest destination in T_0 . The set U of yet unconnected destinations consists of all members of the original set M with the exception of node w and destination nodes that "accidentally" ended up on the path from s to w. In other words $U = M - V_{T_{initial}}$. Our search for a feasible tree terminates when one is found, or when trees based on the first k shortest paths from s to w have been constructed, whichever occurs first. In the latter case, the algorithm will return the tree with the smallest value of δ_T in (4.3).

The three main steps performed by the algorithm are reflected in Figures 5.3, 5.4 and 5.5. In Figure 5.3 we see the partially built tree T and the node $u \in U$ - not yet connected to T. Figure 5.4 demonstrates the selection of a "good" path from node v, already in the tree T, and node u. The algorithm has a choice of at most l paths to make the selection from. The new tree T', which includes u, is shown in Figure 5.5. The details of the algorithm (DVMA) can be found in Figure 5.6.

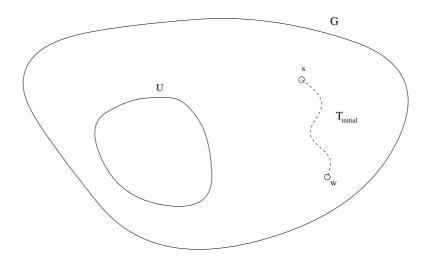


Figure 5.2: The initial tree $T_{initial}$ consisting of a path from s to w and the set $U = M - V_{T_{initial}}$ of unconnected destinations

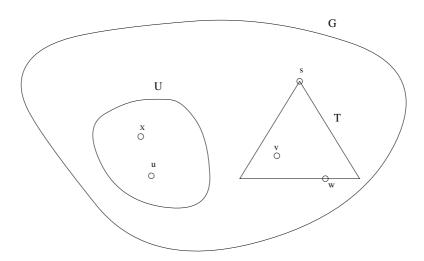


Figure 5.3: The partially-built tree T, graph G and node $u \in U$, $U = M - (M \cap V_T)$ (Step 1)

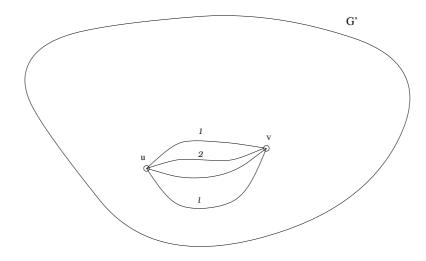


Figure 5.4: The path to $v \to u$ is being selected among the l shortest paths between v and u in G'(Step 2)

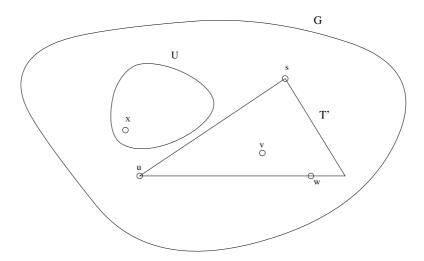


Figure 5.5: The new tree T' after the addition of node u, new set U of still unconnected destinations (Step 3)

Delay Variation Multicast Algorithm (DVMA)

The algorithm is executed if T_0 , the tree of shortest paths, satisfies constraint (4.1) but does not satisfy constraint (4.2). We let $w \in M$ be a node such that $\sum_{\ell \in P_{T_0}(s,w)} \mathcal{D}(\ell) = \max_{v \in M} \left\{ \sum_{\ell \in P_{T_0}(s,v)} \mathcal{D}(\ell) \right\}$.

```
begin
1.
2.
        Let T = T_0
                           //T is the tree returned by the algorithm
3.
        Find the first k shortest paths from s to w in the original graph G = (V, A),
        such that the delay from s to w over these paths is less than \Delta;
        label these paths p_1, \ldots, p_k in increasing order of delay
        for i = 1 to k do
                                // construct a multicast tree T_i for each path p_i
4.
           Initialize T_i = (V_i, A_i) to include all the nodes and links of path p_i;
5.
           obviously, s, w \in V_i
6.
           Let U = M - (M \cap V_i) be the set of destinations not yet connected
           to the tree T_i
           while U \neq \phi do
7.
              Pick any node u \in U
                                            // will connect u to the tree T_i
8.
                                               // find a path from v to u
9.
              for each node v \in V_i do
                  Construct a new graph G' starting with the initial graph G and excluding
10.
                  all nodes in V_i - \{v\}, all links in A_i; also exclude all nodes in U - u and
                  all arcs associated with them
11.
                  Find the first l shortest paths from v to u in the new graph G'
                  Of these l paths choose the best one (as described in Section 5.1)
12.
                  and call it q_v
13.
              end of for each node v \in V_i loop
14.
              Select the best path q among all paths q_v, v \in V_i (as in Step 12 above)
15.
              Update T_i = (V_i, A_i) to include all nodes and links in path q
              Update U = U - u / / \text{ node } u \text{ has now been connected to } T_i
16.
17.
           end of while loop
                                    // construction of tree T_i has been completed
18.
           If tree T_i satisfies constraint (4.2) return T_i and stop
19.
           Let T be the tree among T and T_i with the smallest value of \delta_T in (4.3)
        end of for i loop
20.
21. return T
                     // no tree satisfied the inter-destination delay variation constraint
22. end of the algorithm
```

Figure 5.6: Heuristic algorithm for the *DVBMT* problem

5.3 Correctness and Complexity Analysis of DVMA

The correctness of DVMA is provided by the following lemma. Note, however, that although the algorithm returns the best tree, in terms of maximum delay variation, that it can find, because of its heuristic nature it may fail to discover a feasible tree for the given value of δ even if one exists.

Lemma 5.3.1 (Correctness of DVMA) Algorithm DVMA returns a tree T spanning s and all nodes $v \in M$. The tree T satisfies constraint (4.1), and either satisfies constraint (4.2), or is the one with the smallest value of δ_T in (4.3) among the trees considered by the algorithm.

Proof. We will first show that the algorithm returns a tree T spanning s and the nodes in M. If DVMA returns T_0 , there is nothing to prove. Otherwise, T is one of the T_i 's constructed during one iteration of the loop that starts at line 4. T is initialized to some path p_i at line 5; clearly, at this point T is a tree containing the source s and at least one more destination $w \in M$. New nodes and links are added to T in line 15, where a new path q from a node in $v \in V_T$ to a node $u \in M, u \notin V_T$ is incorporated. The resulting new graph is a tree as path q cannot contain any nodes or links of T other than v itself. Indeed, all other nodes and links of T were removed at line 10, before path q was determined. The new tree T has at least one more node, $u \in M$; since s was in the tree initially, no nodes are ever removed from T, and paths are added to it until all destinations in M are in T, our first claim is true.

That the delay constraint (4.1) is satisfied by the final tree T is now is easy to see. If $T = T_0$ this is true by hypothesis; if $T \neq T_0$ this is also true as no path is ever added to any tree T_i unless the delay constraint is satisfied (refer to lines 3 and 12). Finally, if the algorithm terminates at line 18, the tree returned is a feasible one; otherwise, line 19 guarantees that the tree returned is the one with the smallest value of δ_T among the ones constructed during the execution of the algorithm.

The next lemma determines the running time complexity of DVMA.

Lemma 5.3.2 The worst-case complexity of DVMA is $O(klmn^4)$, where k is the number of paths generated at line 3 of Figure 5.6, l the number of paths generated

at line 11, m = |M| is the number of destinations in the multicast group M, and n = |V| is the number of nodes in the network.

Proof. The running time of DVMA is dominated by the iteration between lines 4 and 20; this outer loop is executed at most k times. During one iteration of the outer loop, the "while" loop at line 7 is executed at most m-1 times. Let t_j be the number of nodes in the tree during the j-th iteration of the "while" loop. Then, the innermost loop starting at line 9 will iterate t_j times; inside this loop the complexity is determined by the l-shortest path algorithm at line 11, which takes time $\mathcal{O}(lN^3)$ [15] for a graph with N nodes. Graph G' has $n-t_j+1$ nodes throughout the innermost loop; the latter then takes time proportional to $lt_j(n-t_j+1)^3$. For a worst case analysis, we let t_j , for all iterations j, take the value that maximizes the quantity $t_j(x-t_j)^3$, where x=n+1. It is straightforward to show that for this value of t_j the complexity of the innermost loop becomes $\mathcal{O}(ln^4)$. After accounting for the "while" and outer loops, we conclude that the overall complexity of the algorithm is, in the worst case, $\mathcal{O}(klmn^4)$.

Regarding parameters k and l, note that the maximum value they can take is, in the worst case, equal to the maximum number of paths of delay at most Δ between any two nodes in the network. If Δ is not very loose, we expect the maximum value of both k and l to be a small constant. The actual values of k and l were left unspecified in the description of the algorithm, as in any particular implementation they will be determined by the desired compromise between the quality of the final solution of the algorithm and its speed.

5.4 A Family of Algorithms for DVBMT

The algorithm for minimizing the delay variation of the multicast tree in fact comes in two "flavors": DVMA, which was described above and another one which we called DVMA2. DVMA2 differs from DVMA only in two steps: step 10 and step 16 in Figure 5.6. Graph G' is formed by excluding only nodes in $V_i - v$ and all links in A_i (not excluding all nodes in U - u). In step 16, the set U of yet unconnected

destinations has to be updated more carefully than just by removing u from it: since we didn't remove other destinations from the original graph, some of them may have inadvertently ended up on the new path, linking u to the tree. Therefore, Step 16 should read: $U = M - (M \cap V_i)$. Otherwise DVMA2 performs exactly as its "sibling".

In addition, we have attempted to improve the performance of DVMA by making it more greedy. This new heuristic instead of simply adding destinations to the tree in an arbitrary order, as DVMA does, in one iteration finds "good" paths for all so far unconnected destinations in U, and then connects the destination whose path is the "best" among all "good" paths (using the same criterion as in the selection of a "good" path for a particular destination). This added complexity costs an extra m factor in the time complexity estimate, making it $O(klm^2n^4)$.

This heuristic also comes in two "flavors", differences between which are the same as the differences between DVMA and DVMA2. Surprisingly, the added complexity does not pay off in the performance. Comparisons of the average performance of these new heuristics (we will call them DVMAv2 and DVMA2v2) and the DVMA family showed no significant improvement in the quality of the routing tree (See Chapter 7 for numeric results).

5.5 Summary

We have just described several heuristics which attempt to solve the *DVBMT* problem. We saw that the same idea can be used to produce a variety of heuristics. We will see the performance comparison of these heuristics in Chapter 7. In the next chapter we will show that the same approach, that we used in these static heuristics, can be used to produce a heuristic that will allow nodes to join and leave the multicast group dynamically, after the connection has already been established.

Chapter 6

Dynamic Reorganization of the Multicast Tree

In Chapter 5 we presented DVMA, an algorithm that can be used during connection establishment to construct a feasible tree for a given destination set. For certain applications, however, it is conceivable that nodes join or leave the initial multicast group during the lifetime of the multicast connection 1 . More specifically, we assume that nodes currently in the multicast group may leave the group after issuing a leave request LEAVE(u). Similarly, nodes that wish to join an ongoing multicast session must first issue a join request JOIN(u). Under such a scenario, it is necessary to dynamically update the multicast tree in response to changes in multicast group membership, in order to ensure that constraints (4.1) and (4.2) are always satisfied for the current destination set.

Let T be the multicast tree of an ongoing multicast session with destination set M, and suppose that as a result of a join or leave request the new destination set is M'. One possible approach to this dynamic version of the DVBMT problem 2 would be to run DVMA anew to obtain a feasible tree T' for set M', and, following a transition period, use the new tree for routing subsequent packets of this session. Note that there

¹For instance, teleconferencing is an application where the ability to dynamically add or drop parties is highly desirable.

 $^{^{2}}$ As opposed to the static version we have considered so far, whereby all the destinations in the multicast group are known in advance.

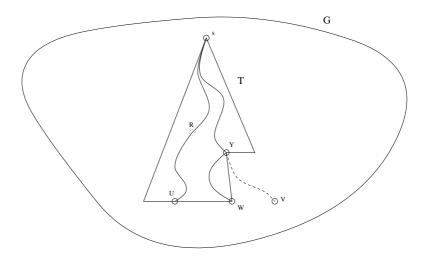


Figure 6.1: Vertices r and v have left the multicast group

is a certain overhead associated with this approach, including the computational cost of running DVMA, and the cost of the network resources involved in the transition from T to T' (i.e., the cost of tearing down old paths and establishing new ones). Since the new tree T' can be significantly different than T, this overhead can be very high. Furthermore, such a radical approach may cause receivers totally unrelated to the destination nodes added or deleted to experience disruption in service. All these drawbacks make the strategy just described inappropriate for real-time environments and applications where frequent changes in the destination set are anticipated.

We now adopt a different strategy, one that attempts to minimize both the cost incurred during the transition period, and the disruption caused to the receivers. More specifically, the multicast tree is never modified unless it is absolutely necessary to do so. Even then, the new tree is not computed from scratch, rather, a feasible tree for the new multicast group is constructed by making incremental and localized changes to the old tree. We now describe in detail how the join and leave requests are handled under our approach.

6.1 Handling leave Requests

Let us first consider leave requests, and assume that node $v \in M$ decides to end its participation in the multicast session. If v is not a leaf node in the current multicast tree T no action needs to be taken. The new tree T' can be the same as T, with the only difference being that node v will stop forwarding the multicast packets to its local user. If, however, v is a leaf node of T, and in order to avoid wasting bandwidth, tree T has to be pruned to exclude v and, possibly, relay nodes and links used in T solely for forwarding packets to v. The new tree T' is essentially the same as T except in parts of the path from the source to v. Both these cases are depicted in Figure 6.1. Notice that since node v was a leaf, the tree was pruned up to the nearest junction point at node v. Node v on the other hand, has simply stopped forwarding multicast packets to its local client, without causing any changes in the tree structure. We conclude that leave requests are easy to handle, and no destination node (other than v, the node leaving the session) needs to notice any difference in terms of the multicast session.

6.2 Handling join Requests

Let us now turn our attention to the actions taken whenever a node $u \notin M$ announces its intention to join the multicast group. We distinguish three cases, as follows.

• u ∉ V_T, i.e., the new node is not part of the multicast tree T. Our approach is to augment T to include a path from a node V ∈ V_T to the new node u. This can be easily accomplished by letting T_i = T and U = {u} at lines 5 and 6, respectively, of DVMA (see Figure 5.6) and executing the code between lines 7 and 17 to search for a path that would result in a feasible tree for the set M∪ {u}. Hence, the transition phase involves only the establishment of a new path and does not affect any of the paths from the source to nodes already in the multicast group, allowing the connection to proceed smoothly and without

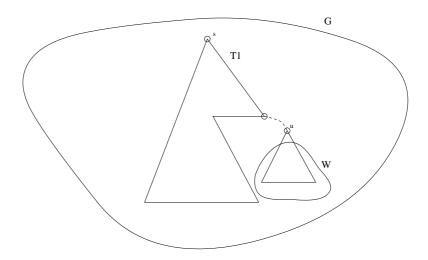


Figure 6.2: Paths to destinations which are in W will change

any disruption ³. The time complexity of this part, determined similarly to Lemma 5.1.2 is $O(\ln^4)$.

- $u \in V_T$, i.e., u is a relay node of T, and the path from the source node s to u is such that the delay variation constraint (4.2) is satisfied for the new multicast group $M' = M \cup \{u\}^4$. Tree T is then a feasible tree for the new set M', and can be used without any change other than having node u now forward multicast packets to its user, in addition to forwarding them to the downstream nodes.
- u ∈ V_T, but the path from s to u is such that the delay variation constraint (4.2) is not satisfied for the new set M ∪ {u}. Consequently, a longer path from s to u has to be found. Let W ⊆ M be the destination nodes in M that are downstream of u (i.e., those destination nodes in the subtree of T rooted at u). Finding a new path from s to u will definitely affect the paths to these nodes, however, the paths to nodes in M − W need not be affected. Let T₁ be the tree T after excluding its subtree rooted at u (this is depicted in Figure 6.2). Our

³If executing this piece of code fails to discover such a path, there are two possible courses of action: (a) run DVMA from scratch for the new multicast group, or (b) deny node u its participation in the multicast session; which course of action to be taken may depend on several factors, such as the nature of the application, the cost of rerouting the connection, etc.

⁴Note that the path from s to u will necessarily satisfy the delay constraint (4.1), as u cannot be a leaf node of T.

approach then is to let $T_i = T_1$ and $U = W \cup \{u\}$ at lines 5 and 6, respectively, of DVMA in Figure 5.6. We then execute the code between lines 7 and 17 to connect the destination nodes in U into tree T_1 . As a result, packets will be routed from s to the nodes in W over new paths in the final tree T', but none of the paths to nodes in M - W will change. The worst case time complexity of this part is $O(lmn^4)$.

As a final observation, besides being minimally disruptive, this approach has the additional advantage that the algorithm used during set-up time to construct an initial tree for the multicast connection, can also be used to reorganize the tree during the lifetime of the session.

Chapter 7

Numerical Results

7.1 Performance of DVMA Family of Algorithms

We now consider five different algorithms that can be used to construct multicast trees for a given source and destination set, and compare their performance in terms of the maximum delay variation δ_T among the source-destination paths in the final tree T, as defined in (4.3). The five algorithms studied are:

- 1. DVMA, the algorithm described in Figure 5.6. We run this algorithm with $\Delta = 0.05s$ and $\delta = 0$. This value of δ was used in order to force the algorithm to go through all possible iterations of the outer for loop and return the tree with the smallest value of δ_T it can find. This last value represents the tightest delay variation tolerance for which a tree can be found using DVMA.
- 2. DVMA2, an algorithm very similar to DVMA (see Chapter 5); The values of parameters Δ and δ used are the same as for DVMA above.
- 3. Dijkstra's algorithm [14] which constructs the shortest paths from the source to any node in the network. The resulting tree is pruned to exclude paths that do not terminate at a destination node, and will be referred to as *shortest path tree* (SPT).

- 4. Prim's algorithm [16] which constructs a tree of minimum weight spanning all nodes in the network; in our case, the weight of each link is the delay incurred along the link. This minimum spanning tree (MST) is also pruned, as discussed above.
- 5. The tradeoff algorithm between the minimum spanning tree heuristic ¹ for the Steiner tree problem [17] and SPT, as presented in [18]. The algorithm operates as follows. First a tree is constructed using the heuristic ²; then the destinations with the largest difference between the delay of their shortest path and the delay of their path in this tree are found, and the tree paths are replaced by the shortest paths from the source to those destinations. This algorithm (which we will call TDF) is studied because it was conjectured in [6] that it may yield good results in terms of maximum inter-destination delay variation.

We are interested in studying the average case behavior of the five algorithms. To this end, we have generated random graphs for a wide range of values for the total number n of nodes, the average degree of each node, and the number m of destinations in the multicast group as a percentage of n. These graphs were constructed to resemble real-world networks using the method described in [19]; the nodes of the graphs were placed in a grid of dimensions $4900 \times 4900 \text{ Km}^3$ (roughly the size of the continental United States), and the delay along each link was set to the propagation delay of light along that link ⁴. The random networks were then fed as input to each algorithm, and the maximum inter-destination delay variation δ_T of the final tree was computed as in expression (4.3). All figures in this chapter plot δ_T against the number of nodes n in the network, for the five algorithms discussed above. Each point plotted represents the average over three hundred different graphs for the stated values of n, m, and the average degree of each node. In addition to the mean estimates for the delay variation, we present the 95% confidence intervals of these estimates.

¹Not to be confused with Prim's algorithm for constructing an MST.

²The delay along each link is used as the cost of the link.

³Hence, the value of $\Delta = 0.05s$ corresponds to the time it would take light to travel, at a speed of $2 \times 10^8 m/s$, over a distance approximately equal to 1.5 times the diameter of this square.

⁴Note that, unless the links operate at or close to their capacity, propagation delays are expected to dominate over queueing and transmission delays in a high-speed environment.

We first consider networks with average nodal degree equal to 2.5. The results shown in Figures 7.1 - 7.4 correspond to multicast groups of sizes equal to 5\%, 10\%, 15%, and 20% of the total number of nodes in the network, respectively ⁵. From these figures we can make the following observations regarding the relative performance of the five algorithms. The trees constructed by DVMA and DVMA2 have a maximum delay variation that is always smaller than that of the SPT, TDF, and MST trees. Furthermore, DVMA outperforms DVMA2 in most cases shown. A plausible explanation for this fact would be that the small change in the way graph G' is constructed at line 10 of the algorithms allows DVMA to explore a larger number of candidate paths and discover a better overall solution. On the other hand, the MST is by far the worst tree in terms of δ_T ; this is expected as Prim's algorithm minimizes the total weight of the tree, without paying any attention to the individual source-destination paths. The tree of shortest paths SPT results in values of δ_T that are between those of the MST and those of the DVMA and DVMA2 algorithms. Note that, in this tree, the value of δ_T is determined by the difference between the delays along the paths to the destinations that are closest and farthest away from the source. Finally, the tradeoff algorithm TDF constructs trees with maximum delay variation larger than that of SPT, a result that is in contrast to the expectations expressed in [6].

Let us now turn our attention to how the size m of the multicast group relative to the size n of the network affects the performance of the algorithms. From Figure 7.1 where m is a small percentage (5%) of n, we can see that the DVMA and DVMA2 trees represent an improvement of roughly an order of magnitude over the SPT and TDF trees. As m increases as a percentage of n, the magnitude of improvement decreases, as seen in Figures 7.2, 7.3, and 7.4 which show results for m equal to 10%, 15%, and 20% of n, respectively (however, even in Figure 7.4 with m = 0.2n, DVMA constructs trees which, on the average, have a value of δ_T at least 16% lower than that of SPT). This behavior can be explained by noting that the smaller the size of the multicast group, the easier it is for DVMA and DVMA2 to find alternative (i.e., longer) paths for the nodes physically closer to the source. At the other extreme,

⁵There is no point plotted in Figure 7.1 for n = 20 as in this case a multicast group of size 0.05n contains only one node, and the quantity δ_T is not meaningful.

when m = n (a broadcast scenario) it is easy to see that no tree can do much better than SPT in terms of the maximum delay variation. In fact, as the trend in Figures 7.1 - 7.4 suggests, when the size of the multicast group is larger than 25-30% of n, it makes sense to simply use the SPT, as running DVMA or DVMA2 would not yield a significant improvement in terms of δ_T . In a typical multicast scenario however, the size of the destination set for any single session would be small compared to the total number of nodes (especially for large networks); it is in these situations that the algorithms presented here would really make a difference in terms of the maximum delay variation of the final tree.

Finally, Figures 7.5 and 7.6 investigate how the value of δ_T for the various trees changes as a function of the average nodal degree. By comparing these figures to Figure 7.1 which presents plots for the same size of destination set (m = 0.05n), we see that for the DVMA and DVMA2 trees δ_T decreases dramatically as the average nodal degree increases from 1.5 (Figure 7.5) to 2.5 (Figure 7.1) and then to 4 (Figure 7.6). This is a result of the fact that a higher nodal degree translates into a larger number of paths between any two nodes, and a larger number of trees for our algorithms to choose from. Another important observation is that for an average nodal degree equal to 4, both of our algorithms are able to construct trees with $\delta \approx 0$, independently of the number of nodes in the network. As such, these trees would be able to meet the delay variation requirements of even the most demanding applications. The behavior of the other algorithms is not significantly affected by the nodal degree, as none of these attempt to optimize in terms of δ_T . In SPT, for instance, δ_T is determined by the relative distance of the various destinations from the source, which is almost independent of the nodal degree.

Overall, the results presented in this chapter suggest that *DVMA* and *DVMA2* achieve their best performance under conditions that are typical of multicast applications running in high speed networks, namely, when (a) the size of the multicast group is relatively small compared to the total number of nodes in the network, and/or (b) the number of incoming/outgoing links at each node is relatively large.

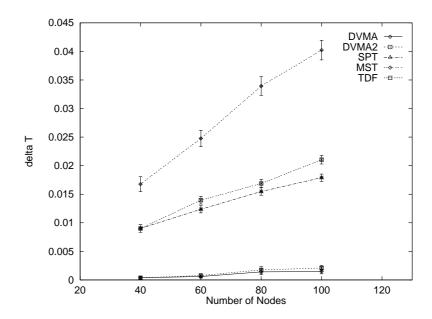


Figure 7.1: Algorithm comparison for networks with average node degree equal to 2.5, and multicast group size equal to 5% of the number of nodes

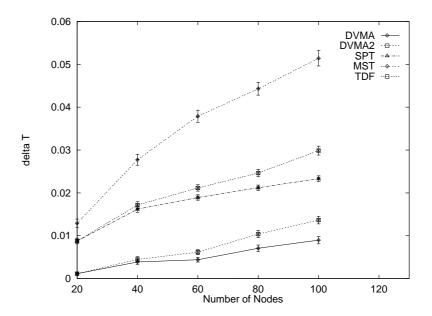


Figure 7.2: Algorithm comparison for networks with average node degree equal to 2.5, and multicast group size equal to 10% of the number of nodes

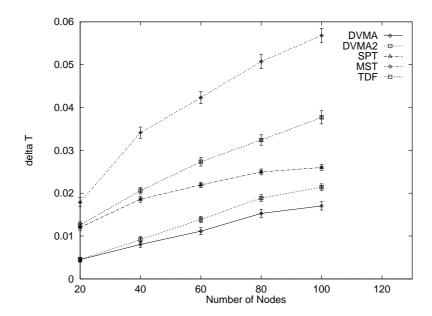


Figure 7.3: Algorithm comparison for networks with average node degree equal to 2.5, and multicast group size equal to 15% of the number of nodes

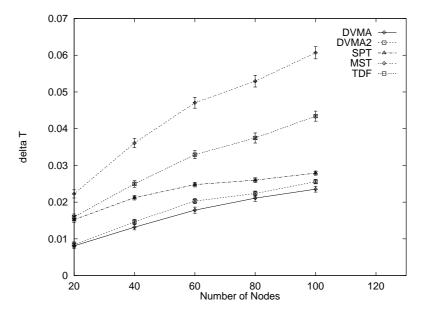


Figure 7.4: Algorithm comparison for networks with average node degree equal to 2.5, and multicast group size equal to 20% of the number of nodes

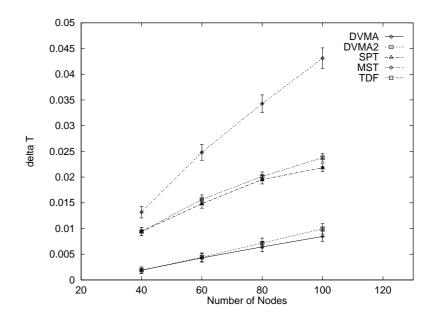


Figure 7.5: Algorithm comparison for networks with average node degree equal to 1.5, and multicast group size equal to 5% of the number of nodes

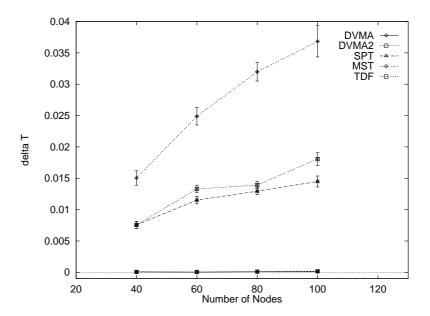


Figure 7.6: Algorithm comparison for networks with average node degree equal to 4, and multicast group size equal to 5% of the number of nodes

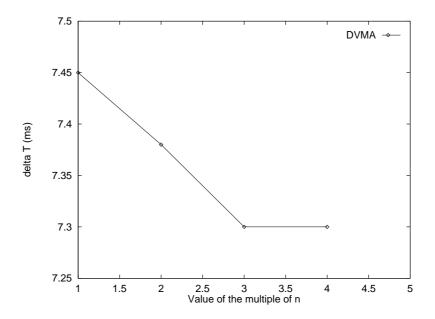


Figure 7.7: DVMA performance comparison for different values of parameter k (k is a multiple of n), average node degree equal 2.5, and multicast group size equal 10%.

7.2 Selection of Parameters k and l

In Chapter 5 we established that the complexity of DVMA is $O(klmn^4)$. Recall that k is the total number of paths between the source s and the farthest destination node w, that are calculated. Parameter l is, similarly, the total number of paths that are calculated between some node v of the partially-built tree T, and the destination $u \in U$ that is currently being connected to the tree T. However, in the description of the algorithm, the values of k and l were left unspecified. In general, the choice of particular values will be a compromise between the quality of the final tree (in terms of maximum delay variation δ_T) and speed.

We now show how the values of k and l were selected for the performance evaluation described in this chapter. A comparison of performance of DVMA family of algorithms under different values of k and l revealed the optimal values to be:

- k = 3n
- \bullet l=n

The data that motivated the selection of k is presented in Figure 7.7. The algo-

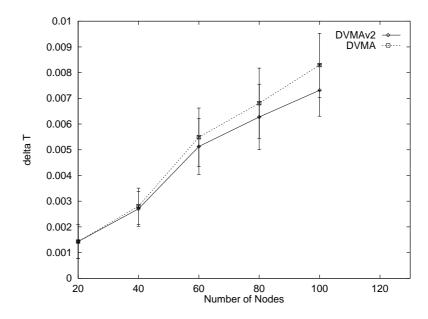


Figure 7.8: Performance comparison between DVMA and the higher complexity heuristic DVMAv2

rithms were executed with different values of k on 100 random graphs with 100 nodes and 10 destinations. The average maximum delay variation δ_T was recorded. It is clear from this figure that when the value of k has reached 3n, the average maximum delay variation stopped decreasing. This means that most of the time a tree with a minimum δ_T can be found among the trees, built on the first 3n shortest paths, connecting source s to destination w. The optimal value for l was attained similarly.

7.3 Performance of Higher Complexity Heuristics

In Chapter 5 we have mentioned an attempt to improve the performance of the DVMA family of heuristics by increasing their complexity. The two higher complexity heuristics, which we will call DVMAv2 and DVMA2v2 were tested against DVMA and DVMA2 on a series of random graphs. As in section 7.1, the graph size varied from 20 nodes to 100 nodes; the average nodal degree was 2.5. The comparative data for DVMA and DVMAv2 from 100 graphs was averaged, and is presented in Figure 7.8 (with 95% confidence intervals).

It is not hard to see that the performance of DVMAv2 is not significantly better on the average, than that of DVMA. In fact, the presented 95% confidence intervals for the mean estimates intersect.

7.4 Performance of the Dynamic Algorithm

Our final set of data refers to the performance of the dynamic algorithm, described in Chapter 6. Recall that the algorithm consists of two separate operations on node $u \in G$: JOIN(u), when node u wants to join a multicast session, and LEAVE(u), when u wants to leave the current session(tree). Recall also that JOIN(u) has three special cases:

- 1. $u \notin V_T$ node u is not part of the current multicast routing tree T
- 2. $u \in V_T$ node u is a relay node in the tree T and its addition to the multicast group does not violate the maximum delay variation δ_T
- 3. $u \in V_T$ node u is a relay node in the tree T and its addition to the multicast group violates δ_T

As with the *DVMA* algorithms, we were interested in the average behavior of the dynamic algorithm. Tests were set up as follows: after creating an initial tree *T* from a random graph using algorithm *DVMA*, a sequence of *join/leave* operations was performed on that tree. The choice of whether a *join* or a *leave* operation was performed at a given moment was random, governed by the Bernoulli distribution (e.g., 50 or 75 percent of all operations had to be *join* operations). After the operation was performed, the maximum delay variation of the new tree was measured and recorded. In addition, the current set of destinations was updated after every operation, and after the update, the *DVMA* algorithm was run on this new set, so that the "optimal" delay variation value from this algorithm could be compared to the delay variation value produced by the dynamic algorithm. Both these values were averaged over 100 random graphs with 100 nodes and 10 multicast destinations each (as before, we also present the 95% confidence intervals for our mean estimates). All

graphs had an average nodal degree of 3. A total of 10 join/leave operations was performed on each tree.

In our experiments we have varied the percentage of join operations over the total number of operations performed. We have studied the cases of 50, 75 and 100 percent, the latter consisting only of join requests. The resulting data is presented in Figures 7.9 (50/50 mix of join/leave operations), 7.10 (75/25) and 7.11 (100/0).

The dynamic algorithm behaves as one would expect - the resulting new tree has the maximum delay variation greater than that of a tree created "from scratch". With 50/50 mix of join/leave operations the average maximum delay variation of the trees after 10 operations was 2.73 times larger than "optimal" (rather, smallest attainable with DVMA). In 75/25 this ratio fell to 2.33, and with 100/0 it became 1.64. This could be explained by the fact that the dynamic algorithm employs a minimal disruption approach, which causes the tree to "deteriorate" over the time. Since the leave operations do not cause the tree to reshuffle, there is usually little improvement in the value of δ_T after a leave operation. At the same time, an algorithm, run "from scratch" finds a better solution, working with a smaller set of destinations. On the other hand, join operations almost always cause the value of δ_T to increase both when the dynamic algorithm is executed and when the DVMA algorithm is run on the enlarged destination set. For these reasons, the smaller the share of leave operations in the total number performed, the closer the performance of the dynamic algorithm to the optimal.

The average behavior of the dynamic algorithm is not indicative of some particular cases. For instance, we have observed some cases, where the value of δ_T of the multicast tree has improved after a *join* operation. This happens when the node, which wants to join the tree, is already a relay node of the existing multicast tree and is located close to the source (the last case for the *join* operation in Chapter 6). If the addition of the node to the multicast group causes the δ_T of the tree to deteriorate beyond the specified constraint, all members of the multicast group, which are positioned downstream from the new node in the tree T, will have to be reconnected. This causes a major "reshuffling" of the tree and in some cases results in an improvement of the value of δ_T .

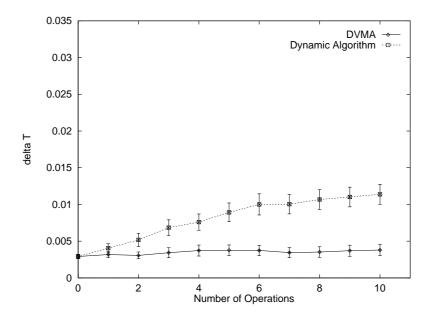


Figure 7.9: Performance of the dynamic algorithm compared to the optimal with $50/50 \ join/leave$ mix

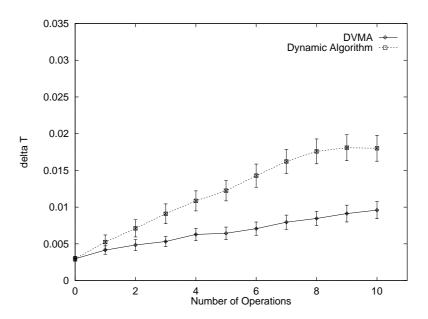


Figure 7.10: Performance of the dynamic algorithm compared to the optimal with $75/25 \ join/leave$ mix

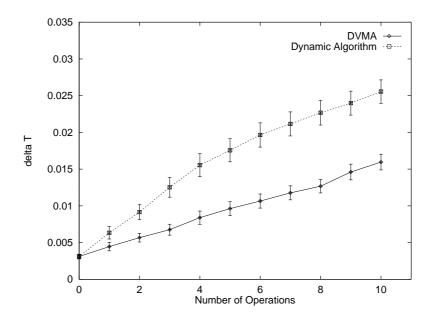


Figure 7.11: Performance of the dynamic algorithm compared to the optimal with $100/0 \ join/leave \ mix$

It should be noted, that despite the fact that routing trees calculated by the dynamic algorithm are not as good as the ones, calculated "from scratch", their maximum delay variation is still much lower than that of other algorithms we considered (*TDF*, *SPT*, *MST*), and is attained at a lower cost, than if they were calculated by *DVMA*.

Chapter 8

Summary and Future Work

8.1 Summary

We have considered the problem of determining multicast trees that guarantee certain bounds on the end-to-end delays from the source to the each of the destination nodes, as well as on the variation among these delays. The bounds are directly related to the quality of service requirements of the higher level applications performing the multicast. After establishing that the problem of constructing such constrained trees is \mathcal{NP} -complete, we developed heuristics that exhibit good average case behavior. Our heuristics perform especially well under conditions typical of multicast scenarios in high-speed networks, namely, when the network is not too sparse, and when the number of destination nodes is relatively small compared to the total number of nodes.

In addition we have shown that the strategy employed by the heuristic is applicable to the problem of reorganizing the tree in response to changes in multicast group membership. The resulting dynamic algorithm also performs well in scenarios which model real-life connections.

8.2 Future Work

Our work can be extended in several directions. Recall that our algorithms do not attempt to optimize the tree in terms of cost; in fact, since their strategy is to

choose longer paths for some of the destination nodes, the cost of the final tree may be somewhat high. One straightforward approach to dealing with cost is to modify our algorithms to seek the least cost path among the candidate paths they consider, and/or the least cost tree among all feasible trees they construct. However, this issue warrants further investigation.

Another area of possible research which has not been widely investigated, is the construction of multicast trees for multipoint-to-multipoint connections. Under this scenario any node belonging to the multicast group can send multicast messages to the rest of the group. The transmitting node expects that the network will provide it with an acceptable quality of service level (in our case - bounded delay and delay-variation). An obvious (and inefficient) solution would be to run *DVMA* algorithm for every node in a multicast group as a source, since in our work we considered delay variation and delay in the tree only from the point of view of the source node. This approach would create one multicast tree per member of the multicast group which would be wasteful in terms of network resources. A different approach would be to create a single routing tree which will attempt to satisfy the quality of service requirements of all members of the multicast group. Whether this approach is a feasible one, remains to be seen.

Bibliography

- [1] J. S. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 24(10):8-15, October 1986.
- [2] S. L. Hakimi. Steiner's problem in graphs and its implications. Networks, 1:113– 133, 1971.
- [3] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing steiner minimal trees. SIAM Journal of Applied Mathematics, 32(4):835–859, June 1977.
- [4] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.
- [5] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves. A source-based algorithm for near-optimum delay-constrained multicasting. In *Proceedings of IEEE Infocom* '95, March 1995.
- [6] H. Salama, D. Reeves, Y. Viniotis, and T. Sheu. Comparison of multicast routing algorithms for high-speed networks. Technical Report IBM-TR29.1930, IBM, September 1994.
- [7] H. Salama, D. Reeves, Y. Viniotis, and T. Sheu. Evaluation of multicast routing algorithms for distributed real-time applications in high-speed networks. In Proc. of 6th IFIP Conf. on High Speed Networks, September 1995.

- [8] H. Salama, Y. Viniotis, D. Reeves, and T. Sheu. Multicast routing algorithms for high-speed networks. Technical Report IBM-TR29.1883, IBM, May 1994.
- [9] Sara Baase. Computer Algorithms. Addison Wesley, 1988.
- [10] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. Acta Informatica, 15:141–145, 1981.
- [11] V.J.Rayward-Smith. The computation of nearly minimal steiner trees in graphs. Intl. J. Math. Educ. Sci. Technol., 14(1):15-23, 1983.
- [12] M. R. Garey and D. S. Johnson. Computers and Intractability. W. H. Freeman and Co., New York, 1979.
- [13] D. Bertsekas and R. Gallager. Data Networks. Prentice Hall, Inc., 1992.
- [14] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [15] E. Lawler. Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, 1976.
- [16] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, 36:1389–1401, Nov 1957.
- [17] E. N. Gilbert and H. O. Pollak. Steiner minimal tree. SIAM Journal on Applied Mathematics, 16, 1968.
- [18] K. Bharath-Kumar and J. M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Transactions on Communications*, COM-31(3):343-351, March 1983.
- [19] B. W. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.