International Journal of Foundations of Computer Science

THE EFFECT OF ASYMMETRY ON THE ON-LINE MULTICAST ROUTING PROBLEM

MICHALIS FALOUTSOS*
Computer Science, University of California Riverside
Riverside, California, 92521, USA

RAJESH PANKAJ
Qualcomm Inc.,6455 Lusk Blvd
San Diego, California, 92121, USA

KENNESTH C. SEVCIK
Computer Science, University of Toronto
Toronto, Ontario, M5S 3G4, CANADA

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

In this paper, we study the problem of multicast routing on directed graphs. We define the asymmetry of a graph to be the maximum ratio of weights on opposite directed edges between a pair of nodes for all node-pairs. We examine three types of problems according the membership behavior: (i) the *static*, (ii) the *join-only*, (iii) the *join-leave* problems. We study the effect of the asymmetry on the *worst case performance* of two algorithms: the Greedy and Shortest Paths algorithms. The worst case performance of Shortest Paths is poor, but it is affected by neither the asymmetry nor the membership behavior. In contrast, the worst case performance of Greedy is a proportional to the asymmetry in a some cases. We prove an interesting result for the join-only problem: the Greedy algorithm has near-optimal on-line performance.

Keywords: Multicast routing, Steiner, on-line, directed

1. Introduction

In this paper, we examine how network asymmetry and membership behavior affect the performinace of algorithms for the multicast routing problem. Multicasting involves the distribution of the same information stream from one node to many nodes concurrently over a tree. In recent years, multicast routing has attracted significant attention with the emergence of applications such as video-on-demand, teleconferencing, and tele-education. Finding the minimum-cost multicast tree is

*This work was supported by the National Science Foundation under CAREER Grant No. 9985195, DARPA award N660001-00-1-8936. Email: michalis@cs.ucr.edu

equivalent to the NP-hard Steiner tree problem [7]; given a weighted graph $G(V, E)$ we want to find the minimum-cost tree that spans a subset of nodes, S, rooted at a node $s \in S$. In the rest of this paper, we will use the terms Steiner and multicast problem interchangeably.

Directed graphs are more accurate representations of a network for practical purposes. However, most previous work on the multicast problem focuses on undirected weighted graphs. In most real networks, adjacent nodes are connected with a pair of opposite and independent edges. The cost of an edge is an aggregate metric of various factors such as available bandwidth, or delay. We define asymmetry, A, of a directed graph to be the maximum ratio of the cost of opposite edges between a pair of nodes over all adjacent node-pairs. Clearly, asymmetric graphs include both previous graph models: undirected graphs for $A = 1$, and directed graphs for $A = \infty$. Consequently, our bounds include bounds on undirected graphs as special cases.

In our work, we define the dynamic Steiner tree problem as follows. Assume a weighted directed graph, a source node and group-members that join and leave the tree arbitrarily. Our objective is to minimize the cost ratio of the generated tree over the optimal tree in the worst case. which is known as the competitive ratio of the algorithm. Note that we do not assume any knowledge of the future membership and we do not allow rerouting the tree. We can distinguish three types of problems. The **static or off-line multicast problem** assumes that all the destinations of a session are known in the beginning of a session. The **join-only on-line problem** allows destinations to join arbitrarily, but they all stay until the end of the session. The **join-leave on-line problem** allows destinations to join and leave arbitrarily. For uniformity, we define M to be the number of the join requests plus one (one accounts for the source). Note that each problem includes its predecessors as a special case.

We examine the worst case performance of two approximation algorithms as function of the asymmetry. We focus on the two algorithms that are most widely used in practice in multicast routing [15, 8, 4, 1, 17, 9]. The first algorithm, which we call **Shortest Paths**, constructs the tree as the union of the shortest paths between the source and each destination. The second algorithm, which we call **Greedy**, is based on an efficient greedy heuristic for the Steiner tree problem [21]; Among all new destinations, we connect the one with the minimum cost path to any point of the current tree.

In this paper, we examine the effect of the asymmetry of the graph and the membership behavior on the dynamic Steiner tree problem. More specifically, we prove the following bounds. It is easy to show that the competitive ratio of Shortest Paths is tightly bounded by $\Theta(M)$. In addition, we prove the following bounds: for the competitive ratio of the algorithms:

- For the *static* problem, the approximation ratio of Greedy is tightly bounded by $\Theta\left(\min\left(A, M\right)\right)$.
- For the *join-only*, the competitive ratio of *any* on-line algorithm is bounded

below by: $\min\left(M, A\cdot\frac{\log M}{\log(A+1)}\right)$

- For the *join-only*, Greedy has near-optimal on-line performance. Namely, Greedy is $\log(M+1)$-competitive compared to *any* on-line algorithm.
- For the *join-leave* problem, we prove that Greedy has a competitive ratio proportional to the asymmetry and exponential in M: $\max(A, 2^M)$.

Putting things in perspective, our work is a generalization of the work of Imase and Waxman [12] on undirected graphs. Furthermore, our results include their results for asymmetry $A = 1$. Earlier versions of this work have been presented in conferences [10, 11].

The rest of this paper is structured as follows. In section 2, we present our model and previous work. In section 3, we discuss the static problem and prove a tight bound for Greedy. We also show the tight bound of the Shortest Paths algorithm for all problems. In section 4, we address the join-only problem and we present an upper bound for Greedy. In section 4.2, we prove a lower bound for *any* on-line algorithm for the join-only problem. In section 5, we discuss the join-leave problem, and we prove a lower bound for the Greedy algorithm. In section 6, we give an overview of all previous and new results.

2. Model and Previous Work

A common measure of the quality of an approximation algorithm is the **approximation ratio**, which is defined as the maximum ratio of the cost of the tree of the algorithm, T, over the optimal one, OPT. The same metric is used for on-line approximation algorithms, and it is typically refer to as **competitive ratio** (CR), $CR = T/OPT$ where the optimal cost is the cost of the off-line optimal algorithm.

We assume a weighted directed graph $G(V, E)$. We denote the total number of nodes by N, and we denote by S, the set of the destinations and the source, which we call **participant** nodes. We define M to be the number of the join requests plus one for the source, s. We want to establish connections from the source towards the destinations. We denote the weight of an edge (u, w) by the function $e(u, w) : V \times V \to \mathcal{R}^+$. The cost of a path (tree) is the sum of the weights of the directed edges of the path (tree). We say a node v *reaches* node w if there is a directed edge or path from v to w. The distance between two nodes is denoted by by $d(u, w) : V \times V \to \mathcal{R}^+$. Similarly, the distance between a node and a tree is the minimum distance between the node and any node of the tree.

We quantify the asymmetry of directed graphs with the following metric.

Definition 1 *The asymmetry of a graph $G(V, E)$, A, is defined as*

$$A = \max_{(v,w)\in E}\left(\frac{e(v, w)}{e(w, v)}\right)$$

We call **opposite** edges the pair of edges between two nodes, i.e., for $v, w \in V$, (v, w) and (w, v) are opposite edges. We use the term *unbounded directed* graph to refer to the special case of $A = \infty$.

The Greedy algorithm was originally suggested by Takahashi and Matsuyama [21] for the Steiner problem. The algorithm can be generalized in a straight-forward way for join-leave requests that arrive on-line. A description of the algorithm in pseudo-code is presented in Appendix 1. For leave requests, we prune the unnecessary parts of the tree. For join requests, we connect the node with the minimum cost path from any point in the tree. The Shortest Paths algorithm takes the union of the shortest paths between the source and each destination. Observe that we route every destination independently of the other destinations. For this reason, this algorithm can be used for any on-line problem and in any kind of graph.

Most previous work focuses on the static problem or on undirected graphs.

Undirected case. For the **static undirected problem**, the CR of Greedy is bounded by $2 \cdot (1 - \frac{1}{M}) \leq 2$ [21]. For the **undirected join-only** problem, Bharath-Kumar and Jaffe [3] and later Imase and Waxman [12] proved that the CR of the Greedy is bounded by[†] $\log(M)$. Westbrook and Yan [26] refined this bound to $O(\log(\frac{d_{max}}{OPT} \cdot M)$, where d_{max} the maximum distance among the destinations and/or the source. For the **undirected join-leave** problem, Imase and Waxman [12] prove that M is a lower bound, which we increase to 2^M. Several experimental studies of this problem exist [6] [25] [24].

Directed case. For the **static directed** problem, Ramanathan [19] proved that $2 \cdot A$ is an upper bound for Greedy. For $A = \infty$, Wong proposed a dual ascent approximation algorithm[28]. For the same problem, Voss showed that M is an upper bound for Greedy and the Shortest Paths [22].

Other Heuristics for the Static Problem. For the static undirected problem, Zelikovsky has the best known approximation algorithm with a ratio of $1 + \ln 2$ [30]. For the static directed Steiner problem, Charikar et al. propose an algorithm with an approximation ratio of $i(i-1)M^{1/i}$) where i a constant [5]. Note that the above algorithms do not address the on-line problem. Various interesting heuristics exist for the static Steiner tree problem [14, 16, 20, 29, 23] for a survey see [27, 18].

Comments and Assumptions. We examine the worst case performance of the algorithms. We consider deterministic algorithms only, and so our worst case is equivalent to the case where we are so "unlucky" that our routing decisions are the worst with respect to the future requests. Thus, in our examples, once we have routed a destination, the on-line sequence will select the "worst" node as the next destination. In addition, we do not consider rerouting the multicast tree, which can help us recover from bad choices in the past [12] [13] [2].

3. The Static Problem

In this section, we study the static multicast problem on directed graphs, and we prove upper and lower bounds on the performance of Greedy. In addition, we prove a tight bound for the Shortest Paths algorithm.

Theorem 1 (Upper bound I) *The following statements hold:*

a) For the static problem, the approximation ratio of the Greedy algorithm in

[†]All the logarithmic functions in this paper have a base of two.

any directed graph is no worse than

$$\frac{T}{OPT} \leq \frac{\sum_{v \in S} d(s,v)}{d_{max}} \leq M - 1$$

where d_{max} is the maximum distance of any destination from the source.

b) The same bound holds for the approximation ratio for Greedy in the join-only problem, and for Shortest Paths in the static, the join-only and the join-leave problems.

PROOF.

a) The cost T of a Greedy tree is bounded above by the sum of the distances between the root, s and all the other nodes in the tree. Therefore:

$$T \leq \sum_{v \in S} d(s,v) \leq (M-1) \cdot d_{max}$$

The cost of the optimal tree OPT is at least as large as the largest distance from the source to any destination d_{max}, $d_{max} \leq OPT$, and the result follows.

b) The proof of part (a) does not make any assumption about the order in which the destinations join. Therefore, with the same arguments, we can prove the bound for Greedy in the join-only problem, and Shortest Paths for static, join-only, and join-leave problems. □

Ramanathan [19] proved the following upper bound for Greedy with respect to the asymmetry of the graph.

Theorem 2 (Upper bound II) *The approximation ratio of the Greedy algorithm in any directed graph of asymmetry A is bounded above by*

$$\frac{T}{OPT} \leq 2 \cdot A$$

We combine the two upper bounds in the following corollary.

Corollary 1 *The approximation ratio of Greedy on a directed graph is*

$$\frac{T}{OPT} = O\left(min\left(M, A\right)\right)$$

We prove that the above asymptotic bound is also a lower bound. We identify a family of graphs for which the approximation ratio of Greedy coincides with this upper bound.

Theorem 3 (Lower bound) *Given the asymmetry A, and the multicast group size, M, there exists a directed graph and a multicast group such that the approximation ratio of the Greedy algorithm is at least*

$$\frac{T}{OPT} = \Omega\left(\frac{A\,M}{A+M}\right)$$

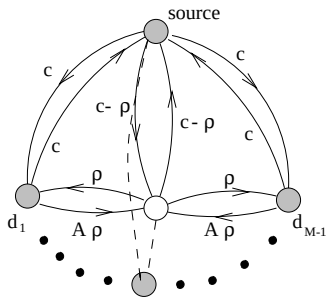


Figure 1: An example of a bad case of a directed graph for Greedy in the static problem.

PROOF. Consider the graph depicted in fig. 1. The source reaches all the destinations with an edge of cost c. There is only one non-participant node that we call the **middle node** (the white node in fig. 1). This node reaches all the participants with an edge of cost ρ, while the opposite edges are of cost $A\ \rho$. The source reaches the middle node with a cost of $c - \rho$, with $\rho \leq c$.

The optimal tree includes the middle node, since $\rho \leq c$:

$$OPT = c - \rho + (M-1)\rho = c + (M-2)\cdot\rho \tag{1}$$

The Greedy will not use the paths through the middle node, if the weights of the edges are as below:

$$c < \rho + A\rho \tag{2}$$

The cost T of the tree of Greedy is given by: $T = (M-1)\cdot c$ and, the approximation ratio of Greedy becomes:

$$\frac{T}{OPT} = \frac{(M-1)c}{c + (M-2)\rho} \tag{3}$$

The approximation ratio of Greedy is maximized for a large c and a small ρ. From eq. 2, the minimum value of ρ for a given c is : $c/(A+1) < \rho$. Taking the limit of the approximation ratio for $\rho \to c/(A+1)$ completes the proof. □

Note that $O(min(M, A)) = O\left(\frac{M\ A}{M+A}\right)$. Therefore, the upper and lower bounds coincide (corollary 1 and theorem 3) and we have the following tight bound.

Corollary 2 *For any A and M, there exists a directed graph of asymmetry A, and a multicast group of size M, such that the approximation ratio of Greedy has complexity*

$$\frac{T}{OPT} = \Theta\left(\min\left(M, A\right)\right)$$

For the Shortest Paths algorithm we can prove the following tight bound.

Theorem 4 *For the multicast problem, the approximation ratio of the Shortest Paths algorithm is $\Theta(M)$.*

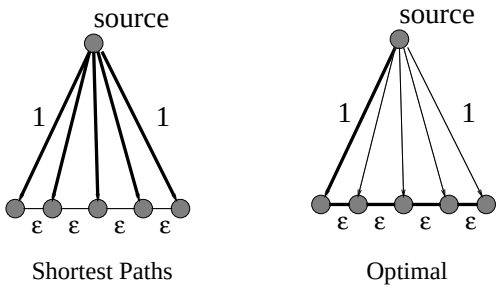


Figure 2: A worst case example for the Shortest Paths algorithm.

PROOF. Theorem 1 proves that M is an upper bound for Shortest Paths. We prove that it is also a lower bound. Assume a graph where the source is connected with each destinations with an edge of unit cost (see fig. 2). Assume $M-2$ edges of small weight ϵ between destinations, in a way that they form a path. Clearly, Shortest Paths creates a tree of cost $T = M - 1$; while the optimal tree has a cost of $OPT = 1 + (M-2)\cdot\epsilon$. Letting $\epsilon \to 0$ concludes the proof. □

Thus, for small asymmetry ($A \leq M$) Greedy has a better approximation ratio than Shortest Paths in the worst-case. However, the approximation ratio of Greedy becomes similar to that of the Shortest Paths in highly asymmetric graphs.

4. The Join-Only Problem

We show that the Greedy is near-optimal compared to any on-line algorithm for the join-only problem. We prove an upper bound for the Greedy algorithm, and then prove a lower bound for any on-line algorithm. We show that the two bounds differ only by a factor of $\log M$.

4.1. The Join-Only Upper Bound

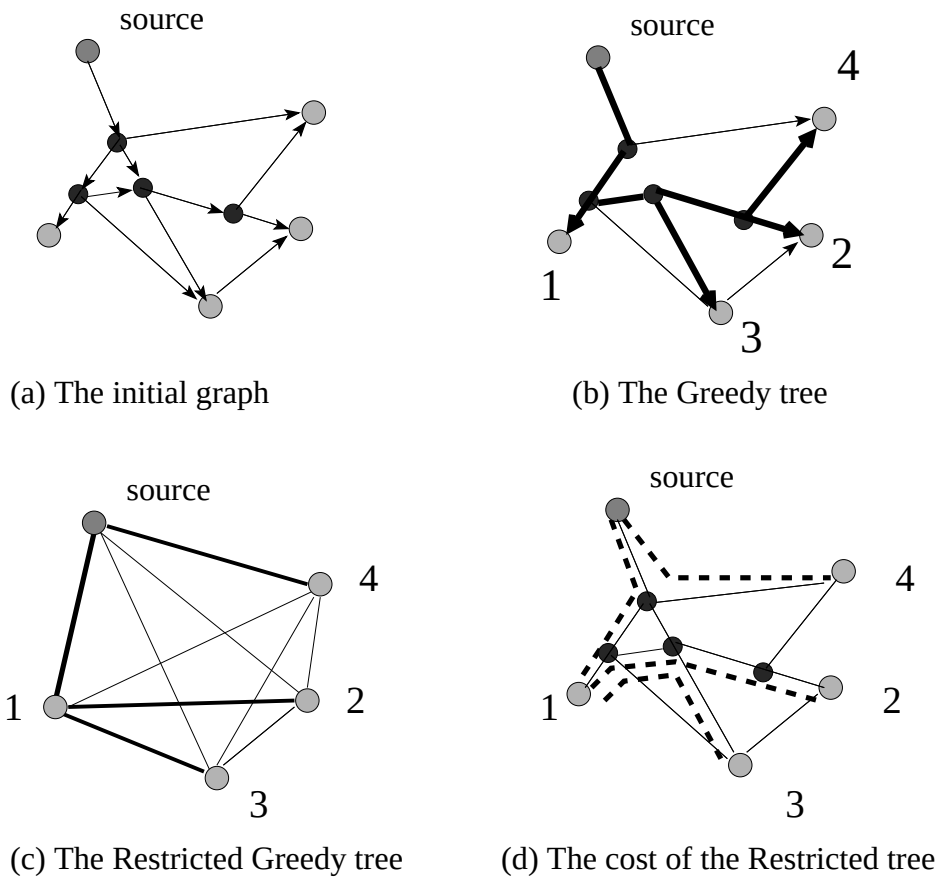
We prove that $A \log M$ is another upper bound for Greedy in addition to the bound of M of theorem 1. Both bounds are valid for any asymmetry value, and so we can combine them by taking the minimum of the bounds (see corollary 3).

Theorem 5 *For any weighted directed graph with asymmetry A and a multicast group of size M, the competitive ratio of the Greedy algorithm in the join-only problem is bounded above by:*

$$\frac{T}{OPT} \leq A \cdot \log M$$

where T is the cost of the Greedy tree and OPT the cost of the optimal one.

PROOF. The basic steps of this proof are as follows. First, we transform the directed graph into an undirected graph in which the weight of each edge is the minimum of the two directed edges. Then, we apply a restricted version of the Greedy to the undirected graph, which we define below. Intuitively, it is easier to compare the restricted Greedy on the undirected graph with Greedy on the directed. Finally, we



(a) The initial graph

(b) The Greedy tree

(c) The Restricted Greedy tree

(d) The cost of the Restricted tree

Figure 3: Graphs for theorem 5. Note that the directed graph has opposite edges between nodes, but they are not shown for simplicity. When node 4 joins, after nodes 1,2 and 3, restricted Greedy chooses edge (1,4) with weight P in the complete undirected graph, which corresponds to a path P' in the directed graph. In the worse case, the directed path has a cost $P' = AP$. When Greedy on the directed graph connects node 4, it will never choose a path more expensive than P'.

prove that the Greedy tree of the *directed* graph is no more than A the cost of the restricted Greedy on the undirected graph

Let $G(V, E)$ be the directed graph. Let us create an undirected graph $G_u(V, E_u)$ as follows. For every pair of opposite directed edges in G, we create an undirected edge in G_u with the minimum weight of the pair: $e_u(v, w) = \min(e(v, w), e(w, v))$, $v, w \in V$. Since OPT_u is the cost of the optimal tree on G_u, the following holds:

$$OPT_u \leq OPT \tag{4}$$

Let us define a variation of Greedy that we call **restricted Greedy** (see fig. 3). First, we create the graph $G'_u(V'_u, E'_u)$, to be the complete undirected distance graph of the participant nodes on G_u, i.e., $V'_u = S$, and the weight of the edge $(v, w) \in E'_u$ is equal to the distance of the nodes in G_u. Then, we execute Greedy on G'_u for the on-line problem. We call T_u the cost of the resulting tree. Note that T_u is a tree of the complete G'_u, and each of its edges corresponds to a paths in G_u (see fig. 3).

For the undirected on-line problem, Imase and Waxman (in theorem 2 [12]) proved that the cost of the restricted Greedy solution, T_u, is bounded as follows:

$$T_u \leq \log M \cdot OPT_u \tag{5}$$

Intuitively, the paths that restricted Greedy considers are a subset of the the paths that Greedy uses. If the restricted Greedy connects a destination with a path of cost P, the Greedywill use a path of at most AP. Note that both algorithms add the nodes in the same order in the join-only problem. For each destination, the path that Greedy chooses can be: a) the path P_d that corresponds to the undirected path that restricted Greedy chose for the same destination, or b) a directed path of cost less than $A\ P_d$ (see fig. 3). In the worst case, each directed path is at most A times more expensive than the corresponding undirected path, and thus we have:

$$T \leq A \cdot T_u \tag{6}$$

We can combine equations (4), (5), (6), to conclude that:

$$T \leq A \cdot T_u \leq A \cdot \log M \cdot OPT_u \leq A \cdot \log M \cdot OPT$$

□

We combine the two upper bounds from theorem 1 and theorem 5 in the following corollary.

Corollary 3 *For any directed graph with asymmetry A and a multicast group of size M, the competitive ratio of the join-only Greedy algorithm in the multicast problem is no worse than*

$$\frac{T}{OPT} = O(\min(M, A \cdot \log M))$$

We can see that for small asymmetry, $A \leq M/\log M$, the bound is $A \log M$, while, when the asymmetry becomes larger, the bound becomes M.

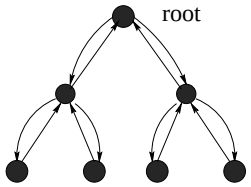


Figure 4: The d-tree : a complete binary asymmetric tree.

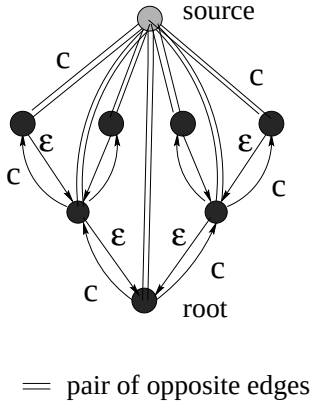


Figure 5: The worst case example for highly asymmetric graphs. Edges from or towards the source are of cost c. Elsewhere, curved edges are of cost c, while their opposite edges are of cost ϵ.

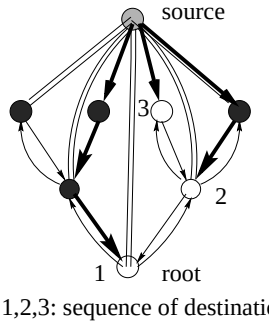


Figure 6: The worst case example for highly asymmetric graphs in the join-only problem for any algorithm.

4.2. A Lower Bound for the Join-Only Problem

In this section, we prove a lower bound for any algorithm for the join-only problem. We distinguish two cases: a) **highly asymmetric** ($A \geq M$), and b) **less asymmetric** ($A < M$) graphs. In both cases, we construct the worst case graph and an execution scenario. In highly asymmetric graphs, the worst case example is based on an asymmetric binary tree. In less asymmetric graphs, we need a more elaborate construction of that uses the previous binary tree as a building block.

4.2.1. Highly Asymmetric Graphs

We prove that, for highly asymmetric graphs, the worst case ratio of any on-line algorithm is proportional to the number of destinations.

Theorem 6 *Given a bound on the size of the multicast group, M, there exists an asymmetric graph, and a sequence of join requests such that the competitive ratio of any on-line algorithm is at least*

$$\frac{T}{OPT} \geq \frac{1}{2} \cdot (M-1)$$

where T is the cost of the tree of the algorithm and OPT is the optimal off-line cost.

PROOF. Intuitively, we want to create graphs that can be "bad" for the routing decision of any algorithm. For this, we have to ensure that the on-line algorithm must have at least two routing choices for every destination. In addition, for either routing choice, the arrival sequence must have the possibility to be equally bad. In other words, we need "two identical paths" for every arriving destination, and this suggests a binary tree configuration (see fig. 4).

Definition 2 *A* **d-tree** *is a complete binary directed tree that has pairs of opposite directed edges instead of undirected edges. The root of the d-tree is the node with exactly two adjacent nodes.*

Assume a d-tree with $M-1$ distinct levels. We assign weight c to the edges "pointing away" from the root (curved edges), and weight ϵ to the opposite edges (straight edges). Let the source be a node outside the d-tree and connected to each node of the d-tree with a pair of opposite edges of cost c (see fig. 5).

The sequence of destinations. We choose the root of the d-tree as the first destination. In each step, we choose a child of the previous destination that does not belong to the multicast tree (see fig. 6). This way, every new destination increases the cost of the multicast tree by c. In more detail, each destination can join the multicast tree with a) a path along the d-tree, b) an edge from the source, or c) an edge from the previous destination. Each of these choices has cost of at least c, and the total cost becomes

$$T \geq (M-1) \cdot c \tag{7}$$

By the definition of the sequence, each new destination is a child of the previous destination, and therefore there exists a path from the source to the root of the d-tree that contains all the destinations. Therefore, the optimal cost is

$$OPT = c + (M-2)\cdot\epsilon \leq 2\cdot c \tag{8}$$

For the last equality, we assume $\epsilon \leq c/M$. Using equations (7) and (8), we calculate the competitive ratio and complete the proof. □

In the example above, the asymmetry is at least $\Omega(M)$. This asymmetry appears at the edges of the d-tree : c/ϵ, given that we assume $\epsilon \leq c/M$. This is the constraint that makes this example apply only to "highly asymmetric" graphs.

4.2.2. Less Asymmetric Graphs

As we saw in the upper bounds, the relative performance of the Greedy algorithm improves as the asymmetry decreases. In less asymmetric graphs ($A = O(M)$), the competitive ratio of Greedy is bounded above by $A\cdot\log M$. Two questions are of interest: i) how tight is this upper bound (for our Greedy algorithm), and ii) can we expect other on-line algorithms to do better? The answer to both questions is that the competitive ratio of *any* on-line algorithm cannot be better than $O(\log(A)) = O(\log(M))$ the competitive ratio of Greedy.

Theorem 7 *Given the size of the multicast group, M, and a bound on the asymmetry $A = O(M)$, there exists a graph and a sequence of join requests such that the competitive ratio of any on-line algorithm is at least*

$$\frac{T}{OPT} = \Omega\left(A\,\frac{\log M}{\log(A+1)}\right)$$

where T is the cost of the tree of the algorithm and OPT is the cost of the optimal tree.

PROOF. The proof consists of three parts. First, we describe how we construct a graph given the asymmetry and multicast group size. Second, we define the on-line sequence in which nodes join. Third, we calculate a lower bound on the cost of the tree for any algorithm.

PART I: Graph construction. We present the intuition behind the **family of graphs** that we denote by F_g. The building block of our graphs is the d-tree of depth k as defined in section 4.2.1. The d-tree guarantees that the on-line algorithm has two identical routing choices for each destination. We construct the graph in a recursive manner, and we call each step a **generation**, g. We start from a pair of nodes, and at each step we insert a d-tree between adjacent pair of nodes. The depth k is a parameter, which we set to $k = \lfloor A \rfloor$ eventually.

We introduce the *layer* to identify nodes with their distance from the source.

Definition 3 *The* **layer** *of a node in the graph F_g is equal to the length (number of edges) of the shortest path from the source to the node.*

A node of the F_g graphs is denoted by $v_{g,l,x}$, where g is the generation, l is the layer , and x an index that identifies nodes of the same layer. For example, in fig. 7, in F_1 we have four nodes in layer one: $v_{1,1,1}, v_{1,1,2}, v_{1,1,3}, v_{1,1,4}$, and two nodes in layer two: $v_{1,2,1}, v_{1,2,2}$. In this proof, we do not need to specify values for x, and we use it as a placeholder.

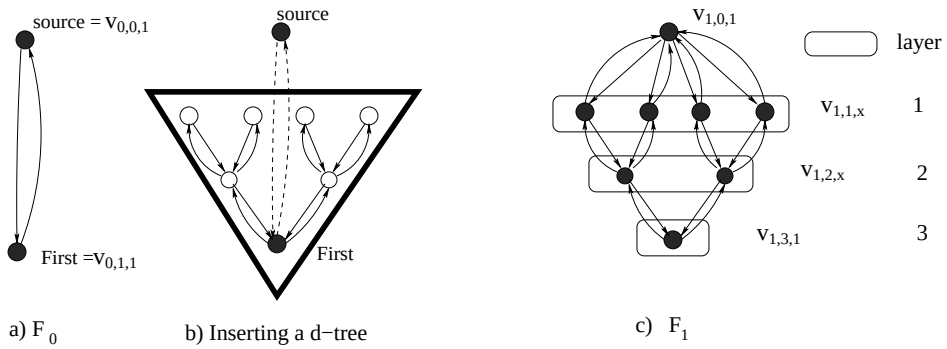


Figure 7: The family of F_g with $k = 2$: F_0 and F_1

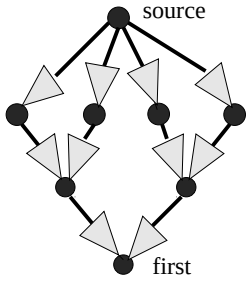


Figure 8: The family of F_g with $k = 2$: F_2. We replace the edges between nodes of F_1 with d-trees just like in Figure 7.b. The thick edges represent multiple pairs of edges for visual clarity.

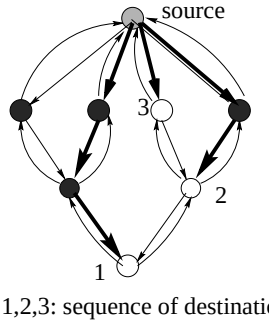


Figure 9: A worst case on-line sequence for graph F_1 for less asymmetric graphs in the join-only problem for any algorithm.

We outline briefly the creation procedure. We start with a pair of nodes, one of which is the source. For every generation, we insert a d-tree between every pair of nodes of the previous generation (see fig. 7 and fig. 8). The weights of the edges are set in such a way that the distances are preserved: all paths between the two nodes of F_0 are of equal cost. Intuitively, the downwards straight edges are light, while the upwards curved edges are heavy.

Initial Graph. The initial graph, F_0 has two nodes, the source node, $v_{0,0,1}$ and node and $v_{0,1,1}$ which we refer to as **First** (see fig. 7). There are two directed edges between the nodes with weights:

$$e(v_{0,0,1}, v_{0,1,1}) = \rho_0, \quad e(v_{0,1,1}, v_{0,0,1}) = A \cdot \rho_0$$

Creating the Next Generation F_g. Between every pair of adjacent nodes in F_{g-1}, we introduce a d-tree (see fig. 7 and fig. 8). Assume the pair of nodes $v_{g-1,l,x}$ and $v_{g-1,l+1,x}$ of F_{g-1}. Node $v_{g-1,l+1,x}$ becomes the root of the new d-tree. Node $v_{g-1,l,x}$ is connected to all the leaves of the tree. We refer to the nodes introduced in generation g as **new-nodes** of the generation. All the "old" nodes can be identified alternatively with indices of the newer generations; the new layer index of a node is $k+1$ times the old index: $v_{g-1,i,x} = v_{g,i(k+1),x}$.

The weights of the edges between adjacent nodes are set as follows.

$$e(v_{g,l,x}, v_{g,l+1,x}) = \rho_g \tag{9}$$

$$e(v_{g,l+1,x}, v_{g,l,x}) = A\ \rho_g \tag{10}$$

$$\rho_g = \frac{\rho_0}{(k+1)^g} = \frac{\rho_{g-1}}{k+1} \tag{11}$$

With this construction, we guarantee the following important properties of the F_g graphs. First, the asymmetry of the graph is A. Second, the distance between any pair of nodes is the same in each generation. As we can see in equation (11), we replace an edge of cost ρ_{g-1} with (multiple paths of) $k+1$ edges of $\rho_{g-1}/(k+1)$ cost.

Observation 1 *All paths between a pair of nodes are of equal cost.*

Finally, we can prove the following relationship between the number of layers and the generation of a graph using induction on g.

Observation 2 *The number of layers, l, of graph F_g is exponential in the generation index g:*

$$l = (k+1)^g + 1$$

PART II: The on-line sequence of joining destinations. Intuitively, we choose one node from each layer and once it is connected we choose the next node in a way that maximizes the total tree cost. The selection process follows the worst case sequence of Theorem 6 that used on one d-tree. The only difference is that we do many such processes for each of the d-trees in each generation. First, we select nodes from the first generation in the worst case sequence (see fig. 9). Then, we repeat the process for selected d-trees of the next generation.

Given a graph F_g, we choose one node from each layer along a path from the First (first node to join) to the source. We choose the nodes in order of increasing

generation number, and within the same generation, in order of decreasing layer. Note that, in each generation, *we choose nodes only in d-trees that are between destinations of the previous generation.* We present the beginning of the sequence of nodes:

$$\underbrace{v_{0,1,1}}_{g=0}\ \underbrace{v_{1,k,x}\ v_{1,k-1,x}\ \cdots\ v_{1,1,x}}_{g=1}\ \underbrace{v_{2,m,x}\ v_{2,m-1,x}\ \cdots\ v_{2,1,x}}_{g=2}\ \cdots$$

where we use the abbreviation $m = k\cdot(k+1)$.

Given the above sequence, the total number of destinations is equal to the number of layers. Using Observation 2, we can prove the following.

Observation 3 *In a graph F_{g_m}, if we select M destinations with the above sequence, the following holds*

$$g_m = \Theta\left(\frac{\log M}{\log(k+1)}\right)$$

How do we pick nodes within each layer? Let us focus on a d-tree. We follow the worst case sequence of Theorem 6. The root of the d-tree is connected already since it is part of the previous generation (see fig. 9). We pick a child of the root that has not been included in the tree as part of another path. Once this node is joined in the tree, we pick among each children one that does not belong to the multicast tree (see fig. 9).

After choosing nodes from all the levels of a generation, we choose nodes from the levels of the next generation *between* the previous destinations. This way, it is easy to see the following:

Observation 4 *Given the above on-line sequence on graph F_g, there exists a path from the source to the First node that contains all the destinations.*

PART III: Calculation of the Cost. For the above on-line sequence, we will find a **lower bound**, T, on the cost for any on-line algorithm on a graph F_{g_m}. We denote the cost of connecting the new-nodes of generation g by T_g. Given Observations 1 and 4 there exists a path from source to First that contains all the destinations of cost ρ. Therefore, the cost of the optimal solution is given by:

$$OPT = \rho_0 \tag{12}$$

We the depth of the d-tree , k, to be equal to the asymmetry:

$$k = \lfloor A \rfloor \tag{13}$$

Given that from each layer we select a destination[‡], k is less than M. Combining this with equation (13), we get an **upper bound on the asymmetry of the graph**:

$$A \leq M \tag{14}$$

[‡]Recall that M is equal to the number of layers, l, and that $k \leq l$ (Observation 2 for $g \geq 1$), thus, $k \leq M$.

The lower bound T of the cost of any on-line algorithm is equal to the sum of the lower bounds of the costs of each generation, T_g for $g = 0, ..., g_m$:

$$T = T_0 + \sum_{g=1}^{g_m} T_g \tag{15}$$

It is easy to see that $T_0 = \rho_0$. We will now calculate the value of T_1 (cost of the destinations added in F_1), and then generalize the result for the other generations. Consider the destination of level l, and note that $l \geq 1$, since zero level corresponds to the source. This destination can either connect: a) to the source or b) to another destination (see fig. 9). In the first case, the cost is $(k + 1 - l)\cdot\rho_1$, while in the second case, the cost is $A\cdot\rho_1$. Given that $k = \lfloor A \rfloor$ from equation 13, the minimum cost connection is $(k + 1 - l)\cdot\rho_1$ for all layers ($l \geq 1$). Furthermore, all possible paths are edge-disjoint, because of the way we choose our destinations. Therefore, the sum of all these paths is a lower bound of the cost of this generation:

$$T_1 = \sum_{i=1}^{k} i\cdot\rho_1 = \frac{k(k-1)}{2}\cdot\rho_1 = \frac{k(k-1)}{2}\cdot\frac{\rho_0}{k+1} \tag{16}$$

For the cost of generation g, it is not difficult to see that for every added d-tree, we can calculate the cost in a similar way, and we will end up with an expression similar to the one above only with ρ_g instead of ρ_1. At each step, the number of these d-trees (that contain destinations) is equal to the number of layers of F_{g-1} graph minus one. Using Observation 2, we have $(k + 1)^{g-1}$ such trees. Thus, the total cost of generation g is given below:

$$T_g = (k+1)^{g-1}\cdot\rho_g\cdot\frac{k(k-1)}{2} = \frac{\rho_0}{k+1}\cdot\frac{k(k-1)}{2} \tag{17}$$

In the above equation, T_g is constant for all the generations. For $k = A$, the total cost becomes:

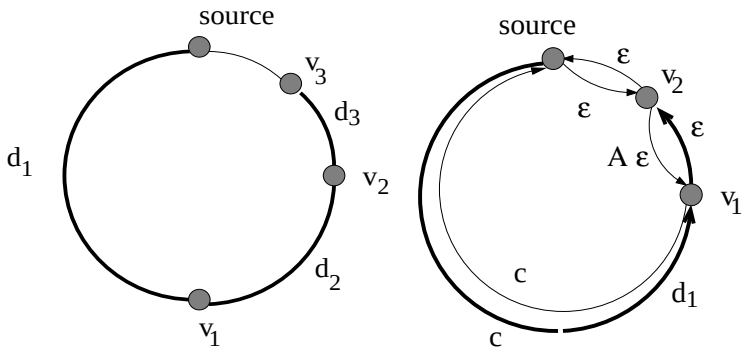
$$T = T_0 + \sum_{g=1}^{g_m} T_g = \rho_0 + \frac{\log M}{\log(k+1)}\cdot\frac{\rho_0}{k+1}\cdot\frac{k(k-1)}{2} \Rightarrow$$

$$T = \Omega\left(A\cdot\frac{\log M}{\log(A+1)}\right)\cdot\rho_0 \tag{18}$$

Recall that $OPT = \rho_0$ (eq. 12), and that T is a lower bound for any on-line algorithm. Therefore, equation 18 completes the proof. □

Note that the two lower bounds correspond to different kinds of graphs: $\Omega(M)$ for highly asymmetric graphs ($A > M$) and $\Theta\left(A\cdot\frac{\log M}{\log(A+1)}\right)$ for less asymmetric ones ($A \leq M$). The high asymmetry bound does not hold for less asymmetric graphs and vice versa. Given this, the lower bound for any specific graph is the minimum of the two bounds.

Corollary 4 *(Lower Bound) Given the size of the multicast group, M, and the asymmetry A, there exists a graph and a sequence of join requests such that the*



(a) Undirected and directed graph: $\Omega\left(2^M\right)$.

(b) Directed graph: $\Omega(A)$

Figure 10: Join-Leave: An example of bad case for Greedy.

competitive ratio of any on-line algorithm is

$$\frac{T}{OPT} = \Omega\left(\min\left(M, A{\cdot}\frac{\log M}{\log(A+1)}\right)\right)$$

Observe that the above bound includes the previous bounds for undirected and directed graphs as special cases. For undirected graphs, $A = 1$, the bound becomes $O(\log M)$ [12]. For directed graphs, $A = \infty$, the bound becomes $O(M)$ [22].

Recall the upper bound from Corollary 3, $O(\min(M, A\log M))$. The upper and lower bound differ only by the term $\log(A+1)$ in the second clause which refers to the the case where $A \leq M$. Therefore, we can state the following corollary.

Corollary 5 *For any weighted asymmetric graph with asymmetry A, a multicast group of size M, for the the join-only problem, the Greedy is* $\log(M+1)$*-competitive compared to any* on-line *algorithm.*

Intuitively, even if we consider the best possible on-line algorithm, Greedy will be within a logarithmic factor of the group size. For many practical applications with limited group size, the Greedy algorithm will have guaranteed good performance.

5. The Join-Leave Problem

In this section, we study the case where participants can join and leave anytime during the session. We find a lower bound for the Greedy algorithm. First, we improve the lower bound for the undirected case [12], which applies in the directed case as well. Then, we prove that the competitive ratio of Greedy can be proportional to the asymmetry. Combining the two lower bounds, we conclude that Greedy can lead to very poor multicast trees in asymmetric graphs.

First, we consider undirected graphs. Recall that we denote the number of join requests by $M-1$. The competitive ratio of the Greedy algorithm is proven to be

as large as M [12]. Here, we prove that the competitive ratio can be exponential in M.

Theorem 8 *Given the number of join requests, M, there exists an undirected graph and a sequence of join-leave requests such that the competitive ratio of the Greedy algorithm is*

$$\frac{T}{OPT} = \Omega\left(2^M\right)$$

PROOF. Let us suppose that the network is a cycle of M nodes $v_0, ..., v_{M-1}$ in that order, and let $v_0 = s$ be the source (see Fig. 10(a)). Assume that the sum of all the weights of the cycle is C. Assume that the weights of the edges between nodes $d_i = e(v_i, v_{i-1}), i = 1, ..., M-1$ decrease by half as follows: $d_1 = \frac{C-\epsilon}{2}$, $d_2 = \frac{C-\epsilon}{4}$ and in general $d_i = \frac{C-\epsilon}{2^i}$, where ϵ is an arbitrarily small constant. The purpose of ϵ is to bias node v_i to be closer to v_{i-1} than the source s: $d(v_{i-1}, v_i) < d(s, v_i), \forall i > 2$.

We distinguish two phases: the joining and the leaving phase.

Joining Phase. Assume that the nodes join in order of increasing index: v_1, ..., v_{M-1}. In the Greedy algorithm, the first node will join to the source. Given the way we set the weights, each new destination joins to the destination that joined just before, e.g., v_i joins to $v_{i-1}, i = 1, ..., M-1$.

Leaving Phase. We can assume that all destinations leave the session except for v_{M-1}. So, in the Greedy tree, we have the destination v_{M-1} connected to just the source along the longest part of the cycle: $T = C - d(s, v_{M-1})$. The optimal tree consists of the edge between the source and the last node in the tree v_{M-1}: $OPT = d(s, v_{M-1})$.

By taking the ratio of the costs of the two trees we get:

$$\frac{T}{OPT} = \frac{C}{d(s, v_{M-1})} - 1$$

For $\epsilon \to 0$, we have: $d(s, v_{M-1}) = C - \sum_{i=1}^{M-1} d_i = C - \sum_{i=1}^{M-1} \frac{C-\epsilon}{2^i}$ and finally, $d(s, v_{M-1}) \simeq \frac{C}{2^{M-1}}$, which concludes the proof. □

The bound of theorem 8 holds even for directed graphs, since an undirected graph corresponds to a directed graph of asymmetry $A = 1$

Let us consider how asymmetry can create a bad case for Greedy without the need for many destinations. Furthermore, we prove that the competitive ratio can be proportional to the asymmetry A.

Theorem 9 *Given a number of join requests, M, and an asymmetry, A, there exists a directed graph and a sequence of join-leave requests such that the competitive ratio of the Greedy algorithm is*

$$\frac{T}{OPT} = \Omega\left(A\right)$$

PROOF. Intuitively, using the asymmetry, we do not need a large M to create the previous worst case example. Let us consider the source and two destinations v_1, v_2 (see fig. 10(b)). The weights of the edges between v_1 and the source are c both

ways. The weights of the edges between v_2 and the source are ϵ. The asymmetry appears only between v_1 and v_2:

$$e(v_1, v_2) = \rho - \epsilon, \qquad e(v_2, v_1) = A \cdot (\rho - \epsilon), \qquad \rho > \epsilon, \;\; \rho, \epsilon \in \mathcal{R}^+ \tag{19}$$

We choose c to be of lower cost than the path from the source to v_1 that includes v_2:

$$c = A \cdot (\rho - \epsilon) \tag{20}$$

Joining Phase. Node v_1 joins first and connects directly to the source (s, v_1) according to Greedy. When connecting v_2, Greedy chooses edge (v_1, v_2), which is less costly than (s, v_2).

Leaving Phase. We remove destination v_1. The cost of the Greedy tree is:

$$T = c + (\rho - \epsilon) \stackrel{eq.20}{\Longrightarrow} T = (A + 1) \cdot (\rho - \epsilon) \tag{21}$$

Clearly, the optimal solution is:

$$OPT = \rho \tag{22}$$

For $\epsilon \to 0$, the competitive ratio becomes:

$$\frac{T}{OPT} = A + 1$$

□

The significance of the above theorem is that when the asymmetry is unbounded, the competitive ratio of Greedy is also unbounded.

Both previous bounds are applicable for any asymmetry value, since both examples apply to any directed graph without any assumptions about the asymmetry. Thus, the combined lower bound is the maximum of the two bounds.

Corollary 6 *The competitive ratio of the Greedy algorithm in the on-line join-leave problem on directed graphs is*

$$\frac{T}{OPT} = \Omega\left(\max(A, 2^M)\right)$$

6. Conclusions

In this paper, we prove bounds for the worst case performance of Greedy and Shortest Paths for the multicast routing problem. We study the effect of the asymmetry and the membership, which have been neglected up to now. With our definition of asymmetry, directed and undirected graphs can be treated in a uniform way.

In general, the Greedy has lower upper bounds than Shortest Paths for the static and the join-only problem. However, for the join-leave problem, the opposite is true: the competitive ratio of Greedy is unbounded (equal to $A = \infty$), while that of Shortest Paths is not affected by the early departures. The reason is that Greedy makes the best choice given an existing tree, but as destinations leave, the choice

Table 1. The worst case bounds of Greedy and Shortest Paths.

	Undirected			Directed		
	Static	Join	Leave	Static	Join	Leave
Sh. Paths	M	M	M	M	M	M
Greedy Up	2	$\log M$	–	$\min(A, M)$	$\min(M, A \cdot \log M)$	–
Greedy Low	2	$\log M$	2^M	$\min(A, M)$	$\min(M, A \cdot \frac{\log M}{\log A+1})$	$\max(A, 2^M)$

can turn out to be bad. On the other hand, Shortest Paths routes each destination independently of the other destinations.

We present the bounds for the Greedy and the Shortest Paths algorithms for all three problems in table 1. The first two columns present previously known bounds, and the rest of the bounds are proven in this paper. Note that the previous bounds are special cases of our bounds for $A = 1$, as we have already mentioned. The bounds for the Shortest Paths are tight, while for the Greedy algorithm, we have separate lines for the upper and lower bounds. The asymptotic bounds of the table do not hide "big constants": in most cases a factor of two is missing and all the logarithmic functions have a base of two.

Several interesting observations become apparent from table 1. First, the Greedy bounds on asymmetric graphs is a factor of A higher than that of the undirected case in most cases. Second, the Greedy competitive ratio is bounded by M for any (even unbounded) asymmetry in the static and join-only problem. Finally, Greedy performs poorly in the join-leave problem: the bound can become as large as the asymmetry or exponential in the number of joins.

Our work leads to the following general conclusions regarding the effect of asymmetry and membership behavior.

- Asymmetry degrades the performance of Greedy by a multiplicative factor A. Therefore, network modeling and simulation should use asymmetric graphs.
- The Greedy algorithm has near-optimal on-line performance for the join-only problem. We show that Greedy is $\log(M + 1)$-competitive compared to *any* on-line algorithm.
- In the join-leave problem, it is the leave requests that hurt the performance of Greedy. For this reason, we should be cautious in problem settings with intense membership activity.

Future Work. For the join-only problem, for less asymmetric graphs, we see a discrepancy between the lower and upper bound of the Greedy algorithm. It would be interesting to prove a tight bound for this case.

Acknowledgments

The authors would like to thank Yossi Azar for suggesting simplifications to the proof of Theorem 5. The authors are grateful to Panayotis Tsaparas, Allan Borodin, and Anindo Banerjea whose criticism and keen comments improved the paper significantly. Special thanks to Christos Faloutsos for his suggestions.

References

1. A. Ballardie. Core Based Trees (CBT: An architecture for scalable inter-domain multicast routing. *ACM SIGCOMM*, 1993.
2. F. Bauer and A. Varma. ARIES: A rearrangable inexpensive edge-based on-line Steiner algorithm. *IEEE Journal of Selected Areas in Communications*, 15(13):382–397, April 1997.
3. K. Bharath-Kumar and J.M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Trans. on Communications*, 31:343–351, 1983.
4. K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. *ACM Computer Communication Review*, pages 5–11, January 1997.
5. M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed steiner problems. *ACM/SIAM Proc. of Symposium on Discrete Algorithms (SODA)*, pages 192–200, January 1998.
6. M. Doar and I. Leslie. How bad is naive multicast routing? *Proc. IEEE INFOCOM*, pages 82–89, 1993.
7. S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.
8. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, F. Sharma, and L. Wei. The PIM architecture for wide area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–161, April 1997.
9. M. Faloutsos, A. Banerjea, and R. Pankaj. QoSMIC: a QoS Multicast Internet protoCol. *ACM SIGCOMM*, Sep 2-4, Vancouver BC 1998.
10. M. Faloutsos, R. Pankaj, and K. C. Sevcik. Multicasting in directed graphs. *18th Biennial Symposium on Communications, June 2-5*, pages 29–32, 1996.
11. M. Faloutsos, R. Pankaj, and K. C. Sevcik. Bounds for the on-line multicast problem in directed graphs. *Proceedings of 4th International Colloquium on Structural Information and Communication Complexity (SIROCCO '97), Monte Verita', Ascona, Switzerland July 24-26*, pages 81–98, 1997.
12. M. Imase and B.M. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4:369–384, 1991.
13. J. Kadirire and G. Knight. Comparison of dynamic multicast routing algorithms for wide-area packet switched (ATM) networks. *Proc. IEEE INFOCOM*, pages 212–219, 1995.
14. L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica, Springer-Verlag*, 15:141–145, 1981.
15. T. Maufer and C. Semeria. Introduction to IP multicast routing. Internet-Draft: draft-ietf-mboned-intro-multicast-02.txt, available from ftp://ftp.ietf.org/internet-drafts/, 1997.
16. K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, pages 125–128, 1988.
17. M. Parsa and J. J. Garcia-Luna-Aceves. A protocol for scalable loop-free multicast routing. *IEEE Journal of Selected Areas in Communications*, 15(13):316–331, April 1997.
18. J. Plesnik. Heuristics for the Steiner problem in graphs. *Discrete Applied Mathematics*, pages 451–463, 1992.
19. S. Ramanathan. An algorithm for multicast tree generation in networks with asymmetric links. *Proc. IEEE INFOCOM '96*, pages 337–344, 1996.

20. V. J. Rayward-Smith. The computation of nearly minimal Steiner trees in graphs. *International Journal of Mathematical Education in Science and Technology*, 14(1):15–23, 1983.

21. H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

22. S. Voss. Worst case performance of some heuristics for Steiner's problem in directed graphs. *Information Processing Letters*, 48:99–105, 1993.

23. L. Wang and T. Jiang. An approximation scheme for some Steiner tree problems in the plane. *Networks*, 28:187–193, 1996.

24. B. M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communications*, pages 1617–1622, 1988.

25. B. M. Waxman. Performance evaluation of multipoint routing algorithms. *Proc. IEEE INFOCOM*, pages 980–986, 1993.

26. J. Westbrook and D. Yan. Greedy algorithms for the on-line Steiner tree and generalized Steiner problems. *Mathematical Systems Theory*, 28:451–468, 1995.

27. P. Winter. Steiner problem in networks: a survey. *Networks*, 17:129–167, 1987.

28. R. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

29. Y.F. Wu, P. Widmayer, and C.K. Wong. A faster approximation algorithm for the steiner problem in graphs. *Acta Informatica*, 23:223–229, 1986.

30. A. Zelikovsky. Better approximation bounds for the network and euclidean steiner tree problems. Technical Report Tech. Rep. CS-96-06, University of Virginia, Charlottesville, 1996.

Appendix A: The Greedy Algorithm

We present a description of the Greedy algorithm. Sets of join and leave requests arrive in an on-line fashion. In our paper, we assume that each set contains one request in the on-line case. The static problem corresponds to the case where we have one set of join requests.

1. START from the source.
2. REPEAT for each new set of requests (on-line)
   (a) Leave requests: Prune the unwanted parts of the tree
   (b) Join requests: REPEAT for all Join requests:
      i. SELECT among non-accomodated requests the one whose destination is closest to the multicast tree, d_{new}.
      ii. JOIN d_{new} to the tree with the related minimum cost path.