Bandwidth Maximization in Multicasting

Naveen Garg* Rohit Khandekar* Keshav Kunal* Vinayaka Pandit†

April 2, 2003

Abstract

We formulate bandwidth maximization problems in multicasting streaming data. Multicasting is used to stream data to many *terminals* simultaneously. The goal here is to maximize the bandwidth at which the data that can be transmitted satisfying the capacity constraints on the links. A typical network consists of the *end-hosts* which are capable of duplicating data instantaneously, and the *routers* which can only forward the data. We show that if one insists that all the data to a terminal has to come along a single path, then it is NP-hard to approximate the maximum bandwidth to a factor better than 2. We also present a fast 2-approximation algorithm. If different parts of the data to a terminal can come along different paths, the problem can be approximated to the same factor as the minimum Steiner tree problem on undirected graphs. We also prove that in case of a tree network, both versions of the bandwidth maximization problem can be solved optimally in polynomial time. Of independent interest is our result that the minimum Steiner tree problem on tree-metrics can be solved in polynomial time.

1 Introduction

Multicasting is a useful method of efficiently delivering the same data to multiple recipients in a network. The IP layer has long been considered the natural protocol layer to implement multicast. However concerns related to scalability, deployment etc. continue to dog it. In this context, researchers [5, 7, 8, 2, 3] have proposed an alternative architecture where multicast functionality is supported at the application layer by end-hosts. In the application layer multicast, data packets are replicated only at end-hosts and not at the routers in the network. Conceptually, end-hosts are viewed as forming an overlay network for supporting multicast.

One therefore needs to pick a suitable overlay network so as to minimize the performance penalty incurred due to duplicate data packets and long delays. The different application layer multicast schemes proposed either build a tree [3, 8] or a mesh [2, 5] as the overlay network. In this paper we will consider the setting where we are multicasting streaming data over a tree overlay network and the metric of interest to us is the throughput/bandwidth of the overlay network.

We model the network as an undirected graph on two kinds of nodes called *end-hosts* and *routers*. The end-hosts are capable of generating and replicating the traffic while the routers are used for forwarding the data. We assume that the terminals, the nodes that are required to receive the data, are end-hosts. The multicasting is done via a *multicast tree* in which the root first sends the

*Department of Computer Science & Engineering, Indian Institute of Technology, New Delhi 110016, India. Email: {naveen,rohitk,keshav}@cse.iitd.ernet.in. Rohit Khandekar is supported in parts by a fellowship from Infosys Technologies Ltd.,Bangalore.

†IBM India Research Lab, Block I, Indian Institute of Technology, New Delhi 110016, India. Email: pvinayak@in.ibm.com

streaming data to (one or more) end-hosts. These end-hosts, in turn, make multiple copies of the data instantaneously and forward it to other end-hosts. This continues till each terminal receives the data. Each link between a pair of nodes has a capacity which is the maximum bandwidth it can support in either directions put together. The objective is to maximize the bandwidth that can be routed to the terminals respecting the link capacities.

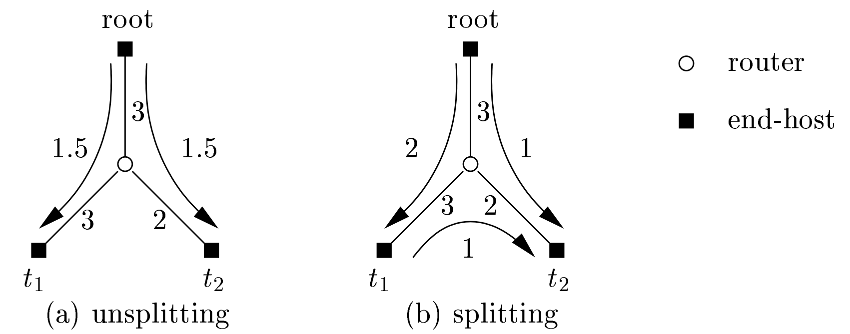

Figure 1: The numbers on the links denote their capacities while the ones on the paths denote the bandwidth routed. (a) A bandwidth of 1.5 each is routed to t_1 and t_2 directly; (b) A bandwidth of 2 and 1 is routed to t_1 and t_2 resp. and t_1 forwards a bandwidth of 1 to t_2

We consider many variants of the bandwidth maximization problem. In the *unsplittable* bandwidth maximization problem (UBM), one desires that the data to any terminal be routed via a single path from the root. On the other hand, in the *splittable* bandwidth maximization problem (SBM), different parts of the data to a terminal can arrive along different paths from the root. Figure 1 shows an example of a tree network in which only a bandwidth of 1.5 units can be routed to the terminals in an unsplitting manner while a bandwidth of 2 units can be routed in a splittable manner. We prove that even if all end-hosts are terminals, it is NP-hard to approximate UBM to a factor less than 2. We prove this by reducing the Hamiltonian cycle problem on 3-regular graphs which is NP-hard [4]. On the positive side, we present a very simple and efficient 2-approximation algorithm for UBM. For the splittable case, we first observe that SBM is equivalent to finding a maximum fractional packing of the *multicast trees* and prove that there is an α-approximation for SBM if and only if there is an α-approximation for the minimum Steiner tree problem on undirected graphs. Our technique is similar to that of Jain et al. [6] who prove that there is an α-approximation for maximum fractional packing of Steiner trees if and only if there is an α-approximation for the minimum Steiner tree problem on undirected graphs. Our results imply that if all the end-hosts are terminals, then SBM can be solved in polynomial time.

We consider yet another variant of the bandwidth maximization problem in which between any pair of end-hosts, the data can be routed only through a pre-specified set of paths. We can think of UBM and SBM problems with this restriction. We prove that SBM with this restriction can also be approximated to within the Steiner ratio, while if all end-hosts are terminals, then it can be solved in polynomial time.

If the network is a tree, UBM and SBM can be solved in polynomial time. First we reduce UBM to the problem of deciding whether a particular bandwidth can be routed to the terminals respecting the link capacities. This problem in turn is solved using dynamic programming to keep track of paths of the multicast tree that go in or out of various sub-trees. Overall, UBM can be solved in $O(n \log n)$ time on a tree on n nodes. We observe that, as in the case of general graphs, SBM can be reduced to solving the minimum Steiner tree problem on tree-metric. We then present a polynomial time algorithm to solve the minimum Steiner tree problem on a tree-metric. This result is of independent interest. Since there is a unique simple path between any pair of end-hosts on the tree network, the pre-specified-path version is identical to the unrestricted version.

A summary of the results obtained in this paper follows.

		SBM		UBM	
		$T \subset H$	$T = H$	$T \subset H$	$T = H$
General Graph Networks	Arbitrary path	$\mathcal{S}$-approx	Polytime	2-approx $(2-\epsilon)$-hard	
	Prespecified paths	$\mathcal{S}$-approx	Polytime	?	?
Tree Networks	Arbitrary path $\equiv$ Prespec. paths	Polytime		Polytime	

Here $\mathcal{S}$ denotes the factor by which the minimum Steiner tree problem can be approximated. The set of end-hosts is denoted by H, and the set of terminals by T. The question marks indicate the open problems.

The rest of the paper is organized as follows. We define the problem formally in Section 2. In Section 3, we address the SBM problem and show its equivalence with the minimum Steiner tree problem. Section 3.1 presents a polynomial time algorithm for the minimum Steiner tree problem on tree metrics which, in turn, implies a polynomial time algorithm for SBM on tree networks. In Section 4.1, we give an $O(n \log n)$-time algorithm for solving UBM optimally on tree networks with n nodes. We prove that it is NP-hard to approximate UBM on general networks to a factor less than 2 in Section 4.2, while Section 4.3 gives a 2-approximation for the same.

2 Problem Formulation

In this section, we formulate the bandwidth maximization problem for multicasting streaming data. We are given an undirected graph $G = (H \cup R, E)$ in which the node-set is a disjoint union of the *end-hosts* H and the *routers* R. We call each element in E as a *link*. Each link $e \in E$ has *capacity* c_e which is the maximum bandwidth it can carry in both directions put together. We are also given a special node $r \in H$ called the *root*, and a set $T \subseteq H$ of *terminals*. Our goal is to multicast the same streaming data from r to all the terminals $t \in T$. The routers can forward data from an incoming link to an outgoing link; they cannot, however, make multiple copies of the data. The end-hosts can perform the function of routers as well as make duplicate copies of the data instantaneously.

Let $K_{|H|}$ be a complete graph on all end-hosts. Let Q be a tree in $K_{|H|}$ rooted at r, which spans T. Conceptually, the data is routed from the root to the terminals via the tree Q. In the unsplittable bandwidth maximization problem, an edge $(h, h') \in Q$ (where h is the parent of h') is realized in the actual network by a path $P_{hh'}$ from h to h' in G. Formally,

Definition 2.1 *A multicast tree is a pair $M = (Q, P)$ where*

- *Q is a tree in $K_{|H|}$ rooted at r which spans $\{r\} \cup T$*
- *P is a mapping of each edge $(h, h') \in Q$, where h is a parent of h', to a unique path $P_{h,h'} \subset G$ from h to h'.*

For a link $e \in E$, let $p_{e,M} = |\{(h,h') \in Q | e \in P_{hh'}\}|$ denote the number of paths of M that go through $e \in E$. Observe that $p_{e,M} \leq |H| - 1$ because Q contains at most $|H| - 1$ edges. Clearly, if a bandwidth of λ is to be routed using M, the capacity constraint of each link $e \in E$ enforces that $\lambda \leq c_e/p_{e,M}$. We denote the maximum bandwidth which can be routed using M by $\lambda_M = \min_{e \in E} c_e/p_{e,M}$.

Unsplittable Bandwidth Maximization Problem (UBM)

- **Input:** $G = (H \cup R, E)$, $c : E \to I\!R^+$, $r \in H$ and $T \subseteq H$.
- **Output:** A multicast tree $M = (Q, P)$.
- **Goal:** To maximize the bandwidth λ_M

Consider a single packet in the splittable bandwidth maximization problem. It reaches each terminal through a path from r. The union of the paths of a single packet to the terminals defines a multicast tree. In SBM, the bandwidth maximization can be thought of as packing as many multicast trees as possible without exceeding the capacity of any link. For each multicast tree M, we associate $\sigma_M \geq 0$ which denotes the bandwidth routed via M.

Splittable Bandwidth Maximization Problem (SBM)

- **Input:** $G = (H \cup R, E)$, $c : E \to I\!R^+$, $r \in H$ and $T \subseteq H$.
- **Output:** An assignment $\sigma_M \geq 0$ of the bandwidth to each multicast tree M such that for any link $e \in E$, we have the following capacity constraint satisfied: $\sum_M \sigma_M p_{e,M} \leq c_e$.
- **Goal:** To maximize the total bandwidth routed: $\sum_M \sigma_M$.

3 Splittable Bandwidth Maximization

The SBM problem on general graphs can be viewed naturally as a linear program. For each multicast tree M, we associate a real variable $\sigma_M \geq 0$ to indicate the bandwidth routed through M.

$$\begin{array}{lrcll} \max & \sum_M \sigma_M & & & \\ \text{s.t.} & & & & \\ & \sum_M p_{e,M}\sigma_M & \leq & c_e & \text{for all } e \in E \\ & \sigma_M & \geq & 0 & \text{for all } M \end{array}$$

This linear program can be viewed as a fractional packing of multicast trees. Since this linear program has exponentially many variables, we consider its dual.

$$\begin{array}{lrcll} \min & \sum_{e \in E} c_e l_e & & & \\ \text{s.t.} & & & & \\ & \sum_{e \in M} p_{e,M} l_e & \geq & 1 & \text{for all } M \\ & l_e & \geq & 0 & \text{for all } e \in E \end{array}$$

The separation problem for the dual program is: Given a length assignment l on the links $e \in E$, determine if there exists a multicast tree M such that $\sum_{e \in M} p_{e,M} l_e < 1$. This can be done by computing the minimum length multicast tree $M^* = (Q^*, P^*)$ for the given length function, and comparing the length of M^* with 1. Observe that for any edge $(h, h') \in Q^*$, the length of the corresponding path $P^*_{h,h'}$ will be equal to the length of the shortest path (among the set of specified paths) between h and h' in G under length l. Thus to compute M^*, we consider the complete graph $K_{|H|}$ on the end-hosts and assign the edge (h, h') a length equal to the length of the shortest path (under lengths l) between h and h' in G. The tree Q^* is now the minimum Steiner tree spanning $\{r\} \cup T$.

Note that the metric on H is defined by shortest paths in graph G. We call such a metric, a G-metric. Formally, for a graph G and a subset H of nodes of G, a metric d on H is called a G-metric if there is a length function l on the links of G such that for any $h, h' \in H$, $d(h, h')$ equals the length of the shortest path between h and h' in G under the length function l. The following theorem holds.

Theorem 3.1 *Let $G = (H \cup R, E), r \in H, T \subset H$ be an instance of SBM. There is a polynomial time α-approximation for SBM on this instance with any capacity function on E if and only if there is a polynomial time α-approximation for the minimum Steiner tree problem on H with a G-metric and $\{r\} \cup T$ as the terminal set.*

The proof of this theorem is similar to the one given by Jain et al. [6] for the equivalence between α-approximations for maximum fractional Steiner tree packing and for minimum Steiner tree problem and is therefore omitted. The above theorem has many interesting corollaries.

Corollary 3.2 *If all end-hosts are terminals, i.e., $T = H$, then SBM can be solved optimally in polynomial time.*

A metric is called a tree-metric if it is a G-metric for some tree G. The following theorem which states that tree-metrics are "easy" for the minimum Steiner tree problem may be of independent interest.

Theorem 3.3 *The minimum Steiner tree problem can be solved optimally on a tree-metric.*

The proof of the above theorem is given in the following section. The theorem above combined with Theorem 3.1 gives the following corollary.

Corollary 3.4 *SBM can be solved optimally on a tree network.*

3.1 Minimum Steiner tree Problem on Tree-Metric

In this section, we prove that the classical minimum Steiner tree problem can be solved optimally on a tree-metric. Given a metric, there are methods known to identify if the metric is a tree-metric and if yes, to construct a weighted binary tree with the points in the metric as leaves that induces the given metric (see [9, 1]). Before we present our algorithm for finding a minimum Steiner tree in tree-metric, we describe a transformation called "load minimization" that will be useful in the analysis.

3.1.1 Load Minimization

Consider a tree network $G = (V, E)$ with L as the set of leaves. We are also given a set $T \subseteq L$ of terminals and a root $r \in T$. The tree G is considered rooted and hanging at r. Consider a multicast tree $M = (Q, P)$ that spans $\{r\} \cup T$. The edges of Q are considered directed away from r. For each edge $(u, v) \in Q$, there is a unique path $P_{u,v}$ directed from u to v in G. For a link $e \in G$, let $G_e \subseteq G$ be the subtree (including e) hanging below link e. Let d_e be the number of paths going into G_e and f_e be the number of paths coming out of G_e. We now describe a transformation called "load minimization" that modifies the multicast tree M to ensure that

- either $d_e = 1$ or $f_e = 0$,
- the tree Q still spans the same set of leaves, and
- the number of paths going through any edge in G does *not* increase.

If $d_e = 0$ then clearly there are no terminals in G_e and $f_e = 0$. Suppose, now, that $d_e > 1$ and $f_e > 0$. There are two cases: (1) $d_e \geq f_e$ or (2) $d_e < f_e$. In the first case, after the transformation, the new values of d_e and f_e would be $d_e - f_e$ and 0 respectively, while in the second case, the new values of d_e and f_e would be 1 and $f_e - d_e + 1$ respectively.

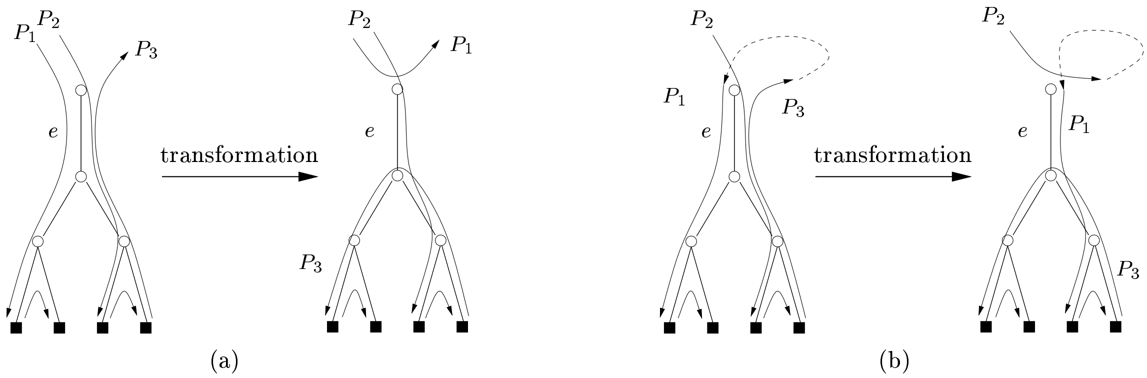

Figure 2: Applying the transformation "load minimization": (a) P_2 is *not* an ancestor of P_1, (b) P_2 is an ancestor of P_1

We first describe the transformation for the case when $d_e = 2$ and $f_e = 1$. There are two paths, say P_1 and P_2, coming in G_e and one path, say P_3, going out of G_e. (Refer to Figure 2.) Let f_1, f_2 and f_3 be the edges of Q that correspond to the paths P_1, P_2 and P_3 respectively. There are two cases:

(a) f_1, f_2 do not have an ancestor-descendant relationship in Q. Note that f_3 is a descendant of one of f_1, f_2. Suppose it is a descendant of f_2. We now change paths P_1, P_3 as follows. The new path P_1 is obtained by gluing together the parts of P_1 and P_3 in $G \setminus G_e$. The new path P_3 is obtained by gluing together the parts of P_1 and P_3 in $G_e \setminus \{e\}$.

(b) $f_2 \in Q$ is an ancestor of $f_1 \in Q$. f_e is then a descendant of f_2 and an ancestor of f_1. In this case, we change paths P_1, P_2, P_3 as follows. The new path P_1 is same as the path P_2 in G_e. The new path P_2 is obtained by gluing the parts of P_2 and P_3 in $G \setminus G_e$. The new path P_3 is obtained by gluing together the parts of P_1 and P_3 in $G_e \setminus \{e\}$.

In both the cases, it is easy to see that this transformation gives a multicast tree $M' = (Q', P')$ with Q' spanning the same set of leaves as Q and $d_e = 1$ and $f_e = 0$. Furthermore, the number of paths of M' going through any link in G is no more than the number of paths of M going through that link. For other values of d_e and f_e, we can apply this transformation by considering two incoming paths and a suitable outgoing path (the edge in Q corresponding to the outgoing path should be a descendant of one of the edges corresponding to the incoming paths). Since by applying the transformation, we reduce both d_e and f_e, we can apply it repeatedly till either $d_e = 1$ or $f_e = 0$.

3.1.2 Algorithm for minimum Steiner tree on tree-metric

Now we describe our dynamic programming based algorithm for finding a minimum Steiner tree on the terminals $T \subseteq L$. We assume, without loss of generality, that the tree underlying the metric is a binary tree. This can be achieved by adding zero-length links, if necessary. Let M^* be the multicast tree that corresponds to the minimum Steiner tree. Note that the transformation of "load minimization" that is described in Section 3.1.1 does not increase the number of paths going through any link. Therefore by applying this transformation to M^*, we get another minimum Steiner tree. Thus we can assume that the minimum Steiner tree satisfies that $d_e = 1$ or $f_e = 0$ for each link $e \in G$. Note that if $n = |L|$ is the number of leaves in G, the number of edges in any Steiner tree is at most $n-1$. Thus, the possible values of (d_e, f_e) are $\mathcal{F} = \{(1,0), \dots, (1, n-2), (2,0), \dots, (n-1, 0)\}$.

The algorithm finds, for any link $e \in G$ and for any possible value of (d_e, f_e), the minimum-length way of routing d_e paths into and f_e paths out of G_e while covering all the terminals (and possibly some non-terminals) in G_e. We denote this quantity by $L_e(d_e, f_e)$. It is easy to find such an information for the leaf links. To find such information for a non-leaf link $e \in G$, we use the information about the child links e_1 and e_2 of e as follows. We route k_1 paths into T_{e_1} and k_2 paths into T_{e_2} such that $k_1 + k_2 = d_e$. We route k_{12} paths from T_{e_1} to T_{e_2} and k_{21} paths from T_{e_2} to T_{e_1}. We route l_1 paths out of T_{e_1} and l_2 paths out of T_{e_2} such that $l_1 + l_2 = f_e$. Thus, the total number of paths coming in and going out of T_{e_1} is $k_1 + k_{21}$ and $l_1 + k_{12}$ respectively, while the total number of paths coming in and going out of T_{e_2} is $k_2 + k_{12}$ and $l_2 + k_{21}$ respectively. We work with those values of $k_1, k_2, k_{12}, k_{21}, l_1, l_2$ that satisfy $(k_1 + k_{21}, l_1 + k_{12}), (k_2 + k_{12}, l_2 + k_{21}) \in \mathcal{F}$. We modify G so that there only one link, g, incident at the root. The cost of the minimum Steiner tree is then equal to $\min_i(L_g(i, 0) + l_g \cdot i)$ and can be computed in polynomial time.

4 Unsplittable Bandwidth Maximization

4.1 UBM on Tree Networks

In this section, we give an efficient algorithm to solve the unsplittable bandwidth maximization problem optimally when the input graph G is a tree. Every multicast tree M has a bandwidth λ_M associated with it, which is the maximum bandwidth that can be routed through it while satisfying the capacity constraints. We want to find the multicast tree which can route the maximum bandwidth. We solve the decision version of the problem, *"Given a bandwidth B, is it possible to construct a multicast tree M of bandwidth $\lambda_M \geq B$.?"* and use this to search for the maximum value of bandwidth that can be routed.

4.1.1 Decision Oracle

We replace every end-host at an internal node in the input tree G by a router and attach the end-host as a leaf node to the router with a link of infinite capacity. It is easy to see that this modification does not change the maximum bandwidth that can be routed through the tree and the number of nodes (and links) is at most doubled. With each link e, we associate a label $u_e = \lfloor c_e/B \rfloor$, which represents the maximum number of paths of bandwidth B that can pass through e without exceeding its capacity. For simplicity, we will use the term *path* to mean, a path of bandwidth B, in this section. The flow of data from root induces a logical direction in the multicast tree. So, we will use the notion of *incoming* and *outgoing* paths, even though the underlying tree is undirected. Note that the same (undirected) link can carry both *incoming* and *outgoing* paths.

The oracle does the verification in a bottom-up fashion. For a link e, let G_e be the subtree below. We shall compute two quantities, d_e, f_e called demand and feedback of the link respectively. The demand d_e represents the minimum number of paths that should enter the subtree G_e to satisfy the demands of the terminals in G_e. The feedback f_e represents the maximum number of paths that can emanate from G_e when d_e paths enter G_e. From Section 3.1.1, it follows that if there is a multicast tree M with $\lambda_M \geq B$, then there is a multicast tree M^* for which $d_e = 1$ or $f_e = 0$ for every link e in G. Since the transformation does not increase the number of paths going through a link, the maximum bandwidth that can be routed does not decrease implying $\lambda^*_M \geq B$.

It is easy to compute (d_e, f_e) for a link e incident on a leaf node. Let $v \in H \cup R$ be the leaf node on which e is incident.

$$(d_e, f_e) = \begin{cases} (0,0) & \text{if } v \in R \\ (0,0) & \text{if } v \in H - T \text{ and } u_e \leq 2 \\ (1, u_e - 1) & \text{otherwise} \end{cases} \tag{1}$$

If the leaf node is a router we simply ignore it (or equivalently set $(d_e, f_e) = (0,0)$), since the link will never be part of a path between two end-hosts. We do the same if the leaf node is a non-terminal end-host and $u_e \leq 2$, because in this case the end-host can act only as a router for the data which is sent along the link. Otherwise we set $(d_e, f_e) = (1, u_e - 1)$. It implies that if a path comes into the end-host (which is clearly its minimum demand), it can replicate the data and send at most $u_e - 1$ paths through the link. If the feedback computed is negative, which can only happen if the node is a terminal and $u_e = 0$, the oracle outputs 'No".

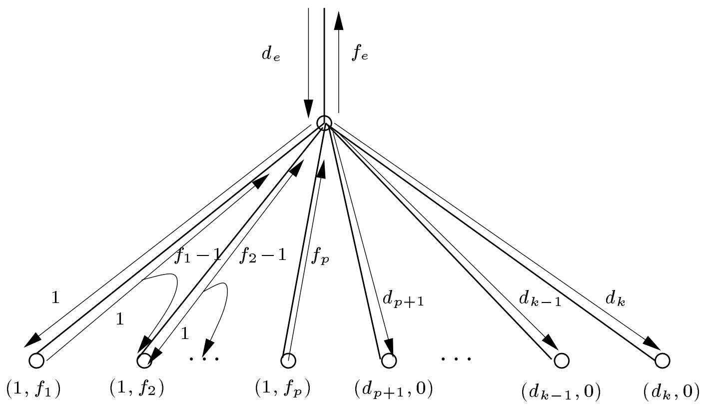

Figure 3: Computing (d_e, f_e) for a link e which has k children. The child links $e_1, e_2 \dots e_p$ have $d_{e_i} = 1$ and $f_{e_i} > 1$, while $e_{p+1} \dots e_k$ have $d_{e_i} \geq 1$ and $f_{e_i} = 0$.

We then compute the values for links incident on internal nodes (routers because of our simplification) in the following manner:

Suppose we have computed (d_{e_i}, f_{e_i}) pairs for the child links $e_1, \dots, e_k$ of a link e. Let $D =$

$\sum_{i=1}^{k} d_{e_i}$, and let $F = \sum_{i=1}^{k} f_{e_i}$. Then

$$(d_e, f_e) = \begin{cases} (D - F, 0) & \text{if } D > F \\ (1, \min\,(u_e - 1, F - D + 1)) & \text{otherwise} \end{cases} \tag{2}$$

The idea is to use only one incoming path to generate all the feedback from child links with positive feedback and use it to satisfy demands of other links with zero feedback. The remaining feedback can be sent up the tree along the link e or if the feedback generated is not enough to meet the remaining demands then the demand for e is incremented accordingly.

Suppose links $e_1, \dots, e_p$ have a positive feedback and $e_{p+1}, \dots, e_k$ have a 0 feedback. Then $d_{e_i} = 1$ for $1 \le i \le p$ and $f_{e_i} = 0$ for $p+1 \le i \le k$. We initialize $(d_e, f_e) = (1, 0)$. We send one path along e_1 and generate f_1 paths. One of these paths is sent along e_2 to generate a further feedback of f_2. We proceed in this manner till we generate a feedback of $\Sigma_{i=1}^{p} f_{e_i}$ out of which $p - 1$ paths are used to satisfy the sub-trees rooted at these p links. Thus, we can use one path coming along link e to meet the demands of $e_1, \dots, e_p$ and generate a surplus feedback of $\Sigma_{i=1}^{p} f_{e_i} - (p-1)$. The demand or feedback of link e is now incremented by the absolute value of the difference of surplus feedback and sum of demands of $e_{p+1}, \dots, e_k$, depending on which of the two terms is greater.

This idea is captured succinctly in the equation (2). Of course the final feedback computed for e can not be more than $u_e - 1$ which explains the *min* expression. Note that our computation ensures that $d_e = 1$ when $f_e > 0$, and $f_e = 0$ when $d_e > 1$.

There is a minor technical detail for the case when we compute $(d_e, f_e) = (1, 0)$. If there is no terminal in the subtree G_e, we need to set $d_e = 0$ because we do not want to send a path into it, unless the feedback is at least 1. This information, whether a subtree contains at least one terminal, can be easily propagated "upwards" in the dynamic computation.

The quantity d_e gives a measure of the minimum capacity required for each link e, and if for some link e, its label $u_e < d_e$, a multicast tree of bandwidth atleast B can not be constructed. If for each link e in the tree, $u_e \ge d_e$, then the oracle outputs "Yes". It is easy to see that the verification algorithm can be easily modified to construct a multicast tree of bandwidth atleast B, when the output is "Yes".

Lemma 4.1 *The bottom-up computation of decision oracle terminates in linear time.*

Proof. No link in the graph is considered more than twice in our traversal, once while computing its demand and feedback and once while computing the values for its parent link. It follows that our algorithm is linear. ∎

4.1.2 Maximizing Bandwidth

It is easy to see that at least one link in a multicast tree goes *tight*, that is, its capacity is completely utilized when the bandwidth routed is maximized. Otherwise, we can increase the bandwidth to utilize the capacity remaining on the link with least residual capacity. Also note that the maximum number of paths that can pass through an link is $|H| - 1$. So, there are $n \cdot (|H| - 1)$ possible values of the form c_e/k for $e \in E, 1 \le k < |H|$, for the optimal bandwidth B^*. We now show how B^* can be found in time $O(n \log n)$ plus $O(\log n)$ calls to the decision oracle. Since our decision oracle runs in linear time, we take $O(n \log n)$ time to compute the maximum bandwidth.

Consider the set of links, $E' = \{e \in E \mid e \text{ lies on the unique path from root to a terminal}\}$. Let c_0 be the capacity of the minimum capacity link in E'. Clearly, c_0 is an upper bound on the maximum achievable bandwidth. If we replace each link e with two anti-parallel links of capacity

$c_e/2$, we can route a bandwidth of $c_e/2$ through the Eulerian tour of this graph. Hence $c_0/2$ is a lower bound on the maximum bandwidth. We also use this idea in Section 4.3, to achieve a 2-approximation for UBM on graphs.

We repeatedly halve the interval $[c_0/2, c_0]$ using binary search, till the length of the interval becomes smaller than $c_0/2|H|^2$. To do this we make $O(\log |H|)$ calls to the oracle. Let I be this final interval containing B^*. We now argue that corresponding to each link e, at most one point of the form c_e/k lies in the interval I. Consider the two cases:

- $c_e < c_0/2$ - No feasible point corresponding to c_e lies in $[c_0/2, c_0]$ and hence in I.
- $c_e \geq c_0/2$ - Consider any value of $k, 1 \leq k < |H|$, such that c_e/k lies in I. For any $k' \neq k$

$$|\tfrac{c_e}{k} - \tfrac{c_e}{k'}| \geq \tfrac{c_e}{|H|-2} - \tfrac{c_e}{|H|-1} > \tfrac{c_e}{|H|^2} \geq \tfrac{c_0}{2|H|^2}$$

Since $c_0/2|H|^2$ is at least the length of the interval I, c_e/k' can not lie in I.

So, we have "filtered" out at most n points, one of which is B^*. We now sort these n points and then make another $\log n$ calls to the oracle to find the value of B^*. As mentioned before, this gives a total running time of $O(n \log n)$.

Theorem 4.2 *There is an $O(n \log n)$-time algorithm to compute the optimal bandwidth multicast tree when the input graph is a tree with n nodes.*

4.2 Hardness results for UBM on general graphs

In this section, we prove the NP-Completeness for UBM on graphs by reducing the Hamiltonian cycle problem on 3-regular graphs to it. In fact, our reduction also gives us a lower bound of 2 on the approximation ratio achievable.

The problem of determining if a given 3-regular graph has a Hamiltonian Cycle is NP-hard [4]. Given an instance of the Hamiltonian Cycle problem on a 3-regular graph, we construct an instance of the bandwidth maximization problem in polynomial time as follows.

Pick an arbitrary vertex s in $G = (V, E)$ and attach two vertices r, t to it. For every other vertex $v_i \in V$, we add three vertices u_i, w_i, z_i and edges $(v_i, u_i), (u_i, w_i), (w_i, z_i)$ and (z_i, v_i). Let $G' = (V', E')$ be the resulting graph (See Figure 4). We designate w_i and r, t as end-hosts and assign a unit capacity to all links in E'.

Lemma 4.3 *The graph G has a Hamiltonian cycle if and only if we can route a bandwidth of more than 1/2 to all end hosts in G' from r.*

Proof. Suppose there is a Hamiltonian cycle C in G. For every edge $(v_j, v_i) \in C$, we add links $(v_j, v_i), (v_i, u_i), (u_i, w_i), (w_i, z_i)$ and (z_i, v_i) to the multicast tree. We also add (r, s) and (s, t) to the multicast tree. It is easy to see that a bandwidth of 1 can be routed through this multicast tree.

Suppose we are able to route a bandwidth of greater than 1/2 through a multicast tree. Note that no link in E' can carry more than one path. Since every $v_i \in G$ has degree 3, at most one path can pass through it. Also, at least one path passes through each v_i as the corresponding terminal w_i has to be visited. So, the multicast tree induces a Hamiltonian cycle in G. ∎

Theorem 4.4 *It is NP-Complete to approximate UBM on graphs to better than 2 even when all links have unit capacity and all end-hosts are terminals.*

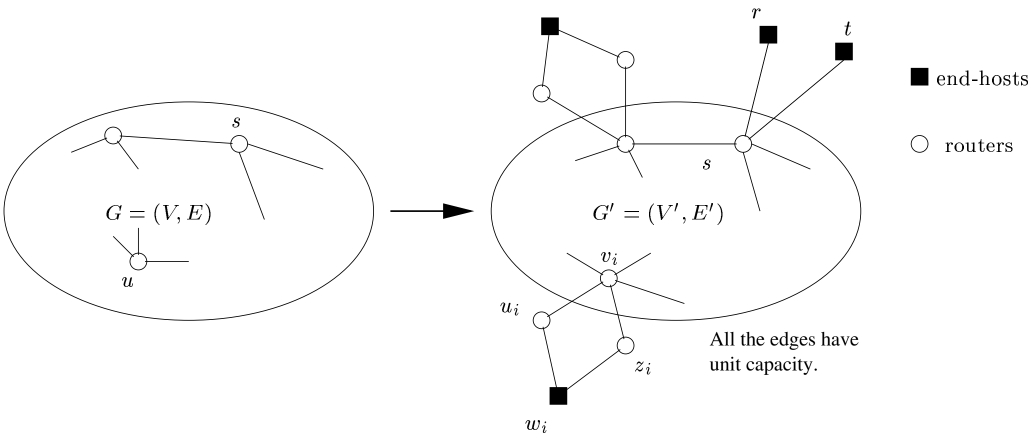

Figure 4: Polynomial transformation of the Hamiltonian Cycle on 3-regular graphs to the bandwidth maximization for multicasting

4.3 A 2-approximation for UBM on general graphs

In this Section, we give a 2-approximation for the bandwidth maximization problem on graphs of arbitrary capacities. Our technique is based on a simple argument for the upper bound on the maximum bandwidth achievable, as used in Section 4.1.2

Lemma 4.5 *Let h be the highest capacity such that links of capacity at least h connect all the terminals in T to the root r. Then, h is an upper bound on the maximum bandwidth.*

Proof. At least one link of capacity h, or lower, is required to connect all the terminals in T to the root. So any multicast tree has to use at least one link, e, of capacity h, or lower, and the paths through e can not carry a bandwidth more than h. ∎

The following algorithm gives a factor two approximation for UBM on general graphs.

Procedure Multicast

1. Find the highest capacity h such that links with capacity at least h connect all the terminals to the root.

2. Let G_h be the subgraph induced by links of capacity at least h. Let S be a tree in G_h spanning the root and all terminals.

3. Replace each link e in S with two anti-parallel links of capacity $c_e/2$. Construct an Eulerian tour of this graph and send a bandwidth of $h/2$ through it.

Theorem 4.6 *There exists a polynomial time 2-approximation algorithm for UBM on graphs.*

Proof. Follows from Lemma 4.5, and the fact that Eulerian tour can be constructed in polynomial time for even degree graphs. ∎

5 Conclusions

We consider the problem of computing an overlay tree so as to maximize the throughput achievable for transmitting streaming data. However, we have ignored the latency incurred in sending the data and it would be interesting to design overlay networks which can take both these aspects into account. We have also left unanswered the question of designing multicast trees when we are permitted to use only a specified set of paths between a pair of end-hosts.

Acknowledgements

We would like to thank Ravi Kannan and Arvind Krishnamurthy for suggesting the problem.

References

[1] K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25:251–278, 1999.

[2] Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastucture Service.* PhD thesis, University of California, Berkeley, 2000.

[3] P. Francis. Yoid: Extending the multicast internet architecture. White paper, 1999. http://www.aciri.org/yoid/.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, San Francisco, 1979.

[5] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast (keynote address). In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12. ACM Press, 2000.

[6] K. Jain, M. Mahdian, and M. Salavatipour. Packing steiner trees. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[7] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th Symposium on Operating System Design and Implementation*, pages 197–212.

[8] Dimitris Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, pages 49–60, San Francisco, CA, USA, March 2001.

[9] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.