Resource Optimization in QoS Multicast Routing of Real-Time Multimedia

EXTENDED SUMMARY

Moses Charikar Joseph (Seffi) Naor Baruch Schieber

Abstract— **We consider a network design problem, where applications require various levels of Quality-of-Service (QoS) while connections have limited performance. Suppose that a source needs to send a message to a heterogeneous set of receivers. The objective is to design a low cost multicast tree from the source that would provide the QoS levels (e.g., bandwidth) requested by the receivers. We assume that the QoS level required on a link is the maximum among the QoS levels of the receivers that are connected to the source through the link. In accordance, we define the cost of a link to be a function of the QoS level that it provides. This definition of cost makes this optimization problem more general than the classical Steiner tree problem. We consider several variants of this problem all of which are proved to be NP-hard. For the variant where QoS levels of a link can vary arbitrarily and the cost function is linear in its QoS level, we give a heuristic that achieves a multicast tree with cost at most a constant times the cost of an optimal multicast tree. The constant depends on the best constant approximation ratio of the classical Steiner tree problem. For the more general variant, where each link has a given QoS level and cost we present a heuristic that generates a multicast tree with cost $O(\min\{\log r, k\})$ times the cost of an optimal tree, where r denotes the number of receivers, and k denotes the number of different levels of QoS required. We generalize this result to hold for the case of many multicast groups.**

Keywords— **Network design, QoS guarantee, Steiner tree, Multicast tree, Generalized Steiner forest.**

Moses Charikar, Computer Science Department, Stanford University, Stanford, CA 94305. Research supported by the Pierre and Christine Lamond Fellowship, NSF Grant IIS-9811904 and NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. E-mail: *moses@cs.stanford.edu*.

Joseph (Seffi) Naor, Bell Laboratories, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974. On leave from the Computer Science Department, Technion, Haifa 32000, Israel. E-mail: *naor@research.bell-labs.com*.

Baruch Schieber, IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. E-mail: *sbar@watson.ibm.com*.

I. INTRODUCTION

THE provision of Quality-of-Service (QoS) guarantees is of utmost importance for the development of future networks. Recent advances in switching and transmission technologies allow the implementation of very high speed networks that carry vast amounts of traffic which is generated by applications that are more sensitive to data quality (such as video or audio), and at the same time less predictable than current fixed rate sources. In the next telecommunication age it will be possible to support new multimedia applications in a global environment and design new services on flexible platforms without upgrading the physical infrastructure. This requires new network architectures capable of offering transport and computation services to communication applications with stringent QoS requirements. A key issue is the provision of network resources so as to meet these requirements.

The multicast backbone of the internet (Mbone) is increasingly used for broadcasting live audio and video in digital form all over the world. However, heterogeneity is an enduring characteristic of the Internet creating difficulties in the transmission of real-time multimedia data across groups. Heterogeneity originates, e.g., from the wide range of network transmission rates, varying across many orders of magnitude, and from the vast differences in computing power. Members of a group (receivers) may vary significantly in their characteristics, e.g., bandwidth availability or computing power. This means that a source would be required to transmit in a way that matches the most constrained receiver. Instead, it would be advantageous to send data to multiple receivers at heterogeneous rates and in a way that matches the capability of each individual receiver. These difficulties are treated in depth by [3], [15]

Bandwidth heterogeneity can be dealt with by adjusting the video and audio stream in a controlled manner that meets the capacity of each link. A source transmits only one signal that is sufficient for the highest bandwidth receiver. The bandwidth of the signal is reduced as it passes through the network. Maxemchuk [15] discusses mechanisms for doing that, either by using a progressive coder [11], [14], or by converting between format encoders such as video gateways [5], [18]. This is also similar to receiver initiated reservations and packet filtering used in the RSVP protocol [20]. Further work in this direction was done by Amir, McCanne and Katz [4] in the development of the SCUBA protocol, and by Amir, McCanne and Zhang [5] through the Active Service framework. Fukuda *et al.* [9] consider multicast video transport to heterogeneous receivers and propose algorithms that are based on aggregating flow corresponding to receivers having similar QoS requirements.

We model multicasting in a heterogeneous environment as a network deign problem, where applications require various levels of QoS while connections have limited performance. The objective is to design a low cost multicast tree that would provide the QoS level requested by the receivers. We assume that the QoS level required on a link is the maximum among the QoS levels of the receivers that are connected to the source through the link. In accordance, we define the cost of a link to be a function of the QoS level that it provides. However, we note that the cost is independent of the utilization of the link. This is a common assumption as indicated in [15].

The multicast tree is essentially a Steiner tree in the network, where the terminals are the receivers. However, the cost function makes the problem more general than the standard Steiner tree problem, where the cost of a link is fixed and does not dependent on its location in the tree. Here, in contrast, the cost of a link is determined by the set of terminals that are connected to the source through it. This means that "local" heuristics for computing a Steiner tree by connecting the terminals to the source one by one will not necessarily yield a low cost solution in this case.

A. The network model

We model the network as an undirected graph $G(V, E)$. The graph G may be a multigraph, i.e., it may contain parallel links. Let $s \in V$ denote the source and let $X \subset V$ denote the terminals or receivers.

We first define our general model which we call the *priority model*. Our network has k *priorities* (or QoS levels) which are denoted by $\{1 \dots k\}$, where 1 is the highest priority and k is the lowest priority. We associate a priority with each link $e \in E$, denoted by $p(e)$. The priority of a link denotes its QoS capability, e.g., its bandwidth. We associate a priority with each terminal $t \in X$, denoted by $p(t)$. The priority of a terminal denotes the QoS level that it requires. For a link $e \in E$, denote by $c(e)$ the cost associated with e. A feasible Steiner tree (multicast tree) T in this model is a tree rooted at the source s spanning all the terminals in X, such that the path in T from s to each $t \in X$ contains only links which have priority at least as high as $p(t)$, i.e., for each link e in the path to t, $p(e) \leq p(t)$. The cost of T, denoted by $c(T)$, is defined to be the sum of the costs of the links in T.

Note that we assume in this model that the cost of a link is fixed. This assumption is valid since the network is modeled by a multigraph, and we can assume that parallel links have different priorities. For a given tree, the appropriate link (among a set of parallel links) is chosen depending on the set of terminals that connect to the source through it. The objective is to find a minimum cost tree among all feasible Steiner trees.

Maxemchuk [15] defined a network design model in the context of a heterogeneous network which we call the *rate model*. In the rate model, we associate a rate with each terminal $t \in X$, denoted by $r(t)$. Let q denote the number of distinct rates. There is a *basic cost* c_e associated with each link $e \in E$. Let T be a given Steiner tree that spans all the terminals in X. For a link $e \in T$, let r_e denote the maximum rate among all terminals connecting to s through e in T. The cost of T is defined to be $\sum_{e \in T} r_e \cdot c_e$.

It is not hard to see that the rate model is a special case of the priority model. Define a priority for each rate value. For each link e, replace it by parallel copies, one for each rate value. The cost of the copy corresponding to rate r is defined to be $r \cdot c_e$.

Our algorithms use a heuristic for computing a low cost (standard) Steiner tree. We denote by α_{ST} the approximation factor of the heuristic chosen. The simple Steiner tree heuristics (see, e.g., [19]) achieve an approximation factor of 2. Alternatively, the current best value known for α_{ST} is by Robins and Zelikovsky [17] (following a long line of work) who suggested a heuristic with approximation factor of $1 + \frac{\ln 3}{2} \approx 1.55$. We use $\textsc{Steiner}(n)$ to denote the time complexity of the Steiner tree heuristic used. The running times of our algorithms will be expressed in terms of $\textsc{Steiner}(n)$. If we use the simple spanning tree heuristic with approximation factor 2, the running time is $O(m + n \log n)$, where n is the number of nodes and m is the number of links in the graph. On the other hand, the heuristic achieving the currently best known approximation factor has a very high running time. The algorithm uses a parameter θ and the running time is $O(n^{f(\theta)})$ for some increasing function $f(\theta)$. The algorithm achieves its stated approximation guarantee in the limit as $\theta \to \infty$.

B. Our results

We present efficient polynomial-time approximation algorithms for the minimum cost Steiner tree problem in several models. For a given optimization problem (say minimization), an algorithm is said to achieve an approximation factor of r, if for every instance of the problem, the algorithm is guaranteed to produce a solution with cost is no more than r times the cost of an optimal solution.

We first consider the minimum cost Steiner tree problem in the rate model. This problem was considered previously by Maxemchuk [15]. He presented a heuristic for computing a minimum cost Steiner tree, but provided only experimental evidence for its performance. We present a heuristic for this problem and show that the cost of the Steiner tree that it generates is no more than $e \cdot \alpha_{ST}$ times the cost of an optimal tree, where e is the basis of the natural log ($e \approx 2.7182$). Plugging in $\alpha_{ST} = 1.55$ [17], we get that the cost of the multicast tree generated by our algorithm is no more than 4.214 times the cost of an optimal multicast tree.

We then turn our attention to the priority model. We present a heuristic for the minimum cost Steiner tree problem in this model and show that the cost of the tree generated is no more than $\min\{2\ln|X| + 2, k \cdot \alpha_{ST}\}$ times the cost of an optimal tree. Plugging in $\alpha_{ST} = 1.55$ [17], we get that the cost of the multicast tree generated by our algorithm is no more than $\min\{2\ln|X| + 2, 1.55 \cdot k\}$ times the cost of an optimal multicast tree.

We generalize our results for the case of many multicast groups, known in the network design literature as the *generalized Steiner problem.* The input to this problem is a set of groups $\{(s_1, X_1), \ldots, (s_\ell, X_\ell)\}$, where group i, $1 \le i \le \ell$, is characterized by a source s_i and a set of receivers X_i. The output is a forest $F \subseteq G$ such that each group is contained in a connected component of F. The cost of F is defined to be the sum of the links in it. This means that if a link is used by more than one group, we still pay for it only once. This scenario is applicable to the case where the same links are used repeatedly for several multicasts each with possibly a different source. This may occur, for example, when a company leases links and once a link is leased, it can be used by several channels belonging to different multicast groups. Indeed, Maxemchuk [15] indicates that Mbone video is a candidate for a pay-per-view service in which an entrepreneur leases facilities to provide coverage of a news or sports event. We present a heuristic for this case that generates a forest with cost no more than $\min\{1 + \log_2(\ell + \sum_{i=1}^{\ell} |X_i|), 2k\}$ times the cost of an optimal forest. (Recall that k denotes the number of priorities.)

We conclude with a proof that our problem is NP-hard in the rate model even in the special case of a spanning tree; i.e., where all the nodes in the network (but the source) are terminals. Observe that in the case of a standard Steiner tree problem this case can be solved efficiently in polynomial time (as it specializes to the standard minimum cost spanning tree problem). This means that the minimum cost spanning tree problem is also intractable in the priority model.

II. THE RATE MODEL

In this section we show how to design a multicast tree in the rate model of cost no more than $e \cdot \alpha_{ST}$ times the cost of an optimal tree. We start with a simpler algorithm that achieves a $4 \cdot \alpha_{ST}$ performance ratio, then we apply the *randomized doubling* technique to get a randomized algorithm that achieves an $e \cdot \alpha_{ST}$ ratio. Finally, we de-randomize this algorithm to obtain a *deterministic* algorithm that achieves the same approximation ratio. A similar randomized doubling idea has been used earlier by several authors in different contexts (cf. [6], [10], [16], [12], [8]).

A. The simpler algorithm

Given an instance of the multicast problem in the rate model, define the *rounded up* instance to be the instance given by rounding up all rates to the nearest power of 2. The algorithm is based on two observations.

Lemma 1: The cost of an optimal solution to the rounded up instance is no more than twice the cost of an optimal solution to the original instance.

Lemma 2: Given a rounded up instance, consider the network given by the union of the Steiner trees computed for the terminals of each rate independently. The cost of this network is at most twice the cost of an optimal solution to the rounded up instance.

Corollary 3: Consider an instance of the multicast problem in the rate model. Then, the cost of the network, given by the union of the Steiner trees computed for the terminals of each rounded up rate independently is at most 4 times the cost of the optimal solution to the original instance.

Before proving the lemmas we show how we apply them to achieve our algorithm. Given an instance of the problem, we first construct the rounded up instance. Then, we solve the (standard) Steiner tree problem for the terminals of each rate separately by applying any of the well known heuristics. Finally, we do a "clean up" process that transforms the network given by the union of these Steiner trees into a tree. We view each Steiner tree as being directed from the source towards the terminals. The clean up is done by simply keeping for each node only the incoming link of maximum rate and deleting the rest of the incoming links. It is not difficult to see that the resulting tree is indeed a valid multicast tree. The lemmas above guarantee that its cost is no more than $4\alpha_{ST}$ times the cost of an optimal multicast tree.

It is not difficult to see that the time complexity of our algorithm is $O(q \cdot \text{STEINER}(n))$, where q is the number of distinct (rounded) rate requirements of the terminals.

We now turn to the proofs of the lemmas.

Proof of Lemma 1: Consider any solution to the original instance and round up the rate of each link in this solution to the nearest power of 2. It is easy to see that the rounded up solution is a solution to the rounded up instance and that the cost of the rounded solution is at most twice the cost of the original solution. ■

Proof of Lemma 2: Consider an optimal solution to a rounded up instance with the rounded rates $2^\ell, \ldots, 2^0 = 1$. Clearly, in this solution the rates of all links are also 2^i, for $i \in \{0, \ldots, \ell\}$. Construct a network by replacing each link of rate 2^i by $i+1$ links of rate $2^i, \ldots, 2^0$. Observe that in this network all the links of a specific rate form a Steiner tree that spans all the terminals of this rate. Also, the cost of this network is no more than twice the optimal cost of the rounded up instance. It follows that the cost of the network given by the union of the Steiner trees computed separately for the terminals of each rate is no more than twice the optimal cost of the rounded up solution. ■

B. *Randomized doubling*

We now show how to reduce the factor $4\alpha_{ST}$ to $e \cdot \alpha_{ST}$. Instead of rounding up to the nearest powers of 2 we choose the basis of these powers differently. Fix a basis a (to be determined later) and randomly choose a "starting point" a^y, where y is chosen uniformly at random in the range $[0, 1]$.

Consider an instance of the multicast tree problem in the rate model. Similarly to the previous algorithm, we define a rounded up instance where each rate is rounded up to the nearest power of the form a^{y+i}, where i is an integer.

Lemma 4: The *expected* cost of the network given by the union of the Steiner trees computed for the terminals of each rounded up rate separately is at most $\frac{a}{\ln a}$ times the optimal cost of the original instance. This factor is minimized for $a = e$.

Proof: Consider an optimal solution to the original instance. Similar to the construction above, we can construct from it a union of Steiner trees, one for each rounded up rate, by replacing each link of rate r_e by links of rates $a^{y+i}, p/a, \ldots, p/a^i = a^y$, where a^{y+i} is the nearest rounded up rate greater than r_e. To estimate the expected cost of this network we need to find the expected costs of the new links. By linearity of expectation this expected cost is no more than the expected cost of the rounded up rate times $\sum_{i=0}^{\infty} 1/a^i = a/(a-1)$. Suppose that $r_e = a^{x+j}$, where j is an integer and $x \in [0, 1]$. If $x \le y$, then the rounded up cost is a^{y-x} times the original cost. Otherwise, it is a^{y+1-x} times the original cost. Since y is chosen uniformly in the range $[0, 1]$, we get that the expected cost is $r_e c_e$ times

$$\begin{aligned}\int_0^x a^{y+1-x}\,dy + \int_x^1 a^{y-x}\,dy &= \frac{a - a^{1-x}}{\ln a} + \frac{a^{1-x} - 1}{\ln a} \\ &= \frac{a-1}{\ln a}.\end{aligned}$$

We get that the expected costs of the replaced links is no more than $r_e c_e \cdot \frac{a}{a-1} \cdot \frac{a-1}{\ln a} = r_e c_e \cdot \frac{a}{\ln a}$. We set $a = e$ to minimize this expected cost and get that the expected cost is $r_e c_e \cdot e$, where e is the basis of the natural log. This implies that the expected cost of the network given by the union of the Steiner trees computed for the terminals of each rounded up rate separately is at most e times the optimal cost of the original instance. ■

This algorithm can be derandomized by replacing the random selection of start point e^y, for y uniformly in the range $[0, 1]$, by a small number of choices for y, one for each distinct rate requirement of the terminals. For each distinct rate r, we run the algorithm for $y = \ln r - \lfloor \ln r \rfloor$. Let the set of such y values be $y_1 < y_2 \ldots < y_q$. We claim that the cost of a solution produced for any $y \in [y_i, y_{i+1})$ is at least the cost of the solution for $y = y_i$. To see this, note that each link in the optimal solution has rate $e^{y_j+\alpha}$, for some $j \in \{1, \ldots, q\}$ and integer α. It follows that for every link rate $e^{y_j+\alpha}$, where $j \le i$, $e^{y-y_j} \ge e^{y_i-y_j}$, and for every link rate $e^{y_j+\alpha}$, where $j > i$, $e^{1+y-y_j} \ge e^{1+y_i-y_j}$. (Also, the cost of the solution for $y \in [0, y_1)$ is at least the cost of the solution for $y = y_q$.) This implies that for some i, the solution for $y = y_i$ has cost which is at most the expected cost of the solution for a randomly chosen y.

The time complexity of the randomized algorithm is the same as the time complexity of the simple deterministic algorithm given above; that is, $O(q \cdot \text{STEINER}(n))$. The derandomized algorithm replaces the randomized selection by q trials for the value of y, and thus its time complexity is $O(q^2 \cdot \text{STEINER}(n))$.

III. THE PRIORITY MODEL

We now present approximation algorithms for the Steiner tree problem and the generalized Steiner tree problem in the priority model. The latter problem is a generalization of the former, yet we present separate algorithms for both, as the algorithm and analysis for the Steiner tree problem is considerably simpler. Our approach for both problems is similar. We consider on-line algorithms for the Steiner tree [2], [13] and generalized Steiner tree problem [7]. The on-line algorithm for the Steiner tree problem takes a sequence of terminals and builds a tree spanning these terminals in an incremental fashion, such that each new terminal is connected to the existing tree by the addition of links. Once added, a link cannot be deleted. In the case of generalized Steiner tree, the input consists of a sequence of pairs and the algorithm incrementally builds a collection of trees such that each pair is connected. In both cases, we sort the terminals (or pairs) in order of priority, from the highest to the lowest, and feed this sorted sequence as input to the on-line algorithm. The algorithms for Steiner tree and generalized Steiner tree were not designed to handle links with different priorities as in the priority model. Yet, we are able to adapt the analysis of the on-line algorithms to give an approximation guarantee in the priority model. The approximation guarantee we obtain is the competitive ratio of the on-line algorithms.

A. Steiner trees

Our heuristic for the Steiner tree problem in the priority model is based on the on-line Steiner tree algorithm. This algorithm was first analyzed by Imaze and Waxman [13] who proved a $\log_2 |X|$ competitive ratio. Here, we use the simpler analysis of Alon and Azar [2] who proved a (slightly inferior) $2 \ln |X| + 2$ competitive ratio.

We consider the terminals ordered by priority from highest to lowest. Let the ordered sequence be $t_1, \ldots, t_{|X|}$. We build the tree incrementally. Terminal t_i is connected to the tree constructed for the first $i-1$ terminals by the cheapest path connecting t_i to the tree. In computing the costs of shortest paths from t_i to nodes in the tree, we use all links of priority at least as high as $p(t_i)$. We obtain the initial tree by connecting t_1 to the source s by the cheapest path from t_1 to s that uses links of priority at least as high as $p(t_1)$.

It is easy to see that the algorithm returns a feasible solution. In other words, the tree constructed by the algorithm has the property that terminal t_i is connected to the source s by a path that uses links of priority at least as high as $p(t_i)$. We now proceed to bound the cost of the tree constructed by the algorithm in terms of OPT, the cost of the optimal tree.

Associate with each terminal t_i, the cost of connecting t_i to the tree in the construction above. We call this cost the *connection cost* of t_i.

Lemma 5: For $1 \leq m \leq |X|$, the mth largest connection cost is at most $2\text{OPT}/m$.

Proof: The proof is by contradiction. To obtain a contradiction assume the contrary; that is, there exists an $m \in \{1, \ldots, |X|\}$ for which the mth largest connection cost is more than $2\text{OPT}/m$. This implies that each of the m largest connection costs is more than $2\text{OPT}/m$. Let $X' = \{x_1, \ldots x_m\}$ be the set of the m terminals with the highest connection costs. For convenience, assume that terminals $x_1, \ldots x_m$ are ordered by priority from highest to lowest. We define a distance function on the set X' as follows: The distance $d(x_i, x_j)$ between x_i and x_j is the cost of the shortest path from x_i to x_j that uses links of priority at least as high as $\max\{p(x_i), p(x_j)\}$. Note that this distance function does not obey the triangle inequality and thus is not a metric. For $i < j$, since x_j was connected to the tree after x_i was connected, the connection cost of x_j is bounded above by $d(x_i, x_j)$. (This is because we connect x_j to the existing tree by the cheapest connection using links of priority at least as high as $p(x_j)$.) As all connection costs are greater than $2\text{OPT}/m$, it follows that all distances $d(x_i, x_j)$ are at least $2\text{OPT}/m$.

Let T be the optimal tree for the instance. Consider the minimal subtree T' of T spanning X' (and including the source s). T' consists of the unions of the paths from x_i to s in T. Note that the cost of T' is at most OPT. Now, we construct a tour of X' by performing a depth-first traversal of the tree T' beginning at s and returning to s. Since each link is used twice, the cost of the tour is at most 2OPT. Suppose x_i and x_j are consecutive terminals visited by this tour, we claim that the links used in the path from x_i to x_j have priority at least $\max\{p(x_i), p(x_j)\}$. To prove this, observe that the path from x_i to x_j can be broken into two parts: a path from x_i to a common ancestor y of x_i and x_j and a path from y to x_j. (One of the two parts may be missing if x_i is an ancestor of x_j or vice versa.) The first part consists of links of priority at least $p(x_i)$ while the second part consists of links of priority at least $p(x_j)$. Thus the links in the path from x_i to x_j have priority at least $\max\{p(x_i), p(x_j)\}$. Recall the distance function $d(\cdot,\cdot)$ defined earlier. It follows that the cost of the path from x_i to x_j is at least $d(x_i, x_j) > 2\text{OPT}/m$. Note that this argument also holds for the path between the last terminal and the first terminal visited in the depth-first traversal. Thus, in the tour constructed, the distance between any two consecutive terminals is greater than $2\text{OPT}/m$. This means that the cost of the tour is greater than 2OPT, which is a contradiction. ■

Theorem 6: The cost of the tree produced is at most $2H(|X|)\text{OPT}$, where $H(m) = 1+\frac{1}{2}+\ldots+\frac{1}{m} \leq 1+\ln m$.

Proof: Since the mth largest connection cost is at most $2\text{OPT}/m$, the cost of the tree is at most $2\text{OPT}\left(1+\frac{1}{2}+\ldots+\frac{1}{|X|}\right) = 2H(|X|)\text{OPT}$. ■

The algorithm can be implemented by running Dijkstra's algorithm from each terminal, giving a running time of $O(nm + n^2 \log n)$.

If the number of priorities k is small, we can get a better approximation guarantee than the above, by computing a Steiner tree for the terminals of each priority separately. When computing the Steiner tree for the terminals of a specific priority we consider only links of priority at least as high as this priority. The final solution is given by the union of the trees constructed. To ensure that the solution produced is indeed a tree, we eliminate cycles in a "clean up" process as described before. The running time of this algorithm is $O(k \cdot \text{STEINER}(n))$.

Let OPT be the cost of an optimal tree for the priority problem. Suppose α_{ST} is the approximation ratio of the Steiner tree algorithm we use. The cost of each Steiner tree constructed is at most $\alpha_{ST} \cdot \text{OPT}$. Hence, the cost of the final solution is at most the sum of the costs of the these trees which is bounded by $k \cdot \alpha_{ST} \cdot \text{OPT}$.

Taking the better of the two algorithms presented in this section, we obtain the following theorem.

Theorem 7: The algorithm presented above computes a multicast tree in the priority model with cost at most $\min\{k \cdot \alpha_{ST}, 2\ln|X| + 2\}$ times the optimal cost.

B. Generalized Steiner Trees

Our heuristic for the generalized Steiner tree problem in the priority model is based on the on-line generalized Steiner tree algorithm of Berman and Coulston [7]. To use their algorithm we first observe that without loss of generality we may assume that the input is a collection of node pairs (a, b) with a priority associated with each pair, and that the goal is to connect each pair by a path with priority at least as high as its associated priority. This is true since each group (s_i, X_i) in the original formulation can be replaced by $|X_i|$ pairs (s_i, a), for every $a \in X_i$. The priority associated with this pair is $p(a)$.

We now describe the algorithm of [7] adapted for the priority model. The algorithm adds the pairs one at a time and maintains the connected components of the forest F created at each stage. We begin with a description of the data structures maintained by the algorithm and some definitions. Each node u received as part of an input pair is associated with an integer $\texttt{class}(u)$. For a component Y of F, define $\texttt{class}(Y) = \max_{u\in Y} \texttt{class}(u)$. For two nodes u and v, define $\texttt{path}_p(u, v)$ to be the cheapest path from u to v using links of priority at least as high as p, and $\texttt{dist}_p(u, v)$ to be the cost of $\texttt{path}_p(u, v)$.

Initially, the connected components of F are singletons, and F contains no links. For each node u we initialize $\texttt{class}(u) = -\infty$. We consider the input node pairs (a, b) in order of priority from highest to lowest. When a new pair (a, b) of priority p is considered, we first place a and b in singleton components. Note that in case a belongs to a larger component Y of F when the pair (a, b) is considered, we can create a copy of a and place it in a singleton component. The copy of a is connected to a with a link of priority 1 and cost 0. For every link e from a to some other node, we place a link from the copy of a to the same node. This new link has priority $p(e)$ and cost $c(e)$. Similarly, we create a copy of b if required. We proceed with the following algorithm.

$\texttt{class}(a) = \texttt{class}(b) = \lfloor \log_2 \texttt{dist}_p(a, b) \rfloor$.
For every component $Y \notin \{\{a\},\{b\}\}$ of F do
1. let u be a node of Y that is closest to a according to distance function $\texttt{dist}_p$.
2. let $m = \min\{\texttt{class}(a), \texttt{class}(Y)\}$.
3. if $\texttt{dist}_p(a, u) < 2^{m+1}$
   then insert $\texttt{path}_p(a, u)$ to F.

For every component $Y \neq \{b\}$ of F do
1. let u be a node of Y that is closest to b according to distance function $\texttt{dist}_p$.
2. let $m = \min\{\texttt{class}(b), \texttt{class}(Y)\}$.
3. if $\texttt{dist}_p(b, u) < 2^{m+1}$
   then insert $\texttt{path}_p(b, u)$ to F.

Theorem 8: The algorithm produces a solution with cost at most $\log_2(\ell+\sum_{i=1}^{\ell}|X_i|)+2$ times the cost of the optimal solution.

Proof: The proof of the above theorem is very similar to the proof of the competitive ratio of the on-line algorithm in [7]. However, we cannot use their result as a "black box" to prove our approximation guarantee. We describe the proof in detail, indicating the modifications required to handle different priorities.

First, we prove that the solution is valid, i.e., each input node pair is connected by a path with the required priority. Consider an input pair (a,b) with priority p. In the first loop of the algorithm we may connect a to other components (except for the component $\{b\}$). As a result, when this loop terminates a may belong to a non-singleton component. Since the pairs are added in order of priority there is a path of priority at least as high as p from a to any other node in its component. In the second loop of the algorithm, node b is connected to all components Y of F for which there exists $u \in Y$ such that $\mathtt{dist}(b,u) < 2^{m+1}$, where $m = \min\{\mathtt{class}(b), \mathtt{class}(Y)\}$. Note that since $\mathtt{class}(a) = \lfloor \log_2 \mathtt{dist}_p(a,b) \rfloor$, the class of the component Y that contains a at the time the second loop is executed is at least $\lfloor \log_2 \mathtt{dist}_p(a,b) \rfloor$. Since $\mathtt{class}(b) = \lfloor \log_2 \mathtt{dist}_p(a,b) \rfloor$, it follows that b is connected to Y by a path of priority at least as high as p and thus is also connected to a by such a path.

We now prove the bound on the cost of the solution. Define $B_p(u,r)$ to be a *sphere*, or *ball*, of radius r around u. The ball $B_p(u,r)$ contains the set of nodes v such that $\mathtt{dist}_p(u,v) < r$, where $\mathtt{dist}_p(u,v)$ is the cost of the cheapest path between u and v that uses links of priority at least as high as p. Note that $B_p(u,r)$ may contain some links (of priority at least as high as p) completely and other such links (belonging to the "periphery") only partially.

Suppose that the input pair (a,b) is associated with priority p, such that $\mathtt{dist}_p(a,b) > r$. Then, for every solution T to this instance, the cost of $T \cap B_p(a,r)$ is at least r. We refer to $B_p(a,r)$ as a *lower bound ball.*

Suppose there exists a collection $\mathcal{C}$ of pairwise disjoint lower bound balls with sum of radii s, then any solution to this instance must have cost at least s. We call $\mathcal{C}$ a *lower bound collection.*

To prove our approximation ratio, we run a *shadow* algorithm which finds a lower bound collection of sufficient size. For every integer j, the shadow algorithm constructs a lower bound collection $\mathcal{C}(j)$, consisting of balls with radius at most 2^j. We denote the sum of radii of $\mathcal{C}(j)$ by $\mathcal{S}(j)$. Let $\mathcal{S} = \sum_j \mathcal{S}(j)$. We ensure the following properties:

$$\mathcal{S} \geq \mathtt{cost}(F) \tag{1}$$

$$\max_j \mathcal{S}(j) \geq \frac{1}{\log_2(\ell+\sum_{i=1}^{\ell}|X_i|)+2}\mathcal{S} \tag{2}$$

Here, $\mathtt{cost}(F)$ denotes the cost of the solution produced by our algorithm. The above two inequalities imply that $\mathtt{cost}(F)$ is within a factor $\log_2(\ell+\sum_{i=1}^{\ell}|X_i|)+2$ of the optimal cost.

To show that (1) holds we perform amortized analysis. For every pair (A,j), where A is a component of F and j is an integer such that $j \leq \mathtt{class}(A)$, we create an account that holds 2^j. (Note that the number of accounts is infinite.) The amortized cost at any point of time is $\mathtt{cost}(F)$ plus the sum of the existing accounts. We start with both $\mathcal{S}$ and the amortized cost being 0. In each step we ensure that the change (increase) of $\mathcal{S}$ is at least as large as the change in the amortized cost. Note that a change in the amortized cost involves the cost of the new edges added to F plus the contents of the new accounts created minus the content of any accounts being liquidated.

To simplify the accounting, we divide the actions performed by the real algorithm and the shadow algorithm into a number of steps. For each step, we verify that the change in $\mathcal{S}$ is at least as much as the change in the amortized cost.

The shadow algorithm starts with all $\mathcal{C}(j)$'s empty. For each pair (a,b) of priority p processed by the algorithm, we have two steps.

Step 1: The real algorithm finds the shortest path $\mathtt{path}_p(a,b)$ from a to b of cost $d_p = \mathtt{dist}_p(a,b)$. Then, it assigns j to both $\mathtt{class}(a)$ and $\mathtt{class}(b)$, where $2^j \leq d_p < 2^{j+1}$. The shadow algorithm performs the following for each $i \leq j$. First, it creates lower bound balls $B_p(a,2^i)$ and $B_p(b,2^i)$ and inserts them into $\mathcal{C}(i)$. Next, it provides components $\{a\}$ and $\{b\}$ with accounts holding 2^i. The amortized cost is increased during this step. The increase amount $\Delta_{\mathtt{cost}}$ is the total holdings in the accounts of $\{a\}$ and $\{b\}$. Thus,

$$\Delta_{\mathtt{cost}} = 2(2^j + 2^{j-1} + \cdots) = 2^{j+2}.$$

On the other hand $\mathcal{S}$ is increased by the sum of the radii of the new balls created, which is also 2^{j+2}. Note that at the end of this step, some of the new balls created might intersect existing balls in the lower bound collections. This will be fixed in the next step.

Step 2: Suppose that in some collection $\mathcal{C}(i)$, we inserted a ball $B_p(a,2^i)$ that intersects an existing ball, say

$B_{p'}(u,r)$, for some $r \le 2^i$ and $p' \le p$. We call this a *conflict* between a and the component of F that contains u. We define a conflict between b and a component of F similarly. The shadow algorithm removes such conflicts while the real algorithm processes the connected components of F in the first and second loops.

Consider a component Y processed in the first loop. Let $d = \min_{u \in Y} \mathtt{dist}_p(a,u)$, and $m = \min\{\mathtt{class}(a), \mathtt{class}(Y)\}$. We distinguish between two cases.

CASE 1: $d \ge 2^{m+1}$.
In this case we claim that there is no conflict between a and Y. To obtain a contradiction suppose there is a conflict. Then, some lower bound collection $\mathcal{C}(i)$ must contain two intersecting balls, $B_p(a,r_a)$ and $B_{p'}(v,r_v)$ for some $v \in Y$. Note that $p' \le p$ (i.e., v has priority at least as high as a) because of the order in which the pairs are processed. Also, the radii r_a, r_v are both at most 2^i. This implies that $d' = \mathtt{dist}_p(a,v) < r_a + r_v \le 2^{i+1}$. But $i \le \mathtt{class}(Y)$ and also $i \le \mathtt{class}(a)$, so $i \le m$. This gives a contradiction, as $d \le d' < 2^{i+1} \le 2^{m+1}$. In this case the real algorithm does not do anything and since there is no conflict, the shadow algorithm does not have to do anything either.

CASE 2: $d < 2^{m+1}$.
In this case, the real algorithm merges Y with the component of a, while the shadow algorithm shrinks (in a minimal fashion) the balls centered at a so that all conflicts are eliminated between a and Y. Also, the shadow algorithm removes the accounts that become redundant as a result of the merger of components.

The change in the amortized cost is the cost of the new connection, i.e., d minus the amount in the accounts that are liquidated. Suppose A is the component that contains a at the beginning of this step. For each $i \le m$, we replace accounts (A,i) and (Y,i) with a single account $(A \cup Y, i)$. Thus,

$$\Delta_{\mathtt{cost}} = d - (2^m + 2^{m-1} + \cdots) = d - 2^{m+1}.$$

The sum of the radii in the lower bound collections also changes, because we shrink the balls centered at a if they intersect balls from the same collection with centers in Y. For $i > m$, the collection $\mathcal{C}(i)$ either contains no ball centered at a or it contains no ball with a center in Y. For $i \le m$, there may be conflicts. The decrease in the sum of the radii is largest if for every $i \le m$, the collection $\mathcal{C}(i)$ contains both $B_p(a,2^i)$ and the largest possible conflicting ball $B_{p'}(u,2^i)$, for some $u \in Y$ that is closest to a amongst nodes in Y, and $p' \le p$. (Note that since u is processed before a, the priority p' of u is at least as high as the priority p of a.) From now on we assume that this is indeed the situation. Let $j = \lfloor \log_2 d \rfloor$ and $f = d - 2^j < 2^j$. It is not difficult to see that to remove the conflicts, for all $j < i \le m$, the balls $B_p(a,2^i)$ need to be shrunk to degenerate balls $B_p(a,0)$, while the ball $B_p(a,2^j)$ needs to be shrunk to $B_p(a,f)$. The balls in collections $\mathcal{C}(i)$ for $i < j$ do not cause any conflicts. It follows that the decrease in the sum of the radii in this case is

$$\begin{aligned}\sum_{i=j}^{m} 2^i + (2^j - f) &= (2^{m+1} - 2^{j+1}) + (2^j - f)\\ &= 2^{m+1} - (2^j + f) = 2^{m+1} - d\,.\end{aligned}$$

This implies that the change in $\mathcal{S}$ is always at least as large as $\Delta_{\mathtt{cost}}$.

The above analysis of cases 1 and 2 holds also for the case of components processed in the second loop. Note that since component $\{b\}$ is *not* processed in the first loop, every conflict between b and a component before the execution of the first loop remains a conflict between $\{b\}$ and either the original component or some superset of this original component after the execution of the first loop.

Observe that if we rescale all the distances by multiplying them by a power of 2, the algorithm still behaves the same, and the same lower bound collections are generated by the shadow algorithm (with rescaled balls). Therefore, we may assume without loss of generality that the $\mathcal{C}(0)$ collection contains the largest lower bound balls. Then, when all the pairs are added there must exist a component Y of F with $\mathtt{class}(Y) = 0$. This component has accounts holding $2^0, 2^{-1}, 2^{-2}, \ldots$ summing to 2. This implies that the shadow algorithm may do some additional operations, as long as they are amortized by this amount. This amount would be used to liquidate some accounts and remove lower bound collections, so that the number of non-empty lower bound collection is minimized.

Recall that the maximum number of nodes in the input pairs to the problem is $n_g = \ell + \sum_{i=1}^{\ell} |X_i|$. Let $\lambda = \lceil \log_2 n_g \rceil$. Each lower bound collection consists of at most n_g balls. For $i \ge 0$, The radius of each ball in the lower bound collection $\mathcal{C}(-\lambda - i)$ is at most $2^{-\lambda-i} = 1/(n_g 2^i)$. It follows that sum of the radii of all lower bounds balls in these collections is at most $\sum_{i=0}^{\infty} 2^{-i} = 2$. Hence, we can remove all these collections amortizing it with the existing account holdings after all node pairs are connected.

After this "clean up" phase the only non-empty lower bound collections are $\mathcal{C}(-\lambda+1), \mathcal{C}(-\lambda+2), \ldots \mathcal{C}(0)$ a total of at most $\log_2 n_g + 1 = \log_2(\ell + \sum_{i=1}^{\ell} |X_i|) + 1$ collections. This implies Eq. (2), and the theorem follows. ■

The running time of the algorithm is dominated by the complexity of computing arrays `dist(.)`. This can be done in $O(n^2 m)$ by adding the links one by one in order of priorities when computing the arrays.

Another heuristic for the generalized Steiner tree problem in the priority model is to generate k generalized Steiner trees separately, one for each priority p and then perform a "clean up" step similar to the Steiner tree algorithm in the rate model. We use the 2-approximation heuristic of Agrawal, Klein and Ravi [1] to generate the generalized Steiner trees for each priority. The cost of the forest generated by this heuristic is at most $2k$ times the optimal cost and the running time is $O(km \log n)$. We conclude that the cost of our heuristic is no more than $\min\{\log_2(\ell + \sum_{i=1}^{\ell} |X_i|) + 1, 2k\}$ times the cost of an optimal forest.

IV. The NP-Hardness proof

In this section we consider the case where the Steiner tree needs to span all the nodes in the graph, i.e., the set of terminals is $X = V \setminus \{s\}$. Recall that in the case where all rates are equal, finding an optimal spanning tree can be done in polynomial time, while finding an optimal Steiner tree is NP-hard. We prove that finding a spanning tree in the rate model is NP-Hard. Since the rate model is a special case of the priority model this also proves the hardness for finding a spanning tree in the priority model.

The NP-Hardness is proved by a reduction from the 3SAT problem to the decision problem corresponding to the problem of finding a minimum spanning tree in the rate model. In this decision problem we are given a graph $G(V, E)$, a source node $s \in V$, a rate $r(v)$ associated with every node $v \in V \setminus \{s\}$, a basic cost c_e associated with every link $e \in E$, and a parameter w. We need to determine whether there exists a feasible tree of cost at most w. From now on, we call this decision problem the RATE-MST problem.

Recall that in the 3SAT problem we are given n logical variables $(x_1, \ldots, x_n)$ and m clauses each consists of three literals (i.e., a variable or its complement). We have to decide whether there exists a valid 0-1 assignment to the variables so that each of the clauses has at least one literal that is assigned 1. (In a valid 0-1 assignment the value of the complement of a variable is the complement of its value.) The 3SAT problem is well known to be NP-Hard.

Theorem 9: The RATE-MST problem is NP-Hard.

Proof: To show the reduction from 3SAT to the RATE-MST problem we consider an input instance to the 3SAT problem and show how to construct a respective instance to the RATE-MST problem such that the answer to the 3SAT instance is "yes" if and only if this is also the answer for the corresponding RATE-MST instance. Suppose that the 3SAT instance consists of the clauses $C_1, \ldots, C_m$, where C_j consists of three literals. We denote the literals of C_j by $y^t_{j(1)}$, $y^t_{j(2)}$ and $y^t_{j(3)}$, where $t \in \{0, 1\}$, $j(1), j(2), j(3) \in \{1, \ldots, n\}$, and y^0_i (y^1_i) denotes $\bar{x}_i$ (x_i).

The corresponding instance of the RATE-MST problem consists of a graph with $1+3n+m$ nodes and $4n+3m$ links. The nodes consist of the source node s, $2n$ "literal" nodes y^0_i and y^1_i with rate 1, for $i = 1 \ldots, n$, n "variable" nodes $x_1, \ldots, x_n$ with rate 2, and m "clause" nodes $C_1, \ldots, C_m$ with rate 2. We now define the links.

1. Each of the literal nodes y^t_i is connected to the source s by a link of basic cost 1.
2. Each variable node x_i is connected to both its corresponding literal nodes y^0_i and y^1_i by a link of basic cost 1.
3. Each clause node $C_j = \{y^t_{j(1)}, y^t_{j(2)}, y^t_{j(3)}\}$ is connected to the literal nodes $y^t_{j(1)}$, $y^t_{j(2)}$ and $y^t_{j(3)}$ by a link of basic cost 3.

We set the cost bound w to be $5n + 6m$.

Suppose that the answer to the instance of the 3SAT problem is "yes". We show how to find a valid spanning tree of cost $5n + 6m$.

1. For each variable x_i, the tree has a link from the source s to y^1_i if the value of x_i in the 3SAT instance is 1 and to y^0_i otherwise. The rate of this link is 2. The tree has a link from the source to the other literal with a link of rate 1. The total cost of these $2n$ links is $3n$.
2. For each variable x_i, the tree has a link of rate 2 connecting the variable node x_i to y^1_i if the value of x_i in the 3SAT instance is 1 and to y^0_i otherwise. The rate of this link is 2. The total cost of these n links is $2n$.
3. For each clause C_j, the tree has a link of rate 2 connecting the clause node C_j to one of the literal nodes that corresponds to a literal in this clause with the value 1 in the 3SAT instance. We are guaranteed to have at least one such literal since the answer to the 3SAT instance is "yes". The total cost of these m links is $6m$.

Note that the total weight of the tree is $5n+6m$ as required.

We still need to show that each node is connected to the source via a path consisting of links with the correct rate. This is clearly the case for the literal nodes each of which require rate 1. Observe that each clause node and each literal node are connected to the source via a literal node corresponding to a literal which is assigned the value 1 in the corresponding 3SAT instance, and that both links along this path have rate 2 as required.

In the full version we show the other direction; that is, we show that if there exists a valid spanning tree of cost at most $5n + 6m$, then the answer to the corresponding instance of the 3SAT problem is "yes". ∎

V. Conclusion

We presented heuristics with provable performance guarantees for the Steiner tree problem in the rate model and the priority model. An interesting open problem is to develop a heuristic with constant approximation ratio for the Steiner tree problem in the priority model. Currently, we also do not have any hardness of approximation results for this problem, as well as for the problem in the rate model. We remark that the Steiner tree problem in the priority model can be formulated in a natural way as a linear integer program; for example, by requiring that every cut separating the root from a subset of the terminals is covered by at least one edge of priority at least as high as the highest priority among the separated terminals. Yet, we are not aware of any example in which the gap between the integral and rational optima in this formulation is greater than two. Note that this is also the integrality gap for the classical Steiner tree problem in this formulation.

References

[1] A. Agrawal, P. Klein, and R. Ravi. *When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks.* SIAM Journal on Computing, Vol. 24, pp. 445–456, 1995.

[2] N. Alon and Y. Azar. *On-line Steiner trees in the Euclidean plane.* Discrete and Computational Geometry Vol. 10, pp. 113–121, 1993.

[3] E. Amir, S. McCanne, and R. Katz. *An active service framework and its application to real-time multimedia transcoding.* Proceedings of ACM SIGCOMM, 1998.

[4] E. Amir, S. McCanne, and R. Katz. *Receiver-driven bandwidth adaptation for light-weight sessions.* Proceedings of ACM Multimedia, 1997.

[5] E. Amir, S. McCanne, and H. Zhang. *An application level video gateway.* Proceedings of ACM Multimedia, 1995

[6] A. Beck and D. Newman. *Yet more on the linear search problem.* Israel Journal of Math., Vol. 8, pp. 419–429, 1970.

[7] P. Berman and C. Coulston. *On-line algorithms for Steiner tree problems.* Proceedings of 25th ACM STOC, 1997.

[8] S. Chakrabarti, C. Phillips, A. Schulz, D.B. Shmoys, C. Stein, and J. Wein. *Improved scheduling algorithms for minsum criteria.* Proceedings of 23rd ICALP, 1996.

[9] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara. *On flow aggregation for multicast video transport.* Proceedings of 6th International Workshop on Quality of Service, pp. 13–22, 1998.

[10] S. Gal. *Search Games.* Academic Press, 1980.

[11] M. Ghanbari. *Two-layered coding of video signals for VBR networks.* IEEE J. on Selected Areas in Communications, Vol. 7, pp. 771–781, 1989.

[12] M. Goemans and J. Kleinberg. *An improved approximation ratio for the minimum latency problem.* Proceedings of 7th ACM-SIAM SODA, pp. 152–157, 1996.

[13] M. Imaze and B.M. Waxman. *Dynamic Steiner tree problem.* SIAM Journal on Discrete Mathematics, Vol. 4 No. 3, pp. 369–384, 1991.

[14] G. Karlsson and M. Vetterli. *Packet video and its integration into the network architecture.* IEEE J. on Selected Areas in Communications, Vol. 7, pp. 739–751, 1989.

[15] N. F. Maxemchuk. *Video Distribution on Multicast Networks.* IEEE J. on Selected Areas in Communications, Vol. 15, pp. 357–372, 1997.

[16] R. Motwani, S. Phillips and E. Torng. *Non-clairvoyant scheduling.* Proceedings of 4th ACM-SIAM SODA, pp. 422–431, 1993. see also: Theoretical Computer Science, Vol. 130, pp. 17–47, 1994.

[17] G. Robins and A. Zelikovsky. *Improved steiner tree approximation in graphs.* Proceedings of 11th ACM-SIAM SODA, 2000.

[18] T. Turletti and J.-C. Bolot. *Issues with multicast video distribution in heterogeneous packet networks.* Proceedings of the 6th International workshop on packet video, 1994.

[19] P. Winter. *Steiner problem in networks: A survey.* Networks, Vol. 17, pp. 129–161, 1987.

[20] L. Zhang, S. Deering, D. Estrin, S. Shenker, and Zappala. *RSVP: A new resource reservation protocol.* IEEE Network, Vol. 7, pp. 8–18, 1993.