Sublogarithmic Approximation for Telephone Multicast *

Michael Elkin † Guy Kortsarz‡

Abstract

Consider a network of processors modeled by an n-vertex graph $G = (V, E)$. Assume that the communication in the network is synchronous, i.e., occurs in discrete "rounds", and in every round every processor is allowed to pick one of its neighbors, and to send it a message. The *telephone k-multicast* problem requires to compute a schedule with minimal number of rounds that delivers a message from a given single processor, that generates the message, to all the processors of a given set $T \subseteq V$, $|T| = k$, whereas the processors of $V \setminus T$ may be left uninformed. The case $T = V$ is called *broadcast* problem. Several approximation algorithms with a polylogarithmic ratio were suggested for these problems, and the upper bound on their approximation threshold stands currently on $O(\log k)$ and $O(\log n)$, respectively.

In this paper we devise an $O(\frac{\log k}{\log \log k})$-approximation algorithm for the k-multicast problem, and, consequently, an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the broadcast problem. Even stronger than that, whenever an instance of the k-multicast problem admits a schedule of length br^*, our algorithm guarantees an approximation ratio of $O(\frac{\log k}{\log br^*})$. As br^* is always at least $\log k$, the ratio of $O(\frac{\log k}{\log \log k})$ follows. In addition, whenever $br^* = \Omega(k^\delta)$ for some constant $\delta > 0$, we obtain a *constant* $O(1/\delta)$-approximation ratio for the problem.

Our results have implications for network design. The poise of a spanning tree is the sum of its depth and maximum degree [R94]. We improve the $O(\log k)$ approximation algorithm [BGN+98] for the poise problem to $O(\log k / \log k \log k)$, and obtain an improved $(O(\log k / \log \log k), O(\log k / \log k \log k))$ bicriteria approximation for the depth-degree problem.

We also derive results concerning the *edge-dependent heterogeneous* k-multicast problem.

---

*A preliminary version of this paper was published in SODA 2003 [EK03].

†Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, elkinm@cs.bgu.ac.il.

‡Computer Science department, Rutgers University, Camden, NY, USA. Email: guyk@crab.rutgers.edu

1 Introduction

Consider a network of processors modeled by an n-vertex graph $G = (V, E)$. Assume that the communication in the network is synchronous, i.e., occurs in discrete "rounds", and in every round every processor is allowed to pick one of its neighbors, and to send it a message. The *telephone k-multicast* problem requires to compute a schedule with minimal number of rounds that delivers a message from a given single processor, that generates the message, to all the processors of a given set $T \subseteq V$, $|T| = k$, whereas the processors of $V \setminus T$ may be left uninformed. The case $T = V$ is called *broadcast* problem.

The telephone multicast and broadcast are basic primitives in distributed computing and computer communication theory, and are used as building blocks for various more complicated tasks in these areas (cf. [HHL88]). The optimization variants of the multicast and broadcast primitives were intensively studied during the last decade [BGN+98, KP95, R94, F01]. Several approximation algorithms with a polylogarithmic ratio were suggested for these problems [BGN+98, R94, EK02], and the upper bound on their approximation threshold stands currently on $O(\log k)$ and $O(\log n)$, respectively [BGN+98, EK02]. The problems are known to be inapproximable within a factor of $3 - \epsilon$, for any $\epsilon > 0$ [EK02, S00].

In this paper we devise an $O(\frac{\log k}{\log \log k})$-approximation algorithm for the k-multicast problem, and, consequently, an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the broadcast problem. Even stronger than that, whenever an instance of the k-multicast problem admits a schedule of length br^*, our algorithm guarantees an approximation ratio of $O(\frac{\log k}{\log br^*})$. As br^* is always at least $\log k$, the ratio of $O(\frac{\log k}{\log \log k})$ follows. In addition, whenever $br^* = \Omega(k^\delta)$ for some constant $\delta > 0$, we obtain a *constant* $O(1/\delta)$-approximation ratio for the problem. We remark that unlike all the previous approximation algorithms for these problems except the algorithm of [EK02], our algorithm is *combinatorial*, i.e., it does not use Linear Programming.

We believe that the main contribution of this paper is in demonstrating that the telephone multicast and broadcast problems both admit *sublogarithmic approximation* algorithms. In view of several different approximation algorithms that achieved logarithmic or slightly higher than logarithmic approximation ratios for the multicast and broadcast problems [R94, BGN+98, EK02], it was natural to assume that these problems are at least as hard as the Set-Cover problem. However, we prove that this is not the case, which suggests that a constant approximation may be attainable.

Implications for network design: A large body of research deals with network design problems in which the objective is to optimize more than one optimization criteria a simultaneously (see, e.g., [MRR+93]). Such optimization problems are called *bicriteria optimization* problems. Consider the degree-depth network design problem. Given an undirected graph G and two parameters h, d so that G admits a Steiner tree of depth at most h and maximum degree at most d, the goal is to find this Steiner tree. Alternatively, we can bound (say) h and request the minimum degree subgraph among Steiner trees of height at most h. Ravi [R94] has shown that this problem is closely related to the telephone broadcast problem, and designed the first bicriteria approximation algorithm for it that provides a simultaneous polylogarithmic approximation to both h and d. Defining *poise* of a tree to be the sum between its maximum degree and its depth [R94], the result of [R94] implies directly a polylogarithmic approximation guarantee for the problem of constructing a spanning tree with minimum poise (henceforth, the *poise* problem). The algorithm of [BGN+98] can be adapted to provide a logarithmic bicriteria approximation for the degree-depth problem, implying, consequently, a logarithmic approximation algorithm for the poise problem.

A version of our algorithm guarantees $O(\log k/\log\log k)$ approximation ratio for the minimum poise problems, as well as an $(O(\log k/\log\log k), O(\log k/\log\log k))$ bicriteria approximation for the degree-depth approximation problem, improving the results of [R94, BGN+98]. In particular, given a graph G and bounds h on the depth and d on the degree, so that G admits a spanning tree of degree at most d and depth at most h, our algorithm produces a Steiner tree of depth at most $O(\log k/\log\log k)\cdot h$ and degree at most $O(\log k/\log\log k)\cdot d$. Alternatively, it can accept a bound h on the height and return a subtree of height at most $O(\log k/\log\log k)\cdot h$ and degree at most $O(\log k/\log\log k)\cdot d$ with d the minimum degree of any Steiner tree of height at most h.

Finally, some our results generalize to the *edge-dependent heterogeneous* telephone multicast problem [BGN+98, EK02] (see Section 5.1 for its definition).

2 Preliminaries

2.1 Definitions

The input of the telephone multicast problem is an undirected connected graph $G = (V, E)$, a source vertex $s \in V$, and a set $T \subseteq V$ of terminals. During the execution of a broadcast algorithm, the vertices of V are split into two complementary disjoint subsets: the subset I of *informed* vertices, and the subset $U = V \setminus I$ of *uninformed* vertices. Another set of

particular importance is the set $W = U \cap T$ of *uninformed terminals*. Before the first round of a broadcast schedule all vertices but the source s are uninformed, i.e., $U = V \setminus \{s\}$ and $I = \{s\}$. Also, as s is never a terminal, $W = T$.

A broadcast schedule is a sequence of rounds, and a round can be seen as a matching $M = M(X, Y)$ between a subset $X \subseteq I$ of informed vertices and a subset $Y \subseteq U$ of uninformed ones. Intuitively, the round defined by $M(X, Y)$ is a time slot on which the vertices of X "inform" the vertices of Y using the edges of the matching M. After this time slot the vertices of Y become informed; in other words, they are removed from the subset U and are added to the subset I.

A schedule is called *admissible*, if after its last round all terminals of T are informed. We also say that such a schedule "informs" T. The goal is to devise an admissible schedule with as few rounds as possible.

The *length* of a schedule is the number of rounds it contains. Throughout the paper we use the notation k to denote the cardinality of the set of terminals $|T|$, and the notation br^* to denote the length of the optimum schedule for the instance at hand. Our algorithm assumes the knowledge of the value of br^*; in fact, strictly speaking it accepts as input a guess parameter b. When this guess parameter b is smaller than the actual value of br^*, the algorithm may returns $NULL$ or return a $O(\frac{\log k}{\log br^*})b$ length schedule. If $NULL$ is returned we are guaranteed that $b < br^*$. If $b > br^*$ the algorithm *always* returns a schedule of length which is no greater than $O(\frac{\log k}{\log br^*}) \cdot b$. Hence an actual implementation of our algorithm should conduct a binary search over the value of this guess parameter to obtain the desired approximation ratio.

The graph induced by a set of nodes U is denoted $G(U) = (U, E(U))$. Given a subgraph G', the distance between a pair of nodes u and w in G' is the number of edges in the shortest path connecting u and w in G', and it is denoted $dist_{G'}(u, w)$. For a vertex u, let $\Gamma(u)$ denote the set of neighbors of the vertex u.

For a rooted tree (Q, v), $Q = (V, E)$, ($v \in V$ is the root), the *depth* of Q is the maximum distance in Q from the root v to some leaf z. It is denoted $h(Q)$. The set of leaves of the tree (Q, v) is denoted $L(Q)$. Also, let $B(Q)$ denote the set of vertices of degree at least 3, and $D(T)$ denote the set of vertices of degree exactly 2.

Lemma 2.1 *For a tree Q,*

$$\sum_{v \in B(Q)} (deg(v) - 2) = |L(Q)| - 2 \ .$$

Proof: Let $L = |L(Q)|$, $B = B(Q)$, $D = D(Q)$. Then

$$\sum_{v \in V} deg(v) = |L| + 2|D| + \sum_{v \in B} deg(v) = 2n - 2 \ .$$

Hence

$$\begin{aligned} \sum_{v \in B} deg(v) - 2B &= 2n - 2D - 2B - L - 2 \\ &= 2n - 2D - 2B - 2L + (L-2) \ = \ L - 2 \ . \end{aligned}$$ ∎

2.2 Basic algorithmic tools

Procedure *Busy_Schedule*: An important algorithmic tool that we use extensively is called *busy* schedule. In a busy schedule on each round for each informed node u, we consider its set of neighbors. If there exists an uninformed neighbor of u, then u chooses an *arbitrary* uninformed neighbor and sends it the message. (In [BGN+98] schedules of this kind are called *non-lazy*; these schedules guarantee that as long as they have anything useful to do uninformed vertices are never idle.)

Another tool that we use is called *fork-tree*. Intuitively, a fork-tree is a rooted tree that contains many terminals that are close to the root. The root of a fork-tree is a terminal too.

Definition 2.2 *Given an (undirected) graph $G = (V, E)$, two sets of vertices $U, W \subseteq V$ such that $W \subseteq U$, and a positive integer br^*, the tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ rooted in a terminal $z \in V_\mathcal{T} \cap W$ is a* (U, W, br^*) fork-tree *in G, if $L(\mathcal{T}) \subseteq W$, $V_\mathcal{T} \subseteq U$, $E_\mathcal{T} \subseteq E(U)$, $h(\mathcal{T}) \leq 2 \cdot br^*$, and $|V_\mathcal{T} \cap W| = br^*$.*

When U and W can be understood from the context, we use the shortcut "br^* fork-tree" for (U, W, br^*) fork-tree.

The following lemma provides a motivation for using fork-trees.

Lemma 2.3 *Let Q be an undirected tree rooted at s with leaf set L and depth h. Assume that s is informed. Then Procedure Busy_Schedule forms a broadcast schedule that informs all the vertices of Q within at most $h + |L|$ rounds.*

Proof: Let ℓ be a leaf, v be an ancestor of ℓ, and u the vertex adjacent to v that belongs to the path connecting v with ℓ in Q. All rounds can be divided to two categories. Either v sends the message to u, in which case the round is called a round of type one. Observe that such a round brings the message one edge closer to ℓ. Or v sends the message to a brother w of u, in which case we say that this is a round of type two. Note that if v participates in

a round of type two, and v is not the root, $deg_T(v) \geq 3$. There are at most h rounds of type one, and at most $1 + \sum_{v \in B(T)}(deg(v) - 2) = L - 1$ rounds of type two (by Lemma 2.1). (The extra 1 in the sum accounts for the case that the root of Q has two children.) The lemma follows. ∎

Informing a well-spread set of vertices efficiently: We say that a set $W \subseteq U$ is *well-spread* in $G(U)$ if for every $z, w \in W$, $dist_{G(U)}(z, w) > 2 \cdot br^*$. In our previous paper [EK02] we devised an efficient broadcast schedule for well-spread sets. Specifically, the following lemma from [EK02] will be used in our analysis.

Lemma 2.4 [EK02] *There exists a polynomial algorithm that computes a schedule that starts with a set I of informed vertices, and informs a well-spread subset W of $U = V \setminus I$ within at most $2 \cdot br^*$ rounds. (The algorithm does so if br^* is an upper bound on the actual length of the optimum schedule; otherwise the algorithm returns NULL.)*

3 The algorithm: an informal description

3.1 Overview

Phase 1 of the algorithm: (Procedure $Disjoint\_Trees$): Procedure $Disjoint\_Trees$ operates in the following way. First it finds a br^* fork-tree Q_1 in $G(U)$, removes its vertices from U and iterates on the new $G(U)$. As long as there is a terminal z in $G(U)$ that has at least br^* terminals at distance at most $2 \cdot br^*$ from z (namely, a fork-tree exists), the next fork-tree is built. The procedure outputs a forest $\mathcal{F} = \{Q_1, Q_2, \ldots\}$ of *vertex-disjoint* br^* fork-trees.

Phase 2 (recursion): On this phase we inform the roots of the trees of the forest $\mathcal{F}$. This is done by recursion: we recursively call the main procedure on the roots of $\mathcal{F}$. The stopping condition is when $\mathcal{F}$ has a single tree T with some root r. Then a shortest path from s to r is used. When the recursion terminates we use a busy schedule to deliver the message from these roots to all the vertices of the set $\bigcup_{Q \in \mathcal{F}} V(Q)$.

Note that by Lemma 2.3, it is possible to broadcast efficiently over a single given fork-tree, because its depth is at most br^*, and its number of leaves is at most br^*. Also, as the trees of the forest $\mathcal{F}$ are vertex-disjoint, one can broadcast efficiently *in parallel* throughout all the trees of $\mathcal{F}$.

Phase 3 (Procedure *Intersecting_trees*)**:** We call U a *fork-tree free* set if it does not contain a (U, W, br^*) fork-tree. In other words, for every terminal $z \in W$, the number of uninformed terminals at distance at most $2 \cdot br^*$ from z in $G(U)$ is smaller than br^*. When the first phase terminates, the resulting set U is fork-tree free. Let W be the set of terminals that are not covered by the trees of $\mathcal{F}$.

Procedure *Intersecting_Trees* builds a family $\mathcal{J} = J_1, J_2, \ldots$ of trees. The edge sets of these trees are all contained in $E(U)$. These trees are *not necessarily vertex-disjoint*. However, later we will transform the family $\mathcal{J}$ into a forest.

The procedure iteratively picks a terminal $z \in W$, and builds a shortest path tree J_z in $G(U)$ rooted at z that spans *all* the terminals of W at distance at most $2 \cdot br^*$ from z. Now, (unlike Procedure *Disjoint_Trees*), the procedure *does not* remove J_z from U. It picks a terminal $w \notin J_z$, and let J_w be the shortest path tree in $G(U)$ rooted at w that spans all the terminals of W at distance $2 \cdot br^*$ or less from w.

We continue this way until all the terminals of W are covered by the trees of $\mathcal{J}$. Note that as all the trees of $\mathcal{J}$ are built in the same graph $G(U)$, they are not necessarily disjoint.

Note that $W \subseteq \bigcup_{J \in \mathcal{J}} V(J)$. Hence, a schedule that informs all the vertices of the set $\bigcup_{J \in \mathcal{J}} V(J)$, informs all the terminals of W.

Informing $\bigcup V(J)$**:** This task can be divided into two subtasks. The first subtask is informing the roots of the trees of $\mathcal{J}$. It is easy to see that these roots are well-spread in U. Hence, by Lemma 2.4, they can be informed within at most $2 \cdot br^*$ rounds. Thus, we are left with the subtask of informing the vertices of $\bigcup_{J \in \mathcal{J}} V(J)$ assuming that the roots are already informed.

By construction, each tree $J \in \mathcal{J}$ satisfies $depth(J) \leq 2 \cdot br^*$ and $|L(J)| \leq br^*$. However, since the trees of $\mathcal{J}$ may intersect, a busy schedule may fail to inform all the vertices of $\bigcup_{J \in \mathcal{J}} V(J)$ within a small number of rounds. We next show that $\mathcal{J}$ can be transformed into a forest (a family of vertex-disjoint trees) $\mathcal{P}$ that covers the entire set W, and moreover, such that each tree P of $\mathcal{P}$ satisfies $depth(P) \leq 2 \cdot br^*$, $|L(P)| \leq br^*$. Once this is proved, we use Lemma 2.3 to show that a busy schedule can be utilized to inform W efficiently. This transformation is performed by Procedure *Make_Disjoint* that we describe next.

Procedure *Make_Disjoint*: For a rooted tree (J, r) and a vertex $v \in V(J)$, the *level* of v in J, denoted $level_J(v)$, is defined as follows. The level of the root r is 0, and the level of $v \neq r$ is the level of its parent in (J, r) plus one. (The *parent* of v in (J, r), denoted $par_J(v)$,

is the neighbor w of v that lies on the path connecting r and v in J.) Also, for a vertex $v \in \bigcup_{J\in\mathcal{J}} V(J)$, let the *min-level* of v be $minlevel(v) = \min\{level_J(v) \mid J \in \mathcal{J}\}$. For a vertex v that appears in more than one tree $J \in \mathcal{J}$, let $par^*(v)$ be the parent of v in the tree $J \in \mathcal{J}$ in which v has the smallest level (i.e., $level_J(v) = minlevel(v)$).

Let $roots(\mathcal{J})$ denote the set of roots of all trees of $\mathcal{J}$. Let $\mathcal{P} = \{(par^*(v), v) \mid v \in V(\mathcal{J}) \setminus roots(\mathcal{J})\}$. As will be argued below $\mathcal{P}$ is a forest. Note that while all leaves of trees of $\mathcal{J}$ are terminals of W, this is not necessarily true for leaves of trees of $\mathcal{P}$. Leaves that are not terminals of W are next iteratively discarded. In other words, we remove every vertex whose entire subtree in $\mathcal{P}$ contains no terminals of W.

We next analyze some properties of the set $\mathcal{P}$.

Lemma 3.1 *$\mathcal{P}$ is a forest.*

Proof: Suppose for contradiction that $\mathcal{P}$ contains a cycle $C = (v_0, v_1, \ldots, v_h = v_0)$. For each edge, $e = (u, w) \in C$, orient it as $\langle w, u\rangle$ if $w = par^*(u)$. Observe that C is an oriented cycle, as by definition of $\mathcal{P}$, the in-degree of every vertex in $\mathcal{P}$ is at most 1.

Suppose without loss of generality that the arcs of C are oriented in the order of increasing indices, that is, $\langle v_0, v_1\rangle, \langle v_1, v_2\rangle, \ldots, \langle v_{h-1}, v_0\rangle$. It follows that $minlevel(v_0) < minlevel(v_1) < minlevel(v_2) < \ldots < minlevel(v_0)$, contradiction. ∎

The latter lemma shows that the edge set $\mathcal{P}$ decomposes into a family of vertex-disjoint trees. In the next lemma we show that each tree $P \in \mathcal{P}$ contains exactly one root $r = r(P)$ of a tree $J \in \mathcal{J}$, and vice versa.

Lemma 3.2 *Each $P \in \mathcal{P}$ contains exactly one root $r = r(P)$ of a tree $J \in \mathcal{J}$, and each tree $J \in \mathcal{J}$ contains exactly one root $r = r(J)$ of a tree $P \in \mathcal{P}$.*

Proof: Let r be the vertex of P with minimum $minlevel$ (that is, $r = argmin\{minlevel_{\mathcal{J}}(v) \mid v \in P\}$). Suppose for contradiction that $minlevel_{\mathcal{J}}(r) > 0$. But then the vertex $v = par^*(r)$ satisfies $minlevel_{\mathcal{J}}(v) = minlevel_{\mathcal{J}}(r) - 1 \geq 0$, and $v \in P$ (as the edge $(v, r) \in \mathcal{P}$). This is a contradiction.

Hence $minlevel_{\mathcal{J}}(r) = 0$, and so r is a root of some tree $J \in \mathcal{J}$. Suppose for contradiction that P contains two vertices r, r' such that both of them are roots of trees $J, J' \in \mathcal{J}$. Let $\Pi = (r = w_0, w_1, \ldots, w_h = r')$ be the path in P connecting r and r'. For each edge $(u, w) \in P$ with $u = par^*(w)$, orient the edge from w to u. By the same argument as in the proof of Lemma 3.1, the path Π is oriented consistently either from r to r' or from r' to r. Suppose without loss of generality that it is oriented from r to r'. But then $minlevel(r') > minlevel(r)$, but $minlevel(r') = minlevel(r) = 0$, contradiction.

For the opposite direction, consider a tree $J \in \mathcal{J}$, and let r be its root. Observe that $\bigcup_{P\in\mathcal{P}} V(P) = \bigcup_{J\in\mathcal{J}} V(J)$, and so there exists a tree $P \in \mathcal{P}$ such that $r \in V(P)$. Note that $minlevel_{\mathcal{J}}(r) = 0$. By previous argument, P cannot contain another vertex r' with *minlevel* 0, and thus r is the only root of a tree of $\mathcal{J}$ contained in the tree P. ∎

Lemma 3.3 *For every tree $P \in \mathcal{P}$,*

$$|L(P)| \leq br^* , \tag{1}$$

$$depth(P) = O(br^*) . \tag{2}$$

Proof: For inequality (1) note that all the fork-trees were extracted from U on previous phases of the algorithm. Recall that U is the set after the fork-trees phase ends. In particular $G(U)$ admits no br^* for-tree. Hence for any terminal the number of terminals of distance at most $2br^*$ in $G(U)$ is at most br^*. Since every leaf is a terminal, the claim follows.

We next prove by induction on the *minlevel* of a vertex v in $\mathcal{J}$ that $level_{\mathcal{P}}(v) \leq minlevel_{\mathcal{J}}(v)$. (As $\mathcal{P}$ is a forest, $level_{\mathcal{P}}(v)$ is the level of v in the unique tree of the forest $\mathcal{P}$ that contains v.) This will complete the proof of inequality (2) because the depth of a tree is the level of its farthest leaf from the root and the depth of every tree in $\mathcal{J}$ is $O(br^*)$.

The induction base is that v is a root of a tree $J \in \mathcal{J}$. By Lemma 3.2, v is also a root of a tree $P \in \mathcal{P}$, and so $0 = level_{\mathcal{P}}(v) = minlevel_{\mathcal{J}}(v)$.

The induction step:
Consider a vertex v with $minlevel_{\mathcal{J}}(v) = h$, where h is a positive integer. Let $J \in \mathcal{J}$ be the tree such that $level_J(v) = minlevel_{\mathcal{J}}(v) = h$, and let $w = par^*(v) = parent_J(v)$. By induction hypothesis, $level_{\mathcal{P}}(w) \leq minlevel_{\mathcal{J}}(w) = h - 1$. Also, by construction, the edge $(par^*(v), v) = (w, v)$ belongs to $\mathcal{P}$. Consequently, the distance between the root r of P and the vertex w is at most $h - 1$, and so the distance between r and v is at most h, and so $level_{\mathcal{P}}(v) \leq h = minlevel_{\mathcal{J}}(v)$. ∎

4 The formal description of the algorithm

Algorithm *Undirected_Multicast*

Input: A graph $G = (V, E)$, a source $s \in V$, a set of terminals $T \subseteq V \setminus \{s\}$, a positive integer br^*.

Output: A multicast schedule that informs T assuming that s was informed before the first round of the schedule.

1. $\mathcal{R} \leftarrow \emptyset$; /* The set of roots of the forest computed in Procedure $Disjoint\_Trees$ */
2. $U \leftarrow V \setminus \{s\}$; /* The set of uninformed vertices */
3. $W \leftarrow T$; /* The set of uninformed terminals */
4. $\mathcal{F} \leftarrow \emptyset$; /* The forest of (vertex-disjoint) (U, W, br^*) fork-trees */
5. $I \leftarrow \{s\}$; /* The set of informed vertices */
6. **While** there exists a (U, W, br^*) fork-tree in $G(U)$ **do**
   (a) Compute the shortest path tree Q in $G(U)$ rooted at some $u \in W$, having depth at most $2 \cdot br^*$ and containg exactly br^* terminals of W.
   (b) $\mathcal{F} \leftarrow \mathcal{F} \cup \{Q\}$;
   (c) $U \leftarrow U \setminus V(Q)$; /* The nodes of Q can no longer be used for other (U, W, br^*) fork-trees. */
   (d) $I \leftarrow I \cup V(Q)$;
   (e) $W \leftarrow W \setminus V(Q)$;
   (f) $\mathcal{R} \leftarrow \mathcal{R} \cup \{u\}$;
7. If $\mathcal{R}$ contains a single root r then send the message from s to r in a shortest path.
8. Else inform $\mathcal{R}$ recursively;
9. Invoke $Busy\_Schedule(G(\mathcal{F}), \mathcal{R})$ to inform the vertices of $V(\mathcal{F})$ assuming that vertices of $\mathcal{R}$ are already informed;
10. $\mathcal{J} \leftarrow Intersecting\_Trees(U, W, br^*)$;
    /* Forming a family of not necessarily vertex-disjoint trees; see Phase 3 in Section 3.1. */
11. $\mathcal{R}' \leftarrow Roots(\mathcal{J})$;
12. Use the schedule formed in Lemma 2.4 to inform $\mathcal{R}'$;
13. Invoke Procedure $Make\_Disjoint$ with input $\mathcal{J}$; /* Transform $\mathcal{J}$ into a forest $\mathcal{P}$; see the description in Section 3.1 just before Lemma 3.1. */
14. Invoke Procedure $Busy\_Schedule$ with inputs $\bigcup_{P \in \mathcal{P}} V(P)$ and $\mathcal{R}'$ to inform the vertices of $\bigcup_{P \in \mathcal{P}} V(P)$ assuming that the vertices of $\mathcal{R}'$ are already informed;
    /* See Section 2.2. */

4.1 The approximation ratio

In this section we analyze the approximation ratio of our algorithm.

We now prove that Algorithm $Undirected\_Multicast$ is an
$O(\log k/\log br^*) = O(\log k/\log\log k)-$approximation algorithm for the telephone multicast problem on undirected graphs, where $k = |T|$.

Phases 1 and 2: Recall that $\mathcal{F}$ is the forest formed on Phase 1, and $\mathcal{R}$ is the set of roots of trees of $\mathcal{F}$.

The terminals of $\mathcal{R}$ are informed via recursion. We provide an upper bound on the number of rounds used on phases 1 and 2 of any given recursive invocation, and multiply this bound by the depth of the recursion tree. In what follows we analyze the number of rounds required to inform the vertices of $V(\mathcal{F})$ assuming that the vertices of $\mathcal{R}$ are already informed, on a fixed recursive invocation.

Lemma 4.1 *The time required to inform $V(\mathcal{F})$ assuming that all the vertices of $\mathcal{R}$ are already informed is $O(br^*)$.*

Proof: Recall that $\mathcal{F}$ is a forest of (U, W, br^*) fork-trees. It follows that the depth of a tree $Q \in \mathcal{F}$, and $|L(Q)| \leq 2 \cdot br^*$. The lemma follows now from Lemma 2.3, and because the trees are vertex-disjoint. ∎

Lemma 4.2 *For a graph $G = (V, E)$, a vertex $s \in V$, a set of terminals $T \subseteq V$, and an integer br^*, consider the invocation $Undirected\_Multicast(G, s, T, br^*)$. The set $\mathcal{R}$ formed by the while loop (step 6) satisfies $|\mathcal{R}| \leq |T|/br^*$.*

Proof: Note that every tree $Q \in \mathcal{F}$ is a (U, W, br^*) fork-tree, and thus contains exactly br^* uninformed terminals. Moreover, these trees are vertex-disjoint. Only one of these terminals, namely, the root of Q, is inserted into the set $\mathcal{R}$. The lemma follows. ∎

Consider the depth of the recursion. The recursion forms a recursion path as there is at most one recursion in every call for Algorithm $Undirected\_Multicast$. By Lemma 4.2, after i recursive calls, the number of roots in the new $\mathcal{R}$ is at most $|T|/br^i$. Thus after $O(\log k/\log br^*)$ calls or less, the number of roots is 1 and the recursion halts. This means that the recursion depth is $O(\log k/\log br^*)$. In summary, as every call for phases 1 and 2 requires $O(br^*)$ rounds the number of rounds used by phases 1 and 2 over all different recursive invocations is at most
$O(\log k/\log br^*) \cdot br^*$.

Phase 3: We next show that the number of rounds used by the busy schedule formed on step 14 of Algorithm *Undirected_Multicast* is $O(br^*)$. It will follow that on any recursive invocation, phase 3 requires $O(br^*)$ rounds. The desired approximation ratio will follow from the upper bound of $O(\log k / \log br^*)$ on the depth of the recursion.

We need two following lemmas. (Recall that $\mathcal{R}'$ is the set of roots of the trees of $\mathcal{J}$.)

Lemma 4.3 *For each pair v, v' of vertices of $\mathcal{R}'$,*

$$dist_{G(U)}(v, v') \geq 2 \cdot br^* + 1 \ . \tag{3}$$

(In inequality (3), "U" refers to the state of the set U after the execution leaves the while loop on step 6 of Algorithm Undirected_Multicast.)

Proof: Let $J, J' \in \mathcal{J}$ be the pair of trees rooted at vertices v and v', respectively. Assume without loss of generality that the tree J was formed before J'. By construction, J contains all the terminals of W that are at distance at most $2 \cdot br^*$ from v in $G(U)$. Moreover, by construction $v' \notin J$. Hence, $dist_{G(U)}(v, v') \geq 2 \cdot br^* + 1$. ∎

Observe also that the set of roots of $\mathcal{P}$ is equal to the set of roots of $\mathcal{J}$, and so the inequality (3) applies to any pair of roots v, v' of trees of $\mathcal{P}$ as well.

By Lemmas 4.3 and 2.4, it follows that the schedule informs $\mathcal{R}'$ in $O(br^*)$ rounds. Recall also that by Lemma 3.3, for every tree $P \in \mathcal{P}$, $|L(P)| \leq br^*$.

Since $\mathcal{P}$ is a forest (by Lemma 3.1), and each tree $P \in \mathcal{P}$ has depth $O(br^*)$ and at most $O(br^*)$ leaves, it follows that a busy schedule informs $\bigcup_{P \in \mathcal{P}} V(P)$ in $O(br^*)$ rounds. This proves that the algorithm provides the approximation ratio of $O(\frac{\log k}{\log br^*})$.

A straightforward implementation of the algorithm requires $\tilde{O}(|V| \cdot |E|)$ time. In addition, the subroutine from [EK02] used to inform a well-spread set of vertices (see Lemma 2.4) requires $\tilde{O}(|V| \cdot |E|)$ time. We remark also that the direct implementation of our algorithm does not use Linear Programming. Our main result follows.

Theorem 4.4 *Algorithm Undirected_Multicast is an $O(\log k / \log br^*)$-approximation algorithm for the telephone multicast problem that requires $\tilde{O}(|V| \cdot |E|)$ time. As $br^* \geq \log k$, the algorithm also provides the approximation ratio of $O(\log k / \log \log k)$.*

5 Extensions

5.1 The edge-dependent heterogeneous postal model

Bar-Noy et al. [BGN+98] suggested a more general model of telephone communication that they termed *postal model*. In that model each vertex v has a delay parameter $\rho(v)$, which is a real number between 0 and 1. The vertex that sends a message at time moment t is considered to be *busy* during the interval $[t, t + \rho(v)]$, and it is not allowed to send any new messages while it is busy. In addition, in the postal model there is a delay ℓ_e associated with every edge $e \in E$. This delay represents the time elapsing between the moment that one endpoint of e sends a message and the moment that the other endpoint receives it. In our previous paper [EK02] we introduced a yet more general model of telephone communication, specifically, the *edge-dependent heterogeneous postal model*. In that model the delay of a vertex v depends on the edge through which it chooses to send the message.

Our approximation algorithm applies directly to the edge-dependent heterogeneous postal model. However, for this more general model we need to use the algorithm of [LST90] to establish Lemma 2.4. The algorithm [LST90] uses Linear Programming. In the case of the non-edge-dependent heterogeneous postal model (uniform delay over the edges adjacent to a fixed vertex) the extended version of Lemma 2.4 can be proved using flow techniques (the use of [LST90] can be avoided).

With regards to other parts of the analysis, we note that it carries through. The algorithm should use weighted shortest paths (delays of edges serve as distances) of weighted depth at most $2 \cdot br^*$. As in the telephone model, the number of leaves close to any terminal should be at most $O(br^*)$.

In [EK02] we have shown how to extend the combinatorial logarithmic approximation algorithm for the problem to these more general telephone communication models. The extension of our current algorithm to the edge-dependent heterogeneous model is fully analogous to the one described in [EK02].

5.2 Implications for network design

Assume that G admits a Steiner tree of depth h and maximum degree d. Any multicast schedule, naturally defines a multicast tree with the parent of v being the node that informs v. The only difference is that we use d instead of br^* as a bound on the maximum degree of the optimum tree, and use h as a bound on the depth. Theorem 4.4 implies that the algorithm

constructs a tree of depth $O(\log k/\log\log k)\cdot h$ and of maximum degree $O(\log k/\log\log k)\cdot d$.

Corollary 5.1 *The depth-degree spanning tree problem admits an* $(O(\log k/\log\log k), O(\log k/\log\log k))$ *bicriteria approximation. Consequently, the minimum poise problem admits an* $O(\log k/\log\log k)$ *approximation.*

References

[BGN+98] A. Bar-Noy, S. Guha, J. Naor and B. Schieber. Multicasting in Heterogeneous Networks. In *SIAM J. Computing* 30(2): 347-358 (2000)

[BK94] A. Bar-Noy and S. Kipnis, Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems, *Mathematical System Theory*, Vol. 27, pp. 431–452, 1994.

[EK02] M. Elkin and G. Kortsarz, A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem, *34th Symp. on Theory of Computing, 2002*, pp.438-447.

[EK03] M. Elkin and G. Kortsarz. A sublogarithmic approximation algorithm for the undirected telephone broadcast problem: A path out of a jungle. In *Proc. Symp. on Discr. Algorithms*, SODA'03, pp. 76-85, Baltimore, MA, Jan. 2003.

[F01] P. Fraigniaud Approximation Algorithms for Minimum-Time Broadcast under the Vertex-Disjoint Paths Mode In *9th Annual European Symposium on Algorithms (ESA '01)*, LNCS Vol. 2161, pages 440-451, 2001.

[HHL88] S. Hedetniemi, S. Hedetniemi, and A. Liestman. A survey of broadcasting and gossiping in communication networks. *Networks*, 18: 319-349, 1988.

[KP95] G. Kortsarz and D. Peleg. Approximation algorithms for minimum time broadcast, *SIAM journal on discrete methods*, vol 8, pages 401-427, 1995.

[LST90] L. K. Lenstra and D. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Math. Programming*, 46, 259-271. (1990).

[M93] M. Middendorf. Minimum Broadcast Time is NP-complete for 3-regular planar graphs and deadline 2. *Inf. Process. Lett.* 46 (1993) 281-287.

[MRR+93] M. V. Marathe, R. Ravi, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Many birds with one stone: Multi-objective approximation algorithms. In *Proc. ACM Symp. on the Theory of Computing (STOC '93)*, pp. 438-447, 1993.

[R94] R. Ravi Rapid rumor ramification: Approximating the minimum broadcast time. *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS '94)*, 202-213 (1994).

[S00] C. Schindelhauer, On the Inapproximability of Broadcasting Time, In *The 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'00)*, 2000