A Linear Algorithm for Finding the *k*-Broadcast Center of a Tree

Hovhannes A. Harutyunyan
Department of Computer Science, Concordia University, Montreal, Quebec, Canada H3G 1M8

Arthur L. Liestman
School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6

Bin Shao
Microsoft Corporation, 1 Microsoft Way, Redmond, Washington 98052-6399

The term *k*-broadcast indicates the process of disseminating a message from one vertex to all vertices of a graph in such a way that in each time unit, an informed vertex can send the message to up to *k* of its neighbors. The *k*-broadcast center of a graph is the set of vertices that can initiate a minimum time *k*-broadcast within the graph. We present a linear algorithm to determine the *k*-broadcast center of a given tree. From this, we obtain a linear time algorithm for finding the *k*-broadcast time of any vertex of the tree and, thus, the *k*-broadcast time of the tree itself. © 2008 Wiley Periodicals, Inc. NETWORKS, Vol. 53(3), 287–292 2009

Keywords: broadcasting; k-broadcasting; center; graph; linear algorithm

1. INTRODUCTION

Broadcasting is the message dissemination process in which a message is sent from one vertex, the *originator*, to all vertices of a graph by a series of calls over the edges of the graph. Such a series of calls is called a *broadcast scheme*. In any given call, an informed vertex relays the message to the other participant of the call. We assume that each call can only involve one informed vertex and one of its uninformed neighbors.

More generally, for a given positive integer k, *k-broadcasting* is a similar process in which each call involves an informed vertex relaying the message to at most k of its neighbors. The process of *broadcasting*, as described earlier, is called 1-*broadcasting* in this terminology. A *k-broadcast scheme* is a series of calls that accomplishes a k-broadcast. This model allows for "conference calls" of a limited maximum size k [10]. It is useful in the study of DMA-bound systems [23] and also has application in computing functions in networks [1, 3, 5].

Most of the previous work on k-broadcasting has been for $k = 1$. For surveys of results on k-broadcasting and related problems, see Hedetniemi et al. [17], Fraigniaud and Lazard [8], Hromkovič et al. [18], and Hromkovič et al. [19]. Much of the research has involved the design of graphs that allow minimum time k-broadcasting [10, 11, 13, 21, 24–26]. Other recent papers focus on the design of trees that are optimal for k-broadcasting [12, 26, 29].

Here, we are interested in the time required to k-broadcast. We assume that each call requires one unit of time and that a vertex can participate in at most one call per unit of time. However, calls involving disjoint sets of vertices can be made in parallel, that is, at the same time. A *round* is a set of calls that are made in parallel in one time unit. We use the number of rounds required by a k-broadcast scheme to measure the time of that scheme.

The *k-broadcast time $b_k(u, G)$ of an originator u*, or simply $b_k(u)$, is the minimum time over all k-broadcast schemes for originator u in graph G. Slater et al. [30] showed that to determine $b_1(u, G)$ for an arbitrary vertex u in an arbitrary graph G is NP-complete. Jakoby et al. [20] showed that the problem remains NP-complete when restricted to graphs with bounded degree at least 3 or when restricted to planar graphs of degree 3. Slater et al. [30] also gave an algorithm to determine $b_1(u, T)$ for an arbitrary vertex in an arbitrary tree T. Their algorithm runs in time linear in the size of T. Harutyunyan and Maraachlian [14] gave a linear time algorithm to find $b_1(u, G)$ for an arbitrary vertex u in an arbitrary unicyclic graph G. Dessmark et al. [4] gave efficient algorithms for 1-broadcasting in almost trees and in partial k-trees. These are the only algorithms for finding the exact 1-broadcast time of a vertex in a graph of a certain type. There are some

Received September 2007; accepted June 2008
Correspondence to: A. L. Liestman; e-mail: art@cs.sfu.ca
DOI 10.1002/net.20270
Published online 10 October 2008 in Wiley InterScience (www.interscience.wiley.com).

approximation algorithms [6, 7, 9, 22, 27, 28] and heuristics [2, 16] for determining the 1-broadcast time of a vertex in an arbitrary graph.

The *k-broadcast number of a graph* G is $b_k(G) = \min\{b_k(u, G) | u \in V\}$, that is, the minimum k-broadcast time for any originator in the graph. (This is not to be confused with the *k-broadcast time of a graph* which is the maximum of those values.) We use the term *k-broadcast center* to denote those vertices whose k-broadcast time is equal to the k-broadcast number of the graph. For graph G, we use $BC_k(G)$ to denote the k-broadcast center of graph G, that is, $BC_k(G) = \{u \in V | b_k(u) = b_k(G)\}$.

Slater et al. [30] gave an algorithm to determine the 1-broadcast center of a tree. For any tree, the 1-broadcast center is a complete bipartite graph $K_{1,m}$ for some integer $m \geq 1$.

Here, we present a linear algorithm to determine the k-broadcast center of a tree. This yields a linear algorithm for finding the k-broadcast time of an arbitrary vertex of a tree. In Section 2, we describe the structure of $BC_k(T)$ for an arbitrary tree T and show how to k-broadcast in minimum time from an arbitrary vertex of T given its k-broadcast center. In Section 3 we give the algorithm, prove its correctness, and show that it is linear.

2. PROPERTIES OF $BC_K(T)$

As we will need to repeatedly refer to certain subtrees of the given tree T, we will use $T_{v,w}$ to denote the connected component of $T \setminus \{(v, w)\}$ that contains v. We let $\Delta(G)$ denote the maximum degree of the graph G and simply use Δ if the graph is clear from context. We use $\epsilon(G)$ to denote the eccentricity of the graph G and simply use ϵ if the graph is clear from context.

We begin with several obvious facts and simple observations about k-broadcasting in trees.

Because a message can move along a path by no more than one edge per time unit, $b_k(G) \geq \epsilon(G)$.

If k is sufficiently large, then the message can be sent to all of the originator's neighbors at time 1, to all of their neighbors at time 2, and so on. Thus, if $k \geq \Delta$, then $b_\Delta(G) = b_k(G) = \epsilon(G)$.

Any scheme for k is also valid for $k + 1$, so $b_1(G) \geq b_2(G) \geq \cdots \geq b_\Delta(G) = b_{\Delta+i}(G) = \epsilon$ for all $i \geq 1$. It follows that there is a $k_0 \leq \Delta$ such that $b_{k_0}(G) = b_{k_0+1}(G) = \cdots = \epsilon$.

For any graph G on n vertices, $b_k(G) \geq \lceil \log_{k+1} n \rceil$.

Observation 1. *Let v and w be adjacent vertices in T, $k \geq 1$, and assume that $b_k(v, T_{v,w}) \leq b_k(w, T_{w,v})$. Then $b_k(v, T) = 1 + b_k(w, T_{w,v})$ and $b_k(w, T_{w,v}) \leq b_k(w, T) \leq 1 + b_k(w, T_{w,v})$.*

Observation 2. *If v and w are adjacent vertices in T, $k \geq 1$, and $b_k(w, T_{w,v}) \leq b_k(v, T_{v,w})$, then for all vertices x of $T_{w,v}$, $b_k(x, T) \geq b_k(v, T)$.*

The next two results appeared in different form in an earlier paper [15] of two of the authors. These results are useful in developing the algorithm and proving its correctness.

Theorem 1. *For $k \geq 1$, the $BC_k(T)$ of an arbitrary tree T is either a single vertex or a complete bipartite graph $K_{1,m}$ for some integer $m \geq 1$.*

Note that although there are at least two vertices in $BC_1(T)$ for any tree T, when $k \geq 2$, $BC_k(T)$ can be a single vertex.

The following is an analog of Theorem 9 of Slater, et al. [30].

Theorem 2. *For $k \geq 1$ and any tree T, given a vertex v that does not belong to $BC_k(T)$, $b_k(v, T) = b_k(u, T) + dist(v, u)$ where u is the vertex of $BC_k(T)$ nearest v and $dist(v, u)$ is the distance from v to u.*

3. AN ALGORITHM TO DETERMINE THE *k*-BROADCAST CENTER

Slater et al. [30] presented an algorithm (called BROADCAST) to determine the 1-broadcast center of an arbitrary tree T. Their algorithm assigns labels to each vertex in the tree beginning by assigning the label 0 to each of the leaves. The label assigned to a vertex u indicates the time required to 1-broadcast from u to the largest subtree of T rooted at u consisting of vertices that have already been labeled. The algorithm "sweeps in" from the leaves, assigning increasing labels to vertices. In each step, the algorithm labels one vertex for which all but one of its neighbors are already labeled. The order in which the labels are assigned and the values of those labels ensure that the label indicates the 1-broadcast time in the corresponding subtree and, ultimately, leads to a particular vertex in the 1-broadcast center of T. With this information, one can easily determine the other vertices in the 1-broadcast center.

3.1. The Algorithm KBC

We now present an algorithm KBC (k-broadcast center) to determine the k-broadcast center of a given tree T. Our algorithm is based on BROADCAST [30]. It considers the vertices of the tree beginning with the leaves and moving inwards. Each vertex u receives a label $l(u)$ during this process. The value of $l(u)$ indicates the time required for u to k-broadcast to the vertices of a particular subtree of T.

The algorithm KBC first labels all leaves of T with 0. Let T' denote the subtree of T obtained by removing all the leaves of T. KBC labels each leaf of T' as follows: Given a leaf u of T' and its p labeled neighbors (from $T \setminus T'$) c_1, $c_2, \ldots, c_p$, where $l(c_1) \geq l(c_2) \geq \cdots \geq l(c_p)$, then $l(u) = \max_{1 \leq i \leq \lceil \frac{p}{k} \rceil} \{l(c_{(i-1)k+1}) + i\}$. After u is labelled, it is removed from T' and the process repeats.

Intuitively, u can k-broadcast to all of the vertices in the subtree rooted at u with $c_1, c_2, \ldots, c_p$ as its children. To do this, u would send the message first to $c_1, c_2, \ldots, c_k$ at time 1, to $c_{k+1}, c_{k+2}, \ldots, c_{2k}$ at time 2, and so on.

Algorithm ***KBC***
Input: tree T, integer k ($k \geq 2$)
Output: $BC_k(T)$ and $b_k(T)$

0. If $|V(T)| \leq 2$ then $BC_k(T) \leftarrow V$
1. $T' \leftarrow T$
2. For each leaf v of T
   2.1. $l(v) \leftarrow 0$
   2.2. $T' \leftarrow T' - v$
3. For each leaf u of T', $l(u) \leftarrow \max_{1 \leq i \leq \lceil \frac{p}{k} \rceil} \{l(c_{(i-1)k+1}) + i\}$, where $c_1, c_2, \ldots, c_p$ are the p labeled vertices adjacent to u ordered such that $l(c_1) \geq l(c_2) \geq \cdots \geq l(c_p)$.
4. While ($|V(T')| \geq 2$)
   4.1. Choose $v \in V(T')$ such that $l(v) = \min\{l(u) | u \in V(T')\}$.
   4.2. $T' \leftarrow T' - v$
   4.3. Let s be the vertex of T' adjacent to v. If s is a leaf of T', then $l(s) \leftarrow \max_{1 \leq i \leq \lceil \frac{p}{k} \rceil} \{l(c_{(i-1)k+1}) + i\}$, where $c_1, c_2, \ldots, c_p$ are the p labeled vertices adjacent to s ordered such that $l(c_1) \geq l(c_2) \geq \cdots \geq l(c_p)$.
5. Let w be the only vertex in T' and let $c_1, c_2, \ldots, c_p$ be its labeled neighbors ordered such that $l(c_1) \geq l(c_2) \geq \cdots \geq l(c_p)$.
   5.1. $l(w) \leftarrow \max_{1 \leq i \leq \lceil \frac{p}{k} \rceil} \{l(c_{(i-1)k+1}) + i\}$.
   5.2. $BC_k(T) \leftarrow \{w\}$.
6. Let j be the smallest integer $1 \leq j \leq p$ such that $l(c_j) + \lceil \frac{j}{k} \rceil = l(w)$.
   6.1. $BC_k(T) \leftarrow \{w\}$.
   6.2. If $j \equiv 1 \mod k$ and for all $i \not\equiv 1 \mod k$ with $j < i \leq p$ it happens that $l(c_i) + \lceil \frac{i}{k} \rceil \neq l(w)$, then $BC_k(T) = \{w, c_1, c_2, \ldots, c_j\}$.

Figure 1 illustrates the execution of KBC on a particular tree T with $k = 2$. In each depiction of the trees, the vertices of $T \setminus T'$ are indicated in black and the labels known at that point in the execution are shown. Figure 1a shows the result of step 2 of KBC. The leaves of T have been labeled and removed from T'. Figure 1b shows the result of step 3. The leaves of the new tree T' have been labeled. Figure 1c shows the result after four executions of step 4. Four more vertices have been removed from T' and new labels have been assigned. Figure 1d shows the result after step 5. Only one vertex, w, remains in T'. In step 6, the algorithm determines that $BC_2(T) = \{w, a, b, c\}$ which is shown in Figure 2 with the vertices of the 2-broadcast center indicated in black.

3.2. Proof of Correctness

Note that at each step of KBC, T' is a connected subtree of T. We will assume that $|V(G)| \geq 3$ as the case $|V(G)| \leq 2$ is easily dealt with in step 0 of KBC.

The following results are generalizations of results of Slater, et al. [30].

Theorem 3. *Let $v_1, v_2, \ldots, v_p$ be the neighbors of a vertex w in T. For $k \geq 1$, if $b_k(v_i, T_{v_i,w}) \geq b_k(v_{i+1}, T_{v_{i+1},w})$ for $i = 1, 2, \ldots, p-1$, then $b_k(w, T) = \max_{1 \leq i \leq p} \{b_k(v_i, T_{v_i,w}) + \lceil \frac{i}{k} \rceil\}$.*

Proof. Clearly, $b_k(w, T) \leq \max_{1 \leq i \leq p} \{b_k(v_i, T_{v_i,w}) + \lceil \frac{i}{k} \rceil\}$ as we can k-broadcast to T by sending the message to $v_1, v_2, \ldots, v_k$ at time 1, to $v_{k+1}, v_{k+2}, \ldots, v_{2k}$ at time 2, and so on, giving the upper bound.

To show that $b_k(w, T) \geq \max_{1 \leq i \leq p} \{b_k(v_i, T_{v_i,w}) + \lceil \frac{i}{k} \rceil\}$, we observe that $b_k(w, T) \geq b_k(v_i, T_{v_i,w}) + \lceil \frac{i}{k} \rceil$ for any $1 \leq i \leq p$. If v_i is called earlier than at time $\lceil \frac{i}{k} \rceil$, then some v_j where $1 \leq j < i$ must be called at time $\lceil \frac{i}{k} \rceil$ or later. Then, for this scheme, the k-broadcast cannot finish before time $b_k(v_j, T_{v_j,w}) + \lceil \frac{i}{k} \rceil \geq b_k(v_i, T_{v_i,w}) + \lceil \frac{i}{k} \rceil$ since $j < i$ implies that $b_k(v_j, T_{v_j,w}) \geq b_k(v_i, T_{v_i,w})$. The result follows. ∎

Thus, for any $k \geq 1$ there is a minimum time k-broadcast scheme for u in T such that u sends the message to its neighbors $v_1, v_2, \ldots, v_k$ at time 1, to $v_{k+1}, v_{k+2}, \ldots, v_{2k}$ at time 2, and so on.

The following two observations about our algorithm follow from Theorem 3.

Observation 3. *If $v_1, v_2, \ldots, v_{m-1}$ is the sequence of vertices selected in step 4.1 and w is the only remaining vertex of step 5, then $l(v_1) \leq l(v_2) \leq \cdots \leq l(v_m)$.*

Observation 4. *When v is deleted in step 4.2, $l(v) = b_k(v, T_{v,s})$ where s is the vertex of T' adjacent to v in step 4.3.*

The following lemma is analogous to Lemma 5 of Slater et al. [30].

Lemma 1. *At any stage after step 0 in the algorithm KBC, some vertex in T' is in the k-broadcast center of T.*

Proof. Initially, all vertices of T are in T'. Vertices are removed from T' only in steps 2.2 and 4.2. Let v be the vertex being removed. Let s be v's neighbor in T'.

We must show that $b_k(v, T) \geq b_k(s, T)$.

If T' contains only two vertices v and s where $l(v) \leq l(s)$ then either $l(v) = l(s)$ or $l(v) < l(s)$. In the former case, both vertices are in $BC_k(T)$. In the latter case, Observation 1 says that $b_k(s, T) \leq b_k(v, T)$.

Otherwise, T' contains at least 3 vertices. First, we show that $b_k(v, T_{v,s}) \leq b_k(s, T_{s,v})$. If $BC_k(T) = \{v, s\}$, then $l(v) = b_k(v, T_{v,s})$, $l(s) = b_k(s, T_{s,v})$, and $l(v) \leq l(s)$. Thus, $b_k(v, T_{v,s}) \leq b_k(s, T_{s,v})$.

Let x be another leaf of T' and let y be the neighbor of x in T'. (Note that y might be s.) We know that $l(v) \leq l(x)$, in other words $b_k(v, T_{v,s}) \leq b_k(x, T_{x,y})$. Because $T_{x,y}$ is a subtree of $T_{v,s}$, then $b_k(s, T_{s,v}) \geq b_k(x, T_{x,y})$. So $b_k(v, T) \geq 1 + b_k(s, T_{s,v}) \geq b_k(s, T)$ from Observation 1.

Thus, the vertex v (being removed) is not the last vertex of the k-broadcast center of T remaining in the set T'. ∎

Note that because vertex w of step 5 is the last remaining vertex of T', it follows that $w \in BC_k(T)$. Combining Theorem 3 and Observation 3 with this fact, we get the following lemma.

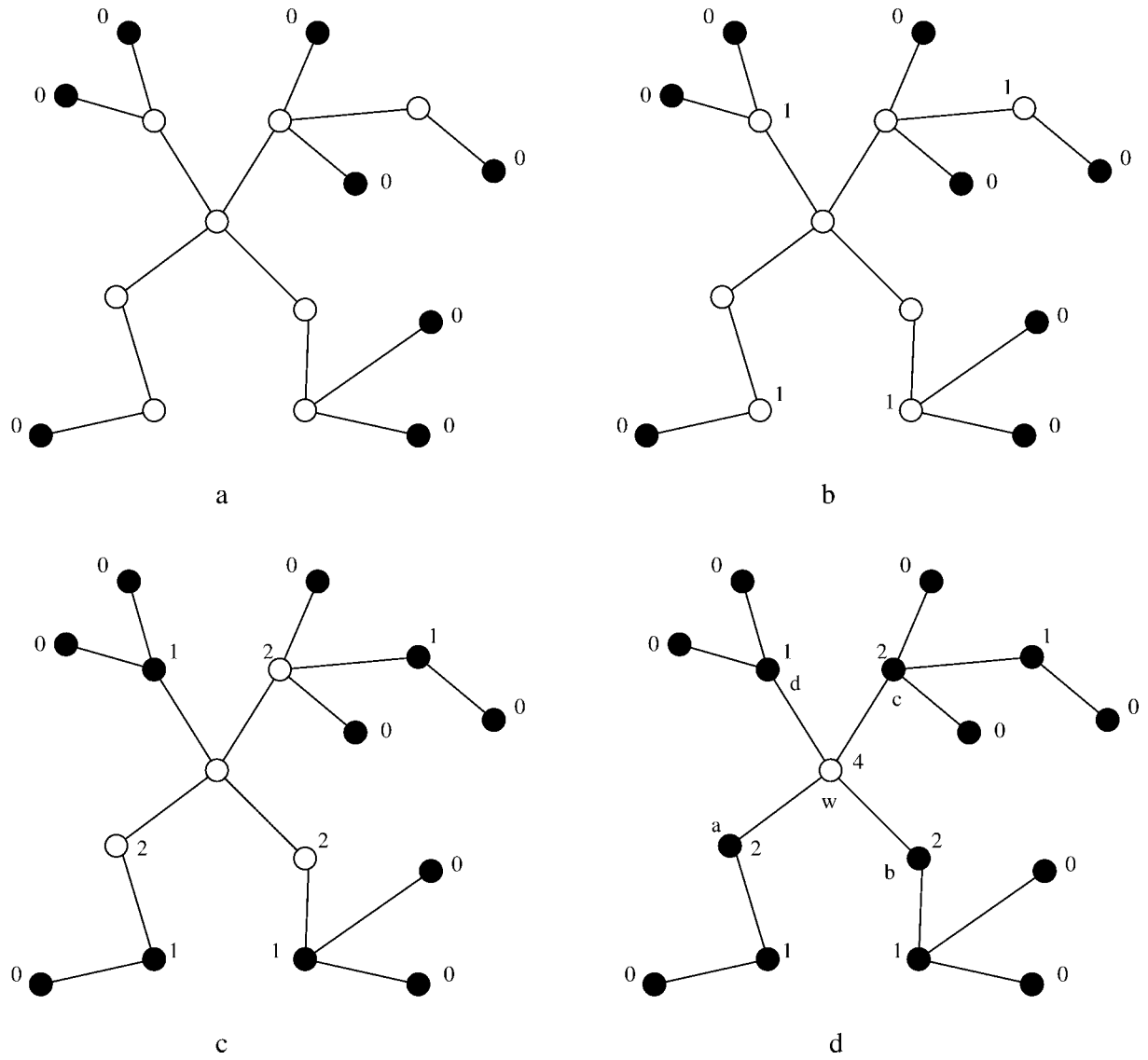

FIG. 1. Snapshots of the execution of KBC.

Lemma 2. *$b_k(T) = l(w)$ where w is the only remaining vertex of T' in step 5.*

Theorem 4. *Let w be the single remaining vertex in step 5 of KBC and let $c_1, c_2, \ldots, c_p$ be its labeled neighbors ordered such that $l(c_1) \geq l(c_2) \geq \cdots \geq l(c_p)$. Then $b_k(T) = l(w)$. Further, let j be the smallest integer such that $l(c_j) + \lceil \frac{j}{k} \rceil = \max_{1 \leq i \leq p}\{l(c_i) + \lceil \frac{i}{k} \rceil\} = l(w)$ and for any $l > j$ with $l(c_l) + \lceil \frac{l}{k} \rceil = \max_{1 \leq i \leq p}\{l(c_i) + \lceil \frac{i}{k} \rceil\}$, $l \equiv 1 \mod k$. If $j \equiv 1 \mod k$, then $BC_k(T) = \{w, c_1, c_2, \ldots, c_j\}$. Otherwise, $BC_k(T) = \{w\}$.*

Proof. Given w and j as described, let i be any integer $1 \leq i \leq j$. Let S_w denote the k-broadcast scheme for originator w corresponding to the ordering $c_1, c_2, \ldots, c_p$ of w's neighbors as in Theorem 3. In fact, Theorem 3 tells us that from any originator in any tree an optimal k-broadcasting scheme exists so that the originator calls its neighbors in

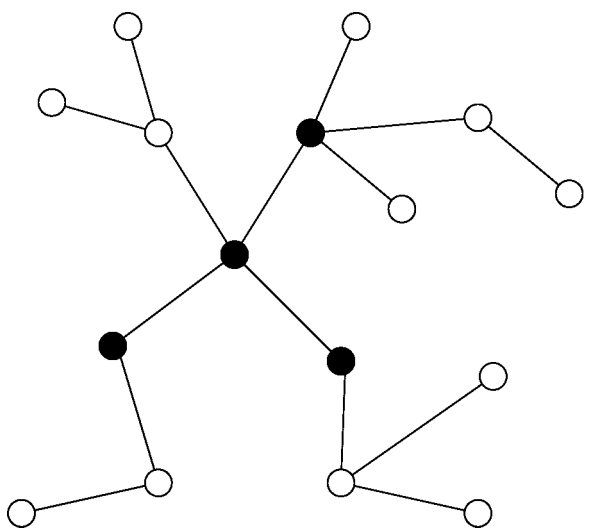

FIG. 2. 2-Broadcast center of tree.

an order consistent with nonincreasing k-broadcast times of these neighbors' subtrees. Thus, we can restrict our attention to such schemes and all subschemes will also have this property.

Consider the following k-broadcast scheme for originator c_i. At time 1, c_i sends the message to w. Beginning at time 2, c_i k-broadcasts within $T_{c_i,w}$ as in S_w. Vertex w sends the message to its neighbors in the same order as in S_w, omitting c_i and calling each vertex as early as possible while maintaining the ordering $c_1, c_2, \ldots, c_p$. The vertices $c_1, c_2, \ldots, c_p$ will receive the message at most one time unit later than in S_w. In particular, the vertices c_l where $l > j$ and $l \equiv 1 \mod k$ receive the message at the same time as in S_w. In any case, each vertex c_m receives the message no later than at time $1 + \lceil \frac{m}{k} \rceil$. The vertices c_l where $l > j$ and $l \equiv 1 \mod k$ receive the message at time $\lceil \frac{l}{k} \rceil$. In each subtree $T_{c_m,w}$, the k-broadcast continues as in S_w. The k-broadcast in each subtree $T_{c_m,w}$ takes time $l(c_m)$. For $l > j$ and $l \equiv 1 \mod k$, the k-broadcast within $T_{c_l,w}$ completes by time $\lceil \frac{l}{k} \rceil + l(c_l) = b_k(T)$. For all other values m, $1 \le m \le p$, the k-broadcast within $T_{c_m,w}$ completes by time $\lceil \frac{m}{k} \rceil + l(c_m) + 1 \le \lceil \frac{j}{k} \rceil + l(c_j) + 1 \le \lceil \frac{j}{k} \rceil + l(c_j) = b_k(T)$. Thus, $w, c_1, c_2, \ldots, c_j \in BC_k(T)$.

To show that no other vertices belong to $BC_k(T)$, consider a minimum time k-broadcast scheme from some vertex c_i (adjacent to w) where $i > j$. Vertex w can learn the message no earlier than at time 1. Recall that an optimal scheme exists consistent with the ordering $c_1, c_2, \ldots, c_{i-1}, c_{i+1}, c_{i+2}, \ldots c_j$. Subtrees $T_{c_m,w}$ for $1 \le m \le j$ must learn the message through w and can complete no sooner than at times $\lceil \frac{m}{k} \rceil + l(c_m) + 1$. In particular, k-broadcast within $T_{c_j,w}$ can complete no sooner than at time $\lceil \frac{j}{k} \rceil + l(c_j) + 1 = 1 + b_k(T) > b_k(T)$.

Theorem 1 implies that no vertex of T nonadjacent to w can be in $BC_k(T)$ unless $j = 1$. In this case, $BC_k(T)$ might be a complete bipartite graph $K_{1,m}$ with c_1 being the singleton vertex. In the case that $j = 1$, consider the neighbors $d_1, d_2, \ldots, d_q$ of c_1 (other than w) ordered such that $l(d_1) \ge l(d_2) \ge \cdots \ge l(d_q)$. Consider a k-broadcast from some $d_i \in BC_k(T)$. Because w and c_1 are both in $BC_k(T)$ and w was the last remaining vertex of T' during the execution of KBC, $b_k(w, T) = l(c_1) + 1 \ge l(d_i) + 1$. If d_i is also in $BC_k(T)$, then $b_k(w, T) = b_k(d_i, T)$. Since c_1 is the only vertex c_i adjacent to w such that $\lceil \frac{i}{k} \rceil + l(c_i) = b_k(T)$, then during the algorithm when T' contains only c_1 and w, $b_k(c_1, T_{c_1,w}) = l^*(c_1) \le l^*(w) = b_k(w, T \setminus T_{c_1,w})$ (where $l^*(c_1)$ and $l^*(w)$ are the values of the labels on c_1 and w, respectively, at that time in the execution of the algorithm) since c_1 is selected before w. Note that there must be such a time when T' contains only c_1 and w since for all $i > 1$, $l(c_i) < l(c_1)$ in this case. A k-broadcast from any vertex $d_i \ne w$ adjacent to c_1 will take at least $2 + b_k(w, T \setminus T_{w,c_1})$ time units. That is, $b_k(d_i, T) \ge 2 + b_k(w, T \setminus T_{c_1,w}) \ge 2 + l^*(w) \ge 2 + l^*(c_1) = 2 + l(c_1) = 1 + (1 + l(c_1)) = 1 + b_k(T)$. Thus, $BC_k\{T) = \{w, c_1\}$ in this case.

This completes the proof that $BC_k(T) = \{w, c_1, c_2, \ldots, c_j\}$ if a j exists as in the condition of the theorem statement.

Otherwise, consider the smallest j such that $j \not\equiv 1 \mod k$ and for which $l(c_j) + \lceil \frac{j}{k} \rceil = \max_{1 \le i \le p} \{l(c_i) + \lceil \frac{i}{k} \rceil\}$. Recall that we are restricting our attention to schemes in which a vertex calls its neighbors in an order consistent with the nonincreasing k broadcast times of those neighbors' subtrees.

If $i \ne j$, a k-broadcast from c_i can reach w no earlier than at time 1 and cannot reach c_j prior to time $\lceil \frac{j}{k} \rceil + 1$. So, c_i cannot be in $BC_k(T)$.

If $i = j$, then consider a k-broadcast for originator c_j. As before, w can learn the message no earlier than at time 1. Note that $l(c_{j-1}) + \lceil \frac{j-1}{k} \rceil \ge l(c_j) + \lceil \frac{j}{k} \rceil = b_k(T)$. Since in any k-broadcast from c_j, c_{j-1} cannot be informed prior to time $\lceil \frac{j-1}{k} \rceil + 1, b_k(c_j, T) \ge l(c_{j-1}) + \lceil \frac{j-1}{k} \rceil + 1 \ge l(c_j) + \lceil \frac{j}{k} \rceil + 1 > b_k(T)$, c_j cannot be in $BC_k(T)$.

Thus, none of the neighbors of w are in $BC_k(T)$ and, from Theorem 1, no other vertices can be in $BC_k(T)$, so $BC_k(T) = \{w\}$. ∎

Corollary 1. *For $k \ge 2$, $BC_k(T)$ is either a single vertex (w from step 5 of KBC) or a complete bipartite graph $K_{1,m}$ where $m \equiv 1 \mod k$ (where w is the singleton vertex of $K_{1,m}$).*

3.3. Analysis of Algorithm

Theorem 5. *Given a tree T and integer $k \ge 1$, the algorithm KBC generates $BC_k(T)$ in time $O(|V(T)|)$.*

Proof. As T is a tree, $O(|E(T)|) = O(|V(T)|)$. KBC can be implemented to run in $O(|V(T)|)$ time. To do this, we maintain for each vertex a list of the labels of its neighbors in non-increasing order. Since we compute the labels in nondecreasing order, we can simply add the label for a newly labeled vertex u to the head of these lists for each neighbor of u. The labels are never changed, so this requires only $O(|E(T)|)$ time during the entire algorithm. In addition to these lists of neighbor labels, we maintain an overall list of labels in nonincreasing order. As a new label is computed, we add it to this list as well and, again, this requires only $O(|E(T)|)$ time during the entire algorithm. As each vertex either becomes a leaf (in step 3 or step 4.3) or becomes the special vertex w (in step 5), we calculate its label by scanning through the labels of its neighbors. During the entire algorithm, this takes $O(|E(T)|)$ time. In step 4.1, we extract the minimum remaining label from the overall list and from the appropriate list of neighbor labels. This also takes only $O(|E(T)|)$ time during the entire algorithm. Finally in step 6, in the worst case we require a constant amount of computation for each neighbor of w, that is, at most $O(|E(T)|)$ time in total. ∎

To determine the k-broadcast time of a particular vertex in a graph is a natural question. The algorithm KBC provides the basis for a linear time algorithm for this problem on trees.

Corollary 2. *Given a tree T and integer $k \ge 1$, the k-broadcast time of an arbitrary vertex u of T can be determined in time $O(|V(T)|)$.*

Proof. The result follows from Theorem 2 and Theorem 5 and the fact that the distance between u and the k-broadcast center can be computed in time $O(|V(T)|)$. ∎

Another natural question is to determine the k-broadcast time of a graph. Again, KBC provides the basis for a linear time algorithm for this problem on trees.

Corollary 3. *Given a tree T and integer $k \geq 1$, the k-broadcast time of T can be determined in time $O(|V(T)|)$.*

Proof. The k-broadcast time of T is simply the maximum value of $b_k(u, T)$ over all vertices u. From Theorem 2 and Corollary 2, it is clear that by finding a vertex at maximum distance from the k-broadcast center, we can easily calculate the k-broadcast time of T. Finding such a vertex can be done in $O(|V(T)|)$ time by using a simple modification of breadth first search starting at the k-broadcast center. ∎

REFERENCES

[1] A. Bar-Noy, J. Bruck, C.T. Ho, S. Kipnis, and B. Scheiber, Computing global combine operations in the multi-port postal model, IEEE Trans Parellel Distrib Syst 6 (1995), 896–900.

[2] R. Beier and J.F. Sibeyn, A powerful heuristic for telephone gossiping, Proc 7th Int Colloquium Structural Info Comm Complexity, Laquila, Italy, 2000, pp. 17–36.

[3] J. Bruck and C.T. Ho, Efficient global combine operations in the multi-port message passing systems, Parellel Proc Let 3 (1993), 335–346.

[4] A. Dessmark, A. Lingas, H. Olsson, and H. Yamamoto, Optimal broadcasting in almost trees and partial k-trees, Proc 15th Ann Symp Theoretical Aspects Comput Sci, Lecture Notes in Computer Science, Vol. 1373, 1998, pp. 432–443.

[5] M. Dietzfelbinger, Gossiping and broadcasting versus computing functions in networks, Discrete Appl Math 137 (2004), 127–154.

[6] M. Elkin and G. Kortsarz, Sublogarithmic approximation for telephone multicast, J Comput Syst Sci 72 (2006), 648–659.

[7] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, Randomized broadcast in networks, Proc 1st SIGAL Int Symp Algorithms, Vol. 450, Springer-Verlag, Lecture Notes in Computer Science, 1990, pp. 128–137.

[8] P. Fraigniaud and E. Lazard, Methods and problems of communication in usual networks, Discrete Appl Math 53 (1994), 79–133.

[9] P. Fraigniaud and S. Vial, Approximation algorithms for broadcasting and gossiping, J Parallel Distrib Comput 43 (1997), 47–55.

[10] M. Grigni and D. Peleg, Tight bounds on minimum broadcast networks, SIAM J Discrete Math 4 (1991), 207–222.

[11] H.A. Harutyunyan and A.L. Liestman, Improved upper and lower bounds for k-broadcasting, Networks 37 (2001), 94–101.

[12] H.A. Harutyunyan and A.L. Liestman, k-broadcasting in trees, Networks 38 (2001), 163–168.

[13] H.A. Harutyunyan and A.L. Liestman, On the monotonicity of the broadcast function, Discrete Math 262 (2003), 149–157.

[14] H.A. Harutyunyan and E. Maraachlian, Linear algorithm for broadcasting in unicyclic graphs, Proc 13th Ann Int Comput Combinatorics Conf, Lecture Notes in Computer Science, Vol. 4598, 2007, pp. 372–383.

[15] H.A. Harutyunyan and B. Shao, Optimal k-broadcast in trees, Congressus Numerantium 160 (2003), 117–127.

[16] H.A. Harutyunyan and B. Shao, An efficient heuristic for broadcasting in networks, J Parallel Distrib Comput 66 (2006), 68–76.

[17] S.T. Hedetniemi, S.M. Hedetniemi, and A.L. Liestman, A survey of broadcasting and gossiping in communication networks, Networks 18 (1988), 319–349.

[18] J. Hromkovič, R. Klasing, B. Monien, and R. Peine, "Dissemination of information in interconnection networks (broadcasting & gossiping)," Combinatorial network theory, D.Z. Du and D.F. Hsu (Editors), Kluwer, Dordrecht, The Netherlands, 1995, pp. 125–212.

[19] J. Hromkovič, R. Klasing, A. Pelc, P. Ružička, and W. Unger, Dissemination of information in communication networks: Broadcasting, gossiping, leader election, and fault-tolerance, Springer, Berlin, 2005.

[20] A. Jakoby, R. Reischuk, and C. Shindelhauer, The complexity of broadcasting in planar and decomposable graphs, Discrete Appl Math 83 (1998), 179–206.

[21] J.-C. König and E. Lazard, Minimum k-broadcast graphs, Discrete Appl Math 53 (1994), 199–209.

[22] G. Kortsarz and D. Peleg, Approximation algorithms for minimum time broadcast, SIAM J Discrete Math 8 (1995), 401–427.

[23] E. Lazard, Broadcasting in DMA-bound bounded degree graphs, Discrete Appl Math 37/38 (1992), 387–400.

[24] S. Lee, Information dissemination theory in communication networks: design of c-broadcast networks, Ph.D. Dissertation, Dept. of Industrial and Manufacturing Eng., Pennsylvania State Univ., University Park, Pennsylvania, 1998.

[25] S. Lee and J.A. Ventura, An algorithm for constructing minimal c-broadcast networks, Networks 38 (2001), 6–21.

[26] S. Lee and J.A. Ventura, Construction of time-relaxed c-broadcast networks, Telecommun Syst 24 (2003), 47–59.

[27] R. Ravi, Rapid rumor ramification: approximating the minimum broadcast time, Proc 35th Symp Foundations Comput Sci, Santa Fe, New Mexico, 1994, pp. 202–213.

[28] P. Scheuermann and G. Wu, Heuristic algorithms for broadcasting in point-to-point computer networks, IEEE Trans Comput 33 (1984), 804–811.

[29] A. Shastri and S. Gaur, Multi-broadcasting in communication networks I: Trees, Proc Int Symp Commun, Hsinchu, Taiwan, 1997, pp. 167–170.

[30] P.J. Slater, E.J. Cockayne, and S.T. Hedetniemi, Information dissemination in trees, SIAM J Comput 10 (1981), 692–701.