SIAM J. ALG. DISC. METH.
Vol. 2, No. 4, December 1981

0196-5212/81/0204-0003 $01.00/0

BROADCASTING IN TREES WITH MULTIPLE ORIGINATORS*

ARTHUR M. FARLEY† AND ANDRZEJ PROSKUROWSKI†

Abstract. Broadcasting is the information dissemination process in a communication network whereby all sites of the network become informed of a given message by calls made over lines of the network. We present an algorithm which, given a tree network and a time, determines a smallest set of subtrees covering sites of the network such that broadcast can be completed within the given time in each subtree. Information developed by the algorithm is sufficient to determine a satisfactory originator and calling scheme within each subtree.

1. Introduction. *Broadcasting* is the information dissemination process in a communication network whereby all sites of the network become informed of a given message by calls placed over lines of the network. We model a communication network by a graph $G=(V, E)$ consisting of a set V of vertices (*sites*) and a set E of edges (*lines*), each edge incident to a pair of vertices. We model processes of information dissemination by the following constraints:

(1) information is disseminated in the form of *messages*;

(2) a message is transferred by a *call* between adjacent sites;

(3) no site can participate in more than one call at any time.

The length of a message determines an associated *time unit*, being the time needed to complete a call transferring the message. As such, we will talk about the number of time units required to broadcast a message.

Broadcasting can be defined more formally as a sequence of sets $S_0 \subseteq S_1 \subseteq \cdots \subseteq S_t = V$, each set representing the sites informed of the broadcast message after time unit $i, 0 \leqq i \leqq t$. For each u in $S_i - S_{i-1} (i > 0)$, there exists an adjacent site in S_{i-1}, not assigned to another site of $S_i - S_{i-1}$, which calls u during unit time i. The elements of S_0 are called the *originators* of the broadcast. The case where $|S_0| = 1$ has received considerable research attention in recent years. The minimum value of t for a given network G over all broadcasts in G is called the *broadcast time* of G; a site from which such a broadcast is possible is an element of the *broadcast center* of G. Slater, Cockayne and Hedetniemi [10] have described an algorithm for determining both parameters in an arbitrary tree network. A *tree* network is a connected, acyclic network.

Farley, Hedetniemi, Mitchell and Proskurowski [3] investigated networks having the fewest lines which allow broadcasting to be completed in the minimum possible time (i.e., $\log_2 |V|$ time units) from any site. Farley [1] discussed construction algorithms for several such minimum-time broadcast networks requiring approximately the minimum number of lines. The general problem of determining the broadcast time for a given network G has been shown to be algorithmically hard (i.e., NP-complete) by Garey and Johnson [5, p. 219]. This motivates approximate results as well as study of restricted classes of networks. Proskurowski [9] has characterized minimum broadcast trees, being rooted trees which allow broadcasting to be completed in $\log_2 |V|$ time units from the root. In [2], Farley considered broadcasting of multiple messages in completely connected networks.

In this paper, we consider a generalization of broadcasting in which several sites may originate the message (i.e., $|S_0| \geqq 1$) within a network. This could arise within practical situations in several ways. A subset of sites may be connected by a broadcast medium (i.e., radio), with message broadcast to be completed by calls over lines. The

* Received by the editors September 12, 1980, and in final form April 3, 1981. This work was supported in part by the National Science Foundation under grant ENG-79-02960.

† Department of Computer and Information Science, University of Oregon, Eugene, Oregon 97403.

situation may also arise from a hierarchical view of broadcasting within a network. A message can be seen to be broadcast through a tree of sites, each such site also being a member of a network at its "level" of the hierarchy. After broadcast in the tree is completed, informed sites originate broadcasting within these level-based networks, each such network having potentially more than one originator. The lines of the tree may be of higher speed and capacity. Networks at each level may likewise have differing communication characteristics.

We present an algorithm which, given a tree network T and a broadcast time t, determines a smallest set of subtrees covering the sites of T such that the broadcast time for each subtree is less than or equal to t. Information developed by the algorithm is sufficient to also determine a satisfactory originator and calling scheme for each subtree. A solution to our problem for $t \leqq 2$ has been given as a special case of decomposing trees into paths by Hedetniemi and Hedetniemi [6]. Our algorithm solves the problem for arbitrary $t > 0$. The algorithm is efficient, requiring time and space proportional to $|V|$, with a constant of proportionality depending linearly on t.

2. Partitioning trees by broadcast time. Trees, being acyclic connected graphs, have several properties which make them suitable for the design of efficient solution algorithms. Most important is that each vertex (and edge) separates the graph into two connected components. Therefore, there is an absence of influence between subtrees of a given vertex other than that transmitted through the vertex itself. This allows efficient algorithms to process a (current) leaf vertex, update information at its single adjacent vertex, make globally correct decisions based upon this local information and prune the leaf vertex, removing it from the tree and further consideration. We follow this paradigm in our solution algorithm for partitioning trees according to broadcast time.

The input tree is represented recursively by a *father array* [8, p. 354], which assumes an arbitrary *root* vertex. The array contains, for each nonroot vertex, a pointer to its unique father (of lesser index) on the path to the root. During execution of the algorithm, certain edges of the input tree are *cut*, disconnecting the tree and forming a subtree of the partition. At any time, the connected component of the input tree containing the root is called the *current tree*. We also refer to the *unprocessed tree*, which initially corresponds to the input tree. During each cycle of the algorithm a leaf vertex of the unprocessed tree is processed. After being processed, the leaf is pruned (i.e., removed) from the unprocessed tree, though it will remain part of the current until a cut disconnects it from the root. With each vertex v of the input tree, we associate the following information:

(i) callees(v)—a list of previously processed, adjacent neighbors, ordered according to the time that v would call them in a minimum time broadcast;
(ii) maxtime(v)—the latest time unit during which v can be called and still complete broadcasting within the required time in the subtree defined by v and subtrees of the current tree rooted by vertices on callees(v);
(iii) mintime(v)—the earliest time unit that v can be called from a (necessary) broadcast originator within a previously processed subtree of v in the current tree;
(iv) caller(v)—the adjacent, processed vertex capable of calling v with the message during time unit mintime(v) from the predetermined originator.

For each vertex v, this information is initialized as follows: callees(v) and caller(v) are set to nil (i.e., empty), maxtime(v) is set to t (as each could potentially be called during the last time unit), and mintime(v) is set to 0 (as each could potentially be an

originator). A vertex v which has a nonnil caller(v) value (has a processed subtree containing a necessary originator) is classified as *heavy*. Otherwise, v is classified as *light*. All vertices are initially light.

The processing of a leaf vertex depends on whether it is heavy or light. If a leaf vertex u is light, the attempt is made to insert u into callees(v) of its father v. If successful, maxtime(v) is updated and processing of u is complete. If u cannot be inserted, a necessary cut is introduced between u and v. If u is heavy (i.e., mintime(u) > 0), then a check is made to see whether a broadcast through u can reach all light subtrees of u (by comparing mintime(u) and maxtime(u)). If not, the subtree rooted at caller(u) is cut from the current tree; u becomes light and is processed as described above. If the light subtrees of u can be accommodated, then consideration of the father v begins. If v is light, it becomes heavy with caller(v) being u. If v is heavy, a cut is introduced, disconnecting the subtree rooted by either u or caller(v) from the current tree.

Each site u has a set of potential timeslots (1 to t) during which it can call elements of callees(u). A function *emptyslots* scans callees(u), determining the set of time slots available below a given maximum time. The maximum or minimum value returned by emptyslots is important in determining whether light subtrees can be accommodated by heavy or father sites. This outlines the approach taken by algorithm BROADCAST, which is formally defined as follows.

```
ALGORITHM BROADCAST
Input Tree T given by the array father, broadcast time t.
Output Partition of T into subtrees of broadcast time at most t represented by
       cut edges of T.
Method begin
{0. Initialize} for each vertex u do
{0.1}     begin maxtime(u) := t;
{0.2}           mintime(u) := 0;
{0.3}           callees(u) := nil;
{0.4}           caller(u) := nil end;
{1. Prune} for each leaf vertex u of unprocessed tree do
        if mintime(u) > 0 {a heavy leaf}
{1.1}      then if maxtime(u) < mintime(u) {u cannot be covered}
{1.1.1}              then begin mintime(u) := 0; cut (caller(u)); {u is light now}
                                updatefather(u)
                           end
{1.1.2}              else upminfather(u)
{1.2}      else {a light leaf} updatefather(u)
         end. {of BROADCAST}

procedure updatefather(u);
{of a light vertex}
begin if maxtime(u) = 0 {root of a broadcast}
        then {a vertex informed at time 0}
             upminfather(u)
        else upmaxfather(u)
end;

procedure upmaxfather(u);
{recompute maxtime requirements}
begin v := father(u);
      s := max[emptyslots(callees(v), maxtime(u))];
```

```
        if s = 0 then cut(u) {v cannot accommodate u}
                 else begin insert (u, callees(v), s);
                        if s ≤ maxtime(v)
                        then maxtime(v) := s − 1
                      end {v calls u at s}
end;

procedure upminfather(u);
{updates mintime info}
begin v := father(u);
      s := min[emptyslots(callees(u), t + 1)];
      if mintime(v) = 0 {a light node}
          then begin caller(v) := u; mintime(v) := s + 1 end {v becomes heavy}
          else if s < mintime(v) {a taller son}
                 then begin cut(caller(v));
                        caller(v) := u;
                        insert(v, callees(u), s);
                        mintime(v) := s + 1
                        end
                 else cut(u)
end;

procedure cut(vertex);
{adds the edge between vertex and father (vertex) to the set of cut edges}
function emptyslots(list, min);
{returns a set of timeslots less than min at which another callee can be informed
    (inserted into list), or 0 if no such slot exists}
procedure insert(vertex, list, slot);
{inserts vertex on the list at time slot maintaining the increasing time order of list}
```

3. Correctness and complexity of the algorithm. In this section we will state and prove lemmas verifying correct computation of vertex (subtree) parameters during execution of the algorithm BROADCAST. These are used to establish correctness of the algorithm. In our arguments, we use the notions of light and heavy vertices, the current tree, and the unprocessed tree, as defined in the preceding section. Additionally, by *pruned subtree* of a vertex v we understand a subtree rooted at a pruned neighbor of v.

We first consider computation of the time parameters: maxtime and mintime.

LEMMA 1. *In the current tree S, the value* maxtime (v) *of an unprocessed vertex v equals the latest time unit (counted from the origination of the message) in which v must be informed in order to complete broadcast in its pruned light subtrees in S by time t.* Callees(v) *is the list of pruned light sons of v ordered by their maxtime values.*

Proof. During initialization, the value of maxtime(v) for each vertex v of T is set to t, which is correct for vertices having no pruned light subtree; callees(v) is initially empty. Let us assume that the values are correct just before a leaf of the unprocessed tree is pruned. If this leaf is heavy and no cut is made, then the values are left correct. A heavy leaf vertex u of the unprocessed tree whose caller must be cut off (step {1.1.1}) becomes a light vertex of the new S. Both for such a vertex and for an originally light vertex the procedure upmaxfather (called from updatefather) correctly updates the values maxtime(v) and callees(v) of the father v. These values stay unchanged if the light leaf vertex u has to be cut off. In this case, the disconnected subtree rooted at u

does not influence the maxtime or callees values of the father. Otherwise, the vertex u is inserted at an appropriate place in the list of callees of v and, if its calling time is now the earliest, it redefines the maxtime of v. □

LEMMA 2. *In a current tree S, a heavy vertex cannot have two pruned sons which are heavy or which have maxtime equal to zero.*

Proof. The broadcast time for a subtree of pruned vertices of S rooted at vertex u is greater than t if u is heavy, or equal to t if $\text{maxtime}(u) = 0$. Thus, the message to be broadcast to u cannot originate outside of its subtree as implied by a heavy brother in S. In the procedure upminfather such a situation is prevented by cutting off one of the two heavy vertices. □

LEMMA 3. *In the current tree S, the value of* mintime (v) *of a heavy vertex v equals the earliest time during which v can be informed of a message originated by a previously determined vertex in a pruned subtree of v. The heavy neighbor of v supplying this message is* caller(v).

Proof. All vertices of T are initially light. A vertex v becomes heavy when one of its pruned sons u in S has maxtime equal to zero or is heavy. In both cases, the procedure upminfather is invoked to update mintime(v) and caller(v) according to the parameters of u. Let us assume that the values are correct just before u is pruned. In upminfather, the earliest available timeslot of u is determined. If v is light, then caller(v) is correctly set to u and mintime(v) is accordingly set. If v is heavy, then this timeslot is compared to the current value of mintime(v) and a cut minimizing the resultant value of mintime(v) is made. If the current caller(v) is cut, then the values are appropriately updated according to parameters of u; otherwise they remain unchanged. □

LEMMA 4. *In the current tree S, a pruned vertex v can have a heavy son u only if u can call v at or before* maxtime(v)

Proof. By Lemma 1, we know it requires $t - \text{maxtime}(v)$ time units to complete broadcasting from v to its light, pruned subtrees in S. Therefore, a broadcast cannot be completed by time t if v is not informed prior to or at maxtime(v). When heavy son u cannot inform v prior to or at maxtime(v), the necessary cut is made in step {1.1.1}. □

THEOREM 1. *Algorithm* BROADCAST *computes a minimum-size partition of tree T such that a message can be broadcast from a single originator in each subtree within time t.*

Proof. Let $b(T, t)$ be this minimum size and $c(T, S)$ be the number of invocations of procedure *cut* in the algorithm on T, when the current tree is S. We have to prove that $c(T, \Phi) = b(T, t) - 1$. This will be shown by establishing the invariant value of $c(T, S) + b(S, t)$ during the execution. Indeed, the current tree changes only when *cut* is invoked in one of three cases: (i) when pruning a heavy vertex which cannot be accommodated (called) from its heavy descendant in S, (ii) when encountering two heavy sons in upminfather, or (iii) when a light vertex cannot be informed by its father in the required time (in upmaxfather). In all these cases, a cut has to be made according to Lemmas 2, 4 and 1, respectively. The cut results in a heavy vertex with smallest possible value of mintime ((ii)), or in a light vertex with largest possible value of maxtime ((i) or (iii)). This ensures that the new current tree, S', has the minimum value of $b(S'', t)$ over all subtrees S'' of S such that the cutoff subtree $S - S''$ has a broadcast time at most t. Thus, $b(S', t) = b(S, t) - 1$ and $c(T, S') = 1 + c(T, S)$. Therefore, $c(T, S') + b(S', t) = 1 + c(T, S) + b(S, t) - 1 = c(T, S) + b(S, t)$. Initially, $S = T$, and this constant value $c(T, T) + b(T, t) = b(T, t)$. After the final cut has been made, the resulting current subtree S''' has broadcast time at most t, and thus it is the only component in its optimal partitioning, $b(S''', t) = 1$. Thus $c(T, S''') + b(S''', t) = b(T, t)$, and as no more cuts are made, $c(T, \Phi) = c(T, S''') = b(T, t) - 1$. □

The pruning strategy employed in the algorithm BROADCAST guarantees that each vertex is processed exactly once, and thus the complexity of the algorithm is defined by the complexity of upmaxfather and upminfather, which are executed at most once per processed vertex. These procedures, in turn, involve at most one call of emptyslots and/or insert which require number of operations in the order of length of the relevant (callee) list. This list of light sons of a vertex in u is never longer than t. Hence, we have the following theorem determining the complexity of BROADCAST.

THEOREM 2. *Given a tree T with n vertices and a broadcast time t, the execution time of algorithm* BROADCAST *is $O(n \cdot t)$.*

4. Conclusion. In this paper, we have presented a linear algorithm for decomposing a given tree into subtrees, each subtree having a broadcast time less than or equal to a given time. This algorithm can be seen as one of a family of linear tree partitioning algorithms [4]. Partitioning techniques can be seen as alternatives to methods determining multiple centers (cf. [7] and [4]).

Partitioning based on broadcast time is well-motivated from an applications perspective. Other models of the information dissemination process would lead to different decomposition problems. For example, associating a call time with each line to reflect average load (i.e., queue length) is a reasonable extension of our model.

REFERENCES

[1] A. M. FARLEY, *Minimal broadcast networks*, Networks 9 (1979), 313–332.
[2] A. M. FARLEY, *Broadcast time in communication networks*, SIAM J. Applied Math., 39 (1980), pp. 385–390.
[3] A. M. FARLEY, S. T. HEDETNIEMI, S. L. MITCHELL AND A. PROSKUROWSKI, *Minimum broadcast graphs*, Discr. Math, 25 (1979), pp. 189–193.
[4] A. M. FARLEY, S. T. HEDETNIEMI AND A. PROSKUROWSKI, *Partitioning trees: matching, domination and maximum diameter*, Internat. J. Comp. Inform. Sci., to appear.
[5] M. R. GAREY AND D. B. JOHNSON, *Computers and Interactability*, W. H. Freeman, San Francisco, 1978.
[6] S. M. HEDETNIEMI AND S. T. HEDETNIEMI, *Broadcasting by decomposing trees into paths of bounded length*, CS-TR-79-16, University of Oregon, Eugene, OR, 1979.
[7] O. KARIV AND S. L. HAKIMI, *Algorithmic approach to network location problems* I: *the p-centers*, SIAM J. Appl. Math, 37 (1979), pp. 513–530.
[8] D. E. KNUTH, *The Art of Computer Programming*, vol. I, 2nd ed., Addison-Wesley, Reading, MA, 1973.
[9] A. PROSKUROWSKI, *Minimum broadcast trees*, IEEE Trans. Computers, C-30 (1981), pp. 363–366.
[10] P. J. SLATER, E. J. COCKAYNE AND S. T. HEDETNIEMI, *Information dissemination in trees*, SIAM J. Comput., 10 (1981), pp. 692–701.