SIAM J. COMPUT.
Vol. 10, No. 4, November 1981

INFORMATION DISSEMINATION IN TREES*

P. J. SLATER,† E. J. COCKAYNE‡ AND S. T. HEDETNIEMI§

Abstract. In large organizations there is frequently a need to pass information from one place, e.g., the president's office or company headquarters, to all other divisions, departments or employees. This is often done along organizational reporting lines. Insofar as most organizations are structured in a hierarchical or treelike fashion, this can be described as a process of information dissemination in trees. In this paper we present an algorithm which determines the amount of time required to pass, or to broadcast, a unit of information from an arbitrary vertex to every other vertex in a tree. As a byproduct of this algorithm we determine the broadcast center of a tree, i.e., the set of all vertices from which broadcasting can be accomplished in the least amount of time. It is shown that the subtree induced by the broadcast center of a tree is always a star with two or more vertices. We also show that the problem of determining the minimum amount of time required to broadcast from an arbitrary vertex in an arbitrary graph is NP-complete.

Key words. algorithm, broadcast center, broadcasting, graph information dissemination, NP-complete, tree

1. Introduction.

Most large organizations are structured in a hierarchical or treelike fashion (cf. Fig. 1), from the highest levels at the top to lower and lower levels as one proceeds down the tree. If there is a need to pass some information, e.g., concerning a new company policy, from one office to all others, then it is natural for this to take place along the organizational reporting lines. Thus, a president might inform each vice-president, each of whom in turn informs all subordinate division heads, who in turn inform their subordinate department heads, etc. In this paper we study the amount of time it takes for this kind of information dissemination to take place in such treelike structures.

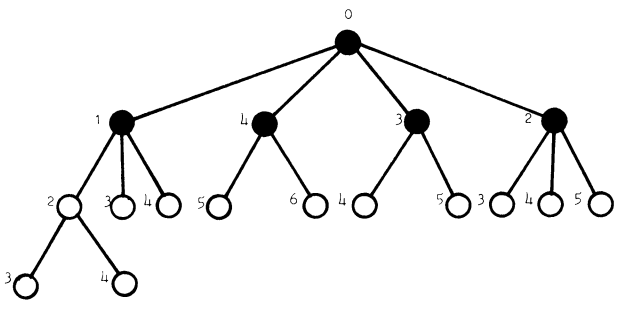

FIG. 1. *Broadcasting in a tree.*

More specifically, we define *broadcasting from a vertex u* to be the process of passing one unit of information from u to every other vertex in a connected graph $G = (V, E)$. This is accomplished by a series of phone calls over the edges of G, subject to the following constraints:

(i) each phone call requires one unit of time (to convey the unit of information);

* Received by the editors June 5, 1979, and in revised form, December 22, 1980.

† Sandia Laboratories, Albuquerque, New Mexico 87185. This work was supported by the U.S. Energy Research and Development Administration (ERDA) under contract AT(29-1)-789. By acceptance of this article, the publisher and/or recipient acknowledges the U.S. Government's right to retain a nonexclusive, royalty-free license in and to any copyright covering this paper.

‡ Department of Mathematics, University of Victoria, Victoria, British Columbia, V8W 2Y2. This work was supported by the National Research Council of Canada under grant A-7544.

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

(ii) a vertex can only call an adjacent vertex; and

(iii) a vertex can participate in only one call per unit of time.

We define *the broadcast number of u in G*, denoted $b(u; G)$ or simply $b(u)$, to equal the minimum number of time units required to broadcast from u. For example, the broadcast number of the topmost vertex in the tree in Fig. 1 is six; the integer label by each vertex indicates the time period during which it receives the information in one possible calling scheme. We define *the broadcast number of a connected graph G*, $b(G)$, to equal the minimum broadcast number of any vertex in G, i.e.,

$$b(G) = \min_{u \in V(G)} \{b(u)\}.$$

The broadcast center of G, $\mathrm{BC}(G)$, is the set of all vertices having minimum broadcast number, i.e.,

$$\mathrm{BC}(G) = \{u \mid b(u) = b(G)\}.$$

In this paper we present an $O(N)$ algorithm for determining the broadcast center of any tree with N vertices, a by-product of which determines the broadcast number of any vertex in the tree. It can be seen, incidentally, that the broadcast center of the tree in Fig. 1 consists of the darkened vertices.

We will also show that the problem of determining $b(u)$ for an arbitrary vertex in an arbitrary graph is NP-complete.

2. An algorithm for determining the broadcast center of a tree.

For a given vertex u in tree T there is not necessarily a unique calling scheme to broacast from u to all other vertices in $b(u)$ time units. We can, however, make the following observations. If (u, v) is an edge in tree T then $T(u, v)$ and $T(v, u)$ will denote the subtrees of T consisting of the components of $T-(u, v)$ containing u and v, respectively. Let $v_1, v_2, \cdots, v_k$ denote the vertices adjacent to u in T, and assume they are labeled so that $b(v_1; T(v_1, u)) \geqq b(v_2; T(v_2, u) \geqq \cdots \geqq b(v_k; T(v_k, u))$. Since $b(v_i; T(v_i, u))$ is the amount of time it will take to pass the information from v_i to the other vertices in $T(v_i, u)$ after a call from u to v_i, one expects an optimal calling sequence from u to consist of first calling v_1, then v_2, then v_3, etc. This is, in fact, the case.

The following algorithm, Algorithm BROADCAST, will identify a vertex v in $\mathrm{BC}(T)$. For any other vertex x in $V(T)$, let x' denote the vertex adjacent to x on the path from x to v. Algorithm BROADCAST will assign a value $t(x)$ to each vertex x with $t(x) = b(x; T(x, x'))$. The final label $t(v)$ for this v in $\mathrm{BC}(T)$ will equal $b(v) = b(G)$. We start by letting $t(x) = 0$ for every endvertex x in T. Subsequently we choose one vertex u at a time to label. At each stage the vertex u chosen to be labeled will be the one whose label is going to be smallest among the remaining vertices.

As indicated, Algorithm BROADCAST proceeds by iteratively assigning to each vertex u in a tree T a value $t(u)$, which equals the minimum time required to broadcast from u to every vertex in a subtree T_u. The subtree T_u in question is the largest subtree of T rooted at u consisting of vertices w which have previously been assigned values $t(w)$. Having initially labeled with zero all endvertices of T, the algorithm proceeds to move "inward" and assigns increasing labels to vertices, all but one of whose neighbors have already been labeled.

ALGORITHM BROADCAST. To determine the broadcast center $\mathrm{BC}(T)$ and the broadcast number $b(T)$ of a tree T. At any point in this algorithm U is the set of labeled vertices which have been removed from T, and W is the set of labeled vertices which have not been removed from T.

```
Step 0. (Does $T$ have only one or two vertices?)
        If $|V(T)| \leqq 2$
          then $\mathrm{BC}(T) \leftarrow V(T)$;
               if $|V(T)| = 2$ then set $b(T) \leftarrow 1$
                             else set $b(T) \leftarrow 0$ fi;
               STOP
               fi.
Step 1. (Label the endvertices of $T$ with 0.)
        Let $U$ be the set of endvertices of $T$;
        for each $u \in U$ do set $t(u) \leftarrow 0$ od;
        set $T' \leftarrow T - U$.
Step 2. (Label the endvertices of $T'$.)
        Let $W$ be the set of vertices of $T'$ with degree in $T'$ at most one;
        for each $w \in W$ do let $u_1, u_2, \cdots, u_k$ be the labeled vertices in $U$ adjacent
          to $w$, ordered so that $t(u_1) \geqq t(u_2) \geqq \cdots \geqq t(u_k)$;
          set $t(w) \leftarrow \max_{1 \leqq i \leqq k} \{t(u_i) + i\}$
          od.
Step 3. (Select the next vertex to be deleted and the next vertex to be labeled, until
        there is only one vertex left.)
        While $|V(T')| \geqq 2$ do
Step 4.        (Select the next vertex $w$.)
               Let $w \in W$ satisfy $t(w) = \min\{t(w_i) | w_i \in W\}$;
               let $v$ be the vertex adjacent to $w$ in $T'$.
Step 5.        (Delete $w$ from $W$ and $T'$, and add it to $U$.)
               Set $W \leftarrow W - \{w\}$;
               set $U \leftarrow U \cup \{w\}$;
               set $T' \leftarrow T' - \{w\}$. (Note, if $T'$ now has one vertex, it is considered to be an
                 endvertex.)
Step 6.        (Label the next vertex.)
               If $v$ is now an endvertex of $T'$
                 then let $v$ be adjacent to labeled vertices $u_1, u_2, \cdots, u_k$ in $U$ ordered so
                   that $t(u_1) \geqq t(u_2) \geqq \cdots \geqq t(u_k)$;
                   set $t(v) \leftarrow \max\{t(u_i) + i | 1 \leqq i \leqq k\}$;
                   set $W \leftarrow W \cup \{v\}$ fi.
               od
Step 7. (There is one vertex left.)
        Let $v$ be the one vertex of $T'$;
        set $b(T) \leftarrow t(v)$;
        let the neighbors of $v$ in $T$ be $u_1, u_2, \cdots, u_k$, where $t(u_1) \geqq t(u_2) \geqq \cdots \geqq$
        $t(u_k)$;
        let $j$ be the smallest integer such that
             $t(u_j) + j = \max\{t(u_i) + i | 1 \leqq i \leqq k\}$;
        set $\mathrm{BC}(T) \leftarrow \{v, u_1, u_2, \cdots, u_j\}$;
        STOP
```

We observe that, if T is not a star, then the final vertex v identified in step 7 will have been labeled twice—once when it became an endpoint of T', and again in step 6 when T' has only the one vertex v. The second label for v is no smaller than the first.

Figure 2 illustrates the application of algorithm BROADCAST to a tree T. In Fig. 2(a) U is the set of endvertices of T, each with label "0," and each of the vertices in W has been labeled. The first vertex which will be moved from W to U (that is, the "W" in

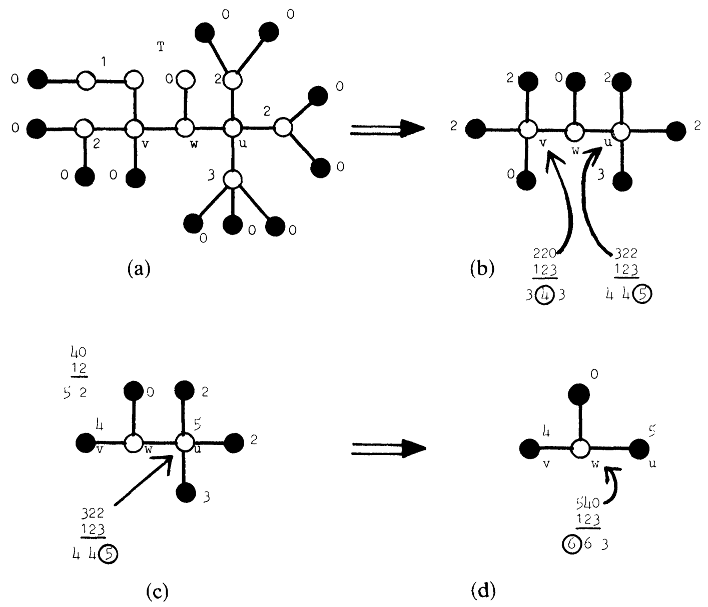

FIG. 2. *Algorithm* BROADCAST *applied to tree T.*

step 4) is the one with label "1." In Fig. 2(b) we see that $t(v)=4$ and $t(u)=5$, and so v is the next vertex moved to U. In Fig. 2(c) vertex u receives the value $t(u)=5$. Finally, in Fig. 2(d) vertex w receives the value 6, and, according to step 7, the broadcast center consists of vertices u and w.

3. Proof of correctness. The following analysis establishes the correctness of Algorithm BROADCAST. With v the root of a subtree T_v of T, $b(v, T_v)$ denotes the broadcast number of v in T_v. Note that, at each step of Algorithm BROADCAST, the subgraph induced by the unlabeled vertices is connected (i.e., is a subtree of T). In what follows we assume that $|V(T)|\geqq 3$. The case where $|V(T)|\leqq 2$ is easily treated in step 1 of Algorithm BROADCAST.

THEOREM 1. *Let $v_1, v_2, \cdots, v_k$ be the neighbors of a vertex w in a tree T, and let $T_i=T(v_i, w)$, for $i=1,2,\cdots,k$. Suppose further that $b(v_i, T_i)\geqq b(v_{i+1}, T_{i+1})$, for $i=1,2,\cdots,k-1$. Then,*

$$b(w,T)=\max\{b(v_i,T_i)+i \mid 1\leqq i\leqq k\}.$$

Proof. Suppose that v_i receives the information from w at time $\pi(i)$, $i=1,2,\cdots,k$, where π is an element of the symmetric group S_k of all permutations of $\{1,2,\cdots,k\}$. Then every vertex in T_i may be called by time $b(v_i, T_i)+\pi(i)$, and we deduce that

$$b(w,T)=\min_{\pi\in S_k}\left\{\max_{1\leqq i\leqq k}\{b(v_i,T_i)+\pi(i)\}\right\}.$$

It is clear that the permutation $\pi(i)=i$ minimizes this expression. □

LEMMA 2. *Suppose in step 5 that Algorithm* BROADCAST *deletes vertex w from W and adds it to U. If v is adjacent to w in T', then $t(w)=b(w, T(w, v))$.*

Proof. This follows immediately from Theorem 1 and the observation that if u is an endvertex of T adjacent to vertex w, then $t(u)=0=b(u, T(u, w))$. □

LEMMA 3. *Let x_1 and x_2 be adjacent vertices in T, and assume $b(x_1, T(x_1, x_2)) \leqq b(x_2, T(x_2, x_1))$. Then*

1) $$b(x_1, T) = 1 + b(x_2, T(x_2, x_1)), \quad \textit{and}$$

2) $$b(x_2, T(x_2, x_1)) \leqq b(x_2, T) \leqq 1 + b(x_2, T(x_2, x_1)).$$

Proof. The obvious proof is omitted.

LEMMA 4: *If $w_1, w_2, \cdots, w_{m-1}$ is the sequence of vertices selected in step 4, and w_m is the one remaining vertex v of step 7, then $t(w_1) \leqq t(w_2) \leqq \cdots \leqq t(w_m)$.*

Proof. Suppose $1 \leqq k < h \leqq m$. It suffices to show that $t(w_k) \leqq t(w_h)$. Vertex w_k is chosen during the kth execution of step 4. If w_h has received the label $t(w_h)$ during one of the first $k-1$ executions of step 6, then $t(w_k) \leqq t(w_h)$ by definition of how w_k is selected in step 4. If w_h receives label $t(w_h)$ during the jth execution of step 6, where $j \geqq k$, it suffices to note that $t(w_h) > t(w_j)$. □

LEMMA 5. *At any stage after step 1 in Algorithm* BROADCAST, *some vertex in T' is in the broadcast center of T.*

Proof. Let w be a vertex selected in step 4 to be removed from the current T' and W and added to the current U, and let v be the vertex adjacent to w in T'. It suffices to show that $b(v, T) \leqq b(w, T)$.

First we show that $b(w, T(w, v)) \leqq b(v, T(v, w))$. If $V(T') = \{w, v\}$, then the labels on w and v are $t(w) = b(w, T(w, v))$ and $t(v) = b(v, T(v, w))$ and $t(w) \leqq t(v)$. Hence $b(w, T(w, v)) \leqq b(v, T(v, w))$. Suppose that T' has an endpoint $x \notin \{v, w\}$, and let x' be the vertex in T' adjacent to x. By Lemma 4, $b(x, T(x, x')) \geqq b(w, T(w, v))$. But $T(x, x')$ is a subtree of $T(v, w)$ implies $b(v, T(v, w)) \geqq b(x, T(x, x'))$.

Now, by Lemma 3, $b(w, T) = 1 + b(v, T(v, w)) \geqq b(v, T)$. □

LEMMA 6. *If v and w are adjacent vertices in a tree T and $b(w, T(w, v)) \leqq b(v, T(v, w))$, then for all vertices x of $T(w, v) - w$ one has $b(x, T) > b(v, T)$.*

Proof. By Lemma 3, $b(v, T) \leqq 1 + b(v, T(v, w))$. Hence $b(x, T) \geqq 2 + b(v, T(v, w)) > 1 + b(v, T(v, w)) \geqq b(v, T)$, as required. □

THEOREM 7. *Let v be the single remaining vertex in step 7 of Algorithm* BROADCAST, *and let $u_1, u_2, \cdots, u_k$ be the neighbors of v ordered so that $t(u_i) \geqq t(u_{i+1})$, $i = 1, \cdots, k-1$. Let j be the smallest integer such that*

$$t(u_j) + j = \max_{1 \leqq i \leqq k} \{t(u_i) + i\}.$$

Then $\mathrm{BC}(T) = \{v, u_1, u_2, \cdots, u_j\}$ *and* $b(T) = t(u_j) + j = t(v)$.

Proof. By Lemma 5 we know that $v \in \mathrm{BC}(T)$, and from Theorem 1 and Lemma 2,

$$b(T) = \max_{1 \leqq i \leqq k} \{t(u_i) + i\} = t(u_j) + j = t(v).$$

Let s be a vertex other than v which is not adjacent to v, and let w be the vertex adjacent to v on the path from v to s. Since w is labeled, by Lemma 4 we have

$$b(w, T(w, v)) \leqq b(v, T(v, w)).$$

Hence, by Lemma 6,

$$b(s, T) > b(v, T),$$

and s is not in $\mathrm{BC}(T)$. Hence, every vertex in $\mathrm{BC}(T)$ is adjacent to v.

We next show that for all h, $1 \leqq h \leqq j$, $u_h \in \mathrm{BC}(T)$.

Let T_i be defined as in Theorem 1. In an optimal series of phone calls originating from vertex v, Theorem 1 asserts that for $i = 1, \cdots, k$, vertex u_i is called at time i. Now suppose that vertex u_h is the originator and v is called first. Vertex v then calls $\{u_i: i=1,\cdots,h-1,h+1,\cdots,k\}$ in this sequential order, i.e., vertices $u_1,u_2,\cdots,u_{h-1}$ are called by v at times $2,3,\cdots,h$, respectively, and vertices $u_{h+1},\cdots,u_k$ are called at times $h+1,\cdots,k$, respectively. Therefore, for $1\leqq i\leqq h-1$, all of the vertices in T_i may be called by time $t(u_i)+i+1\leqq t(u_j)+j$, by definition of h. For $h+1\leqq i\leqq k$, the vertices in T_i may be called by time $t(u_i)+i\leqq t(u_j)+j$, and the vertices in T_h may be called by time $t(u_h)+1\leqq t(u_h)+h\leqq t(u_j)+j$. Hence,

$$b(u_h, T)\leqq t(u_j)+j=b(t) \quad \text{and } u_h\in \mathrm{BC}(T),$$

as asserted.

It still remains to show that for $j+1\leqq h\leqq k$, $u_h\notin BC(T)$. If u_h is the originator, let the times at which $u_1,\cdots,u_j$ are called from vertex v be $t_1,\cdots,t_j$, respectively. We note that each $t_i\geqq 2$. By Theorem 1, in order to call all the vertices in $\bigcup_{1\leqq i\leqq j} T_i$ as quickly as possible from vertex u_h, we can choose $t_i=i+1$, $i=1,\cdots,j$. Hence, $b(u_h,T)\geqq 1+j+b(u_j,T_j)=1+j+t(u_j)>j+t(u_j)=t(v)=b(v,T)$. □

COROLLARY 8. *For any tree T,* BC(T) *consists of a star with at least two vertices.*

An interesting by-product of Algorithm BROADCAST is that it can also find the broadcast number $b(v,T)$ of any vertex v in a tree T.

THEOREM 9. *Let v be a vertex in a tree T which is not in the broadcast center of T, and let the shortest distance from v to a vertex x in* BC(T) *be k. Then $b(v,T)=k+b(x,T)=k+b(T)$.*

Proof. Clearly $b(v,T)\leqq k+b(x,T)$.

For the converse, consider the path in T from v to x, and let the vertex adjacent to x on this path be w (where $v=w$ is possible). Clearly $b(w,T(w,x))\leqq b(x,T)-1$. This implies that if $b(x,T(x,w))<b(x,T)$ then, since one can broadcast from w by first calling x, $b(w,T)\leqq 1+\max\{b(x,T(x,w)),b(w,T(w,x))\}\leqq 1+\max\{b(x,T)-1, b(x,T)-1\}=b(x,T)$. Since this implies that w is also in BC(T), which is a contradiction, we have $b(x,T(x,w))=b(x,T)$. But $b(x,T(x,w))=b(x,T)$ implies that $b(v,T)\geqq k+b(x,T(x,w))=k+b(x,T)$.

Consequently $b(v,T)=k+b(x,T)=k+b(T)$. □

4. Complexity analysis of Algorithm BROADCAST.

In this section we will show that Algorithm BROADCAST has a worst-case time (and space) complexity of $O(N)$ for a tree T with N vertices.

Clearly steps 0, 1 and 2 require at most $O(N)$ time. The endvertices of the trees T and T' in steps 1 and 2 can be found in $O(N)$ time, and the value of the label $t(w)$ assigned in step 2 is simply the number of endvertices in T adjacent to w. This number can be determined by making one pass over the list of vertices adjacent to w. The total time spent examining such adjacent vertices, for all endvertices of T' is $O(N)$, since T' has $O(N)$ edges.

Since at least one vertex w is deleted in every execution of step 5, the test in step 3 is executed in $O(N)$ time. We must therefore show that the total amount of time spent in the iteration involving steps 4, 5 and 6 is $O(N)$.

In order to do this we will create two useful data structures. The first data structure is an array L of N pointers, the ith of which corresponds to a linked list of vertices w of T for which $t(w)=i$. After the value for $t(v)=k$ is determined in step 6, we can, in constant time, add v to the end of the kth list.

The second data structure is another list NBR of N pointers, the ith of which corresponds to a linked list of those vertices w which are adjacent to vertex i *and* which have been given a value $t(w)$. Elements can be added to this list in constant time during the execution of step 4. If vertex w is selected in step 4, it can be added to the *front* of the NBR(v) list for the vertex v adjacent to w in T'. It is important to note two things about this addition. First, the value $t(w)$ will be greater than or equal to the value of any other vertex which has previously been added to v's list; this is guaranteed by Lemma 4. Second, we can use the value $t(w)$ to form an updated value of $t(v) = \max_{1 \leqq i \leqq k} \{t(w_i) + i\}$ in constant time. For each vertex v, we maintain a value MAX (v). As a new vertex w, with value $t(w)$, is added to v's list, the updated value of MAX (v) is determined by

$$\text{MAX}(v) \leftarrow \max\{\text{MAX}(v)+1, t(w)+1\}.$$

For example, if the current values of vertices adjacent to v are 2 2 0, and the current maximum is 4 from $\left(\begin{smallmatrix} 2 & 2 & 0 \\ 1 & 2 & 3 \end{smallmatrix}\right)$, and if the next value added to v's list is 3, then the new maximum is

$$\max\{4+1, 3+1\} = 5 \text{ from } \begin{pmatrix} 3 & 2 & 2 & 0 \\ 1 & 2 & \underline{3} & 4 \end{pmatrix}.$$

Returning to step 4 we see that in order to select the next vertex w, all we need to do is either move a pointer to the next element on list $L(i)$ or find the first nonzero element on the next list $L(i+1)$, $L(i+2)$, etc. The total time spent moving these pointers is clearly bounded by $O(N)$. The total time spent finding the vertices v adjacent to vertices w is also bounded by $O(N)$ since all we must do is examine, once for each vertex, the vertices adjacent to it. Thus the total time spent in step 4 is $O(N)$.

Since the set W in step 5 is essentially the set of unexamined items on the lists $L(i)$, the process of deleting w from W and T' simply involves moving a pointer to the next vertex after w on some list. The process of adding w to the set U is simply a matter of adding w to the front of the list NBR (v). Thus the total time spent in step 5 is $O(N)$.

In step 6 the process of deciding if v is an endvertex of T' is simply a matter of knowing whether all, or all but one, of v's neighbors appear on the list NBR (v). This can easily be determined in constant time by a simple updating process. The process of determining the value of $t(v)$ is simply a matter of getting the current value of MAX (v); and the process of adding v to W involves adding v to the end of list L(MAX (v)). Thus the total time spent per vertex in step 6 is bounded by a constant.

Consequently, the total time spent executing steps 3, 4, 5 and 6 is $O(N)$.

Finally, we must show that the time spent executing step 7 is bounded by $O(N)$. The process of getting the last vertex v remaining in T' is simply that of moving a pointer to the last nonzero element on a list $L(i)$. The process of determining the smallest integer j such that $t(u_j) + j = \max_{1 \leqq i \leqq k} \{t(u_i) + i\}$ is simply one of moving a pointer down the list NBR (w) to the first place where the value MAX $(u_i) + i$ equals MAX (v). This clearly requires at most $O(N)$ time.

Thus, Algorithm BROADCAST has worst-case time complexity of $O(N)$. The space requirements are also $O(N)$ since the lists $L(i)$ and NBR (w) require $O(N)$ words each, and the tree T can be stored via linked lists also requiring $O(N)$ words.

5. NP-completeness.

D. S. Johnson [10] has shown that the problem of determining $b(u)$ for an arbitrary vertex u in an arbitrary graph G is NP-complete. It is with his permission that we present this result here. To begin, we state a more general problem:

BROADCAST TIME. Given a graph $G = (V, E)$ with a specified set of vertices $V_0 \subseteq V$ and a positive integer k, is there a sequence

$$V_0, E_1, V_1, E_2, V_2, \cdots, E_k, V_k,$$

where $V_i \subseteq V$, $E_i \subseteq E$, E_i consists only of edges with exactly one vertex in V_{i-1},

$$V_i = V_{i-1} \cup \{v: uv \in E_i\},$$

and $V_k = V$? (V_i is the set of vertices who are informed at time i with calls along the edges in E_i.)

It is easy to show that this general problem reduces to our special case when $|V_0| = 1$. We will first show that the general problem is NP-complete.

We will now relate this broadcast problem to the three-dimension matching (3DM) problem which has been shown (cf. Garey and Johnson [8]) to be NP-complete.

3DM. Let $X = x_1, \cdots, x_m$, $Y = y_1, \cdots, y_m$, $Z = z_1, \cdots, z_m$ and let $M \subseteq X \times Y \times Z$. Does there exist a subset of M of size m such that each pair of elements of the subset disagree in all three coordinates?

We demonstrate that a solution to the problem 3DM is equivalent to a solution of the BROADCAST TIME problem with $k = 4$ in a certain graph G which can be constructed from the set M in time polynomial in m. The graph G is illustrated in Fig. 3. The independent sets V_0, M are equal in size and the bipartite subgraph induced by these sets is complete. If $(x_i, y_j, z_k) \in M$ then the corresponding vertex of M in G is joined to the vertices x_i of X, y_j of Y and z_k of Z. (For illustrative purposes (x_1, y_2, z_3) is assumed to be an element of M.) All other edges are precisely as depicted.

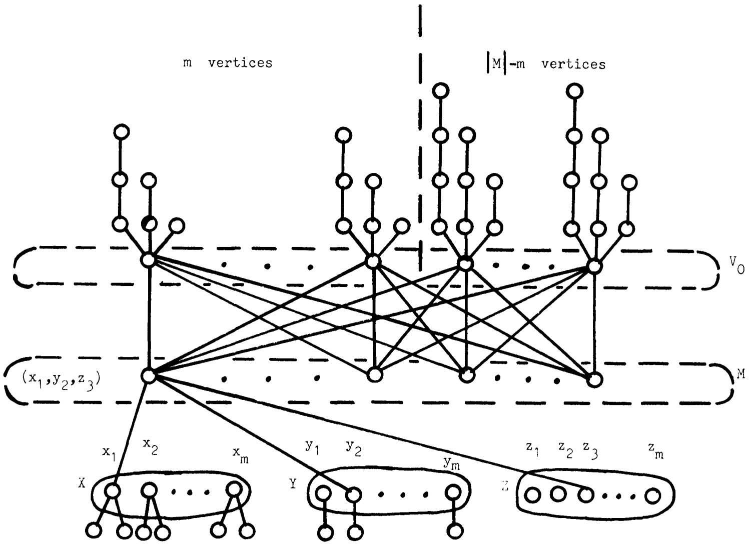

FIG. 3. *The graph G corresponding to the problem 3DM.*

Suppose that we have a solution to BROADCAST TIME in G. In order that the top line of vertices of G may be informed by time 4, $|M| - m$ vertices of V_0 must broadcast in an upward direction during the first time interval. In order that X, Y, Z and the bottom line may be informed by time 4, the remaining m vertices of V_0 must broadcast to an m-subset S of M at time 1. The vertices in S must be able to broadcast to distinct elements of X, Y, Z at times 2, 3, 4 respectively. This is possible if and only if S is a solution to 3DM.

To show that problem 3DM is reducible to the problem of determining $b(u)$ for an arbitrary vertex u in an arbitrary graph, we further extend the graph G to the graph H in Fig. 4. A solution to the broadcast time problem with $k = 4$ and V_0 the independent vertex set of size $m \geqq 2$ in G is equivalent to determining if $b(u) = m + 5$ in H, where H is obtained from G (in time polynomial in m) as follows. Add to G an independent set $U = \{u_1, u_2, \cdots, u_m\}$ and a vertex u and edges (u, u_i) for $1 \leqq i \leqq m = |V_0|$. Append to u_1 paths of length $6, 7, \cdots, m+4$, to u_2 paths of length $6, 7, \cdots, m+3, \cdots$, to u_{m-2} paths of length 6, 7, and to u_{m-1} a path of length 6. Finally, add m edges creating a matching of U to V_0.

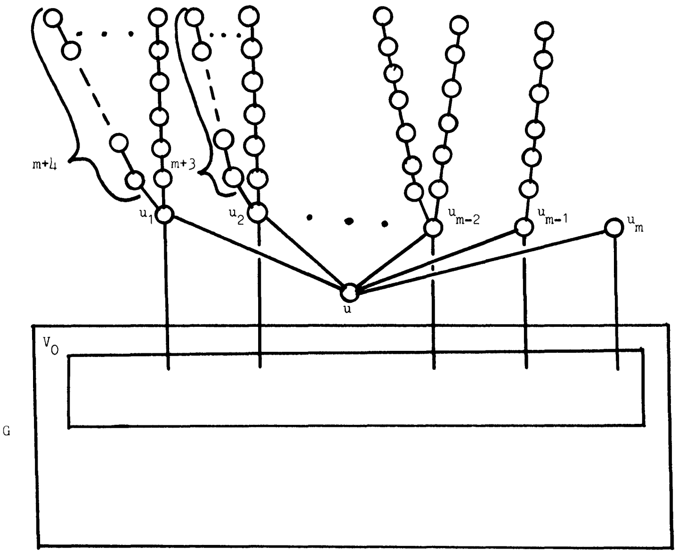

FIG. 4. *Supergraph H of G.*

6. Conclusions.

6. Conclusions. Algorithm BROADCAST for finding the broadcast center of a tree raises a number of questions about the concept of broadcasting, or information dissemination, in communication networks. In particular, how much faster can broadcasting take place if one permits either "conference" calls, instead of two-person calls, or "long distance" calls instead of local, next-neighbor calls? Another class of problems concerns multiple-message broadcasting.

We initiated the study of the problem of transmitting information from one point to all points of a graph as quickly as possible in 1977 [4] when we presented most of the results in this paper. Subsequently much work has been done concerning this problem [3], [5], [6], [7], [9], [12]. In [4] we used the terms "gossip number" and "gossip center" rather than broadcast number and broadcast center. We prefer "broadcast" to "gossip" since gossip problems (for example [1], [2], [11]) involve every vertex (not just one) initiating a piece of information, and one wants to know how many calls are required and/or how long it will take before everyone knows everything.

Acknowledgments. The authors extend their thanks to the referees for many helpful comments and to Terry Beyer for discussions concerning the data structures and time complexity of Algorithm BROADCAST. We also thank Sandra Mitchell for helpful early discussions about broadcasting (and two of the authors congratulate the other on his marriage to her).

REFERENCES

[1] B. BAKER AND R. SHOSTAK, *Gossips and telephones*, Discrete Math., 2 (1972), pp. 191–193.
[2] G. BERMAN, *The gossip problem*, Discrete Math., 4 (1973), p. 91.

[3] E. J. Cockayne and S. T. Hedetniemi, *A conjecture concerning broadcasting in m-dimensional grid graphs*, CS-TR-78-14, Computer Science Department, University of Oregon, submitted.

[4] E. J. Cockayne, S. T. Hedetniemi and P. J. Slater, *The gossip center of a tree* (abstract), in Proc. 8th Southeastern Conference on Combinatorics, Graph Theory and Computing, 1977, p. 658.

[5] A. M. Farley, *Minimal broadcast networks*, Networks, 9 (1979), pp. 313–332.

[6] A. M. Farley and S. T. Hedetniemi, *Broadcasting in grid graphs*, Proc. 9th Southeastern Conference on Combinatorics, Graph Theory and Computing, 1978, pp. 275–288.

[7] A. M. Farley, S. T. Hedetniemi, S. L. Mitchell and A. Proskurowski, *Minimum broadcast graphs*, Discrete Math., 25 (1979), pp. 189–193.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. F. Freeman, San Francisco, 1979.

[9] S. T. Hedetniemi and S. L. Mitchell, *A census of minimum broadcast graphs*, J. Combinatorics, Inform. Systems Sci., 5 (1980), pp. 119–129.

[10] D. S. Johnson, Private communication, May, 1978.

[11] W. Knodel, *New gossips and telephones*, Discrete Math., 13 (1975), p. 95.

[12] A. Proskurowski, *Minimum broadcast trees*, IEEE Trans. Comput., to appear.