Minimum-Time Line Broadcast Networks

Arthur M. Farley
Computer Science Department, University of Oregon,
Eugene, Oregon 97403

Broadcasting refers to the process of message dissemination in a communication network whereby a message, originated by one member, becomes known to all members. Line broadcasting assumes that members may "switch-through" any number of calls during a time unit. An algorithm is presented which produces a calling schedule completing line broadcasting in minimum time from any member of any tree. A discussion of three types of broadcasting and their associated minimum time networks is included.

INTRODUCTION

Broadcasting refers to the process of message dissemination in a communication network whereby a message, originated by one member, is transmitted to all members of the network. We model a communication network with a graph $G = (V, E)$ consisting of a set V of vertices, or *members*, and a set E of edges, or *communication lines*. We assume that a network has n members uniquely labelled by the integers 1 through n. A member can transmit a message to another member by making a *call*. We assume that each call requires one unit of time, where a time unit is understood to be dependent upon message length.

A given broadcast is specified by a corresponding legal calling schedule. A *calling schedule* is a set of statements of the form "Member i calls member j during time unit t," where $1 \leqslant i, j \leqslant n$ and $t \geqslant 1$. Member i is the *sender* and member j is the *receiver* of the corresponding call. A *legal calling schedule* is a calling schedule satisfying the following two constraints:

(i) during a given time unit t, a member is scheduled to participate as sender or receiver in at most one call;
(ii) for each call scheduled a time unit t, the sender is either the message originator or has been the receiver of a call placed during time unit k, $1 \leqslant k < t$.

The second criterion guarantees that a sender is *informed* of the message being broadcast.

Restrictions as to what calls can be placed in a network during a given time unit differentiate three types of broadcasting. In *local broadcasting*, an informed member can only call a member to which it is directly connected (i.e., an adjacent vertex).

0028-3045/80/0010-0059$01.00

This is like the situation in most message-switched networks. In *path broadcasting*, an informed member can call any other member in the network, provided there is an open path between the two members. An *open path* is a path in the network such that every member and line of the path is not involved in another call during that time unit. In *line broadcasting*, an informed member can call any other member, provided there is an open line between the members. An *open line* is a path such that every line of the path is not involved in another call during that time unit. The difference between path and line broadcasting is that in line broadcasting a member may "switch-through" any number of calls. Switching one call in path broadcasting completely occupies a member. Line broadcasting approximates the situation in circuit-switched networks.

The minimum number of time units necessary to complete broadcasting in a network of n members is $\lceil \log_2 n \rceil$ [1]. A *minimum-time broadcast network* is a communication network such that broadcast can be completed in the minimum number of time units regardless of message originator. Each type of broadcasting imposes its own constraints upon the architecture of associated minimum-time broadcast networks. In the case of line broadcasting, the constraints are minimal—that the network be connected. A network is *connected* if there exists a path between every pair of members.

MINIMUM-TIME LINE BROADCASTING

Theorem. Line broadcasting can be completed in minimum time ($\lceil \log_2 n \rceil$ time units) in any connected network, regardless of message originator.

Proof: Proof of the theorem is presented in two phases. First is the presentation of an algorithm which produces a calling schedule realizing minimum broadcast time from any member of any given tree. Following the formal presentation of the algorithm, an example execution of the algorithm is provided to assist in understanding. The second phase of the proof is the verification of the correctness of the given algorithm. Since any connected network contains a spanning tree, successful completion of these two phases constitutes a proof of the theorem.

Phase 1. An algorithm, named MLB, is to be presented. The algorithm accepts as input a tree of n members and a selected member who is to serve as message originator. MLB produces as output a calling schedule which will complete line broadcasting from the given message originator in $\lceil \log_2 n \rceil$ time units.

The calling schedule produced by MLB reflects the following principle: the more calls to be made during a given time unit, the shorter (in number of lines transversed) should each call be. The algorithm develops the calling schedule in reverse time order. It first considers the desired state of the network after time unit $\lceil \log_2 n \rceil$, which is that all members are informed. MLB then works backward in time, making as many calls as possible in each preceding time unit, under the constraint that an informed member can be a participant in at most one call during any time unit. If, after time unit t, m members are informed, it is obvious that at most $\lfloor m/2 \rfloor$ calls can be made during time unit t. MLB determines a set of $\lfloor m/2 \rfloor$ calls which do not involve any line-use conflicts. Thus, $m - \lfloor m/2 \rfloor$ members must be informed after time unit $t - 1$. The algorithm successively considers each preceding time unit, through time unit 1. After time unit 0, only the message originator is informed of the message. The calls that are scheduled during each time unit are between "closest" elements of the set of members of the network to be informed after that time unit. By scheduling calls between such members, all line-use conflicts are avoided.

The surface form and semantics of two control structures used in the formal algorithm presentation deserve explanation. The first has the form

```
REPEAT k TIMES
      ·
      · body
      ·
END-REPEAT
```

The statements of the body are cyclically executed k times. The other control structure has the form

```
CASE
  C1: (condition 1):
      ·
      · body 1
      ·
  C2: (condition 2):
      ·
      · body 2
      ·
  Cn: (condition n):
      ·
      · body n
      ·
END-CASE
```

The conditions are evaluated in the natural order until one is found to be true, at which point the associated body is executed. At most one body is executed. If no condition is true, no body is executed.

While scheduling calls for a given time unit, the algorithm prunes leaf members of the tree, finally reducing it to a single member. A *leaf* member is directly connected to only one member of the tree which is known as its *remote* member. Algorithm MLB is formally defined in Table I. The function make-call has access to the array named time and is defined as follows:

```
FUNCTION  make-call (sender, receiver, time-unit)
  output ("Member" sender "calls member" receiver "during
      time unit" time-unit ".")
  set time[receiver] to time-unit
end
```

The following example of MLB in action will hopefully clarify its operation. Consider the tree in Fig. 1 below. The figure represents the situation after time unit $\lceil \log_2 n \rceil = 4$, with all members informed. Let member 5 be the message originator. Assume that during each cycle the members are pruned in decreasing order of labels. (Note that the algorithm is applicable with differing message originator and pruning

TABLE I. The Algorithm MLB.

ALGORITHM MLB			
Data Structures:			
inputs	*tree*	-	*a tree.*
	n	-	*the number of vertices in the tree.*
	mo	-	*the message originator; an integer, $1 \le mo \le n$.*
local arrays	*time*	-	*time[i] will eventually indicate the time unit during which member i becomes informed; If time[i]=0 during scheduling for a given time unit, member i is informed of the message before or during that time unit.*
	mark	-	*mark[i] indicates which informed member, if any, desires to communicate through or with member i during the current time unit.*
	tr	-	*holds the active, pruned copy of the input tree during scheduling for a given time unit.*
local variable	*unit*	-	*the time unit currently being scheduled.*

Method:

```
STEP #

0          MLB(tree, n, mo)
1              set time[i] to 0 , for $1 \le i \le n$
2              set unit to $\lceil \log_2 n \rceil$
3              REPEAT $\lceil \log_2 n \rceil$ TIMES
3.1                set mark[i] to 0 , for $1 \le i \le n$
3.2                set tr to tree
3,3                REPEAT n-1 TIMES
3.3.1                  prune any leaf lf , with remote member r, from tr
3.3.2                  CASE
3.3.2.1                    C1:(time[lf]=0 and mark[lf]=0 and mark[r]=0):
3.3.2.1.1                      set mark[r] to lf
3.3.2.2                    C2:(time[lf]=0 and mark[lf]=0 and mark[r]=mo):
3.3.2.2.1                      make-call(mark[r],lf,unit)
3.3.2.2.2                      set mark[r] to 0
3.3.2.3                    C3:(time[lf]=0 and mark[lf]=0 and mark[r]>0):
```

TABLE I. *(Continued from previous page.)*

```
3.3.2.3.1              make-call(1f,mark[r],unit)
3.3.2.3.2              set mark[r] to 0
3.3.2.4             C4:(time[1f]=0 and mark[1f]=mo):
3.3.2.4.1              make-call(mark[1f],1f,unit)
3.3.2.5             C5:(time[1f]=0 and mark[1f]>0):
3.3.2.5.1              make-call(1f,mark[1f],unit)
3.3.2.6             C6:(time[1f]>0 and mark[1f]>0 and mark[r]=0):
3.3.2.6.1              set mark[r] to mark[1f]
3.3.2.7             C7:(time[1f]>0 and mark[1f]>0 and mark[r]=mo):
3.3.2.7.1              make-call(mark[r],mark[1f],unit)
3.3.2.7.2              set mark[r] to 0
3.3.2.8             C8:(time[1f]>0 and mark[1f]>0 and mark[r]>0):
3.3.2.8.1              make-call(mark[1f],mark[r],unit)
3.3.2.8.2              set mark[r] to 0

                END-CASE

              END-REPEAT

3.4           prune the last member 1f from tr
3.5           CASE
3.5.1            C1:(time[1f]=0 and mark[1f]=mo):
3.5.1.1              make-call(mark[1f],1f,unit)
3.5.2            C2:(time[1f]=0 and mark[1f]>0):
3.5.2.1              make-call(1f,mark[1f],unit)
              END-CASE
3.6           set unit to unit -1
            END-REPEAT
          END
```

order.) Each succeeding figure (Figs. 2-5) shows the calls scheduled during a time unit. Those members informed during a time unit are indicated by the setting of time[i] to that time unit.

Before considering the formal verification of algorithm MLB, note that the time complexity of the algorithm is $O(n\lceil\log_2 n\rceil)$. The algorithm prunes a tree, reducing it

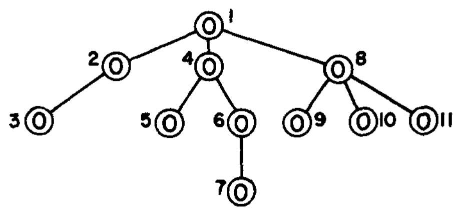

FIG. 1. Line broadcast network after time unit $\log_2 n = 4$ with all members informed.

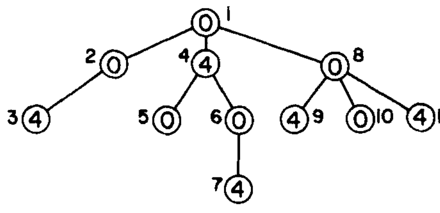

FIG. 2. Member 10 calls member 11 during time unit 4. Member 8 calls member 9 during time unit 4. Member 6 calls member 7 during time unit 4. Member 5 calls member 4 during time unit 4. Member 2 calls member 3 during time unit 4.

to a single member, $\lceil \log_2 n \rceil$ times. An arbitrary order for pruning can be determined in linear time prior to executing algorithm MLB.

Phase 2. The formal verification of the correctness of algorithm MLB is realized through the proof of five lemmas concerning the algorithm and the calling schedule it produces. The first lemma verifies that each member is scheduled to be involved in at most one call during any given time unit. The second verifies that the calling schedule involves no line-use conflicts during any given time unit. The third lemma verifies that if m members are to be informed after time unit t, MLB schedules $\lfloor m/2 \rfloor$ calls during time unit t. The fourth verifies that the sequence representing the number of informed members after time unit i (i.e., $a_{\lceil \log_2 n \rceil} = n, a_{i-1} = a_i - \lfloor a_i/2 \rfloor$) converges to exactly one after time unit 0 (i.e., $a_0 = 1$). The fifth lemma verifies that any member scheduled to make a call during a given time unit either is the message originator or has received a call prior to that time unit.

Lemma 1. During any time unit, a member is scheduled to participate as sender or receiver in at most one call.

Proof: A member may become a participant in a call during a given time unit either when it is pruned from tr or after it has been pruned. A member becomes a participant when pruned from tr as member lf in case C2, C3, C4, or C5 of Step 3.3.2 or in case C1 or C2 of Step 3.5. Since a member is pruned from tr only once during scheduling for any time unit, such a member can be involved in at most 1 call. A member becomes a participant in a call after being pruned from tr by first marking its remote member when being pruned (as member lf) in case C1 of Step 3.3.2. This marking

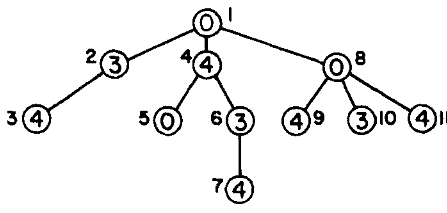

FIG. 3. Member 8 calls member 10 during time unit 3. Member 5 calls member 6 during time unit 3. Member 1 calls member 2 during time unit 3.

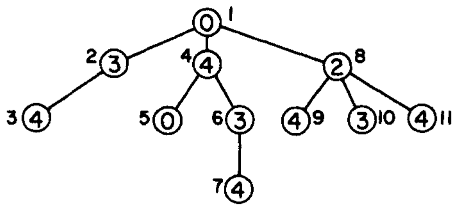

FIG. 4. Member 5 calls member 8 during time unit 2.

may be transmitted to another member by case C6 of Step 3.3.2, when that or even a subsequent remote member is itself pruned. The member may finally become a participant in a call in one of two ways. It may be as mark[lf] in case C4, C5, C6, or C7 of Step 3.3.2 or case C1 or C2 of Step 3.5. In such a case, lf is pruned at this point and mark[lf] (the member in question) is removed from further consideration during that time unit. The member may be mark[r] in case C2, C3, C7, or C8 of Step 3.3.2. In such case, r is not immediately pruned. However mark[r] is reset to 0 at this point, eliminating the member in question from further consideration during that time unit. ||

Lemma 2. During any time unit, the scheduled calls do not create a line-use conflict.

Proof: During scheduling activity for any time unit, MLB traverses each line of the tree only once. This transversal can be said to occur when the first of a line's two end members is pruned from tr. At most one member's label (number) is passed over the line (to mark the other end element) as a possible participant in a call during that time unit. This occurs in cases C1, C2, C3, C6, C7, and C8 of Step 3.3.2. In all other cases, no member label is passed over the line at all. In such cases, the line will be inactive during that time unit. ||

Lemma 3. Assuming that m members are to be informed after time unit t, mLB schedules exactly $\lfloor m/2 \rfloor$ calls during time unit t.

Proof: We shall first determine that the number of calls is less than or equal to $\lfloor m/2 \rfloor$. When beginning to schedule calls for time unit t, it is those members j such that time[j] = 0 which are to be informed after time unit t. During any time unit, if member i is a participant in a call, time[i] = 0 as scheduling begins for that time unit. This can be verified by inspection of MLB. Only case C1 of Step 3.3.2 can generate

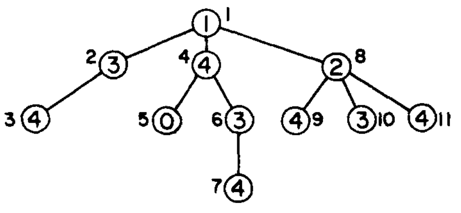

FIG. 5. Member 5 calls member 1 during time unit 1.

a new mark, that mark being a member lf such that time[lf] = 0. An inspection of all cases which generate a call indicates that only a mark or a member lf such that time[lf] = 0 can become a participant in a call. Since each such member participates in at most one call (Lemma 1) and each call requires two participants, at most $\lfloor m/2 \rfloor$ calls are scheduled.

It remains to prove that the number of calls scheduled cannot be less than $\lfloor m/2 \rfloor$. The proof shall be by contradiction. Suppose that fewer than $\lfloor m/2 \rfloor$ calls are scheduled. If this is so, there must be at least two members i and j such that time$[i] = 0$, time$[j] = 0$, and both members are not scheduled participants in a call during time unit t (after executing Step 3.5 for unit = t). Consider the path between j and i, represented as $(jn_1n_2 \cdots n_m \cdots n_{k-1}n_ki)$. During scheduling for time unit t, of those elements of that path either j, i, or an intermediate element n_m is the last to be pruned from tr. Suppose it is member i. Then, j is the first element of the path to be pruned, with remote member being n_1. Since time$[j] = 0$, only case C1 of Step 3.3.2 can apply if j is not to be a participant in a call. Thus, n_1 is marked with j and is the next element of the path to be pruned. When n_1 is pruned it must still be marked with j. Otherwise, j would have become a participant in a call by application of case C2, C3, C7, or C8 of Step 3.3.2. When n_1 is pruned, only case C6 can apply if j is not to be a participant in a call during time unit t. Thus, j is transmitted as a mark to n_2. For each n_p, $2 \leqslant p < k$, the argument is analogous. Only case C6 of Step 3.3.2 can be applicable if member j is not to be scheduled as a call participant. Similarly, when n_k is pruned (with i as remote member), member i becomes marked with member j by case C6 of Step 3.3.2. When member i is pruned, time$[i] = 0$ and mark$[i] > 0$. Therefore, member i will become a participant in a call (with member j) by case C2 or C3 of Step 3.3.2 or case C1 or C2 of Step 3.5. This contradicts our initial assumption. A similar contradiction is easily determined for the case where j is the last pruned or the case in which i and j are adjacent in the tree. Consider then the case when an intermediate element of the path, member n_m, is the last member of the path to be pruned. When j is pruned its remote member is n_1 and when i is pruned its remote member is n_k. When n_p is pruned from tr, n_{p+1} is the remote node for $1 \leqslant p < m$; when n_p is pruned from tr, n_{p-1} is the remote member for $m < p \leqslant k$. As in the argument above, case C1 of Step 3.3.2 must apply when each end element of the path is pruned. As each n_p is pruned, $|p - m| > 1$, i or j is transmitted as mark to the remote member by case C6 of Step 3.3.2. Suppose n_{m-1} is pruned prior to n_{m+1}. Then, n_m is marked with j according to case C6 of Step 3.3.2. When n_{m+1} is later pruned, mark$[n_{m+1}] = i$ and mark$[n_m] = j$. Therefore, case C7 or C8 of Step 3.3.2 will be applicable and member i will become a participant in a call (with member j) during time unit t. The argument is analogous in the case that n_{m+1} is pruned prior to n_{m-1}.

Since there cannot be two such members after Step 3.5 is executed for time unit t, there cannot be less than $\lfloor m/2 \rfloor$ calls scheduled. Therefore, exactly $\lfloor m/2 \rfloor$ calls are scheduled by MLB during time unit t. ||

Lemma 4. Given integer n and the sequence defined as:

$$a_0 = n, a_i = a_{i-1} - \lfloor a_{i-1}/2 \rfloor, \quad i = 1, 2, \ldots .$$

If k is the least integer such that $a_k = 1$, then $k = \lceil \log_2 n \rceil$ for all $n > 0$.

Proof: Obviously, k cannot be less than $\lceil \log_2 n \rceil$ since $a_0 = n$ and $a_i \geqslant a_{i-1}/2$ (as $a_i = \lceil a_{i-1}/2 \rceil$).

Thus, it remains to prove that k cannot be greater than $\lceil \log_2 n \rceil$. The integer k would take on a maximal value when the decreases are minimal. Thus, suppose the decreases are minimal. In other words, let

$$a_i = (a_{i-1} + 1)/2 \qquad (\text{or } a_{i-1} = 2a_i - 1).$$

Let us consider the sequence in reverse, letting $b_j = a_{k-j}$. Thus, $b_0 = 1$, $b_1 = 2$, and $b_j = 2b_{j-1} - 1$, for $j \geqslant 2$. This latter recurrence relation can be solved, yielding as a result: $b_j = 2^{j-1} + 1$, for $j \geqslant 1$. Now, suppose that k is greater than $\lceil \log_2 n \rceil$. Then, $k \geqslant \lceil \log_2 n \rceil + 1$. If $k = \lceil \log_2 n \rceil + 1$ then $b_k = n + 1$. This is the least b_k can ever be made to be due to previous assumption as to the minimal size of the increment (decrement in the a_i sequence). But, $b_k = a_0 = n$, which is a contradiction. Therefore, k cannot be greater than $\lceil \log_2 n \rceil$. ||

Lemma 5. If member i is scheduled to make a call during time unit t, member i is either the message originator or receives a call during time unit j, $1 \leqslant j < t$.

Proof: In case C2 or C7 of Step 3.3.2, mark[r] makes the call and mark[r] is the message originator. In case C4 of Step 3.3.2 or case C1 of Step 3.5, mark[lf] makes the call and is the message originator. In those other cases where calls are scheduled, the member making the call may or may not be the message originator. If it is, then no further consideration is necessary. So, assume it is not. In Lemma 3, it was proven that if member i is eligible to be a participant in a call during time unit t, time[i] = 0 when scheduling begins for time unit t. If member i places a call during time unit t, then time[i] = 0 after scheduling has been completed for time unit t. This is proven, by inspection of the algorithm. Step 2 of the function make-call is the only step which can alter time array values after their initialization to 0. That step changes the value of the recipient's associated time only. Therefore, during time unit $t - 1$, member i remains eligible to be a call participant. If it receives a call, it requires no further consideration. If it does not, it similarly remains eligible for call participation in time unit $t - 2$. Lemma 4 verifies that after time unit 1, only 2 members remain eligible for call participation. If the member in question has still not been called and is not the message originator, he will be called by the message originator during time unit 1. That this occurs results from Lemma 3 and the sequence of cases in Steps 3.3.2 and 3.5 which guarantees that any member participating in a call with the message originator is the recipient of that call. ||

Corollary. The message originator, mo, is the only member informed after time unit 0.

This completes the formal verification of the algorithm. Lemmas 1 and 5 verify that the calling schedule produced by MLB is a legal calling schedule. Lemma 2 verifies that no line-use conflicts occur between calls of that calling schedule. Finally, Lemmas 3 and 4 verify that exactly $\lceil \log_2 n \rceil$ time units are required to complete broadcasting by the calling schedule. Since a calling schedule can be found to complete line broadcasting throughout any tree from any member in minimum time, the conjecture of the theorem is proven. ||

DISCUSSION

The result for minimum-time line broadcasting networks is in contrast to the following result concerning minimum-time local broadcasting networks.

Theorem. No tree of more than three members is a minimum-time local broadcasting network.

Proof: Any tree which has more than three members has at least two members with degree equal to one (i.e., leaf vertices). Consider message broadcast when such a member is the message originator. During time unit 1, that member calls his one adjacent member. Since the message originator can place no further calls, the situation is now equivalent to a message broadcast originated by that adjacent member in a network of $n - 1$ members. If n is not equal to $2^k + 1$ for some k, the overall message broadcast can, at best, be completed in $\lceil \log_2 n \rceil + 1$ time units, which is not minimal.

If n does equal $2^k + 1$, it is possible that broadcast from a member of degree one can be completed in minimal time. That member could call its one adjacent member, from which broadcasting could be completed in $\lceil \log_2 n \rceil - 1$ time units. However, if broadcasting is possible in minimum time from that adjacent member, each member informed during a later time unit must be able to place a call during each successive time unit. Consider the member called by the adjacent member during the next to the last time unit of broadcasting. That member must call a new member during the last time unit. Broadcast originated by that new member can not be completed in $\lceil \log_2 n \rceil$ time units. During time unit 1, it can call its only adjacent member. During the next time unit that member can call its only other adjacent member. A broadcast throughout $n - 2$ members cannot be completed in $\lceil \log_2 n \rceil - 2$ time units for $n > 3$. ||

Minimum-time line broadcasting networks require strictly fewer lines than minimum-time local broadcasting networks for networks of more than three members. Several recent reports provide a closer look at the number of lines required in minimum-time local broadcasting networks [1, 2].

A new measure on broadcasting suggests itself, however. Let us define *broadcast cost* to be the sum of the number of lines traversed by all calls placed during a given broadcast. Each call traverses one line of the network in local broadcasting. Since $n - 1$ calls are placed during any given broadcast, local broadcast cost is constant, being equal to $n - 1$. Cost need not be constant (over message originator or network) for line broadcasting, however. Algorithm MLB traverses the $n - 1$ lines of a tree of n networks $\lceil \log_2 n \rceil$ times. Therefore, broadcast cost for line broadcasting in an n member network has an upper bound of $(n - 1) \lceil \log_2 n \rceil$.

Let us define the *maxi-min line broadcast cost* of a given tree t (MMLBC(t)) to be the maximum of the minimum of broadcast costs achievable from the members of t over calling schedules which realize minimum time line broadcasts. The following two questions arise with respect to this measure:

(1) What is the minimum value of MMLBC(t) over all trees of n members?
(2) Can a class of trees T of n members be determined such that MMLBC(t) is minimal for each $t \in T$?

Answers to these questions have not yet been determined. The result of the following theorem is relevant to answering these questions in the future. Let MC(n) be the minimum value of MMLBC(t) over all trees t of n members.

Theorem. For $n > 3$,

$$n \leqslant \mathrm{MC}(n) \leqslant n + \frac{(\lceil \log_2 n \rceil - 2)(\lceil \log_2 n \rceil + 1)}{2}.$$

Proof: The cost of local broadcasting is defined to be $n - 1$. Since any tree of more than three members is not a minimum-time local broadcasting network, the cost of minimum-time line broadcasting from at least one member of the tree must be greater than $n - 1$. Therefore $\mathrm{MC}(n) \geqslant n$ in a tree of n members for $n > 3$.

The upperbound is determined by consideration of minimum-time line broadcasting within a minimum broadcast tree of n members. A *minimum broadcast tree* is a tree such that local broadcast can be completed from at least two of its members in $\lceil \log_2 n \rceil$ time units. A complete characterization of such trees has been completed [3]. The following calling schedule for line broadcasting produces the upper bound result of this theorem. Let the message originator call the most distant broadcast center of the tree. The *broadcast center* [4] is defined to be the subset of members of the tree from whom local broadcast can be completed in minimum time. This distance can be at most $\lceil \log_2 n \rceil$. That center member can complete the broadcast for its "half" of the tree in $\lceil \log_2 n \rceil - 1$ time units employing only local calls, each of length one. Eliminate that broadcast center and its "half" of the tree and repeat the same process. This time the distance to the center can be at most $\lceil \log_2 n \rceil - 1$. If one continues this process, the maximum line broadcast cost is bounded by:

$$\underbrace{\sum_{1}^{\lceil \log_2 n \rceil} i}_{\substack{\text{cost from message}\\ \text{originator}}} + \underbrace{n - 1 - \lceil \log_2 n \rceil}_{\substack{\text{cost from}\\ \text{others}}} = n + \frac{(\lceil \log_2 n \rceil - 2)(\lceil \log_2 n \rceil + 1)}{2}$$

||

Characteristics of minimum-time path broadcasting networks lie between the extremes established for the local and line broadcasting cases. Every tree is not a minimum-time path broadcasting network, as evidenced by star networks with more than three members. A *star network* is a tree with one central member to which all others are directly connected. Since the central member is an element of every non-null path in a star network, two calls can never be placed at any time during path broadcasting.

Given n members, trees do exist which are minimum-time path broadcasting networks, however. One class is simply the linear tree (i.e., path graph) of n members. Let such a network be called a *chain*. Let the members be numbered from 1 to n following the natural, end-to-end ordering of the chain. During any time unit an informed member places a call within a relevant chain. A *relevant chain* is a contiguous subsection of the original chain which contains no informed member other than the caller. The *initial member* of a relevant chain is the member with the lowest integer label. Each relevant chain consists of two contiguous halves, such that the length of the first half is equal to or one more than the length of the second half.

Given a message originator, a minimum time calling schedule for path broadcasting can be developed in a recursive manner, as follows:

> The message originator begins the process by placing a call within a relevant chain which is the whole network. Let an informed member call the initial member of the other half of its relevant chain and let both members make calls within their respective halves, as relevant chains, during the next time unit.
>
> This process stops when the length of the relevant chain for each informed member is one (i.e., is only that member itself).

CONCLUSION

It requires a minimum time of $\lceil \log_2 n \rceil$ time units to complete broadcasting in a network of n members. As would be expected, each type of broadcasting produces its own set of constraints upon the associated minimum-time broadcasting networks. An arbitrary tree, and thus any connected network, is sufficient to complete line broadcasting in minimum time. However, no tree of more than three members can be a minimum-time local broadcasting network. Finally, an arbitrary tree of n members may not allow minimum time path broadcasting, but a tree with n members does exist which allows path broadcasting to be completed in minimum time.

References

[1] A. M. Farley, "Minimal broadcast networks," *Networks*, **9**, 313–332 (1979).
[2] A. Farley, S. Hedetniemi, S. Mitchell, and A. Proskurowski, "Minimum broadcast graphs," *Discrete Math.*, **25**, 189–193 (1979).
[3] A. Proskurowski, "Minimum broadcast trees," Technical Report CS-TR-78-19, University of Oregon, Computer Science Department, 1978, submitted.
[4] P. J. Slater, S. T. Hedetniemi, and E. J. Cockayne, "Information dissemination in trees," Technical Report, CS-TR-78-11, University of Oregon, Computer Science Department, 1978, submitted.

Received November 1977