Reliable Minimum-Time Broadcast Networks

Arthur M. Farley

CIS-TR-87-06
May 4, 1987

Department of Computer and Information Science
University of Oregon

Reliable Minimum-Time Broadcast Networks

by
Arthur M. Farley
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403

Abstract

The specification of a communication network requires both a topological and an operational aspect. The topological aspect consists of a graph with vertices representing sites and edges their interconnection by lines. The operational aspect consists of calling procedures sufficient to complete various communication tasks. In earlier research, we specified several classes of minimum-time broadcast networks; such networks allow completion of the one-to-all broadcast task from any site as originator in the minimum possible time (i.e., $\lceil \log_2 n \rceil$ time units for n sites). Here we present extensions to the operational specifications of one of these classes to realize broadcast networks that are also immune to single site failures and to isolated site failures. In these networks, broadcast incurs the minimum possible delay of at most one time unit for each failing component encountered.

Introduction

Communication networks provide means for the dissemination of information among a set of *sites* by the transmission of *messages* through *calls* placed over *lines* interconnecting the sites. The sites may be processors in a highly parallel computer, computers in a distributed system, or hosts of a long-distance computer network. Several information dissemination tasks are of interest within these contexts. The basic task is that of *message transfer* between two sites, a sender and a receiver. Here our interest will be on *broadcasting*, which is the task of disseminating a message from one site, the originator, to all other sites as receivers (i.e., a one-to-all process).

The design specification of a communication network consists of two aspects: topological and operational. A network's topological specification represents the physical,

relatively static elements of the network: sites, lines, and any relevant properties of those elements (e.g., length, delay, capacity). A network's operational aspect describes elements of the dynamic behavior of the network, including routing tables and other features of the calling protocol and procedures.

We model the topological aspects of a network design by an undirected graph G = (V,E), where V is a set of *vertices* representing network sites and E is a set of *edges*, each edge connecting a pair of vertices, representing lines of the network. We model the operational aspects by *communication processes*, being *calling procedures* that are performed by each site during completion of an information dissemination task. The *call*, represented by the send procedure, transfers a message between two directly connected sites and is the primitive operation of a calling procedure. We assume that a call takes one unit of time and that a site can participate in at most one call during any time unit.

We design networks to satisfy any of several task-dependent properties of interest. These properties are generally concerned with network effectiveness, efficiency, and reliability. By effectiveness, we mean the ability of a network to successfully complete particular information tasks. Here we will be concerned with networks that are effective for broadcasting. By efficiency, we mean satisfaction of constraints on cost and time; here our concern will be on time required to complete broadcasting. Finally, reliability refers to a network's ability to complete an information dissemination task in the presence of failures in certain site and line components of the network; we say the network is *immune* to such failures. Here we will be interested in immunity to single site failures and to certain sets of multiple site failures.

Broadcast Networks

During message broadcast, a site may be called upon to make zero or more calls, depending upon the broadcast originator. As such, when specifying the operational aspect of a broadcast network, we must define a set of call lists for each site. A *call list* is a sequence of zero or more sites to be called in the given order by the site during a broadcast from a specified originator. The sites to be called by a given site depend upon the originator of a broadcast not the destination, as is the case for message transfer (i.e., the destination is all sites for any broadcast). Therefore, we define **call-list(x,y)** to be the ordered list of sites that are to be called by site **x** when site **y** is the originator of a broadcast. A call list may be empty when site **x** is not required to further the message originated by **y**. The list **call-list(x,x)** is the sequence of sites to be called by site **x** when it is the originator of a broadcast. A broadcast message must carry an indication of its originator so that other sites can perform the correct calling sequence.

Several list processing primitives must be defined to represent processing of the call list. We define **first(lst)** to be the first element of a non-empty list **lst** and **rest(lst)** of a non-empty **lst** to be **lst** with the first element removed. We represent the empty list by the symbol **nil**; **rest(lst)**, where **lst** consists of a single element, is **nil**. We can now specify the following generic broadcast process, to be performed by each site during broadcast of a message:

```
process broadcast (x, m):
;   broadcast process for site x, upon receiving or originating
;   message m of the form <originator, content>

    set clist to call-list(x, originator)
    until clist equals nil do
       begin
            send (first(clist), m)
            set clist to rest(clist)
       end
```

A Class of Minimum-time Broadcast Networks

In results of earlier research, we presented topological specifications of several classes of minimum-time broadcast networks [1]. A *minimum-time broadcast network* is a network in which a message broadcast is completed in the minimum possible time (i.e., $\log_2 n$ time units in a network of n sites) from any site as broadcast originator. One of our topological specifications was based upon a class of graphs known as star polygons. A *star polygon* is a finite graph formed by numbering n vertices from 0 to n-1 and connecting each vertex v (i.e., numbered v) by an edge to each of a finite set of other vertices $\{v+o_1, ..., v+o_k\}$ according to a fixed set of integer offsets $\{o_1, ..., o_k\}$, where all addition is assumed to be modulo n. The topology of one class of minimum-time broadcast networks is determined by star polygons having offsets $\{1, 2, ..., 2^{\lfloor \log_2 n \rfloor}\}$ [1]. We will call such graphs *log-star polygons*. Figure 1 presents an example of a log-star polygon for the case where n is equal to12.

Given the topological specification of a broadcast network, it remains to determine **call-list(x,y)** for each pair of sites **x** and **y** in order to complete its operational specification. For our log-star polygon topology, we define the distance **dist(x,y)** between

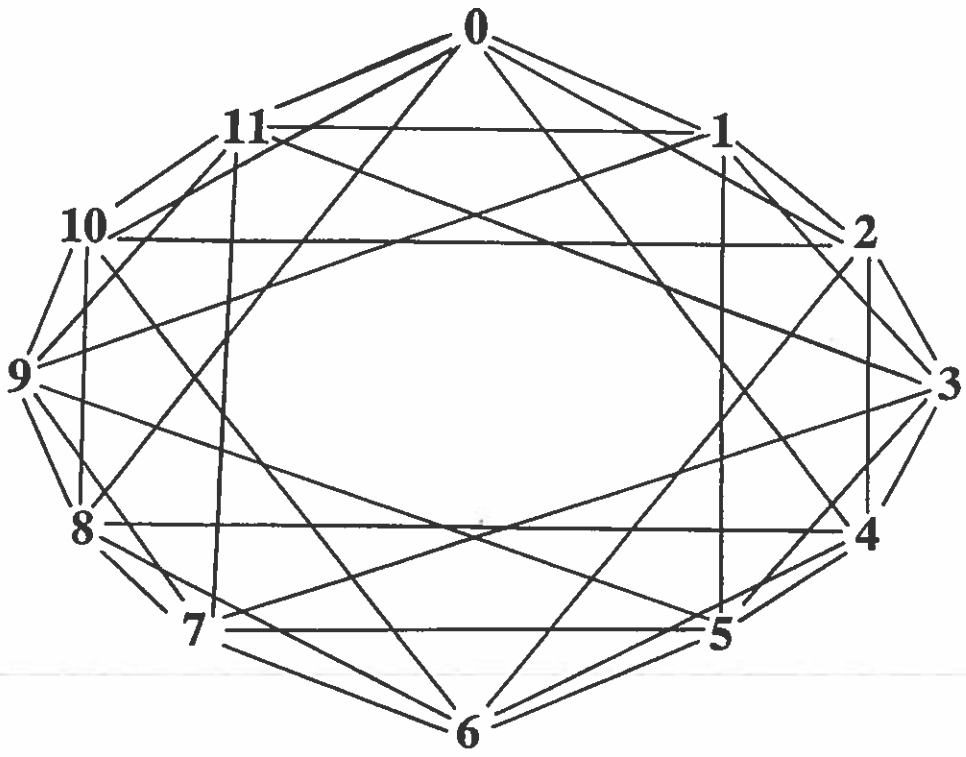

Figure 1. An example of our log-star polygon topology with 12 vertices.

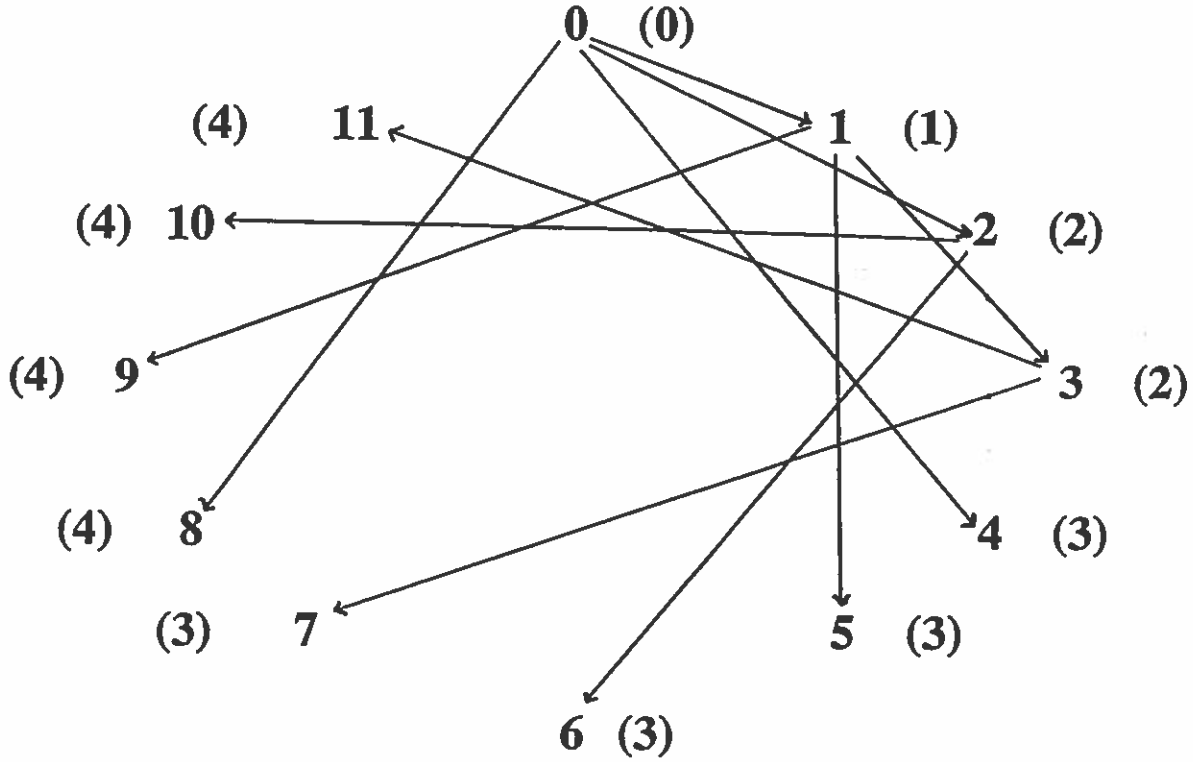

Figure 2. Broadcast network operation indicated by arrows for calls, with the time each site becomes informed in parentheses.

sites numbered x and y to be x-y when x is greater than or equal to y and to (x+n)-y otherwise; conceptually, **dist(x,y)** is the clockwise distance between y and x when the graph is placed on the plane as a star polygon having sites numbered in increasing clockwise order around a circle (as in Figure 1). We define **time(x,y)** to be the least integer t such that 2^t is greater than or equal to **dist(x,y)** + 1. We let T be the maximum value of **time(x,y)**, equal to $\lceil \log_2 n \rceil$.

For our log-star polygon topology, we specify **call-list(x,y)** to be the list of integers determined by the expression $x + 2^i \pmod n$, where i varies from **time(x,y)** to T-1, such that **dist(x,y)** + 2^i is less than n. In our example from Figure 1, n = 12 and T=4; **time(0,0)**=0 and **call-list(0,0)** is 1, 2, 4, 8; **time(1,0)**=1 and **call-list(1,0)** is 3, 5, 9; **time(3,0)**=2 and **call-list(3,0)** is 7,11; and **time(4,0)**=3 but **call-list(4,0)** is empty as its only candidate (i.e., $4 + 2^3$) is equal to n. Figure 2 shows the broadcast process originated by site 0 in our network of 12 sites; arrows show the calls made during the time unit associated with the receiving vertices. Every non-redundant broadcast process traces out a spanning tree of the network topology; we call such trees minimum broadcast trees [6].

Theorem 1: Given a log-star polygon, following the general broadcast process according to the call-lists defined above, each site y becomes informed of a broadcast message originated by x during time unit **time(x,y)**.

Proof: [By induction on **time(x,y)**.] It is obviously true for **time(x,y)**=0, as the originator x knows of the message at time 0. Assume it is true for all sites y such that **time(x,y)** is less than or equal to t. We must show it is true for all sites z such that **time(x,z)**=t+1. All informed sites y call sites $y+2^t \pmod n$ during time unit t+1. As such, all sites at (clockwise) distances less than 2^{t+1} from the originator are informed of the message at the end of t+1 time units. []

This establishes that our log-star polygon graphs are instances of topologies for minimum-time broadcast networks (as previously established in [1]).

Broadcast Immunity to Single Site Failures

What we will establish as new results in this paper are that the log-star polygon graphs described above can be used as topological aspects of minimum-time broadcast networks that are also immune to single site failures or to sets of isolated site failures. We will do this by modifying the operational specification of the networks to realize the desired reliability properties.

If a network is to be immune to failures, sites must know of alternate routings for messages transfers. When we are considering immunity for the broadcast task, a site may need to know more than one alternate call when a failure is encountered, as the failing site may have been responsible for more than one call during a message broadcast from the given originator. However, to minimize the time delay incurred upon encountering a failure and the number of line connections that are needed to realize immunity, we will want to distribute the responsibility for making calls normally made by the failing site among several operational sites.

We will refer to the calls necessary to overcome a failure as the **repair-list** and will append a repair-list field to every message being broadcast. When a site receives a message that contains a non-empty repair-list field (i.e., one not equal to **nil**), it will first call the initial site on the repair-list, forwarding the rest of the repair-list with the message in that call; the site then will complete its broadcast process by calling the sites it normally would call for a broadcast from the given originator. This general immune broadcast process can be described formally as follows:

```
process immune-broadcast (x, m):
;       broadcast process for site x, upon receiving message m;
;        where m is of the form <originator, content, repair-list>.
;       call-list(x, y) is as defined above for minimum time broadcast.

 if repair-list is nil
    then
       do-call-list (call-list(x, originator), m)
    else
       send (first(repair-list),
                  <originator, content, rest(repair-list)>)
       do-call-list (call-list(x, originator), m)
```

What a site does when encountering a failure during broadcast (i.e., during execution of the do-call-list process) will differ depending upon the class of failures for which immunity is to be realized and upon the topology of the network (i.e., what lines and sites are available to overcome the failure). If we consider immunity to single site failures, 2-connected graphs represent a sufficient class of topologies. In a 2-connected graph, there exists two vertex-disjoint paths between every pair of vertices; thus, since there is an alternative path between any originator and every other vertex, broadcast can always be completed when only a single site failure occurs, given an adequate operational

specification. However, graphs that are minimally 2-connected (e.g., single cycles) would most likely encounter significant delays in the broadcasting process if a failure were to occur, as rerouting the broadcast message may require use of long forwarding sequences.

We would like to characterize a class of minimum-time broadcast networks that are immune to single site failures and which incur minimal delay (actually, zero time units) when completing broadcast in the presence of a single failed site. It turns out that our log-star polygon topology is sufficient, given an appropriate specification of the do-call-list process. To understand how a single site failure can be overcome, see Figure 3 where we present the lines contained in the initial part of a normal broadcast tree below an arbitrary site (here numbered zero) as solid and show other relevant lines that exist in the network as dotted. If a descendant site of site **0** has failed, subsequent descendants of **0** are connected to corresponding descendants of the failed site and thus can call the failed site's descendants. Due to our definition of the call-lists as given above, there will always be as many subsequent descendants of site **0** as there are descendants of the failed site. We complete the formal definition of the operational aspect of the immune networks in the following:

```
process do-call-list (clist, m):
;    normal broadcast process after initiating or receiving message m,
;    calling each site in c-list (its call-list for the given originator);
;    if encountering a failure, an appropriate repair process is initiated.
;      call-list(x, y) is as defined above for minimum-time broadcast.

  until clist is nil do
     if (site-state(first(clist)) is down
        then
              do-site-repair (rest(clist), call-list(first(clist)), m)
        else  ; regular call, no failure encountered
              send (first(clist), m)
              set clist to rest(clist)


process do-site-repair (clist, rlist, m):
;    each element on clist is given one of the elements on rlist to call;
;    this allows repair without time delay as each site on clist receives
;    m one time unit earlier than in a normal broadcast.
```

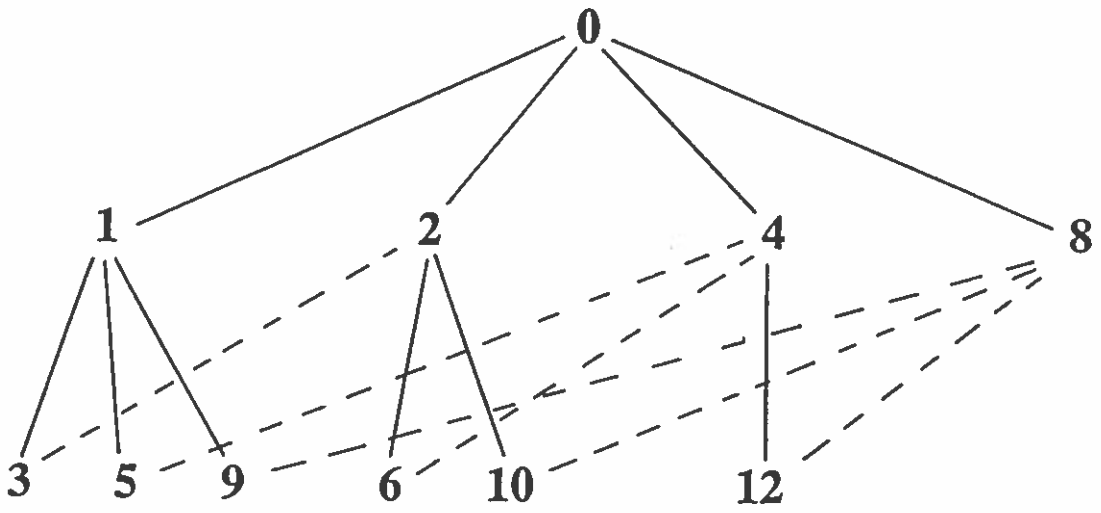

Figure 3. Broadcast tree with other lines of log-star polygons that are used to realize immunity to single site failures.

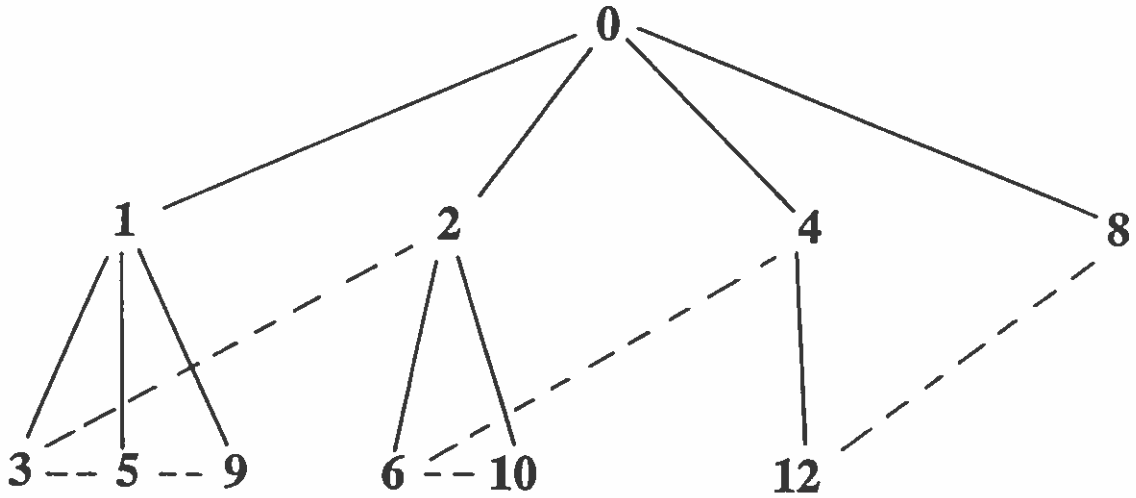

Figure 4. Broadcast tree with other lines of log-star polygons that are used to realize immunity to isolated site failures.

```
until clist is nil
    if rlist is nil
        then    ;no more repair necessary
                send (first(clist), m)
                set clist to rest(clist)
        else
                send (first(clist),
                        <originator, content, [first(rlist)]>)
                set clist to rest(clist)
                set rlist to rest(rlist)
```

Theorem 2: Given a log-star polygon topology, a communication network with the processes immune-broadcast, do-call-list, and do-site-repair as its operational aspect is a minimum-time broadcast network that is immune to single site failures and incurs no delay in repair.

Proof: The network is a minimum-time broadcast network when no failures occur, as its default operation is equivalent to the minimum-time broadcast networks defined earlier. When a failure occurs, we know that **rest(clist)** has the same number of sites as, or one more site than, **call-list(first(clist))**, due to our prior definition of **call-list(x, y)**. Also, the corresponding ith elements of these two lists are connected by a line according to our definition of the log-star polygon topology. Thus, process do-site-repair performs correctly and the network is immune to single site failures. Finally, as each element of **rest(clist)** recieves message **m** one time unit early and calls the corresponding element of **call-list(first(clist))** when that site normally would receive **m**, there is no delay in the completion of the broadcast process. []

Liestman [5] presents alternative operational specifications for our log-star polygon topology that realize immunity to one and two failures (either site or line) with delays of one and two time units, respectively.

Broadcast Immunity to Isolated Site Failures

Immunity to single failures is adequate for networks with highly reliable components. We have previously studied immunity to certain sets of multiple failures which we have called isolated failures [2,4]. A set of site failures is an *isolated set of site failures* if no two

sites that have failed are adjacent to each other (i.e., directly connected by a line of the network). When site failures are isolated, there is a guarantee that other sites in the "neighborhood" around any failure have not failed. The separation between pairs of failures will allow us to define calling procedures that repair the problem by local rerouting of calls.

In the case of immunity to isolated failures for the task of message transfer, each site needed to know only a preferred and an alternate call for each site as recipient. A message transfer process whereby a site first called its preferred neighbor, followed by its alternate neighbor upon failure of the preferred call, was sufficient to guarantee completion of a message transfer (when the recipient was operating) and thus to provide immunity to isolated failures in the network. The class of graphs known as 2-trees has been shown to be the class of topologies having the fewest lines for a given number of sites that could be made immune to isolated site failures [3].

Because we are assuming that multiple site failures are isolated, we can guarantee that all sites involved in the calling of repair-list entries are operating if we rely upon neighbors of the failing site for the repair. When making its normal calls, a site must check to see whether the site it is about to call has failed; if this is the case, the calling site then can initiate repair calls to be made by neighbors of the failed site according to an appropriate repair-list, being the call-list of the failing site for the given originator. Again, to understand the repair process, consider the enhanced broadcast tree shown in Figure 4; we assume, without loss of generality, that the failure occurs at a descendant of **0**. This time we send the complete repair-list to the descendant of **0** that would follow the failed site in the broadcast. That site will initially call the first site normally called by the failed site, with the rest of the repair list appended. Each descendant of the failed site in the broadcast tree will first call the next site that the failed site would have called, before turning to its own call-list. Thus the descendants of the failed site call each other with the broadcast message before forwarding the message to their descendants in the broadcast tree.

Below we specify the operational aspect of a class broadcast networks immune to isolated site failures based upon the log-star polygon topology we have defined above. We assume use of the general immune-broadcast process defined in the previous section, but redefine the do-call-list process to provide immunity to isolated site failures. Our new definition is as follows:

process do-call-list' (**clist, m**):
; normal broadcast process after reception of message **m**
; calling each site in **c-list** (its call-list for the given originator);
; when encountering failures it initiates an appropriate repair-list.

```
until clist equals nil do
    if site-state(first(clist)) is up
        then    ; regular call, no failure encountered
                send (first(clist), <originator, content, nil>)
                set clist to rest(clist)
        else
            if rest(clist) is not nil
                then
                    send (first(rest(clist)),
                            <originator, content,
                                call-list(first(clist), originator)>)
                            set clist to rest(rest(clist))
```

Theorem 3: Given a log-star polygon topology and using the processes immune-broadcast and do-call-list' as operational specification, we have a minimum-time broadcast network that is immune to isolated site failures and that incurs a delay of at most one time unit per failure.

Proof: The default process is as defined for the original minimum-time broadcast networks, so we have a minimum-time broadcast network. We have immunity to isolated site failures as each site involved in the repair process is adjacent to the failed site and thus must be operating when failures are isolated. Our log-star polygon topology provides the necessary line connections among the sites, as illustrated by Figure 4. If there is no next descendant of **0** in the broadcast tree (i.e., **rest(clist)** is **nil**), then no repair is necessary, as the failed site can have no descendants in the broadcast tree according to our prior definition of **call-list(x, y)**. Finally, a delay of one time unit is incurred when repair is necessary, as the descendants of the failed site, though receiving the broadcast message at the time that they normally would, are forced to first make a repair call before turning to their own call responsibilities. []

Conclusion

In this paper we have presented designs for minimum-time broadcast networks that are immune to single site failures and to isolated sets of site failures. We demonstrate the importance of separating topological and operational aspects of network design, as only operational aspects were changed to realize the improved reliability.

References

[1] Farley, A.M. "Minimal broadcast networks", *Networks*, 9 (1979), p313-332.

[2] Farley, A.M. "Networks immune to isolated failures", *Networks*, 11 (1980), p255-268.

[3] Farley, A.M. and Proskurowski, A. "Extremal graphs with no disconnecting matching", *Proc. 14th Southeastern Conference on Combinatorics, Graph Theory and Computing, Conressus Numerantium*, 41 (1984), p153-165.

[4] Farley, A.M. and Proskurowski, A. "Networks immune to isolated line failures", *Networks*, 12, (1982), p393-403.

[5] Liestman, A. "Reliable broadcast networks", *Networks*, 15, 1985.

[6] Proskurowski, A., "Minimum broadcast trees", IEEE Transactions on Computers, (1981), p363-366.