TR-87-12

The Design and Simulation of a Mobile Radio Network with Distributed Control

by

Dennis J. Baker
Anthony Ephremides
Julia A. Flynn

The Design and Simulation of a Mobile Radio Network with Distributed Control

DENNIS J. BAKER, ANTHONY EPHREMIDES, FELLOW, IEEE, AND JULIA A. FLYNN

Abstract—**A new architecture for mobile radio networks, called the linked cluster architecture, is described, and methods for implementing this architecture using distributed control techniques are presented. We illustrate how fully distributed control methods can be combined with hierarchical control to create a network that is robust with respect to both node loss and connectivity changes. Two distributed algorithms are presented that deal with the formation and linkage of clusters and the activation of the network links. To study the performance of our network structuring algorithms, a simulation model was developed. The use of Simula to construct software simulation tools is illustrated. Simulation results are shown for the example of a high frequency (HF) intratask force (ITF) communication network.**

INTRODUCTION

A great deal of interest is focused lately on mobile radio networks that employ packet switching techniques in order to share a common channel efficiently. Indeed several such systems are currently under investigation, including the packet radio network (PRNET) [12], Ptarmigan [16], the joint tactical information distribution network (JTIDS) [20], the battlefield information distribution system (BID) [17], and the high frequency (HF) intratask force (ITF) network [21]. Of great importance in the design of these and other mobile networks are the selection of an appropriate network control architecture and the development of techniques to implement the chosen architecture. Basically, the control architecture specifies the role of each node in performing network control functions, such as routing for example, and it provides a framework in which various networking issues can be studied.

There are several control architectures that can be considered for such networks. From the points of view of decision making and availability of information, we may distinguish four classes of architectures [17], namely

1) centralized,
2) hierarchical centralized,
3) fully distributed, and
4) hierarchical distributed.

An early version of PRNET [12] was based upon a centralized control architecture. The network consisted of three types of nodes: a station, relays, and terminals. All routing paths were assigned by the station, which was the central controller for this network. Relays merely extended

Manuscript received January 15, 1983; revised August 2, 1983. This paper was presented in part at the International Conference on Communications, June 1982, at INFOCOM '83, April 1983, and at the ORSA/TIMS Conference, Orlando, FL, November 1983.

D. J. Baker and J. A. Flynn are with the Naval Research Laboratory, Washington, DC 20375.

A. Ephremides is with the Naval Research Laboratory, Washington, DC 20375, and the Department of Electrical Engineering, University of Maryland, College Park, MD 20742.

the communication range of the station by relaying packets. Terminals acted as sources and sinks of packets. In a hierarchical centralized architecture there can be multiple levels of "centralized" control. A node belonging to one level acts as a central controller for nodes of the next lower layer. Both the centralized and hierarchical centralized architectures are vulnerable to the loss of the central controlling node. For this reason distributed control strategies are often preferred for networks that assign a high degree of significance to network survivability.

One example of a fully distributed control architecture is the network described in [11]. In this architecture, network control is shared equally among all nodes—no node controls any other node—but all nodes must cooperate to effect overall network control. Since fully distributed control techniques tend to require considerable communication overhead to implement, care must be taken when this approach is used in networks where bandwidth is scarce or connectivities are rapidly varying. In [11] a fully distributed control scheme is possible because global paths are provided for only a small subset of the possible node pairs.

The use of a hierarchical distributed architecture can lessen the communication overhead problem associated with fully distributed control. An example of this architecture is the BID network [17] in which a two-level hierarchy is used, consisting of control nodes and data terminal equipments. When a user needs to communicate with another node to which it is not directly connected, a packet is sent to the local control node which routes the packet towards its destination. The routing between control nodes is done in a fully distributed manner. An advantage to the hierarchical approach is that fully distributed routing is done over only a subset of the original nodes. In the BID network, each control node is directly linked to at least two other control nodes and every user is homed to at least two control nodes. If the network is sparse, these connectivity requirements can result in a network with a large number of control nodes. In this paper we will present a variant of the hierarchical distributed control architecture, called the linked cluster architecture, that requires fewer control nodes than does the BID scheme. Having fewer control nodes may reduce the overhead traffic and improve the responsiveness of the network to changing traffic conditions. The linked cluster algorithm has many other features that distinguish it from the BID network, which we will examine later in this paper. It is in fact the architecture proposed for the high frequency (HF) intratask force (ITF) network [21].

The purpose of this paper is threefold: first, to acquaint the reader with the concept of the linked cluster architecture, second, to present distributed algorithms for implementing this architecture, and third, to introduce new software simulation tools for simulations of distributed controlled networks. In the remainder of the paper, these objectives are pursued in this same order. Prior to a summary of our conclusions, we also present simulation results illustrating the use of our distributed control algorithms.

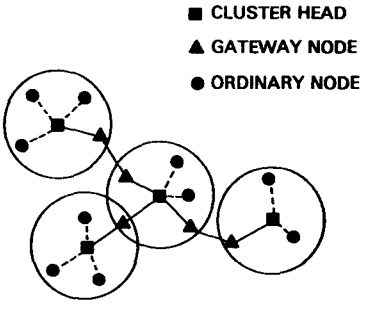


Fig. 1. Example of "linked cluster" organization of a mobile radio network. As shown, a node may be either an ordinary node (solid dot), a cluster head (square), or a gateway node (triangle). Cluster heads may act as local central controllers. The example is for the case of a fixed communication range, as indicated by the circle around each cluster head. Clusters are linked together by gateways, as needed, to form a backbone network for intercluster communication.

The Linked Cluster Architecture

Description of Linked Cluster Structure

The linked cluster architecture consists of node clusters that are linked to each other as shown in Fig. 1. Nodes can assume one of three roles; a node may be a cluster head, a gateway, or an ordinary node. The determination of each node's role is achieved in a fully distributed fashion, as will be described later. Each node has its own cluster head and is said to belong to the cluster of nodes that share this same head. In addition, some nodes may be within communication range of more than one head. The cluster head may act as a local controller for all the nodes belonging to its cluster. Cluster heads are linked via gateways, as needed, to provide overall network connectivity. The cluster heads and gateway nodes and the links that interconnect them form the "backbone network."

An architecture for mobile radio networks should provide

1) communication paths between every pair of nodes,
2) convenient structures for network-wide broadcasts,
3) structures that can avoid the "hidden-terminal" problem associated with multiple access in multihop environments [18], and
4) structures that are robust with respect to node losses and connectivity changes.

The linked cluster architecture achieves all these objectives, as will become apparent later on.

The basic philosophy behind the linked cluster architecture is to invest some communication overhead to develop a linked cluster structure to facilitate routing and channel access control while retaining sufficient distributed control to permit local routing around defective or otherwise undesirable parts of this structure.

The HF ITF Network

The HF ITF network, now under development, is intended to be a general purpose network providing beyond-the-horizon voice and data communication for naval task force units. The ships, submarines, and aircraft that con-

stitute the nodes of this network may be distributed over an area up to 300 mi in diameter and will be linked by radio waves from the HF band (2–30 MHz). Although the task force moves as a group, the positions of the individual platforms with respect to each other can be highly variable.

The network structure shown in Fig. 1 is not, by itself, adequate for the HF ITF network since it is based on a single connectivity map for the network, while over the entire HF band there may be several different connectivity maps due to variations in the HF communication range with frequency. Consequently, the ITF net will likely consist of several overlaid sets of linked clusters, each set being similar to the one shown in Fig. 1 and based on a particular connectivity map. Moreover, these connectivity maps are continually being reformed in order to adapt to the time variation of the HF ITF network connectivities. The HF band is partitioned for this reason into M subbands, for each of which a separate configuration of clusters is created. These separate configurations are formed consecutively during M epochs as shown in Fig. 2. During epoch "i" a network is formed for the ith subband of the HF channel. A connectivity map is formed based on the connectivities that exist within that subband. When the M epochs are completed, the epochs repeat in a cyclic fashion providing a continual updating process. During any epoch only one set of linked clusters is being reorganized—the remaining $M-1$ sets are unaffected. To prevent disruptions in communication traffic flow, the network should route traffic to avoid the subband in which the network is being reorganized. Appropriate message framing provisions must be made in order to avoid interruption of message transmissions at the beginning of the corresponding reorganization epochs.

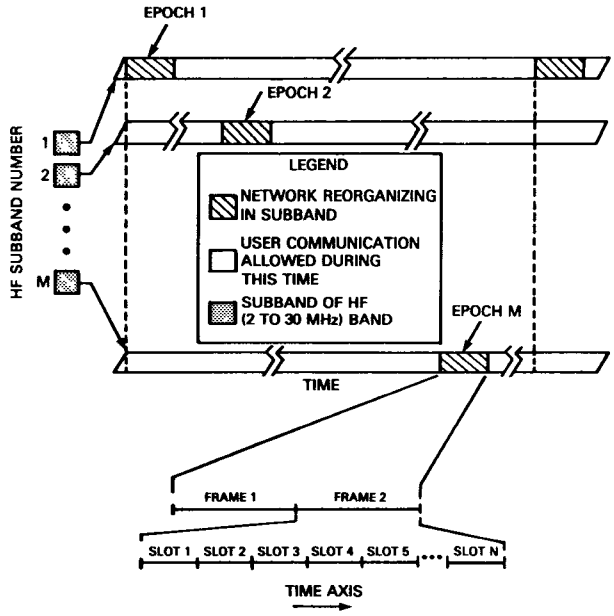


Fig. 2. Timing for network reorganization.

Implementing the Linked Cluster Architecture

An implementation of the linked cluster architecture must address five major tasks:

1) topology sensing,
2) cluster formation,
3) cluster linkage,
4) link activation, and
5) routing.

We seek to implement the linked cluster architecture using distributed control techniques, thus avoiding the vulnerability of relying on a central controller. Consequently, our implementation is based upon distributed algorithms that carry out these five tasks. To this end three algorithms are being developed:

1) the linked cluster algorithm (LCA),
2) the link activation algorithm (LAA), and
3) the routing algorithm.

The linked cluster algorithm performs the first three tasks while the link activation and routing algorithms handle tasks 4 and 5, respectively. Only the first two algorithms are discussed in the present paper.

Topology Sensing—Each node can discover who its neighbors are by the process of probing. When a probe message is broadcast, every node that hears it sends an acknowledgment back to the probing node. A simple strategy for effecting such a probing by every node in a network is to set up two TDMA frames for controlling the transmission of probe and acknowledgment messages. In this approach, each node is assigned its own transmission slot for use in frames 1 and 2 (see Fig. 2). During frame 1 each node broadcasts its probe message and, at the same time, acknowledges the receipt of previously transmitted probe messages that it has heard. During frame 2 each node broadcasts acknowledgments for probe messages that it has received since its own frame 1 transmission. By the time of its frame 2 transmission, each node knows all of the nodes to which it is bidirectionally linked.

Cluster Formation and Linkage—Cluster formation consists of selecting cluster heads and assigning cluster members. The idea of our approach is best illustrated as follows. Suppose that the network topology is known to a single decision maker and the communication range is fixed. The first cluster head is selected arbitrarily (node A in our example of Fig. 3). All nodes that are two-way connected to A are assigned to cluster A. (We follow the convention of identifying a cluster by its head.) Draw a circle, of radius equal to the communication range, around node A; then every node within this circle is a member of cluster A. From the remaining nodes, which do not belong to cluster A, we select the second cluster head—say node B is chosen. Proceeding as before, draw a circle around B; all nodes within this circle that are not already members of a cluster become members of cluster B. The process repeats until every node is a member of a cluster. This procedure produces more widely separated clusters—no cluster heads are directly linked—and a more uniform distribution of clusters than was originally proposed [1].

Once cluster heads have been determined, the designation of gateway nodes becomes a trivial task, provided all

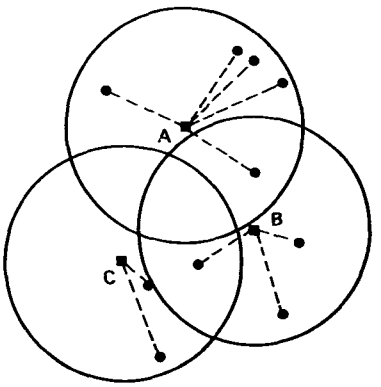


Fig. 3. Example of cluster formation and cluster head selection for case of a fixed communication range.

connectivities are known and all clusters identified. Only two cases need be considered, namely, overlapping clusters and adjacent nonoverlapping clusters. The two cases are illustrated in Fig. 4. Adjacent nonoverlapping clusters are characterized by the existence of at least one pair of nodes —one node from each cluster—that are bidirectionally linked to each other. Candidate nodes for linking together heads that have overlapping clusters consist of those nodes that are bidirectionally connected to both heads—in the case of our example, the nodes in the intersection of the two circles in Fig. 4(a). Any one selection from this list of candidate gateway nodes is sufficient to effect the desired linkage. Nonoverlapping clusters require the selection of a pair of gateways—one node connected to each cluster [see Fig. 4(b)]. Again, if network connectivities are known and all clusters are known, it is straightforward to identify the candidate pairs of gateway nodes and to select a preferred pair.

The task of cluster formation and linkage as described is implemented in a distributed way as follows. (For a fully detailed account of the algorithm see [3].) Each node will now possess only *partial* information about the network. Furthermore, some message exchanges are needed to build up even this limited database.

The procedure for cluster formation, which was just described, comprises four key elements:

1) knowledge of the neighbors of each node (i.e., to whom is each node bidirectionally connected),

2) a rule for selecting a cluster head from a set of candidate nodes,

3) knowledge of the sequence in which clusters are to be formed, and

4) knowledge of each node's own cluster head (i.e., to which cluster does each node belong).

The probing procedure, which we previously described, can be used by a node to learn of its neighbors. We associate an order to the sequence in which cluster heads are formed by assigning the TDMA transmission slots, shown in Fig. 2, according to a node's ranking; thus, the node having rank 1 transmits in slot 1, the node of rank 2 transmits in slot 2, etc. In addition, we require that each node make the determination as to whether it should become a cluster head just prior to its frame 2 transmission. The rule for selecting a cluster head is that a node becomes a cluster head if it is not already bidirectionally linked to a cluster head.[1] Furthermore, each node is required to announce its own cluster head during its frame 2 transmission; this allows each node to learn of the existence of all cluster heads that are one hop away and some that are two hops away. Each node selects as its own cluster head the lowest numbered (highest ranked) head to which the node is bidirectionally connected.

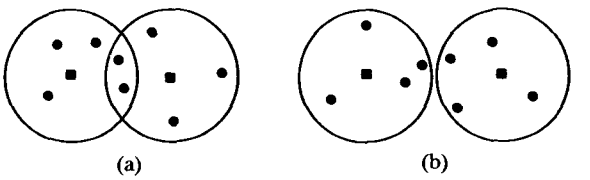


Fig. 4. Examples of the two cases that arise when linking clusters. (a) Overlapping clusters. (b) Adjacent, nonoverlapping clusters.

Two problems arise when attempting to perform cluster linkage with distributed control; they are the identification (recognition) of clusters that are to be linked, and the identification of candidate gateways to effect the linkage. Consider the problem of candidate gateway identification for the example of overlapping clusters shown in Fig. 4. Here, in order that nodes in the intersection of the two clusters know what nodes are candidate gateways, it is sufficient that each node (and therefore each cluster head) also announce, as part of its frame 2 transmission, the list of neighbors to which the node is bidirectionally connected. If nodes are provided with this limited amount of additional connectivity information, the resulting selection of a gateway can be made unambiguously.

To interconnect two adjacent, nonoverlapping clusters, one node from each cluster must become a gateway. There may be several pairs of potential gateway nodes that can perform this linkage. Each node may be "aware" of several of those, but perhaps not all of them. The (arbitrary) deterministic rule chosen for resolving the ambiguity is to select the pair with the lowest sum of identity numbers. In case of a tie, the pair involving the node with the lowest number is chosen. Unfortunately, in the case of adjacent nonoverlapping clusters, our selection procedure (see [3]) does not always result in an unambiguous gateway pair. This ambiguity arises because each node has only *partial* information concerning network connectivity. However, the creation of superfluous gateways is only a nuisance—their existence has little effect on network operation, and, if desired, they can easily be detected and relegated to nongateway status.

Link Activation—After each node has determined what role it should assume, that is, whether it should act as cluster head, gateway, or ordinary node, the next question is, "How are the links of the backbone network activated,

[1]An earlier version of the LCA, which uses a different method for selecting cluster heads, appears in [1]. Comparisons between the version given here and the version in [1] are detailed in [3]. (*Note:* The version described in the present paper is the same as the one referred to as the alternative linked cluster algorithm (ALCA) in [3].)

and how are the links between cluster heads and cluster members set up?" Assuming that cluster reformation occurs periodically, as shown in Fig. 2, then what we are considering here are the rules governing the activation of links during the periods between cluster reformations.

The choice of method for activating links depends upon the number of channels and upon the way channels are implemented and used in the network, where here a channel refers to a frequency–time signaling plan. For example, are there multiple, orthogonal (or nearly orthogonal) channels or must users share a common channel? In the latter case the problem reduces to that of selecting a suitable protocol for channel access. Another important consideration is the number of channels that a node can receive on or transmit on simultaneously.

In this section we describe a conceptual approach to link activation as it might be applied to a network using frequency hopping (FH) code division multiple access (CDMA) spread-spectrum signaling with each node having only a single receiver and a single transmitter. Thus, we assume that there are a large number of quasi-orthogonal channels available for network use, but each node is assumed to be limited to transmitting and receiving on at most one channel (code) at a time.

Link activation requires that the transmitter, at one end of a link, operate on the same FH code pattern as the receiver at the other end of the link. We assume that a unique FH code is associated with each receiver and each transmitter and that these code assignments are known to every node in the network. Additional codes, which are described in [21], may also be used in networks that are based upon the linked cluster architecture, but we do not consider them here. When a node transmits, it may use its own transmitter code, the code of the destination receiver, or some other code. Similarly, a receiver may listen on its own FH code or on any other code in use in the network. Since many codes may be in use simultaneously, some coordination (scheduling) of code usage is needed. The link activation algorithm is intended to perform this coordination without the need for a central controller.

A simple technique for controlling link activation in an FH-CDMA system is to use a single (shared) code channel and a fixed time division multiple access (TDMA) scheme for channel access control. In fixed TDMA, time is divided into frames, which are further divided into slots. If there are N nodes in the network, then there are N slots per frame. Each node transmits during its own slot and "listens" to the channel during the other slots. Unfortunately, in a sparsely connected network, which is often the case in mobile radio networks, receivers that are out of range of the owner of a slot remain idle during that slot, even though there may be nodes within range of these receivers that have data to send to them. The link activation makes use of the multichannel capabilities of CDMA signaling to avoid this sort of inefficiency.

The basic idea of the link activation method described here is as follows. Each node constructs its own TDMA schedule (frame) for establishing communication with its

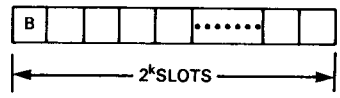


Fig. 5. TDMA structure of link activation schedule. The first slot is reserved for transmissions to cluster heads and for receiving broadcasts from the node's own cluster head; it is referred to as the cluster head's broadcast slot and is denoted by the letter B in schedules.

neighbors. These schedules, which are repeated many times during the period between successive epochs (see Fig. 2), have the following features (see Fig. 5).

1) The number of slots in the schedule is an integer power of 2 (deliberately constrained to be so for implementation purposes) and need not be the same for every node.

2) A slot is assigned for listening to each node that can be heard.

3) A slot is assigned for transmitting to each neighboring node (transmission and reception can occur in the same slot).

4) A special, broadcast slot is assigned for receiving from one's own cluster head and for transmitting to any neighboring head.

Essentially, each node attempts to make an arbitrary assignment of slots to its neighbors and, thus, form a TDMA schedule. Obviously, each neighbor will attempt to do likewise, but as it may have a different number of neighbors than the original node, and as these neighbors set up their own schedules also, inconsistencies will, in general, arise that must be resolved in order to arrive at consistent, conflict-free assignments. The systematic way in which these schedules are set up and the conflicts resolved is provided by the LAA.

In conjunction with the link activation schedules, we assume that the following code usage applies. During broadcast slots, cluster heads transmit using their own transmitter codes, and all other nodes "listen" on the transmitter code associated with their own cluster head. For point-to-point (nonbroadcast) slots, nodes transmit using their own transmitter code and receive on the transmitter code of the node transmitting to them.

We now give a brief description of the link activation algorithm; pseudocode and additional details are given in [4].

The LAA consists of two activities, namely, allocating slots and resolving scheduling conflicts. Slot allocating occurs during frame 1 and conflict resolving occurs in frame 2. These frames are the ones corresponding to the LCA algorithm and shown in Fig. 2. Broadcasts of information relevant to the LAA take place during the same slots that information relating to the LCA is sent. Thus, the LAA rides on the LCA in that it shares the same transmission plan during its communication phase.

At each node, say, for example, node i, the LAA maintains three lists: a list of nodes heard from during frame 1, a list of nodes to which node i is bidirectionally connected, and a list of the link activation schedules. The latter includes node i's own schedule as well as schedules received from neighboring nodes. At the start of frame 1, the list of

schedules contains only node i's own schedule, which initially is only one slot in length—the first slot is always a broadcast slot. The other lists are initially empty.

A node makes its first allocation of slots just prior to transmitting in frame 1. At that time, beginning with the lowest numbered node, a slot is allocated for each node that has been heard from thus far in frame 1. This preliminary schedule is then broadcast. As schedules are received, they are stored in the list of schedules. A node is allocated the first (earliest) available slot, where a slot is available if its allocation does not result in a known scheduling conflict.

After a node has transmitted in frame 1, it continues to update its own schedule as it hears from other, higher numbered nodes during the remainder of this frame. If the link to a higher numbered node is found to be unidirectional (indicated by the absence of the receiving node's identification number in the received schedule), then upon receiving the higher numbered node's frame 1 transmission, the receiving node is free to allocate the first available slot for listening to that node; otherwise the schedule received is examined to see when communication with the higher numbered node should occur, and the receiving node updates its own schedule to reflect that allocation. By the end of frame 1 each node has a slot allocated for communicating with each neighbor.

At the end of frame 1, each node has its own version of the link activation schedule, which may have scheduling conflicts (shared slots). These schedules describe only one activation per link; however, we intend that links should be activated periodically during the time between network reorganizations in the corresponding subband (see Fig. 2). Consequently, the purpose of frame 2 of the LAA is to periodically extend the schedules formed during frame 1 and, at the same time, to resolve any scheduling conflicts that arise. In the LAA, the key to simplifying the periodic extension of the link activation schedules is to constrain these schedules to lengths that are an integer power of 2.

The procedures for managing the link activation schedules require the use of several integer variables indicating different schedule sizes, which we now describe.

Whenever a schedule is received, received_schedule_size refers to the number of slots in this schedule. Schedules transmitted during frame 1 of the LAA can be of any length, but those transmitted during frame 2 are constrained to lengths that are an integer power of 2. In the latter case, the constraint length may have been achieved by appending several unallocated slots to the original schedule.

Own_schedule_size refers to the number of slots currently in a node's own schedule.

During execution of the algorithm, each node keeps track of the maximum schedule size known to it. Maximum _schedule_size equals a node's own_schedule_size or the size of the largest schedule received from a neighbor, whichever is largest. Maximum_schedule_size is monotonically increasing during the two frames of the LAA. Initially, maximum_schedule_size equals 1, and at the start of frame 2 it is rounded upwards to the nearest integer power of 2 (e.g., 8, 16, 32, etc.). Every time a schedule is received or whenever a node transmits or modifies its own schedule, maximum_schedule_size is updated.

Just prior to a node's frame 2 transmission, the node must resolve all shared-slot conflicts. The initial value of minimum_schedule_size is the (constrained) length of the schedule immediately following conflict resolution, which is also equal to the size of the schedule transmitted by that node during frame 2. Each node keeps track of the minimum_schedule_size (i.e., the transmitted schedule sizes) of all of its neighbors. The final value of the maximum _schedule_size is equal to the maximum of all of the minimum_schedule_sizes known to a node.

Beginning with the earliest shared slot, all scheduling conflicts (shared slots) are resolved just prior to each node's second transmission of the LAA. For a given shared slot, the procedure for resolving a scheduling conflict is as follows. Starting with the highest numbered of the nodes that share a slot, the algorithm seeks to allocate different slots to these nodes until the original shared slot has only one remaining node allocated to it. In testing each node to see whether it can be moved to a new slot, the algorithm searches for a new allocation that will not create conflicts in any of the schedules stored in the list of schedules. Here, in order to keep the schedules compact, the maximum number of slots searched is equal to N, if N is an integer power of 2, or to the number nearest N that is both greater than N and is an integer power of 2 (e.g., if $N=10$, then the maximum number of slots searched is $2^4=16$). If, following this approach, nodes still share the slot, then one-by-one the algorithm moves nodes of the shared slot to unallocated slots until only one node remains in the original shared slot. Thus, each node is responsible for resolving conflicts in its own schedule.

Should any scheduling conflicts arise in a node's own schedule after it has transmitted in frame 2 (we have seen this happen only once in our simulations), these conflicts are resolved simply by sequencing, during successive repetitions of the basic schedule, through the list of nodes sharing a slot The affected nodes can be notified of this action without the need for additional communication frames of the LAA.

At the end of frame 2, all schedules that are less than the maximum_schedule_size are expanded, and each node updates its own schedule consistent with the expanded schedules of its neighbors.

Finally, as a result of applying the procedures just described, completely conflict-free schedules are obtained.

Simulation Tools

Digital computer simulation is being used to model the HF ITF network and its environment and to evaluate the various distributed control algorithms that are candidates for implementing the linked cluster architecture. This section describes some of the software tools that we use to simplify the construction of the network simulator.

The HF ITF network simulator is written in the language Simula. Simula is a general-purpose, high-level language comparable in power to PL/I or Algol 68. It is particularly suited to the development of application packages. That is, the basic constructs of Simula are easily extended (see, for example, [6], [19]) to provide powerful software tools designed for specific classes of applications. These application packages are said to form a "context" in which specific problems can be solved [7]. One particular context that is part of the Simula system is called SIMULATION. The HF ITF network simulator uses the facilities provided by SIMULATION and extends the capabilities of this context by providing some tools to simplify the simulation of distributed algorithms. A brief tutorial on the use of SIMULATION is included in [8].

In Simula, only PROCESS objects (i.e. objects that are a subclass of CLASS PROCESS) can be manipulated by the process scheduler. Moreover, a PROCESS can have but one event-notice (activation time) pending. The latter restriction can lead to difficulties when modeling some complex processes [14]. A more generalized process-object, which has the appearance of allowing the scheduling of multiple activation times, has been developed by C. Landwehr [13]; he calls this new class evprocess (i.e., the event process). Originally, the event process concept was used to simplify the simulation of protocols that involved message acknowledgments within some timeout period; we find the event process can also be used to simplify the simulation of distributed algorithms by providing a template for coding these algorithms. In our simulation model both the linked cluster and link activation algorithms are coded as event processes. Our version of the event process, which we describe below, is a modified version of Landwehr's evprocess and also bears resemblance to event handling constructs described in [10].

The new event handling facility is made up of three new class objects, namely, classes event_msg, evprocess, and event. The complete source code for these three classes appears in Fig. 6.

Classes that are prefixed by evprocess should have the format shown in Fig. 7, which is nearly identical to the format for classes prefixed by FUNCTIONBLOCK that appear in [9]. The basic idea is that whenever an event process is activated, it will test to see which event has occurred and then perform the actions appropriate for that event. To simplify the reading of our simulation code, events are given text descriptors. For example, the LCA and LAA respond to several events, including start_of_frame, start_of_transmission_slot, and message_received. When an evprocess is activated, execution should begin immediately after the passivate statement (see Fig. 7). Thus, the programmer should avoid, for example, the use of the HOLD statement within the body of classes prefixed by evprocess. To schedule an event, the code shown in Fig. 8 can be used. The event process that is to receive the event is indicated by the parameter evprocess_name, the name of the event is passed as the event_descriptor, and the time the event is to occur is denoted by event_time. The HF intratask force network simulator makes extensive use of the event process facility because it imposes a structuring that makes the simulation program easy to understand and maintain.

```
CLASS event_msg;
!Use this class as a prefix for messages to be delivered by the;
!  event facility.                                              ;
BEGIN
  !No declarations or statements;
END of event_msg;

Process CLASS evprocess;
!This class provides a prefix to be used by processes that will;
!  use the event facility for scheduling.                       ;
BEGIN
  TEXT evtype;  !Event descriptor;
  REF(event_msg) evmsg; !Pointer to the message that goes with ;
  !  this event.                                               ;
END of evprocess;

Process CLASS event(proc,etype,emsg);
!Each instance of this class is an event;
VALUE etype;
REF(evprocess) proc;    !The process to be reactivated by this ;
!                         event.                               ;
TEXT etype;     !The event descriptor;
REF(event_msg) emsg;   !The message that accompanies this event;
BEGIN
  proc.evtype:-etype;   !Indicate what event has occured;
  proc.evmsg:-emsg;     !Set pointer to  corresponding message;
  REACTIVATE proc;      !Wake up the recipient;
END of event;
```

Fig. 6. Simula code for the classes making up the new event facility.

```
evprocess CLASS class_name;
BEGIN
  !Insert class declarations here, as needed;
  !Insert initializations here, if any are needed;
  WHILE TRUE DO
  BEGIN
    Passivate;
    IF evtype="event1_descriptor" THEN
    BEGIN ... !Statements to be executed when event1 occurs; ... END
    ELSE IF evtype="event2_descriptor" THEN
    BEGIN ... !Statements to be executed when event2 occurs; ... END
      .
      .
      .
    ELSE IF evtype="eventN_descriptor" THEN
    BEGIN ... !Statements to be executed when eventN occurs; ... END
    ELSE ...; !Print "Unrecognized event";
  END;
END of class_name;
```

Fig. 7. Classes prefixed by "evprocess" should have the format shown above.

```
Activate NEW event(evprocess_name,"event_descriptor",NONE)
                        AT event_time;
```

Fig. 8. Example of code for scheduling events.

Simulation Examples

In this section we present simulation examples illustrating the use of the network control algorithms previously described. Here, emphasis is on how the LCA and LAA restructure a network in response to changing topologies rather than on the generation of realistic task force scenarios or detailed HF channel models. Consequently, the results shown in this section are based upon a simple HF channel model and randomly positioned (within a circle of radius 300 km) nodes.

For the examples shown in this section, we use a frequency versus HF groundwave range model that is based upon the polynomial

$$a_0 + a_1 f + a_2 f^2 + a_3 f^3 + a_4 f^4.$$

This polynomial gives the communication range in kilome-

TABLE I
POLYNOMIAL COEFFICIENTS FOR COMMUNICATION RANGE MODEL

Frequency Range [MHz]	a_0	a_1	a_2	a_3	a_4
2 to 10	75.83	219.512	-55.4377	5.449301	-.192016
10 to 20	216.27	33.623	-4.8438	0.224165	-.003497
20 to 30	33.60	34.198	-2.3805	0.064103	-.000583

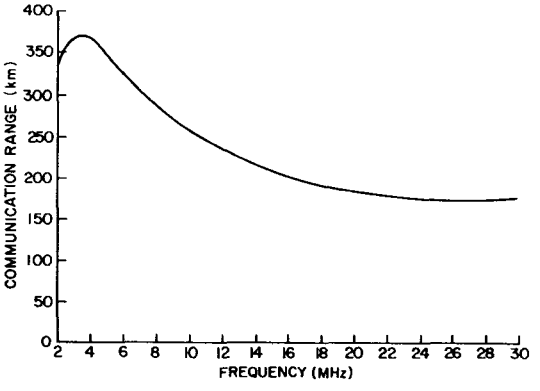


Fig. 9. Groundwave communication range model.

ters as a function of frequency (in megahertz) for the set of constant values given in Table I. A plot of the resultant range versus frequency curve is shown in Fig. 9. The curve of Fig. 9 is meant to be only a representative model. Actual ranges will vary with geographic location, sea state, time of day, season, other user interference, etc. An HF groundwave range model for a specific signaling scheme, which also takes into account the effects of jamming and sea state [5], is described in [2].

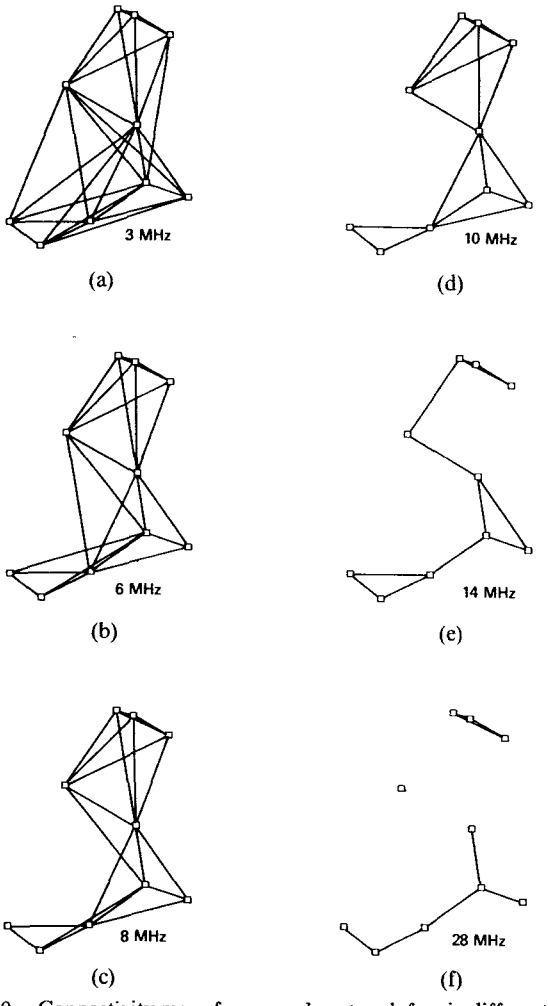


Fig. 10. Connectivity maps for example network for six different frequencies.

Frequency Dependence

As a first example, we consider the results from applying the linked cluster and link activation algorithms at different frequencies in the HF band. Fig. 10 shows the connectivity maps for a sample ten-node network for six frequencies. Based upon these connectivity maps, the LCA produced the network structures shown in Fig. 11, and the LAA produced the link activation schedules shown in Fig. 12. Note that connected networks (i.e., every node is within range of a cluster head and all cluster heads are linked via the backbone network) are formed at all frequencies for which the corresponding connectivity map is connected. The technique of overlaying several sets of linked clusters provides alternative communication paths; this technique is used in the HF ITF network to decrease network vulnerability to the loss of links or nodes. If a backbone network link is lost at one frequency due to jamming, other backbone networks at other frequencies can be used. When the net is reorganized in the subband in which the jamming occurs, a new backbone network will be set up that will not contain the jammed link. The combination of path redundancy and the ability of the network to adapt to node and link losses results in a robust ITF net.

The schedules shown in Fig. 12 illustrate the ability of the LAA to generate schedules that vary in length depending upon the degree of connectivity. Thus, for highly connected networks such as the one shown in Figs. 10(a) and 11(a), schedules of lengths 8 and 16 are produced, whereas for the sparsely connected network of Figs. 10(f) and 11(f), schedule lengths are 4 or less. Moreover, in all but a few cases the period between activations of the same link is less (and in many cases significantly less) than would be obtained with fixed TDMA scheduling.

To illustrate some of the conflicts that can occur during the formation of the link activation schedules, we show in Fig. 13 the partially formed schedules as they are at the end of LAA frame 1 for the example network shown in Figs. 10(d) and 11(d). Because the network is not fully connected, each node will have only a subset of these schedules in its list of schedules.

Each node has its own schedule, which specifies the slots assigned for communicating with particular neighboring nodes. For example, during the fifth slot, node 3 can communicate with node 8. Since the link (3,8) is bidirectional, node 8 will have a 3 in its fifth slot, indicating that it can communicate with node 3 during slot 5. During a broadcast slot, indicated by the letter *B* in the schedules, each node listens for broadcasts from its own cluster head and also is able to transmit to any neighboring head. The

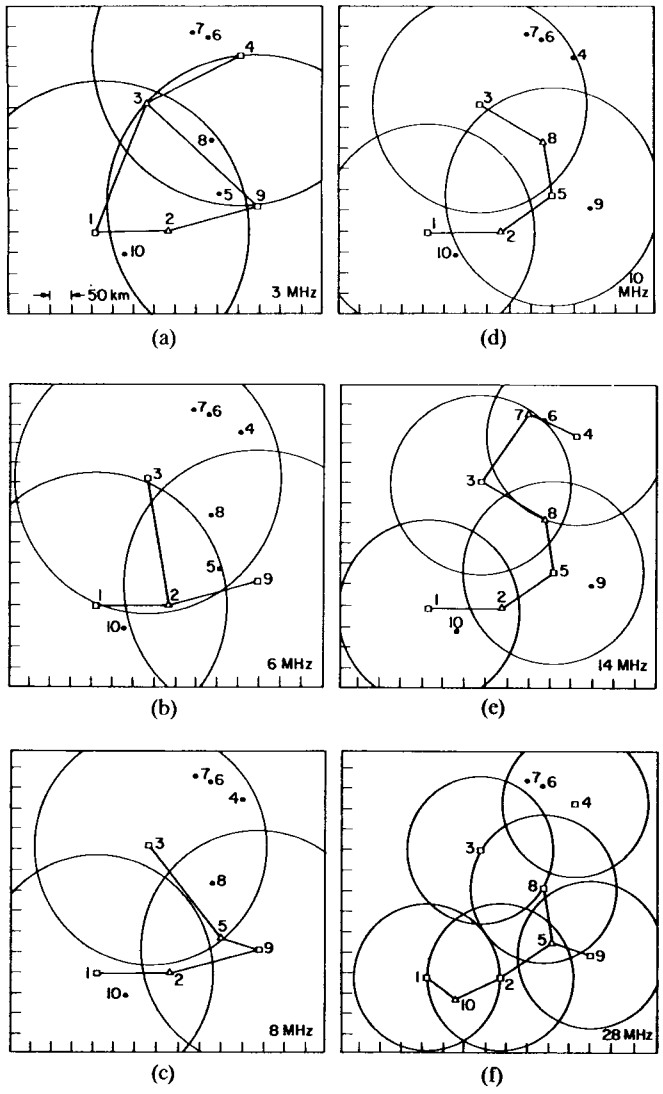


Fig. 11. Network structures produced by the linked cluster algorithm and based upon the connectivity maps shown in Fig. 10.

latter transmissions are, however, subject to interference from other nodes also wishing to access this same head.

Because of the unequal lengths of the schedules shown in Fig. 13, periodically repeating them would result in scheduling conflicts. For example, the periodic expansion of node 3's schedule (B,4,6,7,8, B,4,6,7,8,$\cdots$) and the periodic expansion of node 5's schedule (B, 8, 2, 9, B, 8, 2, 9, B, 8$\cdots$) lead to a scheduling conflict for node 8 during slot 10. The LAA seeks to avoid such conflicts.

The schedules that are formed at the end of frame 1 of the LAA also contain scheduling conflicts that arise due to the network not being fully connected. For example, since nodes 7 and 8 are out of range of each other, each is unaware that the other has chosen slot 3 for communicating with node 4. However, during frame 2 of the LAA, this as well as other conflicts are resolved, and we obtain the link activation schedules shown in Fig. 12(d). Defining "allocation efficiency" as the percentage of slots that are allocated over a period equal to the duration of the longest schedule, our example has an allocation efficiency of 71 percent.

Final Schedules - 3 MHz

NODE SLOT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	B	2	3	5	8	-	10	-								
2	B	1	5	3	9	8	-	-	10	1	5	-	-	-	-	-
3	B	4	1	2	5	7	8	9	6	4	1	-	5	7	-	-
4	B	3	6	7	-	5	-	8								
5	B	8	2	1	3	4	9	10	B	-	2	1	3	4	9	10
6	B	7	4	8	-	-	-	-	3	7	4	-	-	-	-	-
7	B	6	8	4	-	3	-	-								
8	B	5	7	6	1	2	3	4	9	10	7	-	1	-	-	4
9	B	-	-	-	2	10	5	3	8	-	-	-	-	-	5	-
10	B	-	-	-	-	9	1	5	2	8	-	-	-	-	1	5

(a)

Final Schedules - 6 MHz

NODE SLOT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	B	2	5	10	B	-	-	10								
2	B	1	3	5	8	10	9	-								
3	B	4	2	6	7	8	5	-								
4	B	3	6	7	-	-	8	-								
5	B	10	1	2	9	-	3	8								
6	B	7	4	3	-	-	-	-	8	7	4	3	-	-	-	-
7	B	6	8	4	3	-	-	-								
8	B	9	7	-	2	3	4	5	6	9	7	-	2	3	4	5
9	B	8	-	-	5	-	2	-								
10	B	5	-	1	-	2	-	1								

(b)

Final Schedules - 8 MHz

NODE SLOT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	B	2	10	-	B	-	10	-								
2	B	1	5	8	9	10	-	-								
3	B	4	6	7	8	5	-	-								
4	B	3	7	6	-	8	-	-								
5	B	-	2	9	10	3	8	-								
6	B	7	3	4	-	-	-	8								
7	B	6	4	3	-	-	-	-								
8	B	9	-	2	3	4	5	6	7	9	-	2	3	4	5	6
9	B	8	-	5	2	-	-	-								
10	B	-	1	-	5	2	1	-								

(c)

Final Schedules - 10 MHz

NODE SLOT	1	2	3	4	5	6	7	8
1	B	2	10	-	B	-	10	-
2	B	1	5	8	9	10	5	-
3	B	4	6	7	8	-	-	7
4	B	3	7	6	-	-	8	-
5	B	8	2	9	B	-	2	9
6	B	7	3	4	-	8	-	-
7	B	6	4	3	B	-	-	3
8	B	5	9	2	3	6	4	-
9	B	-	8	5	2	-	-	5
10	B	-	1	-	-	2	1	-

(d)

Final Schedules - 14 MHz

NODE SLOT	1	2	3	4	5	6	7	8
1	B	2	10	-				
2	B	1	5	10				
3	B	7	8	-				
4	B	6	7	-				
5	B	9	2	8				
6	B	4	-	7				
7	B	3	4	6				
8	B	-	3	5	9	-	3	5
9	B	5	-	-	8	5	-	-
10	B	-	1	2				

(e)

Final Schedules - 28 MHz

NODE SLOT	1	2	3	4
1	B	10		
2	B	5	10	-
3	B			
4	B	6	7	-
5	B	2	8	9
6	B	4	-	7
7	B	-	4	6
8	B	-	5	-
9	B	-	-	5
10	B	1	2	1

(f)

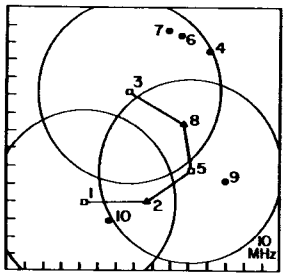
Fig. 12. Link activation schedules for the networks shown in Fig. 10. Each schedule is repeated many times between epochs.

Node Renumbering

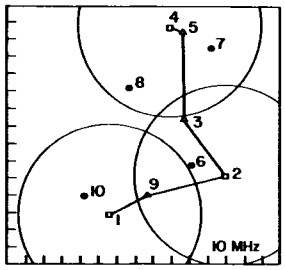
Given the simple strategy used in deciding the identity of a cluster head or a gateway node among a group of candidates, it is clear that number assignment to the nodes is a very important part of the proposed organization. For

NODE SLOT	1	2	3	4	5	6	7	8
1	B	2	10					
2	B	1	5	8,10	9			
3	B	4	6	7	8			
4	B	3	7,8	6				
5	B	8	2	9				
6	B	7	3	4	-	8		
7	B	6	4	3				
8	B	5	4	2	3	6	9	
9	B	-	-	5	2	-	8	
10	B	-	1	2				

Fig. 13. Example of partially formed schedule at the end of frame 1. These results are for the network shown in Figs. 10(d) and 11(d).



(a)



(b)

Fig. 14. Example of node renumbering. (a) Structure obtained with original numbering. (b) Structure obtained with inverted numbering.

example, the lower numbered nodes simply have a greater tendency to become heads or acquire gateway status than higher numbered nodes. If the numbering system is the same for all epochs, these nodes will be overburdened with network management and traffic direction responsibilities.

One way to avoid having the same nodes always becoming cluster heads and gateways is to assign to each node a different number, and hence a different transmission slot, for each epoch. Also, if all nodes do not have comparable equipment, favored numbers should be assigned to nodes with greater capability. A simple strategy that tends to produce disjoint backbone networks, i.e., having no nodes or links in common, is to alternate on successive epochs between number randomization and number inversion. By number inversion we mean that high numbers become low numbers and vice versa, e.g., nodes 1, 2, 3, etc., become nodes N, $N-1$, $N-2$, etc. An illustration of the results of such a strategy on network structure is shown in Fig. 14, and the corresponding link activation schedules are shown in Fig. 15. The results show some separation of the backbone networks; however, nodes 2 and 8 [Fig. 14(a)] still appear in each of the backbone networks—they appear as nodes 9 and 3 in Fig. 14(b). That this situation is unavoidable for the particular example chosen can be seen by considering the complete connectivity map for this case,

Final Schedules - 10 MHz

NODE SLOT	1	2	3	4	5	6	7	8
1	B	2	10	-	B	-	10	-
2	B	1	5	8	9	10	5	-
3	B	4	6	7	8	-	-	7
4	B	3	7	6	-	-	8	-
5	B	8	2	9	B	-	2	9
6	B	7	3	4	-	8	-	-
7	B	6	4	3	B	-	-	3
8	B	5	9	2	3	6	4	-
9	B	-	8	5	2	-	-	5
10	B	-	1	-	-	2	1	-

(a)

Final Schedules - 10 MHz

NODE SLOT	1	2	3	4	5	6	7	8
1	B	9	10	-	B	-	-	-
2	B	3	6	9	B	-	6	9
3	B	2	5	6	8	9	7	-
4	B	5	7	8	B	-	-	8
5	B	4	3	-	7	8	-	-
6	B	-	2	3	9	-	2	-
7	B	8	4	-	5	-	3	-
8	B	7	-	4	3	5	-	4
9	B	1	-	2	6	3	10	2
10	B	-	1	-	-	-	9	-

(b)

Fig. 15. Link activation schedules for the node renumbering example shown in Fig. 14.

which appears in Fig. 10(d). Since nodes 2 and 8 of Fig. 14(a) are cut set nodes, they must necessarily be part of the backbone network. However, if there are no cut set nodes, the strategy of node number inversion followed by number randomization on alternate epochs should produce well-separated backbone networks. Numerous other strategies are, of course, possible. If, despite frequent node renumbering, a node becomes a cluster head or gateway node in several subbands, then this node is likely to be a critical node in the sense that its loss may disconnect the network. The identification of "critical" nodes can be an important byproduct of running the LCA in conjunction with renumbering.

Adapting to Node Losses

Because the LCA and LAA have built into them tests for network connectivities, node losses can be detected and the linked cluster structure and the link activation schedules can be modified to avoid their use. Examples of this adaptibility are shown in Figs. 16 and 17. The first frame of Fig. 16 shows the resultant backbone network for an initial network of 20 nodes. Each subsequent frame in the figure corresponds to a network obtained by deleting five nodes from the network shown in the preceding frame. Since we want to show how effective the network is at adapting to node losses, we consider an extreme case in which the nodes that are lost are those that are more likely to become cluster heads or gateways. Since the LCA favors the selection of the lower numbered nodes as heads and gateways, the five lowest numbered nodes were deleted in successive frames of Fig. 16.

The results of this simulation are interpreted as follows. The original network of 20 nodes, shown in Fig. 16(a), is provided with a single, connected backbone network. Even with the loss of five key nodes, the network still is able to

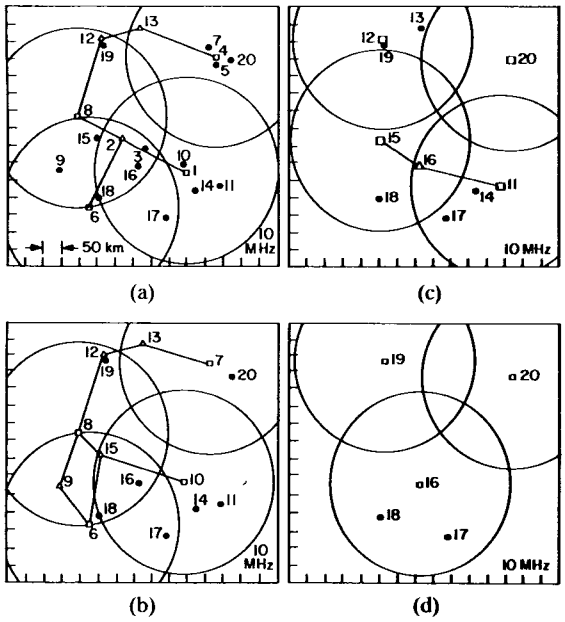


Fig. 16. Example of restructuring to combat node losses.

Final Schedules - No Nodes Lost

NODE SLOT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	B	2	3	10	14	16	17	-	11	-	-	18	-	-	-	-
2	B	1	6	3	9	14	16	18	15	-	8	10	17	-	-	-
3	B	6	1	2	11	9	14	16	-	15	17	8	10	18	-	-
4	B	5	7	13	20	-	-	-								
5	B	4	13	7	-	20	-	-								
6	B	3	2	9	16	18	15	-	-	-	-	-	-	17	-	-
7	B	13	4	5	-	-	20	-								
8	B	-	-	12	19	-	9	-	16	18	2	3	15	-	-	-
9	B	15	16	6	2	3	8	-	18	-	-	-	-	-	-	-
10	B	-	-	1	-	11	-	14	17	16	18	2	3	15	-	-
11	B	-	14	16	3	10	-	17	1	-	-	-	-	-	-	-
12	B	19	-	8	13	-	-	-	B	19	-	-	13	-	-	-
13	B	7	5	4	12	19	-	-								
14	B	16	11	-	1	2	3	10	-	17	-	-	-	-	-	-
15	B	9	18	-	-	-	6	-	2	3	16	-	8	10	-	-
16	B	14	9	11	6	1	2	3	8	10	15	17	18	-	-	-
17	B	18	-	-	-	-	1	11	10	14	3	16	2	6	-	-
18	B	17	15	-	-	6	-	2	9	8	10	1	16	3	-	-
19	B	12	-	-	8	13	-	-	B	12	-	-	-	13	-	-
20	B	-	-	-	4	5	7	-								

(a)

Final Schedules - Nodes 1 to 5 Lost

NODE SLOT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
6	B	9	15	16	18	-	17	-								
7	B	13	20	-												
8	B	12	9	18	16	19	15	-								
9	B	6	8	15	-	16	18	-								
10	B	11	14	17	15	-	16	18								
11	B	10	16	14	17	-	-	-								
12	B	8	13	19	B	-	13	19								
13	B	7	12	-	19	7	12	-								
14	B	16	10	11	-	17	-	-	B	-	10	11	-	17	-	-
15	B	-	6	9	10	18	8	16								
16	B	14	11	6	8	9	10	15	17	18	11	6	8	9	10	15
17	B	-	18	10	11	14	6	-	16	-	-	10	11	14	6	-
18	B	-	17	8	6	15	9	10	-	16	-	8	6	15	9	10
19	B	-	-	12	13	8	-	12								
20	B	-	7	-												

(b)

Final Schedules - Nodes 1 to 10 Lost

NODE SLOT	1	2	3	4	5	6	7	8
11	B	14	16	17				
12	B	13	19	-				
13	B	12	-	19				
14	B	11	17	16				
15	B	16	18	-	B	-	18	-
16	B	15	11	14	17	18	11	14
17	B	18	14	11	16	-	14	11
18	B	17	15	-	-	16	15	-
19	B	-	12	13				
20	B							

(c)

Final Schedules - Nodes 1 to 15 Lost

NODE SLOT	1	2	3	4
16	B	17	18	-
17	B	16	-	18
18	B	-	16	17
19	B			
20	B			

(d)

Fig. 17. Link activation schedules corresponding to the examples shown in Fig. 16.

form a single, connected backbone; see Fig. 16(b). When the network loses nodes 1–5, the roles of heads 1 and 4 are taken over by nodes 10 and 7. The role of gateway node 2, which links clusters 1–8 and 1–6, is assumed by node 15, which links clusters 8–10 and 6–10. The loss of node 2 also results in the creation of the new gateway at 9, which links clusters 6 and 8. The loss of nodes 3 and 5 has no immediate effects.

The additional loss of nodes 6–10 has the effect of disconnecting the backbone network in Fig. 16(c). This is unavoidable since, for example, the loss of node 7 results in the isolation of node 20. Likewise, the set of nodes 12, 13, and 19 are also isolated once nodes 7 and 8 are lost. The cluster head roles of nodes 8 and 10 [see (b)] are taken over by nodes 11 and 15 [see (c)]. The effects of the loss of head 6 are borne by new heads 11 and 15. That is, all the nodes within cluster 6 [see (b)] are now contained within the combination of new clusters 11 and 15 [see (c)]. Also, the disappearance of 6 negates the need for a gateway node at 9. Thus, the gateway role of 9 does not have to be taken over by any other node.

The additional loss of nodes 11–15 causes no further partitioning of the network; it still comprises three isolated parts. However, the roles of heads 11 and 15 and gateway 16 [Fig. 16(c)] are now taken over by the single head at node 16 [Fig. 16(d)].

The link activation schedules corresponding to the examples of Fig. 16 are shown in Fig. 17. In a fixed TDMA system, slots that belong to disabled (lost) nodes go unused; however, such is not the case for schedules produced by the LAA, as is clearly shown in Fig. 17.

Conclusions

We have presented a new architecture for mobile radio networks, called the linked cluster architecture, which can be used to construct survivable networks that provide

1) communication paths between every pair of nodes,
2) convenient structures for network-wide broadcasts,
3) structures that can avoid the "hidden-terminal" problem associated with multiple access in multihop environments, and
4) structures that are robust with respect to node losses and connectivity changes.

Two distributed algorithms are described, which implement cluster formation/linkage and link activation. The first algorithm, the linked cluster algorithm (LCA), enables each node to determine what role it should perform in the network, that is, whether it should become a cluster head, gateway, or ordinary node. The LCA also identifies a backbone network, which connects the cluster heads and provides paths for intercluster traffic. The second algorithm, the link activation algorithm (LAA), is designed for a multichannel code division multiple access (CDMA) signaling scheme and has as its purpose the activation of those links that have been identified as potential links by the LCA. Using the LAA, each node unambiguously and with consistency creates its own TDMA frame for communication with its neighbors. As an illustration of the linked cluster architecture, we have described its applica-

tion to an HF intratask force (ITF) network. Simulation results showing the structuring of the HF ITF network under the control of the LCA and LAA are presented.

REFERENCES

[1] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Trans. Commun.*, vol. COM-29, pp. 1694–1701, Nov. 1981.

[2] D. Baker, J. Wieselthier, A. Ephremides, and D. McGregor, "Distributed network reconfiguration in response to jamming," in *Proc. MILCOM '82*, Oct. 1982, pp. 23.2-1–23.2-7.

[3] D. J. Baker, A. Ephremides, and J. E. Wieselthier, "An architecture for the HF intra-task force (ITF) communication network," Naval Res. Lab., Washington, DC, NRL Rep. 8638, Dec. 22, 1982.

[4] D. J. Baker, A. Ephremides, and J. Flynn, "A distributed algorithm for scheduling the activation of links in the high frequency intratask force (ITF) communication network," Naval Res. Lab., Washington, DC, NRL Rep., 1984.

[5] D. Barrick, "Theory of HF and VHF propagation across the rough sea, 2, Application of HF and VHF propagation above the sea," *Radio Sci.*, vol. 6, pp. 527–533, May 1971.

[6] G. Birtwistle, "Advanced use of SIMULA," in *Proc. Simulation Conf.*, Winter 1981, pp. 293–304.

[7] —, "Introduction to DEMOS," in *Proc. Simulation Conf.*, Winter 1981, pp. 559–572.

[8] B. Eklundh, "SIMULA—A way of thinking," in *Proc. Simulation Conf.*, Winter 1979, pp. 11–20.

[9] —, "Simulation of large and complex systems: Some general trends and an example," in *Proc. Simulation Conf.*, Winter 1979, pp. 145–151.

[10] W. R. Franta, *The Process View of Simulation*. New York: North-Holland, 1977.

[11] A. Hauptschein and M. Kajor, "Recognition and self-organization of nodes into DTDMA nets," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-17, pp. 531–542, July 1981.

[12] R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman, "Advances in packet radio technology," *Proc. IEEE*, vol. 66, pp. 1468–1496, Nov. 1978.

[13] C. E. Landwehr, personal communication, 1978.

[14] —, "SIMULA and events," Naval Res. Lab., Washington, DC, NRL Tech. Memo. 7503-113, Apr. 11, 1978.

[15] —, "An abstract type for statistics collection in SIMULA," *ACM Trans. Programming Lang. Syst.*, vol. 2, pp. 544–563, Oct. 1980.

[16] M. Lawson and S. Smith, "Reconfiguration techniques of a mobile network," in *Proc. Int. Zurich Sem. Digital Commun.*, Zurich, Switzerland, 1980, IEEE 80CH1521-4 COM, pp. B10.1–B10.5.

[17] A. Nilson, W. Chou, and C. J. Graff, "Packet radio communication system architecture in a mixed traffic and dynamic environment," in *Proc. Comput. Networking Symp.*, 1980, pp. 51–66.

[18] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part II—The hidden terminal problem in carrier sense multiple access and the busy tone solution," *IEEE Trans. Commun.*, vol. COM-23, pp. 1417–1433, 1975.

[19] B. W. Unger and D. S. Bidulock, "The design and simulation of a multi-computer network message processor," *Comput. Networks*, vol. 6, pp. 263–277, Sept. 1982.

[20] D. H. Walters, "JTIDS net management model," in *Conf. Rec. EASCON 82*, Sept. 1982, pp. 209–213.

[21] J. E. Wieselthier, D. J. Baker, A. Ephremides, and D. N. McGregor, "Preliminary system concept for an HF intra task force communication network," Naval Res. Lab., Washington, DC, NRL Rep. 8637, Aug. 9, 1983.

Dennis J. Baker was born in Detroit, MI on July 25, 1940. He received the B.S. degree from the University of Detroit, Detroit, in 1963 and the Ph.D. degree from the University of Michigan, Ann Arbor, in 1973, both in physics.

He served as Scientific Consultant to Convair Division, General Dynamics, from 1968 to 1969. Since 1971 he has been with the Naval Research Laboratory, Washington, DC. His research interests include the study of radiation phenomena in ionospheric and magnetospheric plasmas and the design of communication network simulation models. Currently he is engaged in the study of HF communication networks.

Anthony Ephremides (S'68–M'71–SM'77–F'84) was born in Athens, Greece, in 1943. He received the B.S. degree from the National Technical University of Athens, Athens, in 1967, and the MA and Ph.D. degrees from Princeton University, Princeton, NJ, in 1969 and 1971, respectively, all in electrical engineering.

He was a Visiting Professor at the Department of Electrical Engineering and Computer Science, University of California, Berkeley, in 1979. Now he is Professor of Electrical Engineering at the University of Maryland, College Park, where he has been since 1971. He has been a Consultant at the Naval Research Laboratory, Washington, DC, since 1977, and is also the President of Pontos, Inc., a private consulting firm. He has taught several short courses on multiuser information theory and systems under continuing engineering education programs in the United States and abroad. His research interests include communication theory and systems with emphasis on multiaccess problems and on communication networks.

Dr. Ephremides is a member of the Board of Governors of the IEEE Information Theory Group, and was an Associate Editor on Estimation of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL. He was the organizer of the 1983 IEEE Workshop on Multiuser Information Theory and Systems. He is also the Director of the Fairchild Scholars and Doctoral Fellows Program, an academic and research partnership program in satellite communications between Fairchild Industries and the University of Maryland.

Julia A. Flynn was born in San Francisco, CA, on June 25, 1959. She received the B.S. degrees in mathematics and natural science from St. Mary's College of Maryland, St. Mary's City, in 1981.

Currently, she is employed as a Computer Scientist at the Naval Research Laboratory, Washington, DC, where she provides software support for the HF communication network simulation model. Her research interests include computer applications in communication networking and distributed processing.