INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

A Packet Radio Network Design for Field, Mobile Multimedia Communications:

Network Configuration, Radio Channel Access and Network Management

by

François Simard, B.Eng

A thesis presented to the
School of Graduate Studies and Research
of the University of Ottawa in partial fulfillment
of the requirements for the degree of
Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Ottawa, Ontario, Canada
August 1997

Abstract

Current trends in the field of communication systems engineering aim at integrating different communication media into a single, coherent system. In many cases, these communication systems can be implemented because of the availability of a fixed infrastructure capable of supporting the stringent demands of multimedia communications. In some applications, however, a requirement for multimedia communications exists but access to fixed infrastructure is either impractical or unavailable. For these applications, the packet radio network represents an excellent framework for the communication system design.

We are interested in designing a packet radio network which can satisfy the requirements of such ad-hoc, mobile multimedia applications. Specifically, the following broad design issues are addressed in this thesis: determining the network connectivity map and maintaining it as it changes over time; efficiently sharing the common radio channel amongst all users; and managing the overall network such that traffic is reliably routed from source to destination.

Firstly, we identify the linked cluster architecture as a network configuration appropriate for our design. We then consider three different clustering algorithms and select one of them, the lowest ID clustering algorithm, for implementation, based on its structral properties and its robustness under nodal mobility. Secondly, we review the scheduling problem for pure TDMA networks and the two principal approaches, link and broadcast scheduling, which exist to solve it. Of these two approaches, we justify our choice of broadcast scheduling and select a specific broadcast scheduling algorithm, the GRAND algorithm, for incorporation into our network design. Thirdly, we devise a hierarchical routing scheme for the linked cluster architecture. Finally, we integrate these design choices into a concrete, generalized packet radio network which may be customized to fit specific operating environment and user requirements.

Acknowledgments

I wish to express my sincere gratitude to my advisor, Dr. Abbas Yongaçoglu, whose constructive guidance and direction was central to completing this thesis.

Le soutient de mes parents sont, aujourd'hui comme dans le passé, à la source de mes réussites. À vous deux, un grand merci!

Enfin, je tiens à remercier ma conjointe, Marie-Christine, qui me donne, à chaque jour de ma vie, le goût d'aller toujours un peu plus loin ...

Contents

List of Figures

List of Symbols

N	Number of nodes in the network
G	Directed graph of the network
V	The set of all nodes in the network
E	The set of all directed links between neighbors
r	Radio range
d	Distribution range
(u, v)	The directed link going from node u to node v
V_m	The m-th broadcast mode of V
C_u^m	The m-th broadcast zone of node u
E_m	The m-th link activation mode of E
$C_{(u,v)}^m$	The m-th link activation zone of link (u, v)
L_b	Broadcast schedule length
L_l	Link schedule length
s_b	Total number of transmissions in the broadcast schedule
s_l	Total number of transmissions in the link schedule
S_b	Broadcast schedule throughput
S_l	Link schedule throughput
t_s	TDMA slot duration
$D_{i,j}^h$	Optimal route cost of no more than h links from source i to destination j
$d_{i,j}$	Link cost of link (i, j)
b_d	Data channel bandwidth
b_c	Control channel bandwidth
$\left(\frac{E_b}{N_0}\right)$	Energy-per-bit to noise-spectral-density ratio

Y_i^j	Connectivity matrix of node i for the $j-th$ communication mode
γ_j	The power threshold of the j-th communication mode
P_c	Control channel signal power
P_d	Data channel signal power
$P_{i,j}^p$	Node i's probing signal strength as measured at node j
$L_{i,j}$	Propagation loss factor from sending node i to receiving node j
$P_{i,j}^d$	Predicted data signal power at node j from node i
H_i	Node i's clusterhead assignment vector
R_i^k	Node i's role assignment vector for communication mode k
A_i^k	Node i's cluster assignment vector for communication mode k
$GF(q)$	The Galois field of order q
β_i	Element $i+1$ of $GF(q)$
D	Maximum degree of all nodes in V
d_{max}	Maximum D of the GRAND algorithm validity region

Chapter 1

Introduction

Current trends in the field of communication systems aim at integrating different communication media into a single, coherent system capable of reliably transferring information between users. The obvious example of such a system is today's Internet which is capable of simultaneously carrying data, video, and voice, but a multitude of other examples can be found where there exists a requirement for such multimedia communications and where communication systems have been developed to support it.

For a good percentage of cases, these communication systems can be implemented because of the availability of a fixed infrastructure capable of supporting the stringent demands of multimedia communications. In some instances, however, multimedia communications are required but access to the appropriate, fixed infrastructure is either impractical, or simply not available. Examples include the broad field of mobile communications, where, obviously, connection to a fixed infrastructure is not possible, but other interesting cases may be given. These include modern military tactical communication systems for field operations [1]; communication systems for disaster relief and search and rescue operations [2]; and systems for communications in remote locations where little or no infrastructure exists [3], such as the Canadian northern territories. A common theme found in all of these applications is the requirement for reliable multimedia communications, "anytime, anywhere". In almost all cases, this

requirement is complemented by the requirement for mobile operations.

For example, we may consider the general, post-2000 communications architecture proposed in [1] for NATO military forces, as depicted in Figure 1.1. This architecture incorporates a mobile subsystem designed to function as either an independent network or as a component of a larger, tactical military

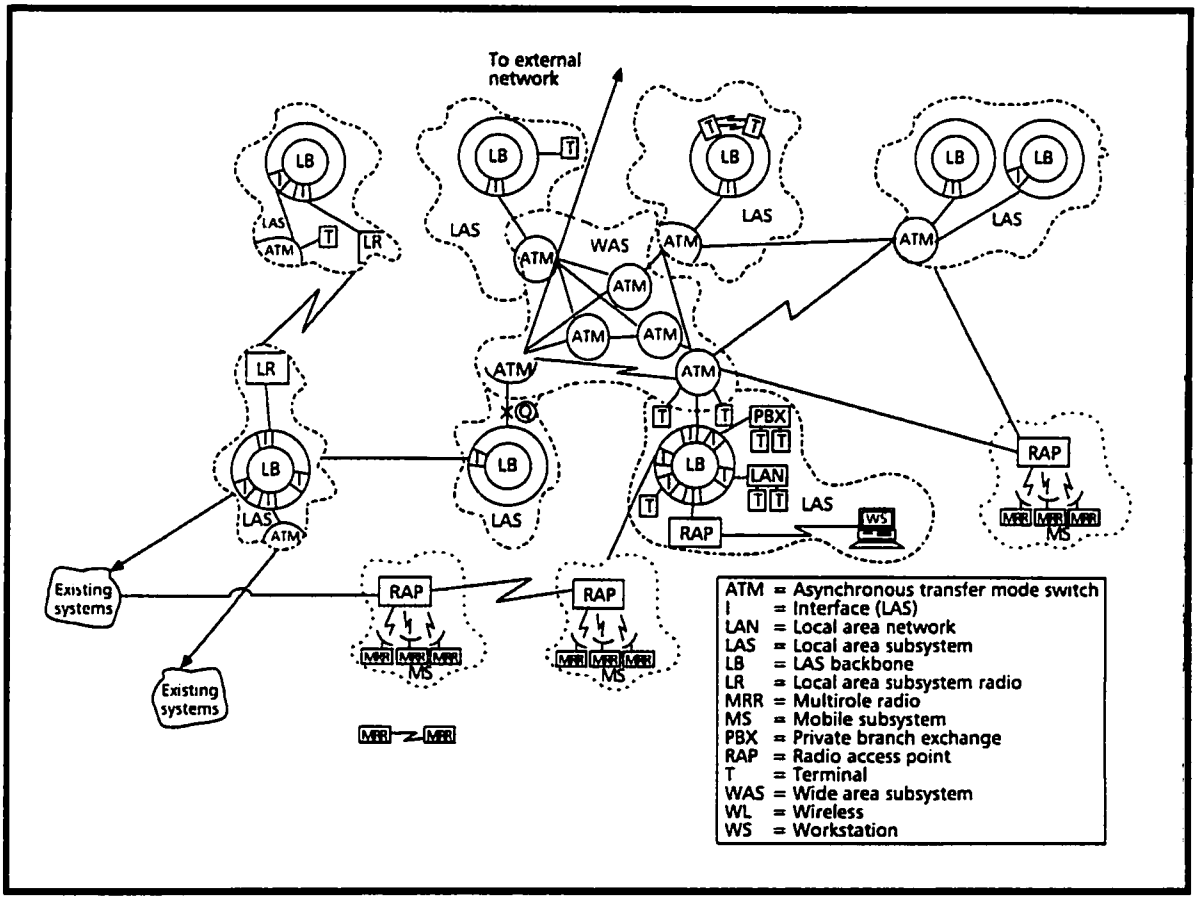


Figure 1.1: NATO post-2000 communications architecture, as reproduced from [1].

communication system. The subsystem supports three modes of operation: combat net radio; mobile telephone; and packet radio. Integrating voice, data, and imagery user services into a single communication system, the radios of the mobile system are envisaged to be compact, digital packet radios with multifunction and multimode capabilities. Although this specific example is given for illustration purposes only, it is fair to say that the shape of the mobile subsystem, and the objectives it is intended to meet, are typical of the general requirements of field, multimedia communications.

The Networking Problem and Thesis Outline

We are interested in designing a communication system which can satisfy the requirements of such ad-hoc, mobile, multimedia applications. To this effect, packet radio networks represent an excellent framework for our system design. These packet radio networks have been and are being designed for a number of varying environments including ground mobile radio; amateur radio; naval HF communications; and satellite communications [4]. The design of these networks share a set of common design issues which are closely related to those characteristics which differentiate packet radio networks from other types of networks: use by all *nodes* [1] of a common transmission medium; and a dynamic topology resulting from both nodal mobility and the typically noisy and fading radio channel. Thus, the important issues in designing an effective packet radio network can be broadly categorized as follows:

- Determining the network connectivity map and maintaining this map as it changes over time;
- Efficiently sharing the common radio channel amongst all users;
- Achieving reliable communications between nodes in the presence of noise and fading; and
- Managing and controlling the overall network such that traffic is reliably routed from source to destination.

[1] A node is simply a communicating entity in the network — a radio — which may or may not have been assigned special tasks to perform.

Finding solutions to these broad problems implies making a multitude of design choices, each dependent on a variety of parameters such as the operating environment of the network, performance requirements, complexity, and associated costs. Furthermore, as technology evolves, the variables governing the design choices will similarly change. Thus, no single design represents the "correct" design. For this reason, our objective is not to wholly specify a network design, down to specifications for each possible variable, but rather to develop a general and flexible framework which may be customized to fit the specific environment, requirements, and constraints of any given user.

The broad issues listed so far can be related to the seven-layer OSI Reference Model. In terms of this model, determining connectivity is a function of the physical layer (layer 1). Similarly, radio channel access and use is a function of the data link layer (layer 2), while overall network management and control are functions of the network and transport layers (layers 3 and 4). It is with this model in mind that we now consider each of the elements of packet radio network design. In doing so, we will identify those specific issues which have been addressed in this thesis. Thus, we will define the elements of our design in relation to the general, complete problem of packet radio networking.

For packet radio networks, the physical layer establishes radio connections between communicating nodes. We say that a link exists between nodes u and v if u can transfer information to v, or vice versa, within some specified performance measures. The problem of defining the physical layer of a packet radio network essentially translates into specifying the following parameters [4]: the radio frequency and required bandwidth; the signaling, encoding, and modulation schemes; and the network topology. Here, network topology refers principally to the number of nodes constituting the network; their locations and connectivities; and their roles with respect to network control.

Chapter 2 of this thesis focuses on the problem of defining a network topology from a set of randomly distributed nodes. Working from a set of initial constraints and design objectives, we identify the *linked cluster architecture* as a network structuring scheme appropriate for our design. We further consider three different clustering algorithms which, when applied, yield three different cluster structures. By simulating each of these algorithms and measuring the robustness

of each resulting structure under different types of nodal mobility, a specific algorithm is selected for incorporation into our network design.

Determining access to the physical medium and using it to achieve reliable communications between adjacent nodes are two important functions of the data link layer. With respect to accessing the common radio channel, two general approaches can be taken: fixed-assignment multiple access; and random multiple access. Fixed-assignment multiple access refers to the three broad types of deterministic multiple access schemes: frequency division multiple access (FDMA); time division multiple access (TDMA); and code division multiple access (CDMA). Similarly, random access schemes can be broadly categorized into two types: ALOHA, and carrier sense multiple access (CSMA) types [4, 5, 6]. This categorization describes each multiple access scheme in its elementary form. Oftentimes, real multiple access schemes are a combination of two or more of these basic elements of multiple access.

With respect to the problem of achieving reliable communications between adjacent nodes, two principal issues arise. The first is determining how to best combine and implement forward error control coding and automatic repeat request schemes so as to achieve desired performance criteria. The second issue is determining how to implement node-to-node acknowledgments [4]. Here, two alternatives are possible. One possibility is an active acknowledgment scheme where explicit acknowledgments are transmitted by the receiving node to the transmitting node on successful reception of each packet. The second possibility is a passive acknowledgment scheme where, because of the broadcast nature of the radio channel, the forwarding of a packet by some intended receiver serves as an acknowledgment to the original sending node of successful reception.

In Chapter 3, some of these issues are reviewed and analyzed. Specifically, the chapter begins by briefly reviewing the different types of multiple access schemes and justifying the choice of TDMA for our network design. With this selection in hand, we review the TDMA scheduling problem and justify our preference for broadcast scheduling over link scheduling based on schedule comparisons in terms of general properties, published delay characteristics, and simulated throughput results. Finally, a choice of a specific broadcast scheduling algorithm is made for incorporation into our network design.

The basic function of the network and transport layers is to allow data packets to be reliably and efficiently routed throughout the overall network. This entails two basic tasks: establishing routes from a given connectivity map of the network; and then forwarding the packets along those routes by using effective flow and congestion control mechanisms [4]. In Chapter 4, we quickly review the broad types of routing schemes applicable to our packet radio network and then discuss in more detail two specific routing schemes which have been found to be particularly relevant: vector-length routing and hierarchical routing. We close this chapter by defining the framework of a hierarchical routing scheme customized to fit the characteristics of linked cluster networks.

Finally, the elements presented in chapters 2 to 4 inclusively are integrated into a concrete packet radio network design in Chapter 5. Specifically, this chapter focuses on specifying the structure of the network's control and data channel and practical implementation of the essential network configuration algorithm and the radio channel access scheme. By the end of Chapter 5, then, we will have developed a solid framework for packet radio network design based on the choices justified in the following three chapters.

Chapter 2

Network Configuration

2.1 The Linked Cluster Architecture

The first challenge in designing a packet radio network for field multimedia communications consists in constructing a network configuration from a collection of randomly distributed nodes. Specifically, starting only with the knowledge that a given number of nodes is distributed within some geographic area, we are seeking to structure the nodes under the following initial constraints:

- No a priori connectivity map is assumed except for the general assumption that an appreciable number of nodes can be connected to form a network;
- No node is preassigned a specific role for the purposes of network control; and
- No specific node has a priori knowledge of the locations of the other nodes nor of its connectivity to any one of them.

Working from these initial constraints, we establish broad design objectives which must be met in the context of field multimedia communications. In

particular, the packet radio network configuration must be designed such that the following general design objectives can be attained:

- Communication paths must exist between any two nodes with the exception of nodes which cannot be connected to the network because of their location;
- The structure must support network-wide broadcasts when so required;
- The structure must avoid the hidden terminal problem associated with multiple access radio networks[1];
- The network configuration must be robust with respect to node additions and losses, node mobility, and changes in connectivity;
- The network must be capable of carrying multimedia traffic including data and real-time voice and video;
- The network must be distributed.

The linked cluster architecture was proposed in [7, 8] as a design solution, consistent with these objectives and constraints, for a U.S. Navy high frequency intra task force packet radio network. Though naval operations differ widely from land, field operations, the objectives and constraints that one must meet in designing a packet radio network for either environment match closely. Given this concordance of objectives and constraints, the linked cluster architecture can serve as the basis of a packet radio network for field operations [2], and we choose to retain it as an architecture appropriate for our network design.

The linked cluster architecture simply consists of a collection of clusters, each containing a mutually exclusive subset of all nodes, which are interconnected in some fashion. Each node assumes one of three possible roles: *ordinary node*, *gateway*, or *clusterhead* [7, 8]. Each cluster is constituted of a single clusterhead and its associated ordinary nodes and gateways. It is envisaged that the

[1]The hidden terminal problem occurs when two transmitters, which are not within each other's radio range, wish to transmit to some common receiver. Becase each transmitter cannot hear the other's transmission, simultaneous transmissions may occur, resulting in a collision at the receiver.

clusterhead acts a local coordinator for the purposes of network control and that the gateways serve to bridge communications between clusters. A node may be *connected*[2] to more than a single clusterhead, but for such cases, this node must select a clusterhead to which it will be associated. Similarly, a node may not be connected to any other node. In this case, the node is excluded from the network until such time as a connection can be established.

We consider the linked cluster network of Figure 2.1. This network is composed of three clusters, at the center of which are three respective clusterheads. Each of the clusterheads controls a subset of all the nodes. For example, the bottom cluster is composed of three ordinary nodes and one gateway. The ordinary nodes are simply communicating elements of the network and perform no special network management functions except for the obvious case where they are responsible, like all nodes, for packet store-and-forward functions. The gateway is a node which has as a neighbor some node which belongs to another cluster. It can thus bridge traffic between clusters. For instance, the bottom cluster contains a single gateway through which traffic intended for the top-left cluster may be routed. Of special interest, however, is the gateway between the two top clusters. In this case, we see that the clusters overlap. Though this gateway falls within the clusters of both clusterheads, we associate it to a single cluster for the purposes of network management. Thus, we say that this gateway belongs to the top-left cluster but falls inside the two clusters.

The linked cluster architecture is somewhat reminiscent of the current cellular radio architecture but differs from it in at least two very important ways. Firstly, the linked cluster architecture allows for connections to be established between all nodes, in contrast with cellular systems where mobile users must necessarily go through the basestation to communicate. In terms of our previous example, we see that, for any given cluster, some connections exist between ordinary nodes in addition to each node being connected to its clusterhead. Secondly, the linked cluster architecture is a dynamic network structure defined from a set of peer nodes, in comparison with cellular networks where the network structure is fixed in space and nodes are not equal, i.e. the basestation is hierarchically

[2]A node is said to be connected to another node if two-way communications between the nodes can be established without intermediaries. The nodes are then said to be *neighbors*.

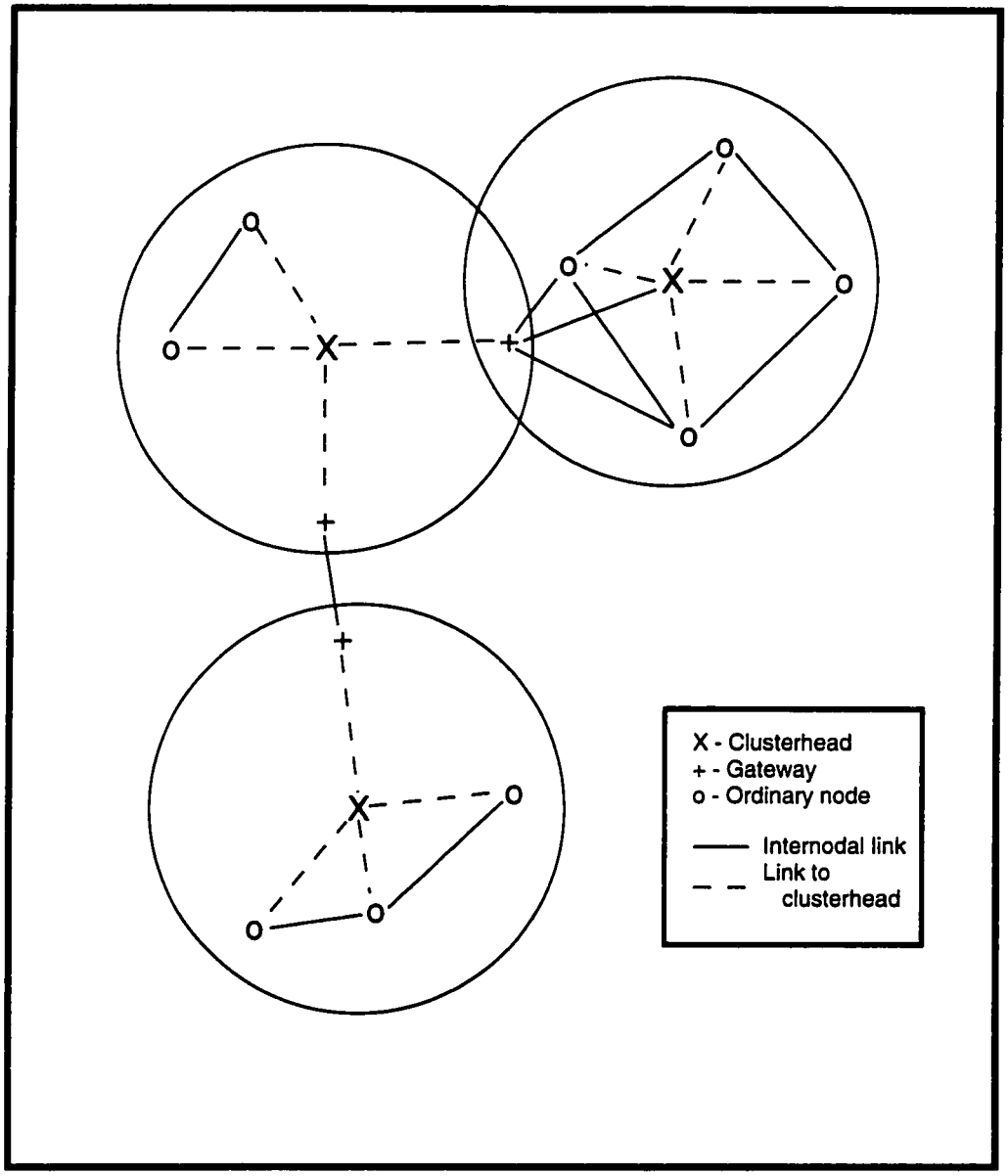


Figure 2.1: The linked cluster architecture.

superior to the mobile radio. As we shall now see, linked cluster networks are constructed by applying some clustering algorithm to a randomly distributed collection of peer nodes. Thus, though some nodes are assigned special roles to play, no node is hierarchically superior.

2.2 Clustering Algorithms

The general structure above and depicted in Figure 2.1 can be achieved through at least three distributed construction algorithms which, when applied, yield different role and cluster assignments. We are interested in identifying which of these four algorithms, if any, yields a more robust configuration under nodal mobility. To begin, we review each of the centralized versions of the algorithms. In all cases, we begin our description with the basic assumption that the **N** nodes are numbered from 1 to N.

2.2.1 The Highest ID Clustering Algorithm

The centralized version of the highest ID clustering algorithm is depicted in Figure 2.2 and is described in [9]. The algorithm starts with the highest numbered node and draws a circle around it of a radius equal to the communication range. If the communications range is not constant in all directions, an appropriate contour may be drawn instead. The highest numbered node becomes a clusterhead, and all nodes that fall inside its circle form the first cluster. A circle is then tentatively drawn around node (N-1). This node becomes a clusterhead if at least one of the nodes that fall within its circle is not already associated with a clusterhead, and all unassociated nodes are assigned to this new cluster. Finally, this procedure is repeated for nodes (N-2), (N-3), etc. until all **N** nodes have been assigned a role and a cluster.

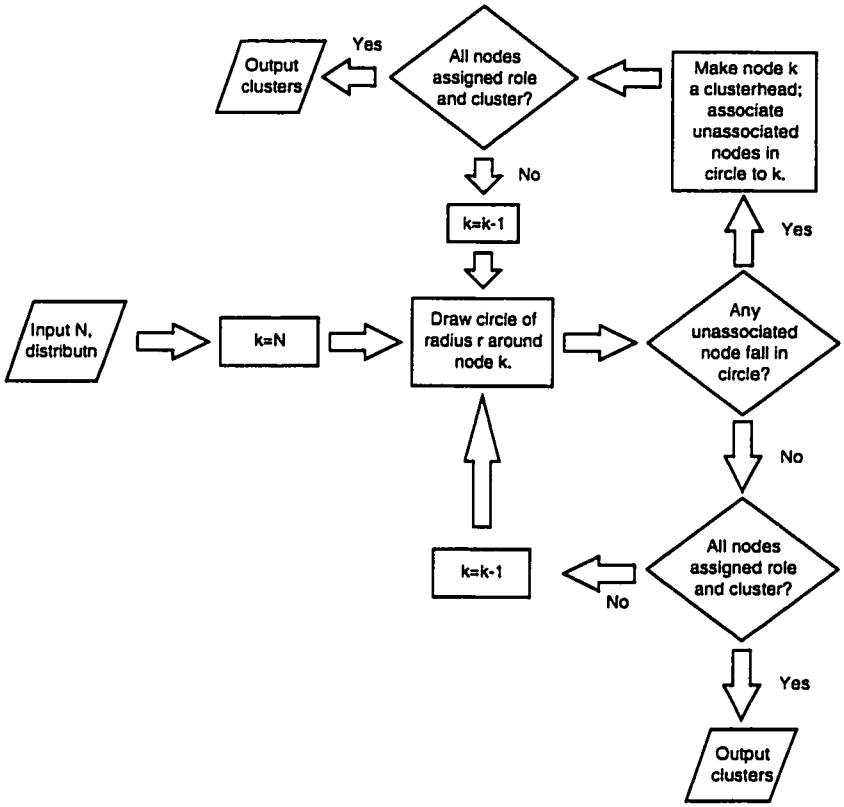


Figure 2.2: The highest ID clustering algorithm.

2.2.2 The Lowest ID Clustering Algorithm

The centralized version of the lowest ID clustering algorithm is depicted in Figure 2.3 and is described in [8, 9]. The algorithm starts with the node 1 and draws a circle around it of radius equal to the communication range. If the communication range is not constant in all directions, an appropriate contour may be drawn instead. Node 1 then becomes a clusterhead, and all nodes that fall inside its circle form the first cluster. Consideration is then given to whether or not node 2 is associated to a clusterhead. If it is already associated to a clusterhead, then it simply remains an ordinary node. However, if it is not already associated, then it becomes a clusterhead and forms a cluster by capturing

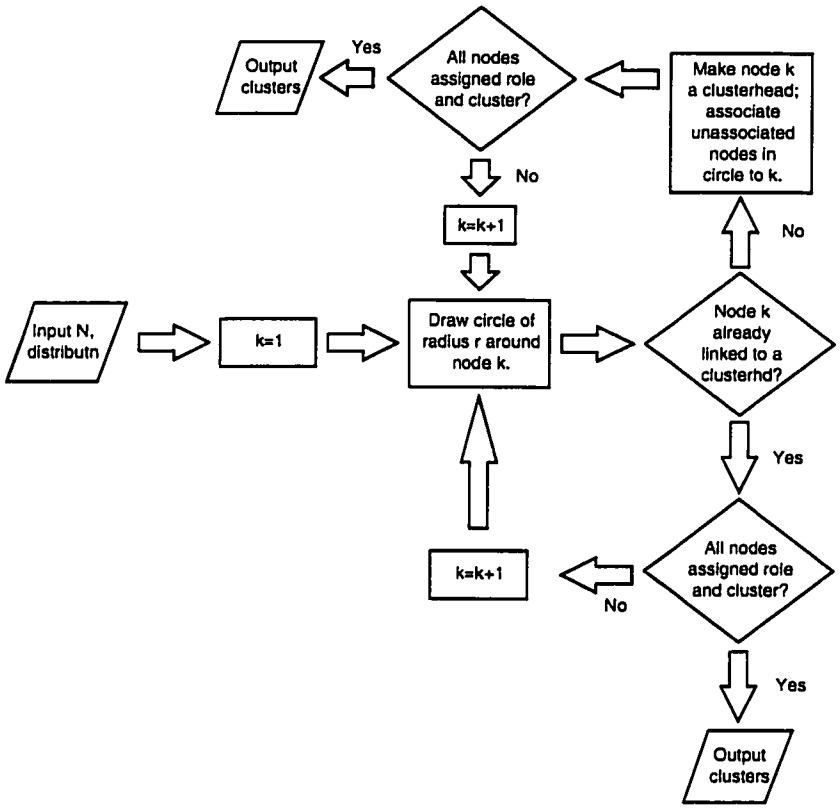


Figure 2.3: The lowest ID clustering algorithm.

all nodes not already assigned to a previously formed cluster. The procedure is repeated for nodes 3, 4, ..., N such that all **N** nodes have been assigned a role and a cluster. Thus, in this algorithm, node *i* becomes a clusterhead unless it is already associated to one. It is principally with this decision rule that the lowest ID algorithm differs from the highest ID. In the case of the lowest ID algorithm, a node becomes a clusterhead unless it is associated to one while in the case of the highest ID algorithm, a node becomes a clusterhead if at least one unassociated node falls inside its communication range.

2.2.3 The Highest Connectivity Clustering Algorithm

The highest connectivity clustering algorithm is described in [2, 10, 11] and is depicted in Figure 2.4 below. The algorithm begins its construction by considering all N nodes and their respective *degrees*.[3] The node of highest degree is chosen to become a clusterhead; in case of a tie, the lowest numbered node becomes the clusterhead. This new clusterhead then forms its cluster by capturing all of its unassociated neighbors. This procedure is simply repeated on the subset of unassigned nodes until all nodes have been typed and associated to a specific cluster.

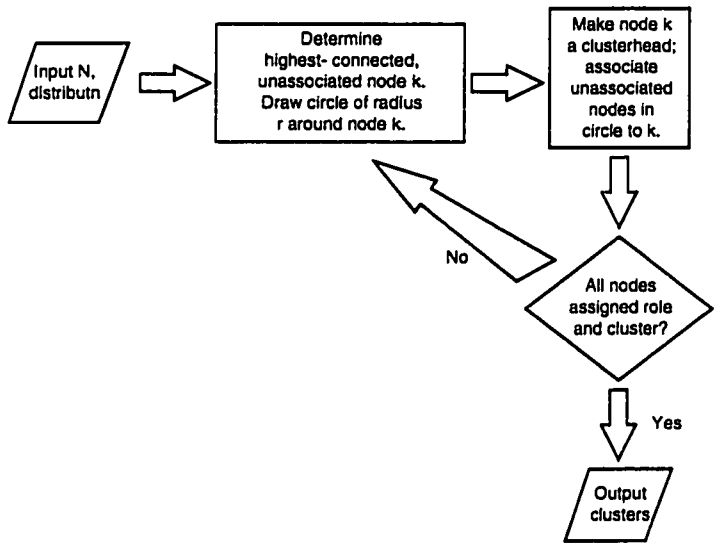


Figure 2.4: The highest connectivity clustering algorithm.

We pause at this point to briefly discuss the properties of the linked cluster networks produced by these algorithms. While it is true that the three

[3]The degree of a node is that node's number of neighbors.

algorithms will configure the network as discussed in the previous section, slight differences exist in the network structure which are worth noting [9]:

- **Clusterhead connectivity** - In all cases, each node in a cluster is connected to the clusterhead. In the case of the highest connectivity and lowest ID algorithms, two clusterheads may never be connected. In the case of the highest ID algorithm, clusterheads may be directly connected.
- **Spatial Reuse** - In the case of the highest connectivity and lowest ID algorithms, clusters may overlap but may never entirely cover each other. The highest ID clustering algorithm may yield clusters which cover each other.

Of these two properties, the second is particularly interesting. For purposes of spatial reuse, it is desirable that the clusters be spread out in space. The highest connectivity and lowest ID algorithms inherently produce such structures, in contrast with the highest ID algorithm which tends to produce more concentrated cluster distributions. Thus, strictly from a structural perspective, we find that the highest connectivity and lowest ID algorithms represent preferable alternatives over the highest ID algorithm.

2.3 Configuration Robustness

As we have seen in the algorithm descriptions above, at least three different algorithms can be applied to construct packet radio networks of the linked cluster architecture type. We have seen that from a structural point of view, the lowest ID and highest connectivity algorithms are essentially equivalent and that both represent preferable choices over the highest ID algorithm. To complement this reasoning in choosing a specific algorithm for our packet radio network design, we are interested in determining if any one of the three algorithms represents a better choice from the perspective of configuration robustness.

What exactly is meant by configuration robustness? The linked cluster architecture is essentially constituted of two components: role assignment, and cluster assignment. We say that a configuration is more or less robust depending

on how well it preserves these assignments over time. Specifically, the more a configuration maintains its assignments over time, the more it is said to be robust.

It is desirable that the network configuration be as robust as possible since we wish to minimize the frequency of the network structuring phase and thus optimize all resources allocated to it. In our case, we wish to construct linked cluster networks which maintain each node's role and cluster assignments for as long as possible. It is important to note at this point that these assignments will necessarily vary over time as the connectivity map changes due to nodal mobility and the varying radio environment. Because each clustering algorithm yields a different configuration, we wish to see if any specific algorithm will yield a structure which is better preserved over time when compared to the structures of the other two.

2.3.1 Simulation

A simulation was designed to measure the robustness of each algorithm under different types of nodal movement. The simulation begins by randomly (uniformly) distributing the **N** nodes in a square area of fixed dimensions. The three different clustering algorithms are then applied to this distribution resulting in three different network configurations. Each of the nodes are then moved according to a specified motion scheme — which we consider further in the next paragraph — and the algorithms are reapplied resulting in three new, possibly different, networks configurations. For each algorithm, the old and new configurations are compared, and changes in either cluster or role assignments are measured.

Motion is broadly categorized into three types, as follows:

- **Fixed** - Each node travels in a fixed direction for the duration of the experiment, or *run*. In our case, each node is assigned a random (uniform) direction between zero and 2π radians at the beginning of each run.
- **Random** - Each node is independently reassigned a new direction at each time interval in a given run. In this simulation, the direction for each node is randomly (uniformly) chosen between zero and 2π radians at each time tick.

- **Hybrid** - Each node independently changes direction at each time tick with probability p. If a node is to change direction at a given time tick, then the new direction is randomly (uniformly) chosen between zero and 2π radians

In all cases, the motion is of one unit per time tick.

Each run lasts for a fixed number of time ticks. At each time tick in the run, after each node is moved and each algorithm has been applied to construct the new configurations, two specific measurements are taken: cluster association; and role assignment. For the case of cluster association, we consider that a cluster change occurred if, for any given node, its new cluster association, i.e. the cluster to which it is associated after it has been moved, is different from the previous association. In the case where role assignment changes, we consider that a clusterhead change has occurred if, for any given node, the new role assignment is different from the previous one, i.e. a node goes from being an ordinary node to being a clusterhead, or vice versa.

The number of role and cluster assignments are averaged over the number of time ticks in the run, yielding an estimate of the average number of role and cluster changes for a given scenario, i.e. a certain number of nodes moving under some type of motion. A large number *runs* of these experiments is then performed, yielding a large collection of estimates which, once averaged over the number of experiments, should represent a good estimate of the average number of role and cluster changes per unit time. These are the figures plotted in the following pages.

2.3.2 Results

Results of the simulation for 10, 25 and 50 nodes, each under fixed, hybrid, and random motion, types are plotted in Figures 2.5 to 2.13 inclusively. Results for 10 and 25 nodes are for 1000 runs; 50 time ticks per run; an initial distribution area of 100 x 100; and a hybrid direction change probability $p = 0.25$. Results for 50 nodes are for 250 runs with all other parameters remaining the same.

The general shape of the plots can be simply explained. For low radio range values, the network tends to be disconnected, with each node constituting its own

cluster. As nodes move in relation to each other, the network remains disconnected, and thus all nodes tend to preserve their cluster and role assignments. Thus, very few changes in either cluster or role assignments occur. The situation is exactly the opposite for large radio range values. In this case, networks tend to be fully connected. Thus, the clusterhead, once it is determined, captures all of the nodes in the network. In essence, the network is constituted of a single large cluster. This remains true even if the nodes move. The cluster structure is thus well-preserved over time and, again, very few changes occur. It is for intermediate values of radio range that the connectivity map truly varies, and it is for these range values that we would expect the most role and cluster assignment changes to occur. This expectation is confirmed by all our plots.

In all cases, we see that the highest connectivity algorithm yields considerably more changes per unit time than the other two algorithms, both in terms of average cluster and clusterhead changes. This result can be explained by the fact that the highest connectivity algorithm uses the degree of each node to configure the network. Thus, any change in just a single node's connectivity can lead to reconfiguration of the entire network. Thus the network configuration produced by the highest connectivity algorithm is inherently unstable.

We further see that in all cases, the average number of cluster changes for the highest and lowest ID algorithms are very close. In terms of average clusterhead changes, the highest ID algorithm presents a very small advantage over the lowest ID algorithm. In terms of average cluster changes, the lowest ID algorithm initially presents a small advantage over the highest ID algorithm, but this advantage is progressively lost as the number of nodes in the network increases. If we strictly consider the case of average changes per unit time, then the highest ID algorithm represents the best choice. We have previously discussed, however, how the lowest ID clustering algorithm yields a more appropriate distribution of clusters in space. Given the clear advantage of the highest and lowest ID algorithms over the highest connectivity algorithm in terms of configuration robustness; our preference for the lowest ID clustering algorithm over the highest ID in terms of spatial reuse; and the insignificant difference between these two algorithms in terms of average cluster and role changes, we choose the lowest ID clustering algorithm for incorporation into our packet radio network design.

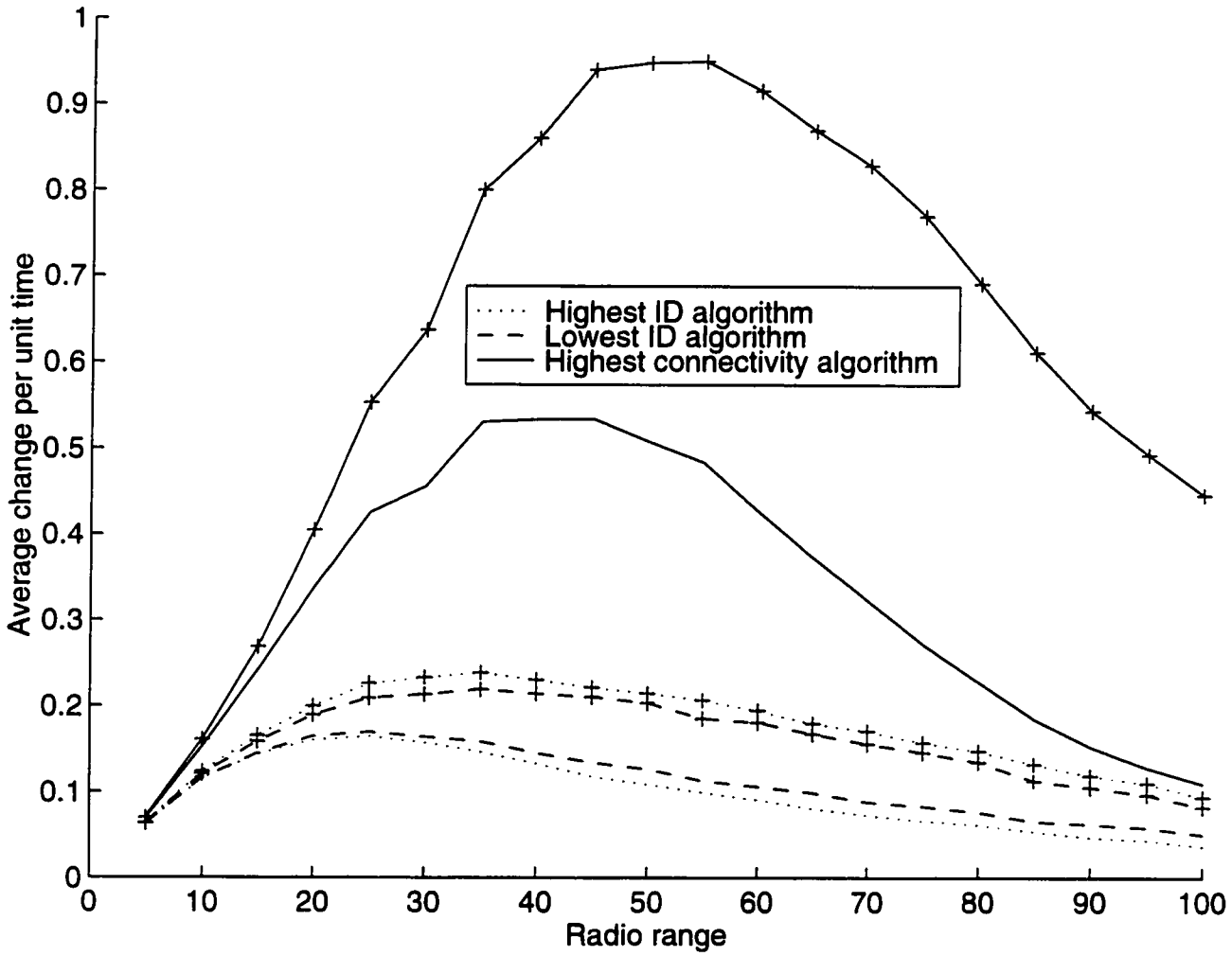


Figure 2.5: Mean cluster and clusterhead changes per unit time for **10 nodes, fixed motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

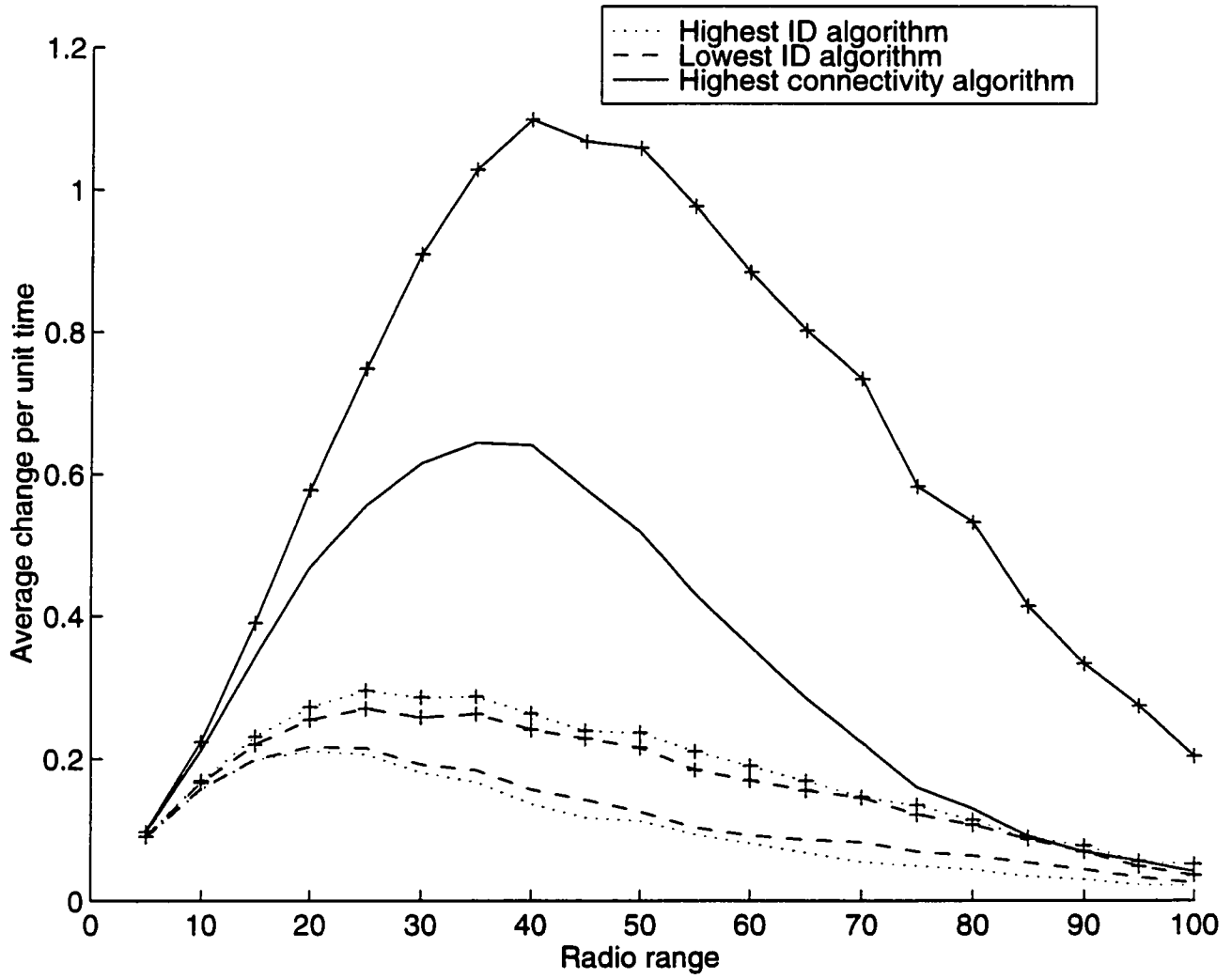


Figure 2.6: Mean cluster and clusterhead changes per unit time for **10 nodes, hybrid motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

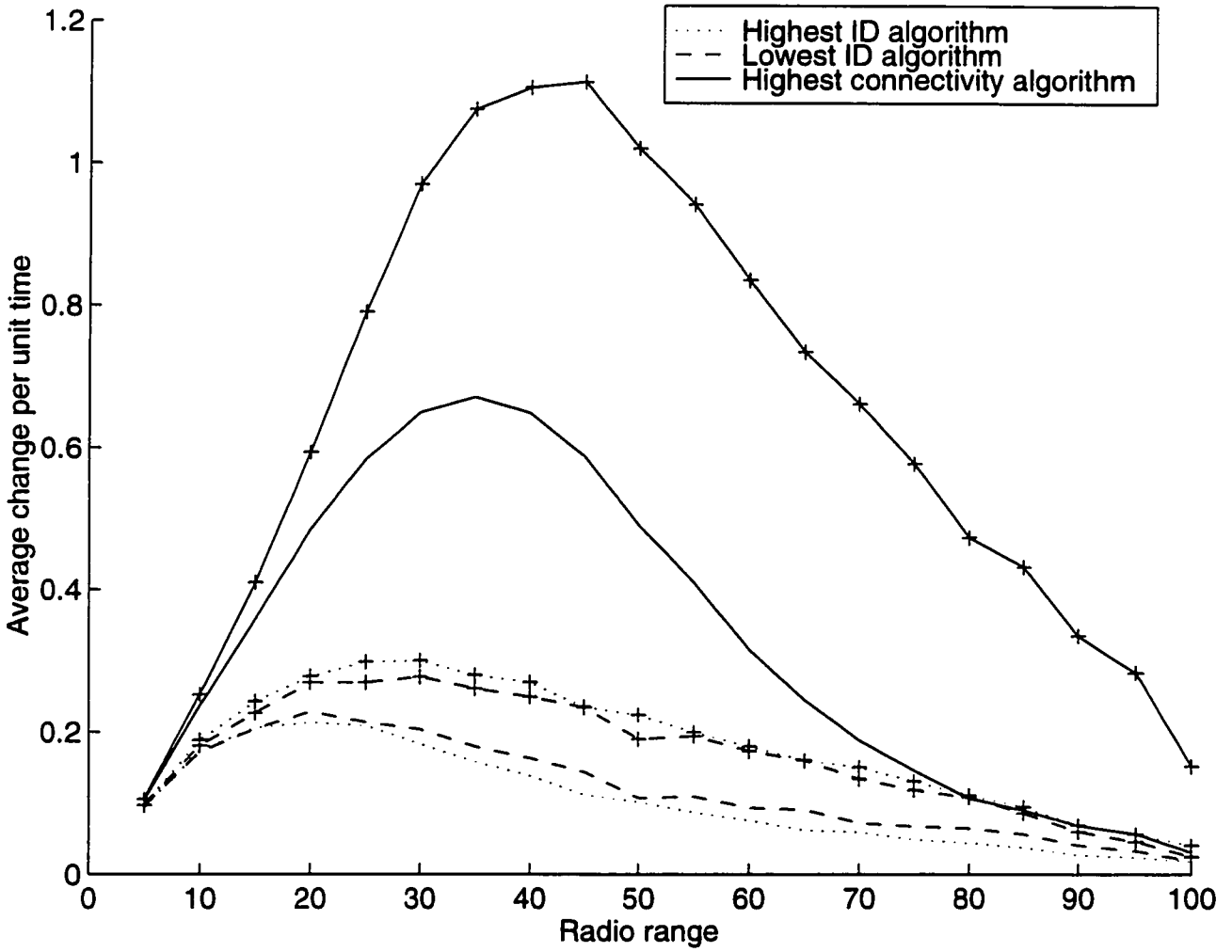


Figure 2.7: Mean cluster and clusterhead changes per unit time for **10 nodes, random motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

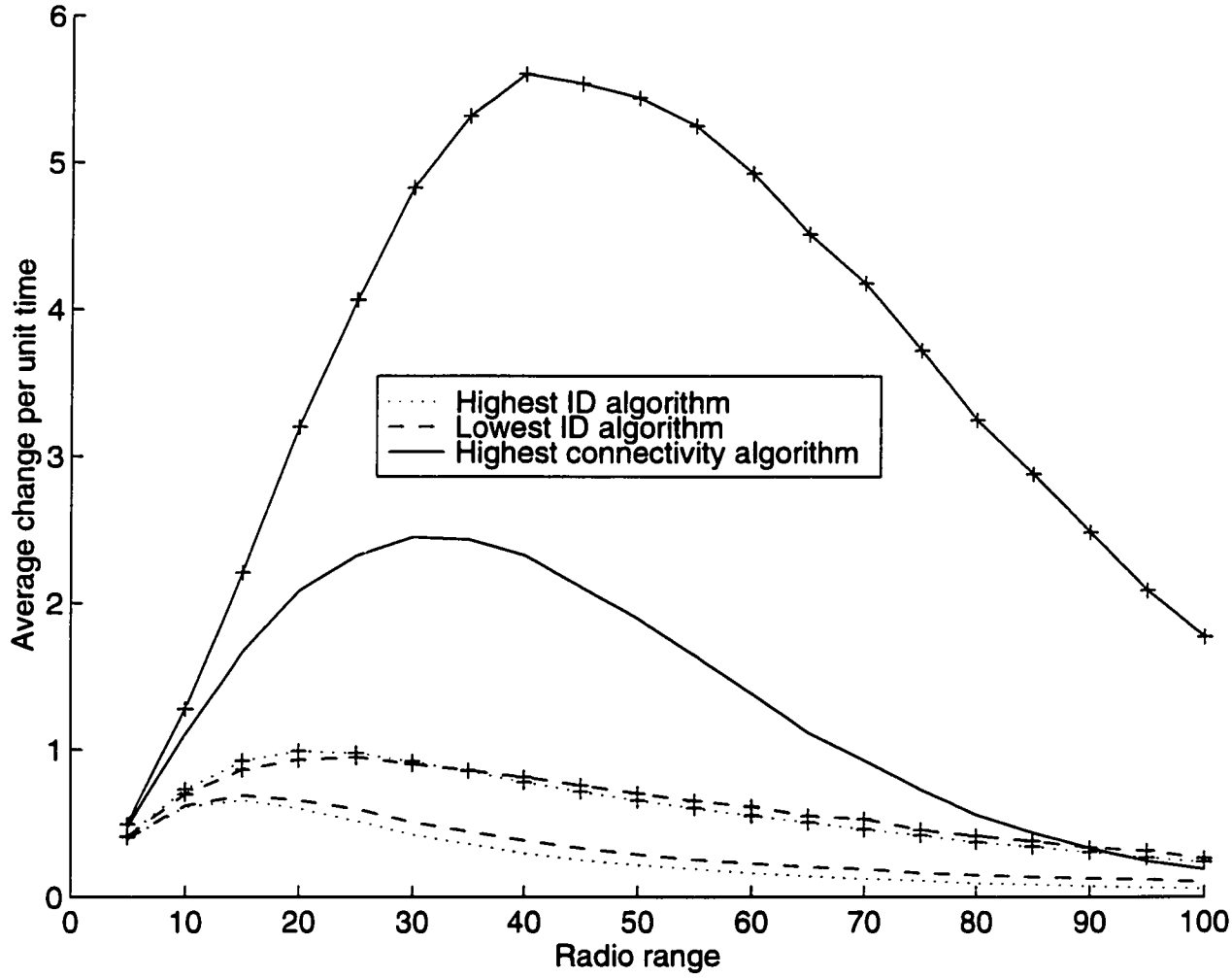


Figure 2.8: Mean cluster and clusterhead changes per unit time for **25 nodes, fixed motion.** Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

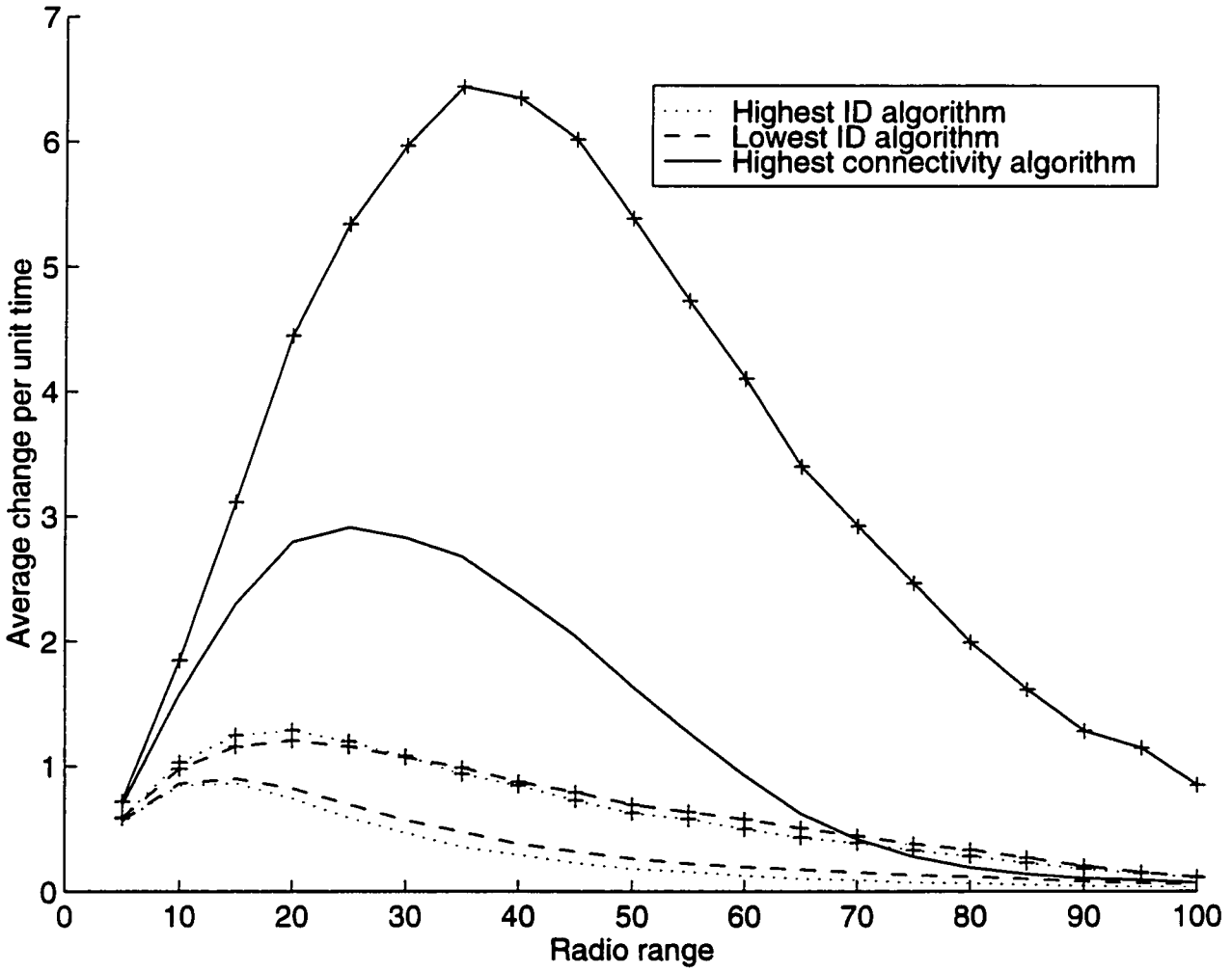


Figure 2.9: Mean cluster and clusterhead changes per unit time for **25 nodes, hybrid motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

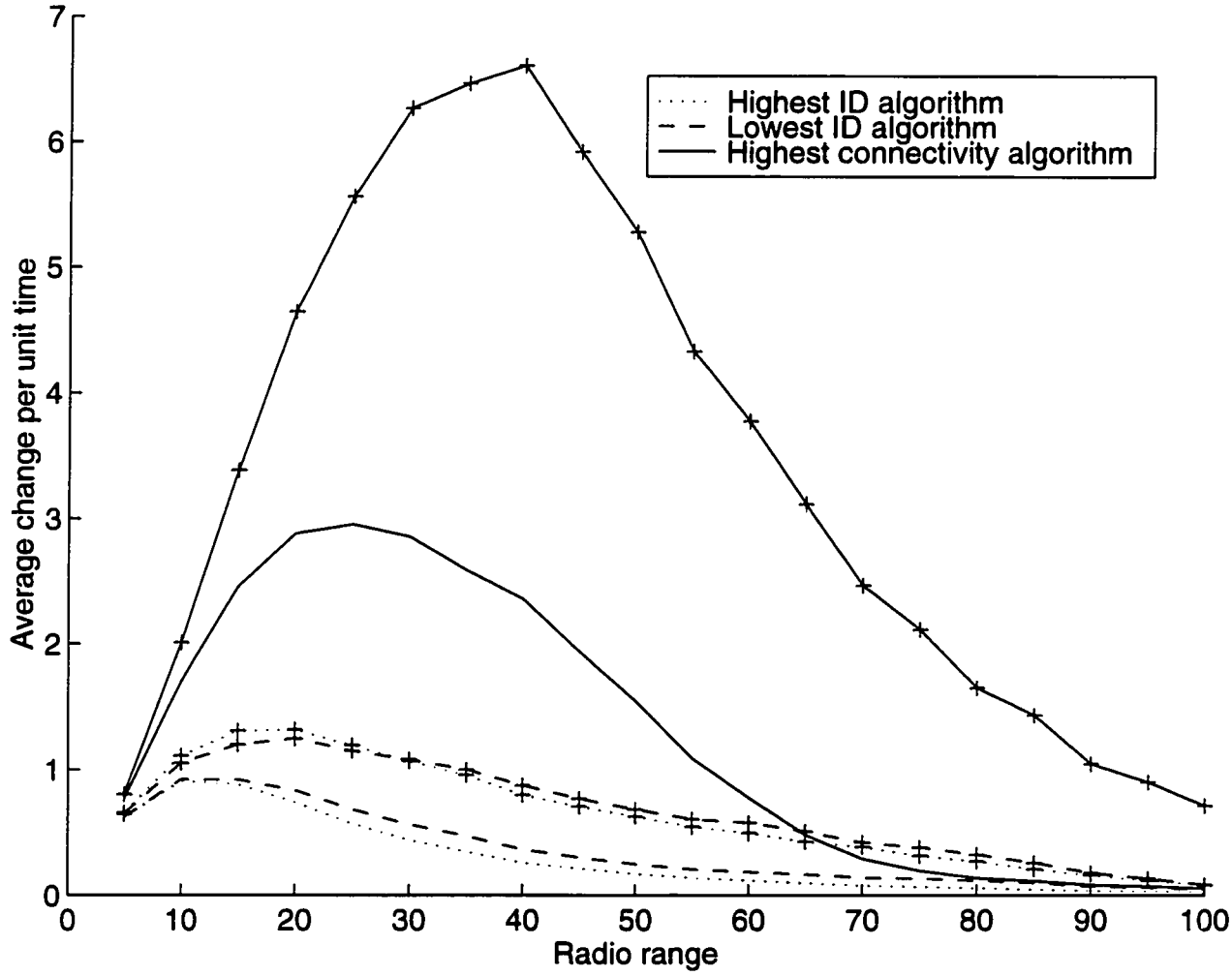


Figure 2.10: Mean cluster and clusterhead changes per unit time for **25 nodes, random motion.** Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

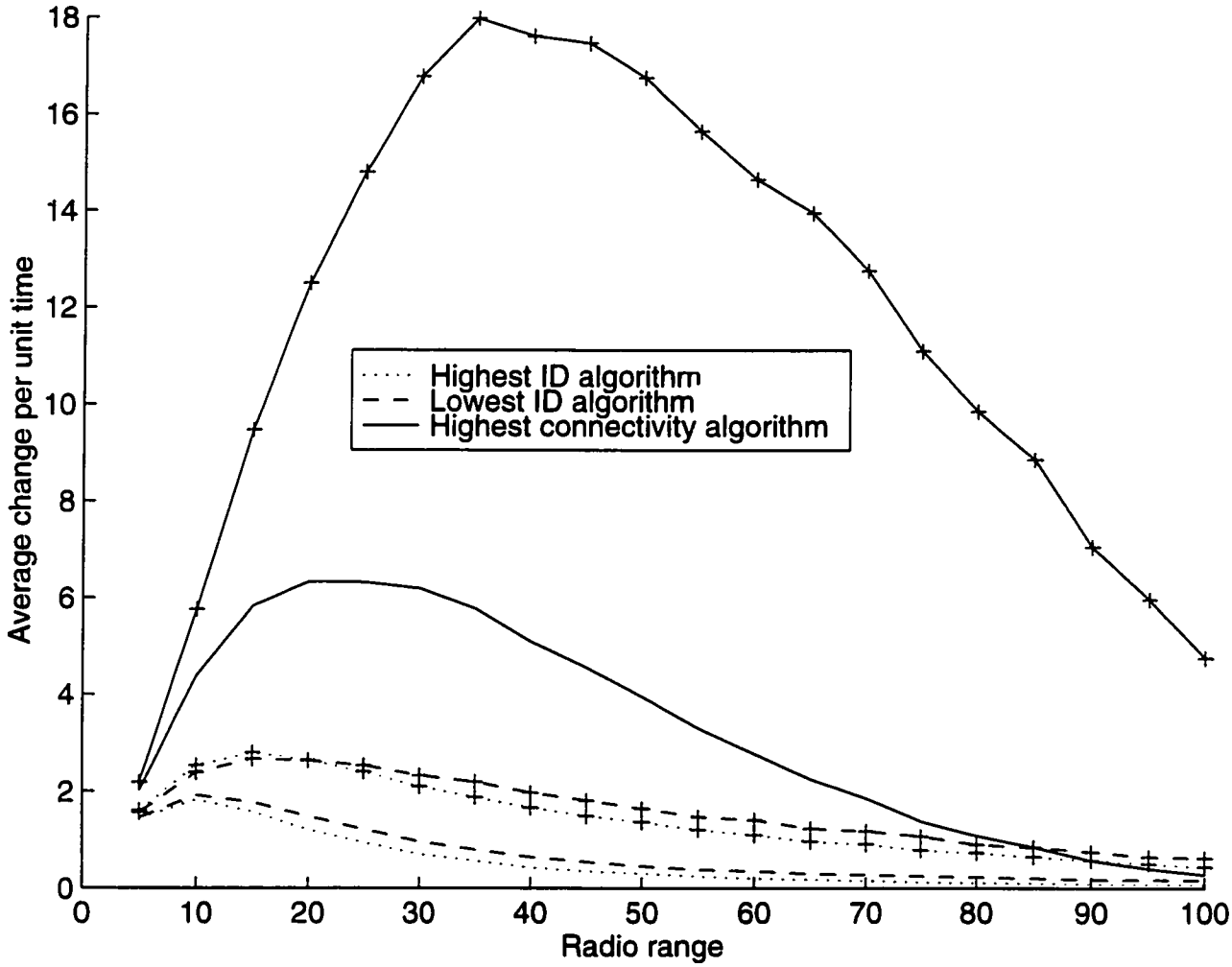


Figure 2.11: Mean cluster and clusterhead changes per unit time for **50 nodes, fixed motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

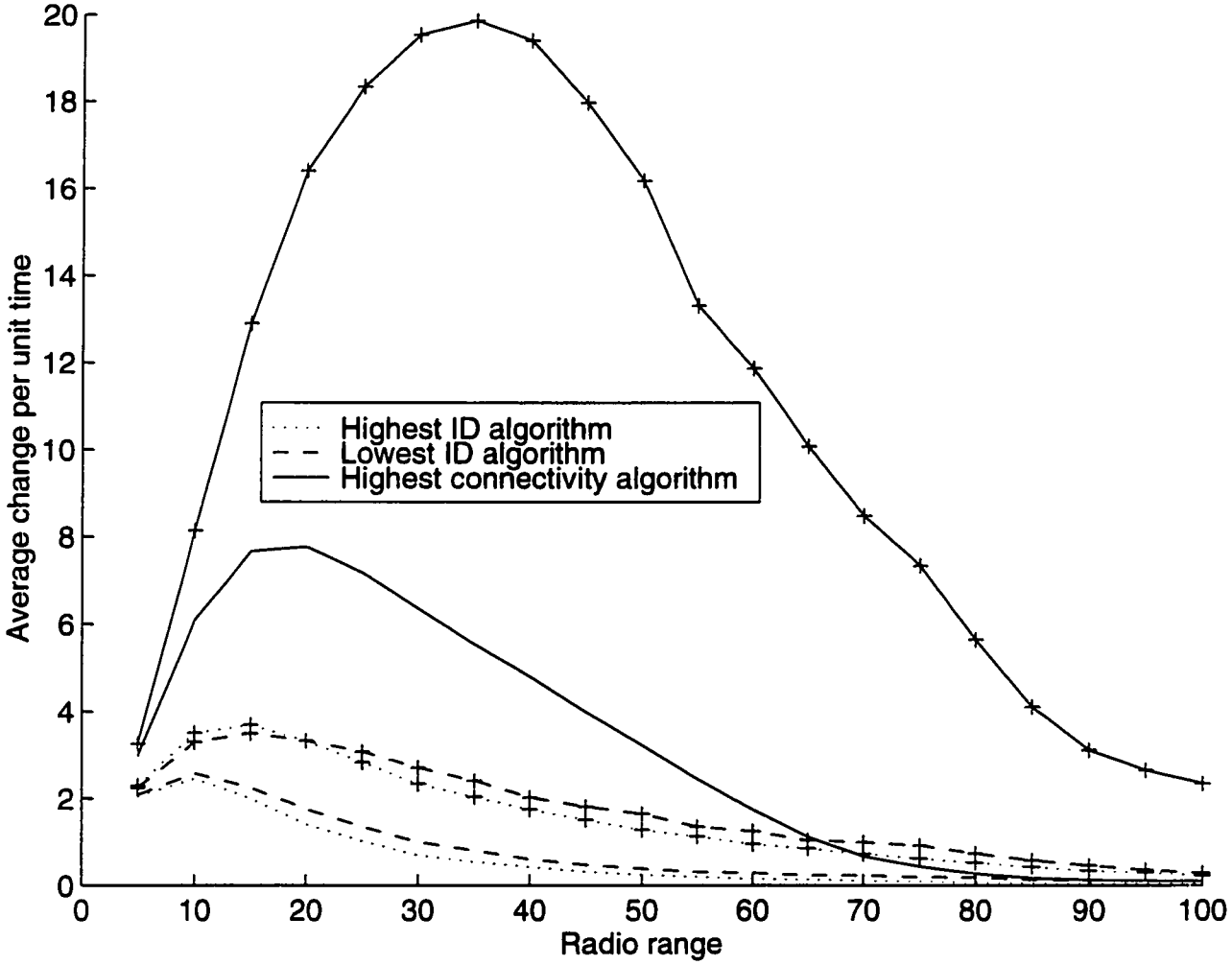


Figure 2.12: Mean cluster and clusterhead changes per unit time for **50 nodes, hybrid motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

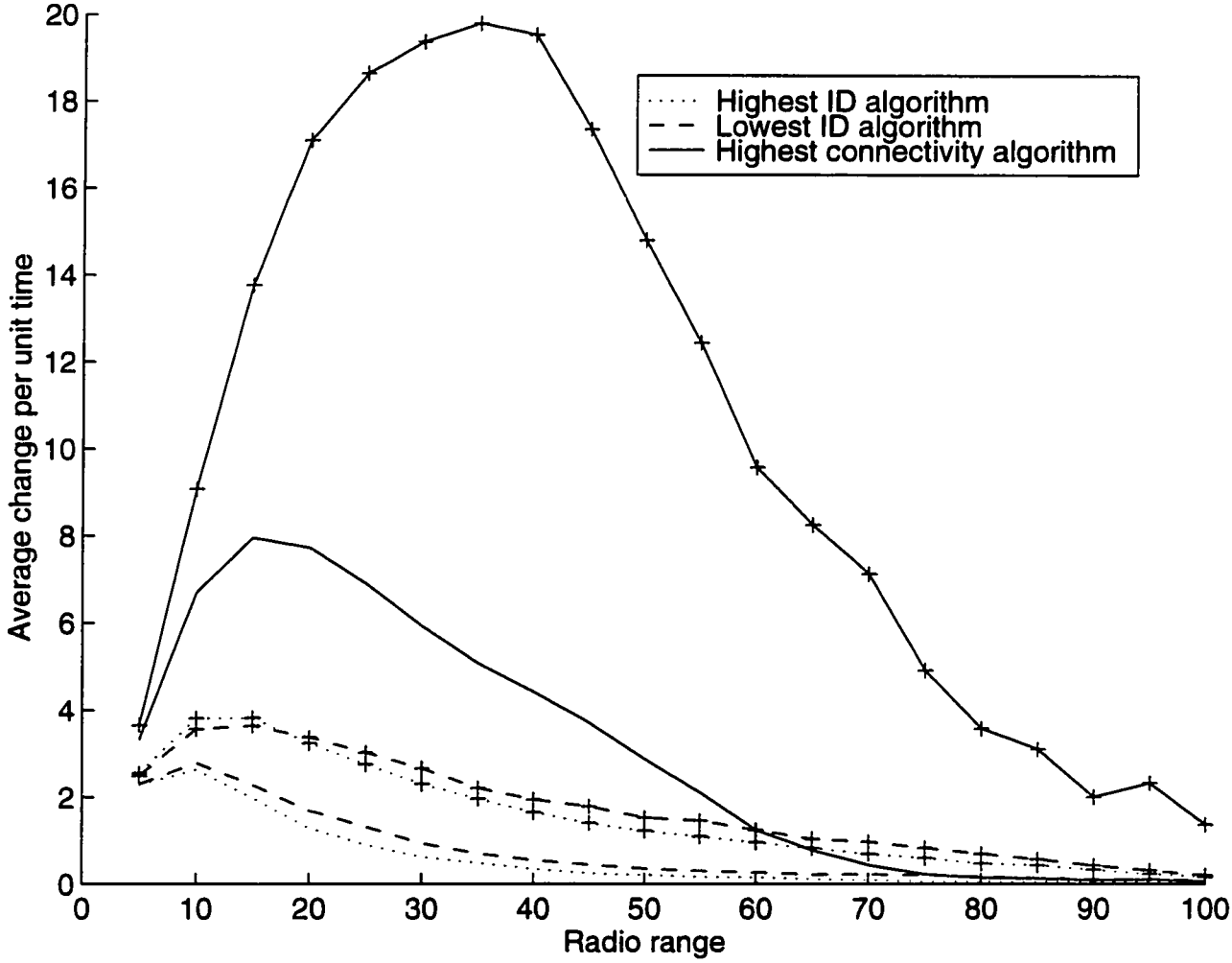


Figure 2.13: Mean cluster and clusterhead changes per unit time for **50 nodes, random motion**. Plots with '+' represent cluster changes while plots with no marks represent clusterhead changes.

Chapter 3

Radio Channel Access

The radio channel access problem is of utmost importance in all types of radio communication systems. The problem can be simply stated: given a specific and common radio channel of fixed bandwidth, how do we distribute channel resources in frequency and time such that any two nodes may reliably communicate? This problem is the problem of multiple access. With the proliferation of data networks, and, more recently, of cellular radio communication systems, many multiple access methods have been designed and studied. We begin this chapter by reviewing the principal multiple access schemes which currently exist.

3.1 Multiple Access

Multiple access schemes are largely categorized in the literature as being of the fixed assignment or random types [4, 5, 6]. We choose to proceed with our discussion of multiple access schemes based on this categorization for the purposes of clarity. We wish to underline, however, that this categorization tends to be misleading in that it implies that the multiple access schemes are mutually exclusive. In fact, current access methods are more often than not a combination of two or more multiple access schemes as they are here categorized. Thus, our

present discussion is more a discussion of elements of multiple access than it is one of well-defined and exclusive techniques.

3.1.1 Fixed Assignment Multiple Access

Fixed assignment multiple access refers to a broad class of multiple access techniques where each user is assigned some, or all, of the system resources for its exclusive use. The three principal methods for sharing this bandwidth are frequency division multiple access (FDMA); time division multiple access (TDMA); and code division multiple access (CDMA).

In FDMA systems, the total bandwidth of the system is subdivided into a set of individual channels which may then be assigned to individual users. Allocations to each user may be determined a priori or may be based on a demand-assignment scheme. In either case, users have exclusivity of the frequency channel when they require to use it, and the scheme is fixed in the sense that the frequency channels are predetermined and fixed in time.

TDMA systems divide the bandwidth in time rather than in frequency. A TDMA frame is divided into a series of slots in which only a single user is allowed to transmit. A sequence of frames thus produces a cyclical repetition of time slots where any particular slot may be viewed as a communication channel for any given user. These slots may be preassigned to the system users or may be dynamically allocated. Thus, users have access to the entire system bandwidth, but only for small, successive periods of time. Here again, the scheme is fixed in the sense that the time slots are predetermined and fixed in time.

In CDMA systems, each user's signal is spread over a very large bandwidth through multiplication by a direct-sequence spreading code. This spreading code is a pseudo-noise sequence with a chip rate much greater than the original signal's data rate. Each user in the system is assigned a pseudo-noise sequence from a set of almost orthogonal sequences and uses this assigned sequence to code its signal. All users can then simultaneously transmit over all of the spread bandwidth. With knowledge of the specific code that a particular sender is using, any receiver may perform time correlation operations to detect a given sender signal since all other coded signals are orthogonal and appear as noise. Thus,

each user can theoretically operate independently of the other users, accessing the entire bandwidth for all time.

Fixed assignment multiple access schemes are relatively efficient when each user has a continuous, steady flow of information to transmit. When the information is inherently bursty, however, these mechanisms can prove to be inefficient because assigned resources may sit idle for prolonged periods of time while others may be fully loaded. Random multiple access schemes have been developed to resolve this efficiency issue. Random multiple access provides a more efficient and flexible approach to sharing channel access in the context of short, spurious communications. We now briefly review the principal random multiple access schemes.

3.1.2 Random Multiple Access

In contrast with fixed-assignment access schemes where no contention exists for channel resources, random multiple access are based on the idea that users must compete for use of the system resources. Thus, contention, and how to manage it, is the primary concern of random multiple access techniques. Depending on how rigorous the contention management mechanisms are, we say that the random access scheme is more or less disciplined. Of the two broad, principal types of random multiple access schemes, ALOHA and carrier sense multiple access (CSMA), ALOHA is the least disciplined.

Pure ALOHA is very simple to describe. If a user has information to transmit, then he simply transmits and gives no consideration to what time it is or to whether other users are transmitting. Of course, there will be collisions when two users choose to transmit at the same time. Thus, after a transmission, any sender waits for an acknowledgment from the intended destination. If it does not receive this acknowledgment, then the sender waits for a random amount of time, after which the message is retransmitted. This process is repeated until such a time as the message is properly sent from source to destination.

Pure ALOHA provides poor throughput and delay performance under heavy offered load. In order to improve its efficiency, variations on the pure ALOHA have been devised: *slotted* and *reservation* ALOHA. In slotted ALOHA,

transmission time is divided into slots. Any user who wishes to transmit must wait for the next available slot to do so. This slot structure in effect halfs the vulnerability period for transmission collisions and doubles the peak throughput of pure ALOHA.

Reservation ALOHA goes a step further by designating two types of slots: *reservation* and *message* slots. Reservation slots are divided in a series of subslots which are accessed by the users on the basis of slotted ALOHA. These subslots are used by the users to reserve one or more of the message slots. Once a message slot has been reserved by any given user, it has exclusive rights to that slot until it relinquishes control. Thus, reservation ALOHA may be viewed as a dynamic TDMA system where rights to a fixed TDMA cycle are assigned to specific users over a contention-based, slotted ALOHA control channel. This provides us with a very good example of how it becomes difficult to categorize some of the more complex multiple access schemes. In this case, we see that reservation ALOHA is not strictly fixed-assignment or random but rather integrates elements of both: the control channel is clearly random but the data channel is fixed. We can further imagine many scenarios which mix and match the elementary approaches to multiple access. It illustrates well why it is better to speak of elements of multiple access rather than of exclusive, well-defined multiple access schemes.

The second important class of random access schemes is the CSMA class. In CSMA, much more effective use is made of the channel by listening to it before transmitting. Thus, a considerable portion of potential collisions are averted because senders will not transmit if the channel is sensed busy. Collisions are still possible, however, because of propagation delays.

There exists three principle CSMA random access schemes: *1-persistent*, *nonpersistent*, and *p-persistent* CSMA. Each is described as follows:

- **1-persistent CSMA** - This is the simplest form of the CSMA schemes and can be either slotted or unslotted. Any user who wishes to transmit listens to the channel. If the channel is idle, the user immediately transmits.[1] If the channel is sensed busy, then the user continuously monitors the channel

[1] If the access scheme is slotted, the user transmits in the next available slot.

until it becomes free, at which time the message is transmitted. The sender then listens for an acknowledgment. If none is received after a certain amount of time, then the sender repeats the transmission process after a random amount of time.

- **Nonpersistent CSMA** - In this variation on the basic, 1-persistent CSMA scheme, a user does not sense the channel continuously after it senses it to be busy. Instead, it waits for a random amount of time, after which it senses the channel again. This tactic eliminates most of the collisions that occur from multiple users transmitting simultaneously upon sensing the channel to be idle.

- **p-Persistent CSMA** - Th p-persistent approach is a generalization of the 1-persistent scheme and is necessarily slotted. When a user has data to send, it senses the channel. If it is sensed idle, then it either transmits with probability p or defers until the next slot with probability $q = 1 - p$. If it chooses to defer, then it repeats the process until the message is transmitted or the channel is sensed busy. In all cases, if the channel is sensed busy, the user senses the channel continuously until it is free.

We see that fixed-assignment and random multiple access schemes offer very different solutions to the problem of sharing a common communication channel. Fixed-assignment multiple access schemes are best suited for continuous, real-time traffic where throughput and associated delay are of primary importance, whereas random multiple access schemes are effective in bursty traffic environments. Hybrid multiple access schemes, such as reservation ALOHA and its close cousin, packet reservation multiple access, aim at integrating elements of these two broad approaches into a single access method capable of supporting mixed traffic types.

3.1.3 A choice of multiple access scheme

We have selected a pure TDMA multiple access scheme for our packet radio network. The primary reason for making a choice in favor of fixed-assignment multiple access over random multiple access is related to the multimedia nature of our packet radio network. Because the network must carry multimedia traffic, it must be capable of carrying both continuous, real-time traffic in addition to

short, bursty data. Those applications which generate real-time traffic, typically voice and video, require certain bandwidth and delay guarantees which are best provided by fixed-assignment multiple access schemes. These techniques will also provide adequate performance for bursty applications, though it is recognized that this approach is not the most effective. Thus, for the specific purposes of our packet radio network design, we have opted for a fixed-assignment multiple access method.

Justification for choosing TDMA over either an FDMA or CDMA-based scheme is more subjective. It is fair to say that current trends in wireless technology have moved away from FDMA towards TDMA or CDMA systems. This trend undoubtedly has to do with the fact that, in general, it eases network deployment and resource allocation and management to have all users tuned to the same frequency [4]. From this perspective, FDMA does not represent a very interesting choice. Furthermore, much wireless research has recently gone into the development of pure CDMA systems. This is particularly true for the design of packet radio networks, where a CDMA approach provides a convenient mechanism for circumventing the central problem of interference. In contrast, few designs are based on a pure TDMA approach. We therefore thought that it would be a challenging research exercise to design a packet radio based solely on this multiple access scheme, though we recognize that a pure TDMA approach may not represent the best of all possible options. This research challenge, combined with the relevance of TDMA in terms of current wireless technologies, justifies our choice of TDMA for our packet radio network design.

3.2 Time Division Multiple Access

3.2.1 The Scheduling Problem

Our discussion and analysis of the scheduling problem begins with an understanding of collision mechanisms in packet radio networks. Specifically, depending on the type of transmission scheduling scheme used in the packet radio network, transmissions may be subject to two types of collisions: *primary* and *secondary* interference. Primary interference occurs when any single node is

expected to do more than a single activity in one time slot. For example, a node might be scheduled to receive and transmit at the same time; or a node might be scheduled to simultaneously receive from two different transmitters. Secondary interference refers to the situation where a certain receiver R, listening for transmissions from transmitter T_1, is within range of a second transmitter T_2 whose transmissions, though not intended for R, interfere with the transmissions of T_1 [12]. It is important to note that this characterization is based on the assumption that each node communicates over a single TDMA channel in a half-duplex mode.

The problem of assigning transmission rights to the nodes of the radio network can be approached in two different ways: *broadcast* (node) scheduling, and *link* (point-to-point) scheduling [12, 13, 14]. For broadcast scheduling, it is the nodes which are assigned time slots during which they may transmit to any of their neighbors. In the case of link scheduling, time slots are allocated to the directed links between pairs of nodes[2] The distinction between the two types of scheduling is illustrated in Figure 3.1. In the following two subsections, we consider each scheduling type separately to precisely formulate the scheduling problem for each case, subject to the common constraint that no primary and secondary interference be tolerated.

Broadcast Scheduling

Let us be given a packet radio network represented by the directed graph $G = (V, E)$. Here, V represents the set of all nodes in the network ($|V| = N$) while E represents the set of all directed links (u, v) between all neighbor nodes $u, v \in V$. The broadcast scheduling problem consists in associating a slot with each node subject to the following constraints:

- Each node is assigned to a single slot;
- Any pair of nodes which are directly linked or which share a common neighbor cannot be assigned to the same slot; and
- The number of slots is minimized.

[2]Any pair of neighbors u and v are connected by two directed links denoted (u, v) and (v, u), where (u, v) represents the directed link of node u transmitting to receiving node v.

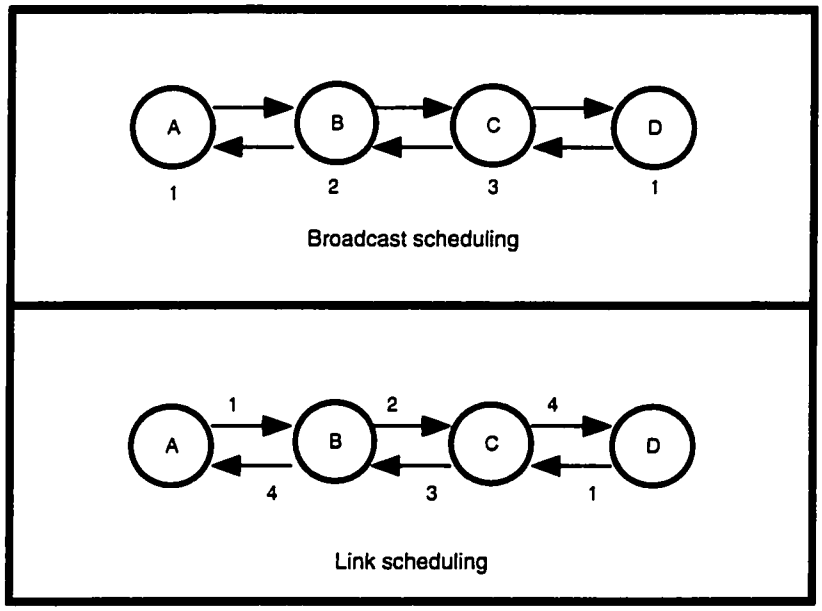


Figure 3.1: Broadcast and link scheduling. Numbers depict the time slot during which transmissions are allowed.

The problem as detailed above consists of the *optimal* broadcast scheduling problem. This problem is closely related to the classic graph theoretic vertex coloring problem and has been shown to be NP-complete [8, 12, 15], and thus the optimal solution can only be determined from an exhaustive search of all possible solutions. In the context of static networks where the network topology is constant in time, such a computational effort might be justifiable. In the context of mobile packet radio networks, however, reevaluating the optimal schedule after each topology change is clearly infeasible. Furthermore, the optimal solution requires global connectivity information [8, 13]. Collecting such information to a central location for computational purposes could prove to be very inefficient and is inconsistent with our initial objective that the network be distributed. For these reasons, we dismiss the optimal broadcast scheduling

problem and instead reformulate the scheduling problem as follows:

- Each node is assigned to at least one slot; and
- Any pair of nodes which are directly linked or which share a common neighbor cannot be assigned to the same slot.

This less constrained broadcast scheduling problem consists of the *maximal* broadcast scheduling problem. Here, maximal is used in the sense that for any given slot in the schedule, no additional node can be allowed to transmit without interfering with the transmission of at least one node already scheduled in that slot.

Let V_m denote the subset of all nodes in V scheduled to transmit in slot m. We call V_m the *m-th broadcast mode*. For each node $u \in V_m$, we let C_u^m denote the *broadcast zone* of u as the union of nodes u and its one-hop and two-hop neighbors[3]. The non-interference condition of the problem can then be expressed as follows [16, 17]:

$$\forall u, v \in V_m, v \notin C_u^m \tag{3.1}$$

This condition guarantees that no primary interference will occur in any given slot[4]. By *blocking* the broadcasting node's one-hop neighbors, we guarantee that this node will not have to both transmit and receive in the same slot. Similarly, by blocking the broadcasting node's two-hop neighbors, we guarantee that its one-hop neighbors will not be scheduled to receive from two different nodes in the same slot. The requirement for these constraints is illustrated in Figure 3.2.

We have thus far discussed and analyzed the significance of the second condition of the problem statement. We now consider the first. The obvious reason for it is that we do not wish to construct schedules which, though they might be maximal, do not allow all users to transmit at least once in the TDMA cycle. Nodes may, however, be allowed to transmit in more than a single slot, subject to meeting the constraints imposed by the non-interference criteria.

[3]The two-hop neighbors of u consist of the neighbors of the neighbors of u.

[4]By the definition we have given to secondary interference, broadcast schedules are not subject to secondary interference [12].

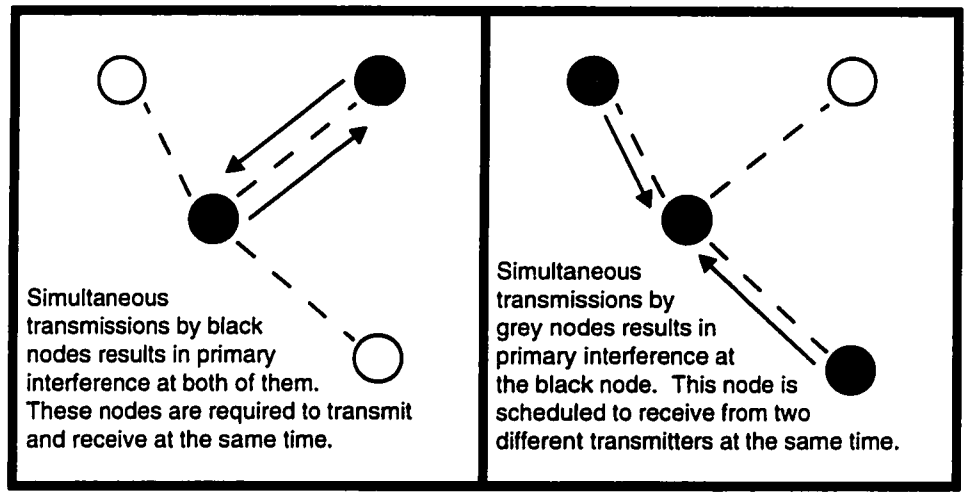


Figure 3.2: Primary interference in broadcast scheduling.

Consider, for example, the simple five - node network depicted in Figure 3.3. A possible broadcast schedule for this network is given as follows:

Node\Slot	1	2	3	4
1	T	R		T
2	R	T	R	
3	R		T	R
4	T	R		
5	R			T

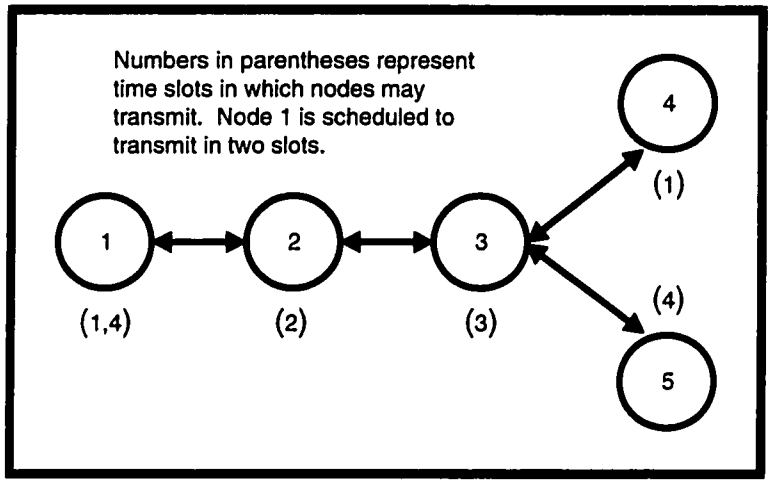


Figure 3.3: Broadcast scheduling and multiple slot assignment.

In this example, a T refers to a node scheduled to transmit, while an R represents a receiving node. We see that all nodes are scheduled to transmit at least once in the TDMA cycle of four slots. In particular, we see that node 1 is scheduled to transmit in both slots 1 and 4. While it is true that node 1, having been scheduled in slot 1, is guaranteed one collision-free transmission per four-slot cycle, we choose to allocate to it a second slot in slot four in order to maximize the point-to-point throughput of the network.

Clearly, it is desirable, though it is not essential, that any node be allowed to transmit as often as possible, subject to the initial conditions set in the problem statement. In terms of efficiency, it is desirable that for each broadcasting mode V_m [16, 17],

$$V = \bigcup_{u \in V_m} C_u^m \tag{3.2}$$

In short, we wish that, for each broadcast mode in the cycle, each node be either transmitting or receiving. If this condition holds for a given broadcast mode, then this mode is said to be *maximal.* When the condition fails, the broadcast mode is not maximal and at least one node may be added to that slot without causing

interference. For example, consider V_4 in the previous network. Assume that node 1 only transmits in slot 1, leaving only node 5 to transmit in slot 4. In this case, V_4 = node $\{1\}$, and $\bigcup_{u \in V_4} C_u^4$ = nodes $\{2, 3, 4, 5\}$, which excludes node 1. Thus, we choose to include node 1 in V_4 = nodes $\{1, 5\}$, and V_4 becomes a maximal broadcast mode.[5]

We have so far discussed and analyzed both the optimal and maximal broadcast scheduling problems. We have seen that in the context of mobile packet radio networks, it is irrealistic if not impossible to consider solutions to the optimal scheduling problem and must thus limit ourselves to the maximal scheduling problem. We continue our discussion by considering the link scheduling problem.

Link Scheduling

As was the case for the broadcast scheduling problem, let us be given a packet radio network represented by the directed graph $G = (V, E)$. Each directed link (u, v) can be represented by its *sending* node u and its *receiving* node v. The *optimal* link scheduling problem consists in associating a slot with each directed link subject to the following constraints:

- Each directed link is assigned to a single slot;
- Links (α, u) and (u, γ) cannot be assigned to the same slot, where α and γ represent neighbors, different or identical, of u, i.e. any single node cannot be both a sending and receiving node at the same time;
- Links (u, v) and (w, z) must be assigned to different slots if link $(w, v) \in E$, i.e. for any single sending node, its associated receiving node's neighbors cannot also be sending nodes in the same slot; and
- The number of slots is minimum.

In much the same manner as in the case for the optimal broadcast scheduling problem, the optimal link scheduling problem is closely related to the classic

[5]Note that the other broadcast modes are also maximal.

graph theoretic vertex coloring problem and has been shown to be NP-complete [8, 12]. For this reason, we choose to dismiss solving this problem and instead reformulate it as the following *maximal* link scheduling problem [13, 18]:

- Each directed link is assigned to at least one slot;
- Links (α, u) and (u, γ) cannot be assigned to the same slot, where α and γ represent neighbors, different or identical, of u, i.e. any single node cannot be both a sending and receiving node at the same time; and
- Links (u, v) and (w, z) must be assigned to different slots if link $(w, v) \in E$, i.e. for any single sending node, its associated receiving node's neighbors cannot also be sending nodes in the same slot.

Here, the schedule is maximal in the sense that for any given slot, no additional link can be scheduled without causing interference with transmissions already scheduled in that slot.

Let E_m denote the subset of all links in E scheduled for activation in slot m. Associated with E_m are the two subsets of all nodes in V,S_m and R_m representing respectively the sending nodes and receiving nodes of slot m. We call E_m, S_m, and R_m the *m-th link activation*, *sending*, and *receiving* modes, respectively. For each link $(u, v) \in E_m$, let $C^m_{(u,v)}$ denote the *link activation zone* of (u, v) consisting of all links incident on sending node u; all links incident on receiving node v; all links incoming to the neighbors of u; and all links outgoing from the neighbors of v. Thus, the non-interference condition of the problem can be expressed as follows:

$$\forall (u, v), (w, z) \in E_m, (w, z) \notin C^m_{(u,v)} \tag{3.3}$$

This condition guarantees that neither primary or secondary interference will occur in any given slot. More specifically, by blocking all links incident on the sending node and receiving nodes, we guarantee that both the sending and receiving nodes will not be subject to primary interference, be it either having to transmit to more than a single node; having to transmit and receive in the same slot; or having to receive from more than one transmitting node. By blocking all links incoming to the neighbors of the sending node, we guarantee that these neighbors will not be subject to secondary interference. Similarly, secondary

interference at the receiving node is eliminated by blocking all links outgoing from its neighbors. The requirement for these constraints is depicted in Figure 3.4.

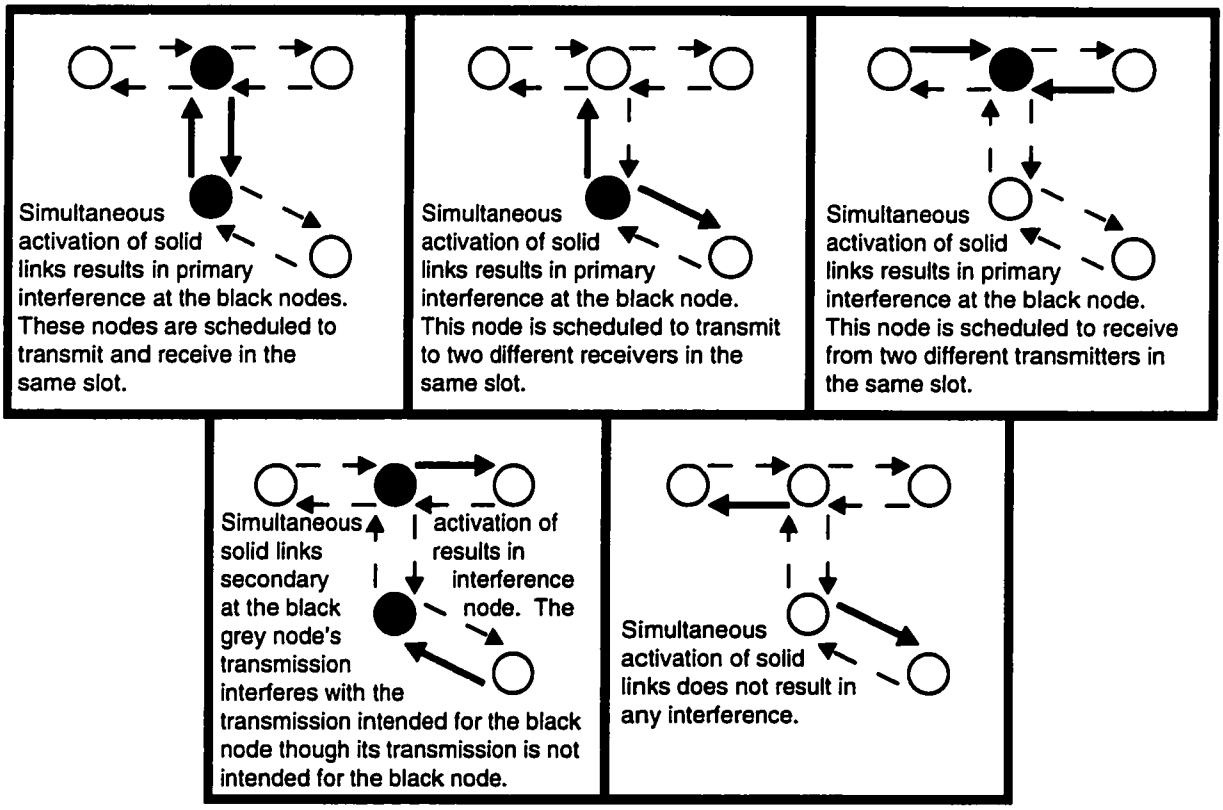


Figure 3.4: Primary and secondary interference in link scheduling.

As was the case for broadcast scheduling, we do not wish to construct schedules which, though they might be maximal, do not allow all links to be activated at least once in the TDMA cycle. Furthermore, in order to maximize point-to-point throughput, it is desirable that links be activated as often as possible in each cycle, subject to meeting the constraints imposed by the

non-interference criteria. For example, consider the link equivalent, depicted in Figure 3.5, of the previous five-node network of Figure 3.3. A possible link schedule is given as follows:

Node\Slot	1	2	3	4	5	6
1	T_2	R_2	I		R_2	T_2
2	R_1	T_1	T_3	R_3	T_1	R_1
3	R_4	T_4	R_2	T_2	T_5	R_5
4	T_3	R_3	I			
5	I				R_3	T_3

In this schedule, $R_i(T_i)$ signifies that a specific node is scheduled to receive (transmit) from (to) node i. An **I** indicates that a node is *idle*, meaning that it cannot be scehduled to transmit without interfering with the transmission of other nodes and that it is not the intended destination of any sending node. Thus, we see that in slot 1, for example, two directed links are activated: links $(1, 2)$ and $(4, 3)$.

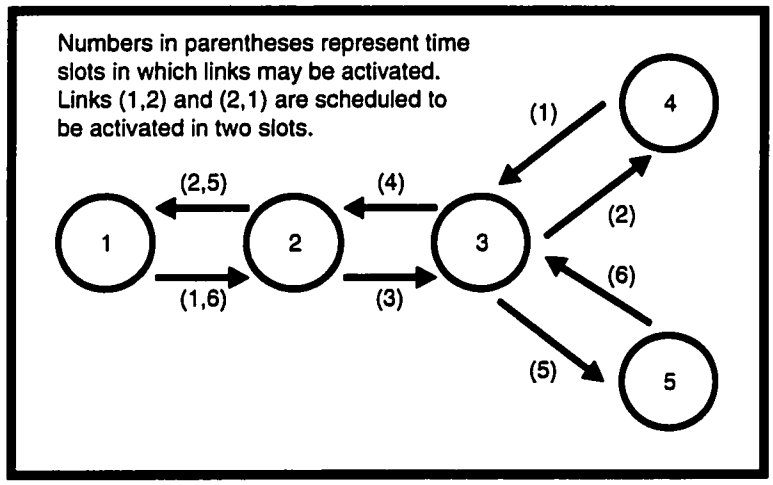


Figure 3.5: Link scheduling and multiple slot assignment.

By taking a closer look at this example, we see that all links are scheduled for activation at least once in the TDMA cycle of six slots and that links $(1, 2)$ and $(2, 1)$ are both activated twice. While it is true that these links are both guaranteed one collision-free activation in the cycle, having been scheduled for activation in slots 1 and 2 respectively, we choose to allocate them secondary slots in slots 6 and 5 in order to maximize the point-to-point throughput of the network.

It is desirable, though not essential, that any link be activated as often as possible, subject to meeting the initial conditions set in the problem statement. In terms of efficiency, it is desirable, that for each link activation mode,

$$E = \bigcup_{(u,v)\in E_m} C^m_{(u,v)} \tag{3.4}$$

In other words, we wish that all links, for any link activation mode, be either active or blocked. The link activation mode is maximal when this condition holds. When the condition fails, at least one node may be added to the link activation mode without causing interference. Referring to our previous example, consider E_1 and assume that only link $(4, 3)$ is scheduled to be active, i.e. link $(1, 2)$ is active only in slot 6. Under this assumption, E_1 = link $\{(4, 3)\}$, and

$$\bigcup_{(u,v)\in E_1} C^1_{(u,v)} = \{(2, 1), (2, 3), (3, 2), (3, 4), (4, 3), (3, 5), (5, 3)\}.$$

We see that link $(1, 2)$ is not included in this list. Thus, we choose to include it in E_1, and thus E_1 = links $\{(1, 2), (4, 3)\}$ constitutes a maximal link activation mode.

This concludes our discussion and analysis of the optimal and maximal link scheduling problems. As was the case for broadcast scheduling, we have seen why we must limit ourselves to the maximal scheduling problem. This problem has been concisely discussed, analyzed, and characterized. The next section of this chapter focuses on justifying a choice of both a scheduling type and algorithm.

3.3 A choice of scheduling algorithm

In deciding which algorithm to use in our packet radio network, we must seek to answer two essential questions: of the two approaches that can be taken to produce schedules, i.e. broadcast and link scheduling, is one approach a preferable choice than the other; and, assuming a given choice of scheduling, which algorithm represents a preferable choice in a mobile environment? In this subsection, we shall provide answers to these two questions in an attempt to identify that algorithm best suited to our network design.

Our evaluation process begins with a determination of design objectives which must be met by the scheduling algorithm. In our case, we establish the essential design objectives as follows:

- The algorithm must not rely on global connectivity information to produce a schedule;
- In the case of broadcast scheduling, the algorithm must produce schedules which allow each node to successfully[6] broadcast at least once in the TDMA cycle; and
- In the case of link scheduling, the algorithm must produce schedules which allow each link to be successfully activated at least once in the TDMA cycle.

These simple objectives constitute the foundation of our evaluation process. We are then faced with the problem of selecting a single algorithm from a set of alternatives. To make this selection, several performance measures can be applied. We choose to consider the following:

- Adaptivity to configuration changes, including changes in the network connectivity and the addition or removal of nodes;
- Adaptivity to varying traffic loads;
- Ability to handle both point-to-point and broadcast traffic;

[6] A successful transmission occurs when neither the transmitter and intended receiver(s) are subject to primary and secondary interference.

- Number of primary and secondary collisions;
- TDMA cycle length;
- Fairness;
- Overhead;
- Throughput; and
- Delay.

3.3.1 Link vs Broadcast scheduling

With these design objectives and performance measures in hand, we attack the problem of determining which type of scheduling, if any, represents a preferable choice. To this effect, results presented in [14] provide us with some insight. In this paper, results indicate that for both point-to-point and broadcast types of traffic, broadcast scheduling consistently offers better delay performance than does link scheduling. Though these results point us in the direction of broadcast scheduling, such a choice cannot be made on the basis of delay performance alone. Other performance measures, principally the ability to handle different types of traffic, cycle length, and throughput, must be considered in making our choice.

Some general observations can be made about each scheduling approach which may be helpful in making our selection. Consider the following points:

- Link scheduling is better suited for point-to-point traffic. In link scheduling, every node can transmit at least one distinct packet to each of its neighbors in at most one cycle since each directed link is activated at least once per cycle. On the other hand, consider a packet radio network, its associated broadcast schedule and one of its nodes u of order n. Assume that u has been allocated s slots per cycle for transmission. Then node u requires $k = \lceil \frac{n}{s} \rceil$ cycles to transmit one distinct packet to each of its neighbors. Only in the best scenario where a node is allocated at least as many slots as it has neighbors will it be able to distinctly communicate with each neighbor in a single cycle.

- Broadcast scheduling is better suited for broadcast traffic. The advantage is easy to see: in the case of broadcast scheduling, only a single transmission is required for all of a node's neighbors to receive a broadcast packet, while in the case of link scheduling, broadcast packets must be routed as separate packets to each neighbor, i.e. the number of transmissions required by the sending node equals its order.

- Link scheduling allows two neighbors to transmit in the same slot if the destination nodes are not neighbors of both sending nodes.[7] For broadcast scheduling, any neighboring nodes cannot simultaneously transmit because such transmissions result in primary interference at both of them.

- In broadcast scheduling, passive acknowledgments are possible, whereas such an acknowledgment scheme is not possible in link scheduling. A passive acknowledgment strategy is possible in broadcast scheduling because all neighbors of a sending node may receive that sending node's transmission. In the case of link scheduling, this strategy cannot apply because the acknowledgment must be transmitted over a scheduled, directed link. Thus, link scheduling requires that an explicit acknowledgment be sent for each packet.

- The cycle length of broadcast schedules is, on average, shorter than the cycle length of link schedules [8] [14].

From the perspective of being able to support both broadcast and point-to-point traffic, we can immediately say that neither broadcast or link scheduling represents a preferable choice. We have seen how broadcast and link scheduling are better suited for broadcast and point-to-point traffic, respectively. Each scheduling type, however, is capable of supporting its opposite type of traffic given an effective protocol design. Thus, if we look at just the question of being able to support both types of traffic, then we must admit that broadcast and link scheduling are equivalent.

Clearly, if the traffic were strictly point-to-point, then we should select link scheduling as that scheduling technique appropriate for our network design. The

[7] Refer to frame 5 of Figure 3.4.

[8] Refer to Figure 3.1 which illustrates this property well.

same logic can be applied to selecting broadcast scheduling for strict broadcast traffic. However, if we wish that the scheduling method adopted be able to support both types of traffic, it is because we expect to see both types of traffic transmitted. That is not to say that both types of traffic will be equally present.

It is fair to assume that in a typical packet radio network, point-to-point traffic constitutes a large portion of total traffic, with broadcast traffic representing a relatively small portion, if it exists at all, of total traffic. Why not choose link scheduling on the basis of this traffic characterization? Indeed, if the traffic to be supported is principally point-to-point, link scheduling might seem to be preferable over broadcast scheduling. Although this argument has some validity and is appealing in its simplicity, it is incomplete and somewhat misleading.

The structural appropriateness of link scheduling for packet radio network traffic is meaningless unless it translates into a significant advantage over broadcast scheduling in terms of throughput and delay. If link scheduling represents a better approach to the scheduling problem because it is better suited to support dominantly point-to-point traffic, then we expect to see a clear advantage in favor of link scheduling over broadcast scheduling in terms of throughput and delay. We ask ourselves if this is the case.

We have already briefly mentioned the results presented in [14]. We now examine them more closely. The object of this paper was to examine the delay experienced by both point-to-point and broadcast traffic in packet radio networks using link and broadcast scheduling strategies. In the case of point-to-point traffic only, results indicate that the average delay[9] experienced is shorter for networks employing broadcast scheduling. Results are similar when 10 and 20 percent of total traffic is broadcast traffic.[10] Thus, we see that despite the apparent superiority of link scheduling, broadcast scheduling translates in lower average delay for both point-to-point and broadcast types of traffic.

[9]In the case of point-to-point traffic, delay is defined as the time interval from when a packet enters the network to when it reaches its destination.

[10]In the case of broadcast traffic, delay is defined as the time interval from when the packet enters the network to when all nodes have received it.

This result is somewhat surprising when we consider that link scheduling provides more slots per cycle to each node than does broadcast scheduling. We neglect, however, to consider that the length of broadcast schedules is, on average, shorter than the length of link schedules. Thus, this shorter cycle length compensates for the additional transmission slots available to each node in link scheduling, producing better overall delay performance for broadcast scheduling.

We are left, then, with determining the advantages of broadcast scheduling over link scheduling, or vice-versa, in terms of throughput. We know of no studies which have looked at the problem from this perspective, and thus we must seek answers to the problem ourselves.

At this point, we define throughput simply as the average number of successful point-to-point transmissions which may occur per unit time. Determining which of the two scheduling approaches represents a preferable choice in terms of this measure is not a simple task. The first observation which can be made is that throughput is simply a function of the average number of transmissions scheduled in a given time frame. From the observation that link scheduling appears to allow for more simultaneous transmissions than does broadcast scheduling, we could be tempted to believe that link scheduling offers better throughput. This analysis, however, would neglect the fact that broadcast schedules are shorter than link schedules, and thus that its simultaneous transmissions can occur more often.

Consider any packet radio network and its associated link and broadcast schedules. The broadcast and link schedules are L_b and L_l slots long, respectively. Let t_s denote the slot duration, and let s_b and s_l denote the total number of transmissions allowed over the lengths of the broadcast and link schedules, respectively.[11] Broadcast and link scheduling throughputs are then given as:

$$S_b = \overline{\left(\frac{s_b}{t_s L_b}\right)} \qquad \text{and} \qquad S_l = \overline{\left(\frac{s_l}{t_s L_l}\right)} \tag{3.5}$$

[11] We assume the schedules to be maximal, and thus each node, or link, may be activated more than once in each cycle.

The precise, mathematical analysis of arbitrary networks is not feasible, and thus we choose to evaluate and compare throughputs of both scheduling strategies by simulation. A general depiction of the simulation which we implemented can be found in Figure 3.6. Broadly described, this simulation evaluates the throughputs of broadcast and link scheduling by averaging over a large set of random network configurations.

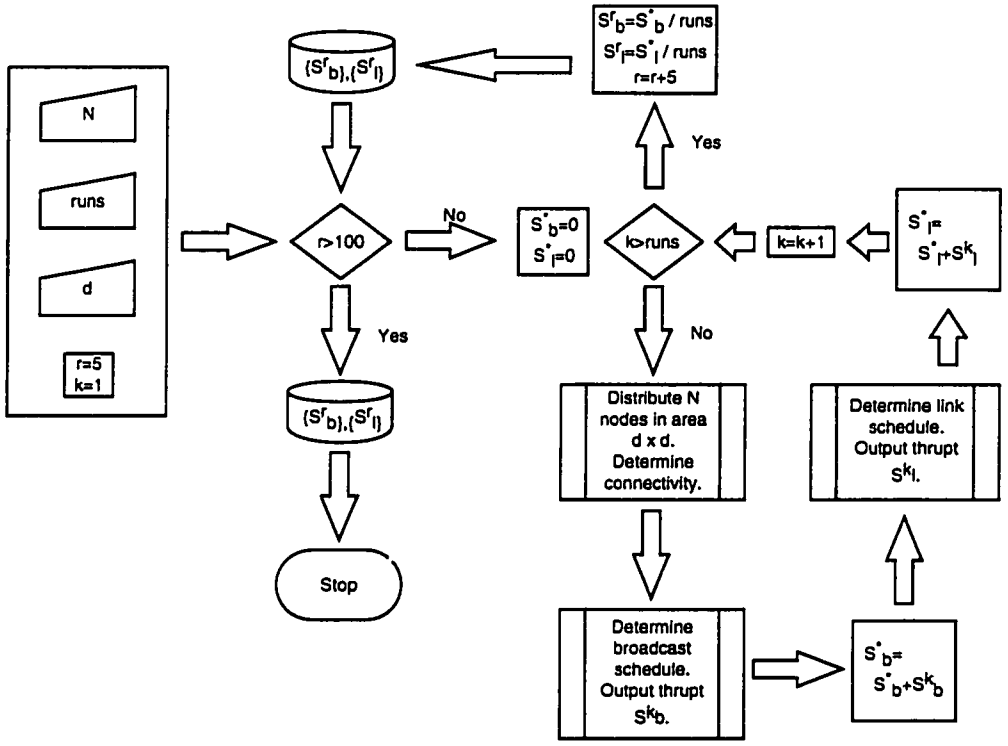


Figure 3.6: Simulation for throughput evaluation of broadcast and link scheduling.

In the simulation, N nodes are randomly (uniformly) distributed in a dxd square area. For a given radio range r, the connectivity map of the distribution is determined, where it is assumed that bidirectional links exist between any two nodes if the distance between them is less than or equal to r. Given the network topology, both maximal broadcast and link schedules are determined, allowing us

to calculate both schedule lengths and the numbers of successful transmissions which can occur over the entire schedules. The link and broadcast throughput is then calculated for this one specific distribution. For each radio range, the throughput figures thus obtained are averaged over *runs* random placements. It is important to note at this point that in the case of broadcast scheduling, any node which is isolated, i.e. any node which has no neighbors, is not scheduled for transmission.

The maximal broadcast schedule is determined as follows. Begin with the lowest numbered node not already scheduled and assign it to the current slot, then block its one-hop and two-hop neighbors from transmitting in this slot. Next, evaluate if the current broadcast mode is maximal. If it is not, schedule in the current slot the lowest numbered node not included in the broadcast mode; reevaluate the mode; and repeat until the mode is maximal. If all nodes have not been scheduled to transmit at least once, then designate a new slot and repeat the process. This algorithm is illustrated in Figure 3.7.

The maximal link schedule is similarly determined. The algorithm begins by associating each link with a unique identifier. Consider any link (u, v). This link is labelled with the unique number $linkid = (u-1)N + v$. Begin with the lowest numbered link not already scheduled and assign it to the current slot. Block all links incident on the transmitting and receiving nodes; all links incoming to the transmitting node's neighbors; and all links outgoing from the receiving node's neighbors. Evaluate if the current link activation mode is maximal. If it is not, schedule in the current slot the lowest numbered link not included in the link activation mode; reevaluate the mode; and repeat until the mode is maximal. If all links have not been scheduled for activation at least once, then designate a new slot, and repeat the process. This algorithm is depicted in Figure 3.8.

We carried out simulations for 10 and 25 nodes, each distributed in areas of 100 x 100 and 250 x 250. For the case of 10 nodes, averages were computed over 10,000 runs for radio ranges between 5 and 100 units, in steps of five units. For the case of 25 nodes, averages were computed over 2,500 runs for radio ranges between 10 and 60 units for the case of $d = 100$, and between 10 and 100 units for the case of $d = 250$ units, both in steps of 10 units. Results for $N = 10$ and 25 nodes are presented in Figures 3.9 and 3.10 respectively.

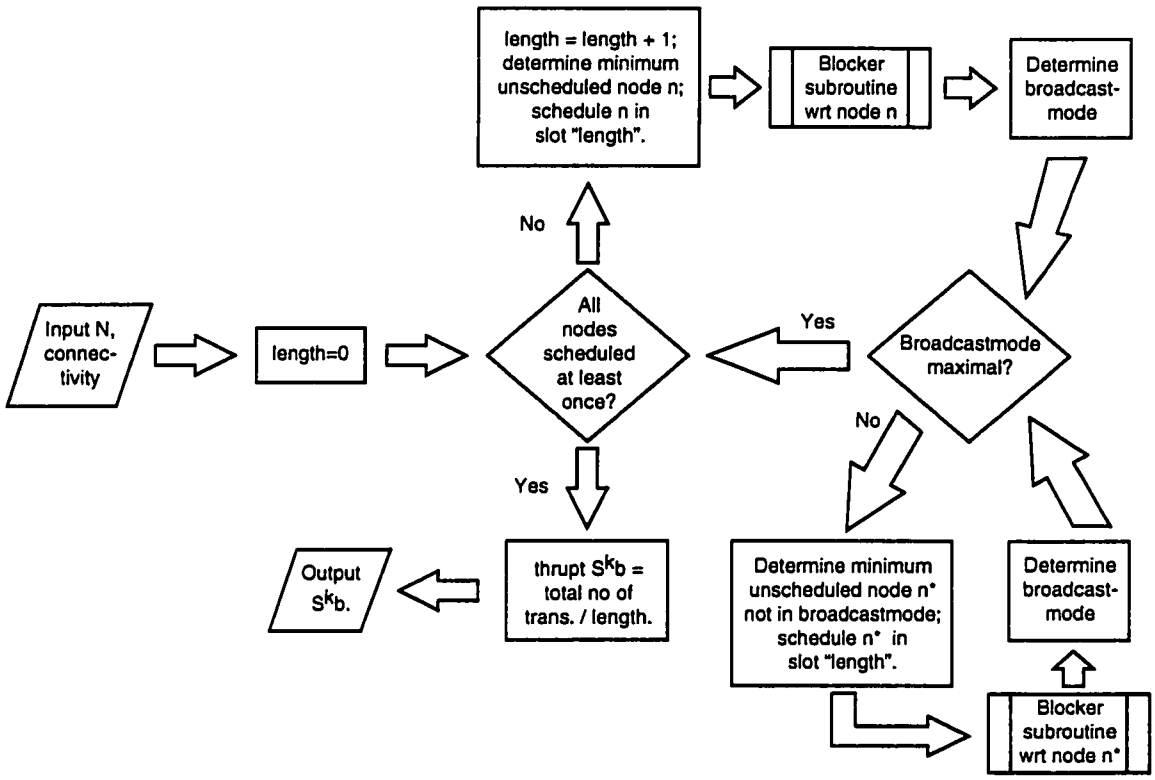


Figure 3.7: Algorithm for determining a maximal broadcast schedule.

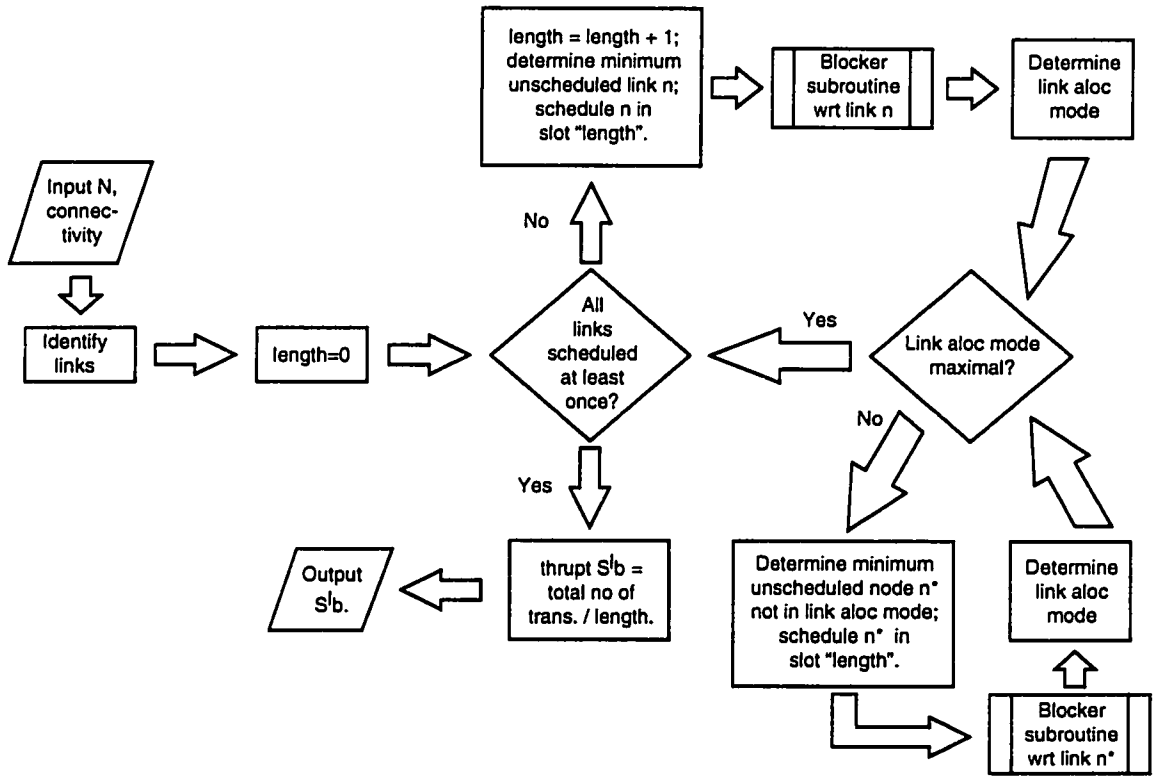


Figure 3.8: Algorithm for determining a maximal link schedule.

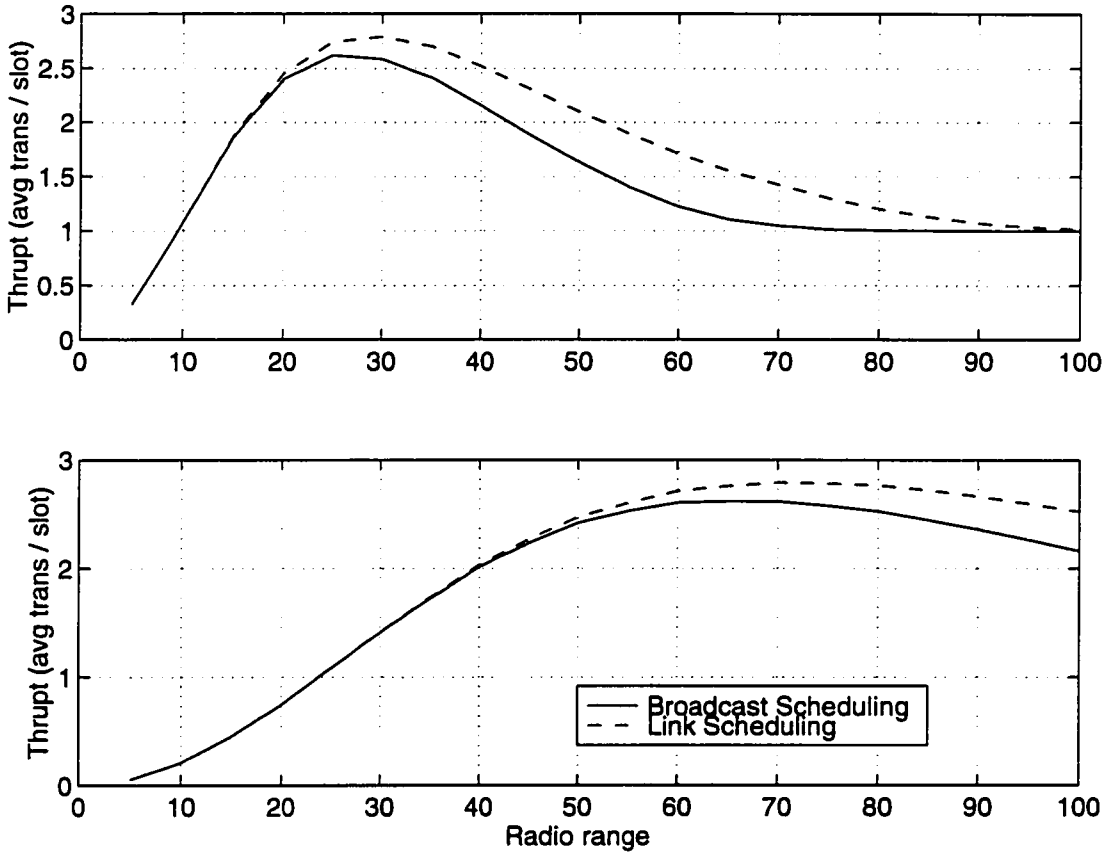


Figure 3.9: Throughput simulation results for 10 nodes. Top and bottom plots are for $d = 100$ and 250 units respectively.

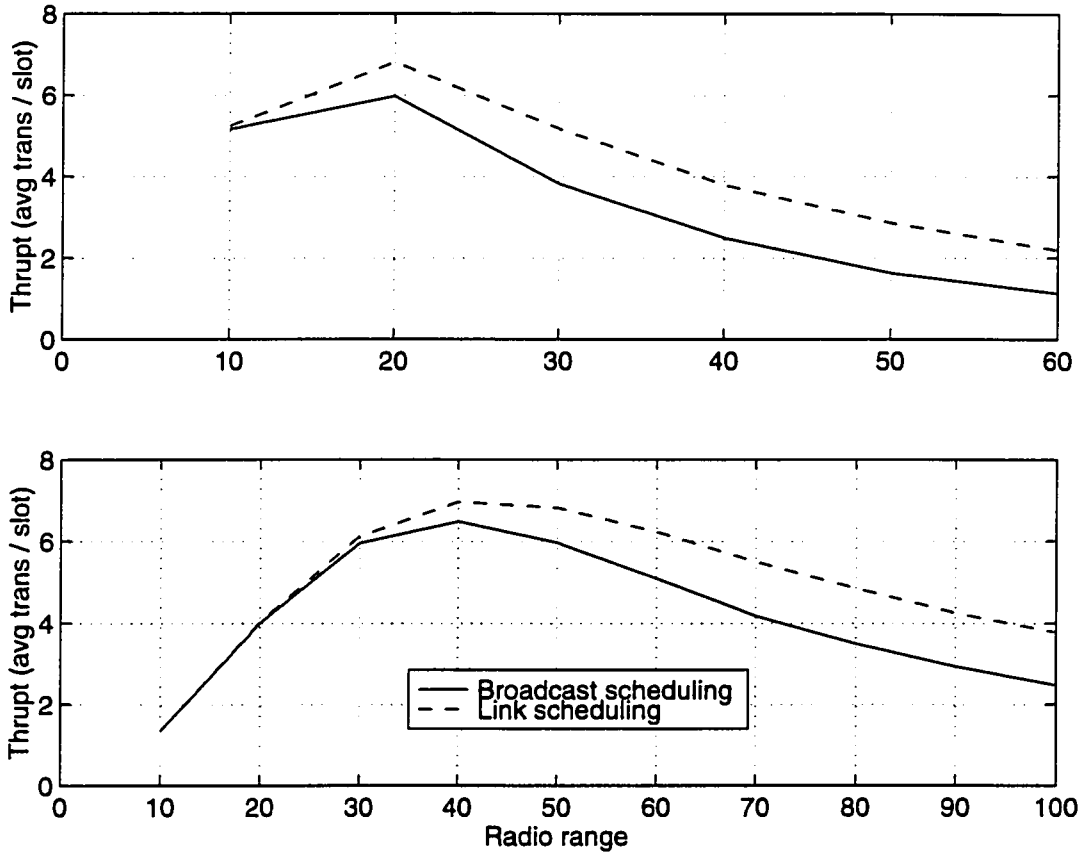


Figure 3.10: Throughput simulation results for 25 nodes. Top and bottom plots are for $d = 100$ and 250 units respectively.

Results indicate that in all cases, the throughput of link scheduling is greater than the throughput of broadcast scheduling as radio range increases. This can be explained by the fact that link scheduling allows for more simultaneous transmissions than does broadcast scheduling. We see, however, that the advantage of link scheduling is not considerable and clearly varies as a function of node density and radio range.

The general behavior of the throughput curves can be easily explained. At the low end of radio range values, nodes tend to be disconnected. The degree of connectivity will of course vary as a function of the distribution area, but it remains true that for a given distribution range, low radio ranges translate into low connectivity. Thus, at low radio range, few connections exist between nodes, resulting in poor throughput. Similarly, high radio ranges result in highly connected networks. It is therefore more difficult for nodes to simultaneously transmit since every node — or just about — is within radio range of every other node. Thus, only a single node may transmit in any given slot, i.e. the cycle length equals the number of nodes, and throughput diminishes to a value of one transmission per slot.

The advantage of link scheduling over broadcast scheduling in terms of the number of allowable transmissions is undermined by the shorter schedule length of broadcast scheduling. In order to gauge the importance of this advantage, we plotted the difference between broadcast and link scheduling in Figure 3.11.In this figure, the ratio $\frac{S_l}{S_b}$ is plotted as a function of radio range for the cases of 10 nodes distributed in an area of 100 x 100 (top plot) and 25 nodes distributed in an area of 250 x 250 (bottom plot). In the case of the top plot, we see that the maximum difference between scheduling types is of the order of 40% of broadcast scheduling throughput with a peak throughput difference lower than 10%. In the bottom plot, we see that the maximum difference is not within the range of simulated radio range values and that the peak throughput difference is also of the order of 10%.

While it is true that the exact dependence of the throughput difference on both radio range and node density requires further study, it is fair to state that the difference between the throughputs of link and broadcast scheduling is not important enough to favor link scheduling. Varying the node density only translates and spreads the general throughput curves over possible radio range

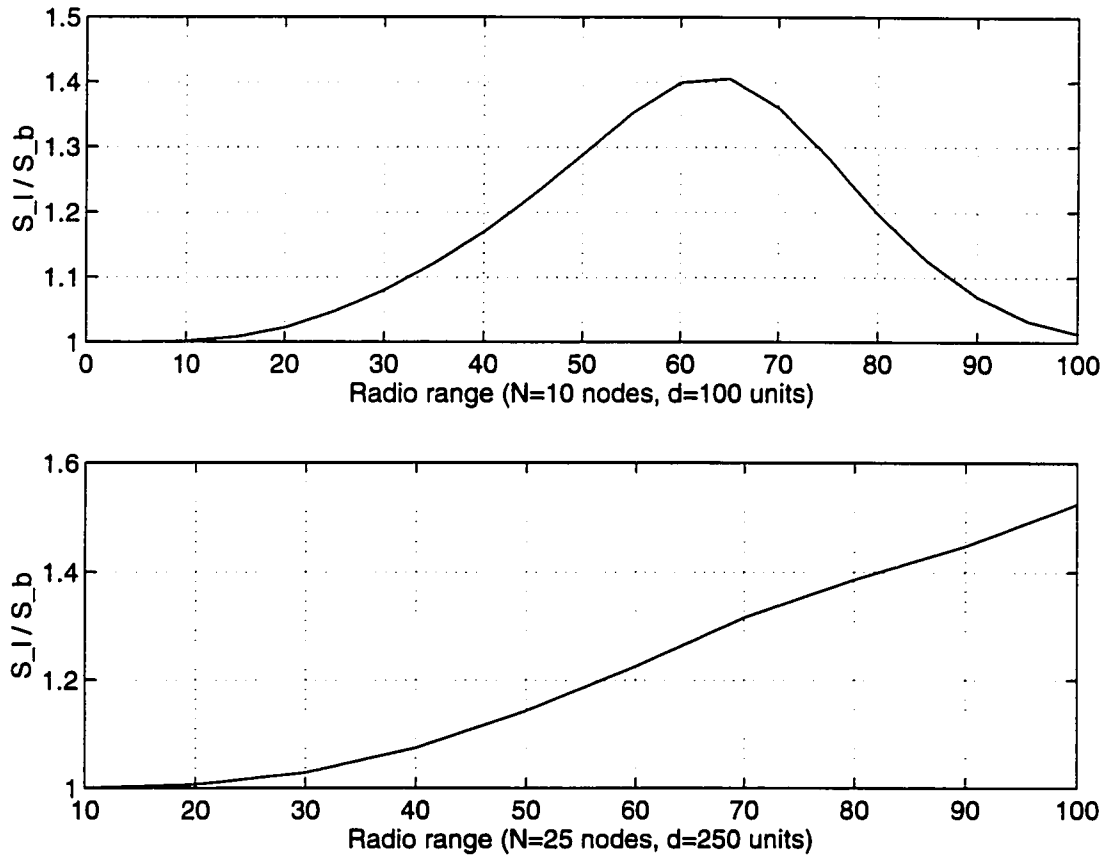


Figure 3.11: $\frac{S_l}{S_b}$ as a function of radio range.

values. It does not alter the relationship between the link and broadcast scheduling values. Thus, for a given node density, it is reasonable to say that for some intermediate value of radio range, throughput will peak to some value, with the difference between scheduling types being of the order of 10% of broadcast scheduling throughput, in favor of link scheduling. After having reached this peak throughput value, the two curves will both decline to a throughput value of one for some value of radio range which could be less than or greater than the upper values reviewed in our plots. For radio range values between the peak value and the value where throughputs equal 1, the broadcast scheduling curve will decline more steeply than the link scheduling curve. Thus, in this interval. the difference between broadcast and link scheduling, expressed as a percentage of broadcast scheduling throughput, will continue to increase[12]; will peak to some value; and will then decrease as both throughput curves converge to a value of 1. It is important to note at this point that the desirable value of transmission range is that which maximizes throughput, not that which maximizes the difference between the scheduling types.

We see, then, that the advantage of link scheduling over broadcast scheduling is of the order of 10%. This advantage is quickly lost when we consider that for the case of link scheduling, half of all throughput is used for acknowledgments. In effect, *effective* throughput for link scheduling is only half of that throughput plotted in Figures 3.9 and 3.10. On the other hand, effective throughput for broadcast scheduling is only slightly lower than its simple throughput since it may use a passive acknowledgment scheme, and active acknowledgments are required in only a small fraction of all transmissions.

In summary, we have seen that both link and broadcast scheduling are similar in terms of the ability to handle point-to-point and broadcast traffic. Broadcast scheduling, however, is superior to link scheduling in terms of TDMA cycle length, delay, and effective throughput. Thus, we choose broadcast scheduling as that type of scheduling appropriate for our packet radio network design.

[12]The difference continues to increase even though the actual throughput values are decreasing

3.3.2 Choosing a broadcast scheduling algorithm

We have selected the broadcast scheduling approach for our packet radio network design. We now attack the problem of selecting a specific broadcast scheduling algorithm. The problem of broadcast scheduling has been widely studied, and many algorithms exist which provide alternative solutions to the problem [13, 16, 17, 19, 20]. These algorithms span a period of over 10 years of research and development in the area of broadcast scheduling, and thus they largely vary in both the approaches they take to solving the problem and the results they produce.

We do not wish to compare each of these algorithms and have thus identified two which represent the most interesting alternatives for incorporation into our network design. These two algorithms are Ephremides' and Truong's broadcast scheduling algorithm [16, 17], which we shall label the *ETBSA*, and Chlamtac's and Farago's *Galois Radio Network Design (GRAND)* algorithm [20].

We cannot exclude the other two algorithms, however, without briefly explaining why they offer less desirable alternatives than the two chosen algorithms. The Transmission Scheduling Algorithm (TSA) proposed in [19] was conceived as a method for constructing broadcast schedules for linked cluster networks. The basic concept is to maintain two sets of parallel schedules, one network-wide fixed TDMA schedule, and a separate set of transmission schedules for each cluster. Since there are generally fewer clusters than there are nodes in the network, it is hoped that the amount of interference resulting from the overlap of the network-wide and cluster schedules will be kept small. This algorithm's main deficiency is that it cannot guarantee interference-free transmissions but rather seeks to minimize interference, thus maximizing the number of valid transmissions. The ETBSA, on the other hand, guarantees interference-free schedules. It is therefore preferred over the TSA.

Cidon's and Sidi's broadcast scheduling algorithm is described in [13]. The idea behind the algorithm is to split the radio channel into control and transmission sub-channels. The control subchannel is used to avoid conflicts amongst nodes and to increase the utilization of the transmission segment. Its main deficiency is that transmission rights are allocated to any given node on a slot by slot basis, resulting in both unfair schedules and high overhead. These

characteristics contrast with the ETBSA which relies, as we shall soon see, on a deterministic access to the control channel, and which produces better schedules with less overhead.

It is interesting to compare the performance of the ETBSA to the GRAND algorithm because the algorithms seek to implement two very different broadcast scheduling schemes. We briefly pause at this point to review these schemes before proceeding to the descriptions of the algorithms. The ETBSA constitutes an implementation of what we could call the *topology-dependent* approach to broadcast scheduling, in contrast to the *topology-independent* scheme implemented in the GRAND algorithm. The topology-dependent scheme is principally characterized by two distinct properties. Firstly, the basis for the broadcast schedule computation is knowledge, be it complete or partial, at any given node of the network connectivity map. Secondly, the topology-dependent approach builds on knowledge of the network connectivity to construct a schedule which is interference-free in every slot of the TDMA cycle. In comparison, the topology-independent approach seeks to construct a schedule based not on the network connectivity but rather on global network parameters such as the number of nodes in the network and the maximum degree [13] of all the network's nodes. Furthermore, the fundamental idea behind the topology-independent approach is to weaken the non-interference criteria of the topology-dependent approach such that nodes be guaranteed at least one interference-free transmission in the TDMA cycle, instead of imposing a non-interference condition in every slot of the cycle.

For the case of mobile networks, the topology-dependent approach suffers from major inconveniences which should make us think twice before adopting and implementing it. Because the topology-dependent approach relies on the network connectivity to construct its schedule, it is very vulnerable in a mobile environment. Because the schedule must be recomputed after each topology change, significant overhead is incurred. Furthermore, if the connectivity changes faster than the schedules can be computed, then the scheduling algorithms will be unable to construct valid schedules. It would thus appear that the topology-independent approach is better suited for the mobile environment.

[13] Maximum degree is the the highest number of neighbors associated to any single node, for all the nodes of the network.

While it is true that the topology-independent approach provides significant advantages over the topology-dependent approach in terms of robustness under nodal mobility and incurred overhead, weakening the non-interference criteria has the potential of translating into higher TDMA cycle lengths, higher delay, and lower throughput. Specifically, the basic premise behind the topology-independent approach is that interference-free transmissions in every slot of the TDMA cycle is not a necessary condition. Instead, it is proposed that a more appropriate, less restrictive, condition would be that for every node, there exists at least one slot in the cycle where each node may successfully transmit to any of its neighbors. Thus, schedule robustness is gained at the expense of increased interference, potentially lower throughput, and higher cycle lengths and delay. We ask ourselves if this cost is important enough to justify implementation of the topology-dependent approach, despite its clear disadvantages. We shall attempt to answer this question by first fully describing each algorithm and then comparing them to each other in terms of adaptivity to varying traffic loads, overhead, fairness, cycle length, and throughput.

The ETBSA is a fully distributed, topology-dependent algorithm and is considered by some to be the "state of the art" [21] algorithm for effectively computing broadcast schedules consistent with the maximal broadcast scheduling problem described earlier. It may be characterized by its simple and distributed implementation and is fully described and demonstrated in [16] and [17]. Here, we are satisfied with a summary description of the algorithm. Consider a N x N matrix S, where each row corresponds to a time slot and each column corresponds to a node. Thus, the entry $S(i, j)$ indicates the status of node j during slot i. This status must take either a *reserved* status, indicating a node is scheduled to transmit in that slot; a *blocked* status, indicating a node may not be scheduled to transmit in that slot; and an *unassigned* status, indicating the node has not yet been scheduled to transmit or has not been blocked.

The algorithm starts with a skeleton schedule for which the ith slot is reserved for the ith node, i.e. the elements of the diagonal of S are all reserved. Given an appropriate mechanism for each node to determine its two-hop connectivity, each node may determine the blocked slots in its column because these slots cannot be assigned to either its one-hop or two-hop neighbors. The remaining slots in the column remain initially unassigned. Each node is also capable of partially determining the column entries of each node in its broadcasting zone. No single

node, however, is capable of fully determining the broadcast schedule.

At this point, each node has initialized its version of the schedule, and the problem narrows down to scheduling transmissions in unassigned slots. This assignment is achieved by having each node transmit some control information in their reserved slots. Specifically, the algorithm requires a sequence of two TDMA cycles, the first of which is used by each node to transmit its schedule column; and the second of which is used to broadcast the columns of each node's neighbors. We refer the reader to the references for an exact description and a complete example of how this assignment process is implemented. Suffice it to say that the process results in a maximal slot assignment of fixed cycle length N. We shall soon consider its principal characteristics when comparing them to those of the GRAND algorithm.

The GRAND algorithm is an innovative, topology-independent algorithm which can be used to construct a broadcast schedule based strictly on the number of nodes in the network and the maximum degree of all nodes. It does not seek to provide a solution to the maximal broadcast scheduling problem but instead seeks to ensure that every node in the network gets at least one slot in which it can successfully transmit to one or more of its neighbors [20]. The broadcast scheduling problem then becomes a problem of determining a subset of slots in which any given node may transmit such that the weaker non-interference criteria may be satisfied.

Chlamtac and Farago provide a solution to this problem based on the properties of Galois fields. Let $GF(q)$ be some representation of the Galois field of order q. Consider a set of N distinct polynomials over $GF(q)$, where each polynomial P_i is associated to a node i. The *graph* of each polynomial can be represented as some subset of all pairs $\beta, P_i(\beta)$, whereβ is an element of $GF(q)$, and the value of P_β is also in $GF(q)$. If we assume that the degree of the polynomials is at most k, then the graphs of the polynomials cannot have more than k points in common. Thus, a one-to-one correspondence between these graphs and the time slots of a broadcast schedule will yield a schedule where each node is allocated q slots, and no two of them will have more than k common slots. In other words, if the parameters are properly chosen, then no subset of all possible slots can be entirely covered by the subsets of any node's one-hop and two-hop neighbors, i.e. at least one slot exists for any given node in which no

interference will occur. We shall not, at this point, enter into the details of how the proper values of q and k may be computed. Suffice it to say that the computation is based on the number of nodes in the network and the maximum degree of all nodes.

The fundamental difference between the two algorithms should be clear at this point. In the case of the ETBSA, the algorithm dynamically allocates slots to the nodes based on partial knowledge of the network connectivity. Thus, all types of interference are eliminated in any given slot of the TDMA cycle. In the case of the GRAND algorithm, slots are pre-assigned to each node on the basis of assigned polynomials derived from global parameters. Thus, the specific slot in the frame in which a node's transmission will be successful is not known in advance. It is only guaranteed, and its location in the cycle will depend on the actual, dynamic connectivity.

We are interested in comparing each algorithm in terms of some of the more important criteria listed at the beginning of this section. Particularly, the algorithms can be easily compared in terms of adaptivity to configuration changes, adaptivity to varying traffic loads, fairness, overhead, cycle length, and throughput. Comparisons for the first four measures are summarized below:

- **Adaptivity to configuration changes** - Both algorithms provide adaptivity to configuration changes, with the GRAND algorithm representing a more robust solution than the ETBSA. Because the ETBSA represents a topology-dependent solution, topology changes translate into having to recompute the schedule after some period of time. This recomputation process may fail in the event that the network topology changes faster than the rate at which schedules may be computed and distributed. On the other hand, the GRAND algorithm is independent of topology. The predetermined schedule thus remains valid despite any changes that may occur in the network connectivity.

- **Adaptivity to varying traffic loads** - The ETBSA provides no adaptivity to varying traffic loads while the GRAND algorithm is inherently traffic adaptive. In the case of the ETBSA, the schedule is computed strictly on the basis of the network connectivity map. While this schedule is maximal in the sense that it maximizes point-to-point throughput, it offers absolutely no guarantees that the schedule is consistent with the traffic

loads in the network. Furthermore, because the schedule, once computed, is fixed in time, the ETBSA cannot dynamically reallocate time slots to different nodes on the basis of varying traffic loads. In contrast, the GRAND algorithm is inherently traffic adaptive because heavily loaded nodes can successfully transmit more than once in each cycle if their neighbors are lightly loaded.

- **Fairness** - The advantage of the GRAND algorithm over the ETBSA, or vice-versa, is not as clear as in the previous cases. We say that any schedule is *unfair* if the spread in the number of slots given to any two particular nodes is large. The schedules produced by the ETBSA are known to be unfair [16, 17, 22]. In the case of the GRAND algorithm, we can argue that it is fair in the sense that all nodes are allocated the same number of slots [20]. While this is indeed the case, it would be a mistake to go a step further and argue that each node is given the same opportunity for successful transmission. While it is true that each node is guaranteed at least one slot per cycle in which it can successfully transmit, some nodes may, depending on the exact topology and traffic load at any point in time, be provided with more or less good slots in which to transmit.

- **Overhead** - The requirement for schedule recomputation inevitably translates into significant overhead due to the exchange of control messages and the complete or partial redistribution of the schedule. This overhead requirement simply does not exist for the case of topology-independent scheduling schemes.

At this point, the advantage is clearly in favor of the GRAND algorithm over the ETBSA. Before finalizing our choice, however, we wish to consider a comparison of the two algorithms in terms of cycle length and throughput. We begin with an analysis of cycle length.

The cycle length of the ETBSA is simply the number of nodes in the network. We have seen that the broadcast schedule of the ETBSA stems from a skeleton schedule of length N. The algorithm does not seek to shorten or lengthen this initial schedule but rather seeks to maximize throughput in each of the N slots. Thus, the cycle length is fixed at N. The case for the GRAND algorithm is not so simple where the cycle length of the schedule is given as $L = q^2$, where q represents the order of the Galois field over which the scheduling polynomials

are generated. We have mentioned that the appropriate value of q will depend on the N and on the maximum degree of all nodes. Thus, the cycle length is also a function of these two parameters.

The cycle lengths of the GRAND schedules are depicted in Figure 3.12 for $N = 25$, 100, 500, and 1000 nodes as a function of maximum degree. These plots also include the cycle length of the ETBSA for each case, represented by a dashed line equal to the number of nodes for all values of maximum degree.[14] From these plots, we see that the cycle length of the GRAND algorithm is considerably shorter than that of the ETBSA under conditions of low connectivity. More specifically, we see that in all cases, the GRAND algorithm has a clear advantage over the ETBSA for small values of maximum degree until it eventually surpasses the cycle length of the ETBSA for maximum degree values greater than some d_{max}. We further see that the value of d_{max} increases with an increasing number of nodes. Thus, for example, d_{max} is equal to 4 for $N = 25$ nodes, while d_{max} is equal to 9 for $N = 500$ nodes.

We assume at this point that the proper connectivity conditions can be achieved to justify the use of the GRAND algorithm. We shall discuss the validity of this assumption in Chapter 5. For the time being, making this assumption allows us to further proceed in our comparison and consider the case of throughput. We consider the case for 100 nodes. For this case, we see that, given the maximum degree of any given node does not exceed $d_{max} = 4$, then we are justified — in terms of cycle length, at least — in preferring the GRAND algorithm over the ETBSA. We further assume that no node is isolated. In such a case, the ETBSA will yield a schedule where 32 to 34 percent of all slots are used for successful transmissions between any node pairs [16, 17]. The number of slots[15] is given as N^2. If we consider the best case scenario where 34% of all slots are used, then the throughput of the ETBSA, for the case of 100 nodes, can be calculated to be $S_{ETBSA} = \frac{.34N^2}{N} = (.34)(100)$ =34 successful transmissions / slot.

In the case of the GRAND algorithm, we must be content with calculating a lower bound on the throughput since the exact number of successful transmissions in any given cycle will depend on both the network configuration

[14]The cycle length is fixed to the number of nodes regardless of the degree of any one of them.

[15]N nodes may transmit in N possible slots yielding a total of N^2 possible transmission slots.

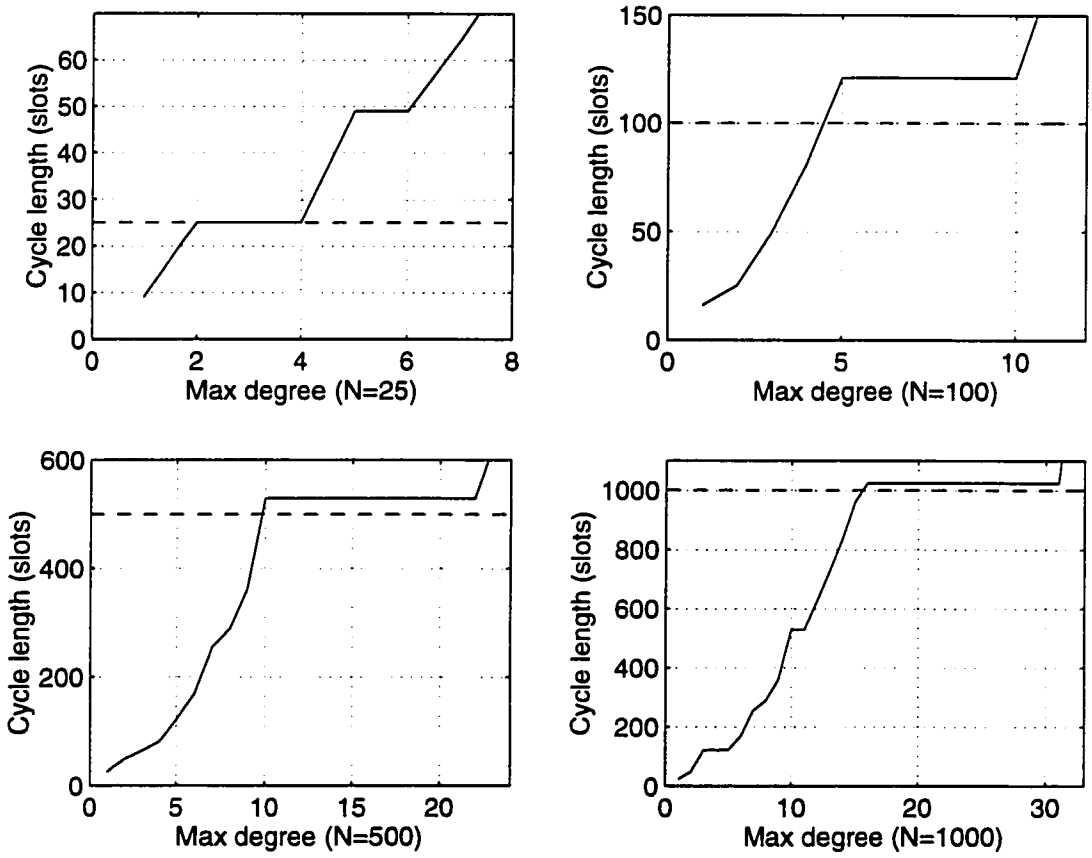


Figure 3.12: Cycle lengths of the ETBSA and GRAND algorithm for 25, 100, 500, and 1000 nodes. Dashed lines represent cycle length of the ETBSA.

and the traffic load. The GRAND algorithm guarantees that every node in the network will successfully transmit at least once in the TDMA cycle. Thus, for a given value of maximum degree, it is possible to calculate the minimum throughput to simply be $\frac{N}{q^2}$. Throughput values thus obtained for the case of $N = 100$ nodes are plotted in Figure 3.13.

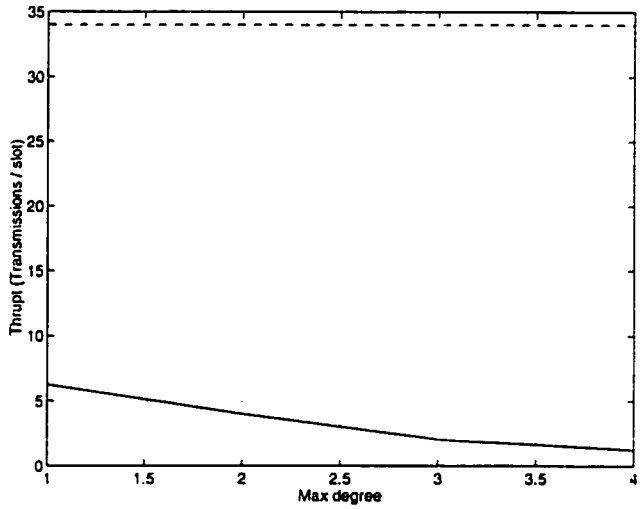


Figure 3.13: Throughputs of the ETBSA and GRAND algorithm for 100 nodes. Throughput of the ETBSA is represented by the flat, dashed line

We see from this plot that a large discrepancy exists between the throughput of the ETBSA when compared to the minimum throughput of the GRAND algorithm. This result is easily explained by the fact that the ETBSA is a maximal scheduling algorithm whose primary objective is to maximize throughput for every slot in the cycle through the strict control of interference. It should therefore not be surprising to see how it yields superior throughput than the minimum guaranteed throughput of the GRAND algorithm. That is not to say that the GRAND algorithm cannot do much better than the minimum throughput guaranteed by its lower bound.

We suspect that if we apply the GRAND algorithm to tag each node with a unique scheduling polynomial, and then randomly distribute the nodes in a fixed

area, then, on average, the throughput of the GRAND algorithm will be appreciably higher than its lower bound while remaining lower than the throughput of the ETBSA. We justify this expectation by the fact that each node will, on average, be able to successfully use more than a single slot for transmission, although the exact amount will vary as a function of the nodes' distribution. We suggest that the cumulative effect of these extra successful transmissions will result in higher overall throughput than the lower bound. Further simulation is required to both confirm this hypothesis and quantify the throughput gain. In any case, we have seen that the GRAND algorithm is superior to the ETBSA in all respects except throughput. Though throughput does represent an important measure of performance, we believe that the cumulative advantage of the GRAND algorithm over the ETBSA, particularly in terms of its innate robustness and adaptivity, make the GRAND algorithm a better choice for our packet radio network. Thus, we choose the GRAND algorithm as that algorithm appropriate for our packet radio network design.

Chapter 4

Network Management

The discussion and analysis of the previous chapter has centered on how two neighbor nodes may communicate over a common radio channel. This discussion has essentially been one of point-to-point communications. The next set of issues pertains to methods for effectively moving packets through the entire network from a source node to a destination. Thus, issues of network management translate into a problem of end-to-end communications.

The fundamental function of the network management algorithms is to efficiently and reliably circulate packets through the network from source to destination. This function can be decomposed into two basic tasks: the establishment of routes (routing); and the forwarding of packets along those routes (packet forwarding) [4]. In this chapter, we shall only consider the routing problem.

The routing problem can be simply stated: given a network connectivity map, which path, if its exists, must a packet travel as it proceeds from a source to destination through a set of intermediate nodes? The design choices that must be made in solving this problem essentially fall into two areas. Firstly, we ask ourselves what type of routing and routing algorithm should be used, and secondly, we must determine how the routing information should be

disseminated to the nodes. We begin our discussion by considering the routing problem in the specific context of packet radio networking.

4.1 Routing in Packet Radio Networks

Routing is a complex and crucial problem common to all types of networks. The unique environment of radio networks, however, evokes a particular set of conditions which, in turn, gives rise to design choices largely different from those of other networks. Specifically, three factors have significant impact: a more or less dynamic network topology; a shared broadcast channel; and a paramount requirement for efficiency. As we have seen, the connectivity picture of a packet radio network will vary as a function of nodal mobility and radio propagation conditions. This configuration variability translates into the requirement for more or less adaptive routing schemes which determine and update routes at least as fast as the topology changes. Secondly, use of a broadcast channel can allow some nodes to benefit from the availability of broadcast information and thus improve the overall routing scheme. Finally, routing algorithms typically generate large amounts of overhead traffic, particularly in the case routing schemes for packet radio networks where algorithmic robustness is a prime consideration. In the context of packet radio networks where limited bandwidth is an omnipresent concern, the efficiency of the routing algorithm becomes a very important design consideration.

Methods for routing packets can be categorized into two broad categories: *flooding*, and *point-to-point* techniques. Flooding refers to that approach where any given packet is transmitted to all nodes in the network. In contrast, point-to-point methods are based on associating each source-destination pair with a route. These last methods can be further characterized as being of either the *connectionless* (datagram) or *connection oriented* (virtual circuit) types.

Flooding is well suited to applications where reliable delivery of packets is a requirement in the presence of uncertain connectivity or a rapidly changing topology. Its main advantages include small overhead requirements, little requirement for network management, and ease of distributed implementation.

Flooding methods, however, tend to utilize the network resources inefficiently since every node is required to receive every packet [4].

In the connectionless approach, each packet is routed independently, with no reference to any packets that have been previously transmitted. Typically, this approach involves the background execution of routing functions at the node level with no specific knowledge of the existence of end-to-end connectivity. In contrast, connection oriented routing schemes require that a fixed, predetermined path be identified before any transmissions are allowed, i.e. end-to-end connectivity is a requirement for transmission. The basic difference between this approach and the connectionless approach is that in the connection oriented approach, nodes do not need to make a routing decision for each packet. Instead, the routing decision is made initially and maintained as long as a virtual circuit is being used.

Connection oriented approaches typically require little routing overhead and thus lead to better utilization of the radio channel. The ability of connectionless routing to locally adapt to connectivity changes is desirable in networks where topology changes rapidly. Thus, we see that the different routing schemes are complementary, each presenting different advantages as a function of the network operating environment. In relatively static or slow-changing networks, it is usually most efficient to implement a connection oriented routing scheme, whereas for more dynamic networks, connectionless routing schemes represent a preferable alternative. Finally, in the most dynamic networks where the rapid rate of change of the network topology precludes the establishment of routes, flooding constitutes the most reasonable approach [4].

We have so far discussed the broad characterization of routing schemes in the context of mobile packet radio networks. We now continue by considering the problem of how to diffuse this routing information to the nodes. For any type of routing scheme, local connectivity information must be processed and distributed to the nodes such that they may forward their packets as required. With respect to this distribution problem, three approaches may be implemented: centralized, decentralized, and distributed routing.

Centralized routing involves the concentration of connectivity information to a unique routing server which uses complete knowledge of the network

connectivity map to compute routes. Routing information is then redistributed to the nodes who may then process and forward packets as required. Because the routing server has knowledge of the entire network connectivity map, it may be very effective in the computation of routes. Relying on a central server, however, presents major disadvantages which do not necessarily make it an attractive alternative. The most important of these disadvantages is that centralized routing is relatively ineffective for cases of rapidly changing topologies. Because the connectivity information must be uploaded to the routing server, and then the routing information must be dowloaded to the individual nodes, this approach is inherently limited in how it can adapt to local change. Furthermore, the exchange of information between nodes and the server necessarily translates into significant overhead. Finally, reliance on a central router is contrary to our initial objective of implementing a fully distributed system. Thus, we shall not consider this alternative any further.

In decentralized routing, each node determines its connectivity and then floods this information to all other nodes in the network. These connectivity updates provide each node with the information to locally compute routes to any other node. In essence, each node becomes its own routing server. While this approach is more adaptive to local changes in topology and is more robust than the centralized scheme, the amount of overhead traffic it generates is appreciable and will increase as a function of increasing rate of changes in topology [4]. Thus, this method, though more attractive than the centralized method, is best-suited for networks where the topology is static or slowly changing.

In distributed routing, each node simply broadcasts to all of its neighbors the value of some cost metric for each destination in the network. Neighboring nodes can then determine, based on the values of the metrics that it hears from all its neighboring nodes, which node it must transmit to en route to a given destination at minimum cost. This technique is inherently well-suited to manage local changes in connectivity. Furthermore, this has been shown to yield appreciably less overhead than the decentralized approach. However, explicit mechanisms required to overcome the robustness problems of distributed routing schemes — principally loop routes and lengthy convergence times — generate some overhead. Thus, recent studies of robust distributed routing algorithms indicate that the amount of overhead generated by these algorithms is comparable to that of decentralized routing techniques [23]. In other words, comparable algorithm

robustness translates into comparable overhead costs. Given that distributed routing is better suited for the dynamic connectivity environment of mobile radio networks, and given that this approach coincides well with our desire to implement a fully distributed packet radio network, we propose that distributed routing serve as the basis of the routing scheme for our packet radio network.

4.2 Two Useful Routing Schemes

The routing requirements of a mobile packet radio network for multimedia communications are very broad and touch on practically all elements that we have so far discussed. Multimedia traffic includes both real time traffic and data. For the case of real time traffic, specific bandwidth and delay must be guaranteed during the transmission period, and this requirement is best met through a connection oriented routing approach. Similarly, data, because of its inherently bursty nature, is best carried in a connectionless routing mode. Thus, we see that our routing scheme must include both types of point-to-point routing schemes. In addition to point-to-point routing, we must provide for a broadcast capability through an effective flooding scheme. Finally, implementation of the scheme must be achieved on the basis of a distributed implementation. We are interested in devising a routing scheme which will address all of these requirements in as effective a fashion as possible and in a manner which builds on the characteristics of our network architecture.

From the perspective of routing, the linked cluster architecture presents interesting characteristics on which we can possibly base our routing scheme. The linked cluster architecture is inherently well-suited to facilitate the development of an effective routing scheme in at least three ways. Firstly, the hierarchical nature of the architecture lends itself well to hierarchical routing schemes which, particularly for the case of large networks, are more bandwidth-efficient than comparable point-to-point routing schemes. Secondly, because clusterheads are connected to every node in their cluster, they may act as broadcast centers for the purposes of a flooding mechanism. If we go one step further and assume that the clusterheads are fully aware of the connectivity for every node in their cluster, i.e. they not only know that they are connected to their nodes but are also aware of how these nodes, within the cluster, are

interconnected, then it is also possible for the clusterhead to act as a routing server. Thirdly, the linked cluster architecture defines a backbone network consisting of the clusterheads, gateways, and links which connect them, over which inter-cluster traffic may be concentrated and managed [8].

At this point, we have briefly discussed those general properties of the linked cluster architecture which could be of benefit in the elaboration of our routing scheme. We now continue by considering two classes of routing schemes which are particularly relevant in the context of our packet radio network design.

4.2.1 Distance-Vector Routing

The general class of distance-vector routing algorithms generate routes that are optimal with respect to some route cost metric. The basic cost metric is simply the number of hops but may be defined as a function of a variety of factors including monetary cost, delay, throughput, or error rate [24, 25]. In the simplest scenario, the total route cost is an additive function of the individual constituent link costs.

The basic implementation of a distributed, vector-length routing algorithm is known as the Bellman-Ford algorithm. Consider a packet radio network with N nodes represented by its directed graph $G = (V, E)$. Let the optimal route cost for the route, of no more than h links, going from source i to destination j be represented as $D^h_{i,j}$. Furthermore, let $d_{i,j}$ represent the cost of any point-to-point link between the transmitting node i and receiving node j. In this case, any two neighbor nodes i and j will have $d_{i,j} \geq 0$, while, by definition, $d_{i,i} = 0$ and $d_{i,j} = \infty$ when no directional link exists from node i to node j. Finally, we let h represent the maximum number of links in a route at the current iteration of the algorithm. The Bellman-Ford algorithm is then given as follows [24, 25]:

1. Initialize $D_{i,j}$ as follows:
   - $D^0_{i,j} = \infty$ for all $i \neq j$
   - $D^h_{i,i} = 0$ for all h
2. For each $h \geq 0$
   - $D^{h+1}_{i,j} = \min_k(d_{i,k} + D^h_{k,j})$ for $i \neq j$

The principal advantage of this simple algorithm is that it readily lends itself to a distributed implementation. We may imagine a scheme where in any given iteration, each node k transmits to each of its neighbors i an estimate of its route costs $D_{k,j}$. If we assume that each node i knows of the link cost $d_{i,k}$, then node i may, after having received estimates from all of its neighbors, compute estimates of $D_{i,j}$ through any neighbor k and select the smallest one.

This implementation scheme is the basis for Routing Information Protocols (RIPs). In this family of routing protocols, each node maintains a routing table in which it stores the cost of the least-cost route from itself to any other destination node in the network and the address of the neighbor node en-route to the destination. This information is replaced when a lower-cost route is discovered or when the currently stored route has failed.

The general algorithm that we have so far discussed will converge to a set of routes which are optimal with respect to some static cost metric. In the case of mobile packet radio networks, the link metric is likely to change as nodes move in and out of each other's radio range and as the propagation conditions vary over time. It is thus essential that some mechanisms exist to maintain the routing table such that each table reflects the network topology as accurately as possible while averting the two important problems of counting to infinity and loop routing. We do not wish to enter into the details of these mechanisms at this point. It is important to underline, however, that these maintenance mechanisms play a critical role in the context of mobile radio networks and that they must be an integral part of the overall routing scheme.

4.2.2 Hierarchical Routing

At first glance, hierarchical routing would appear to be a routing scheme of choice in light of the natural hierarchical structure of the linked cluster architecture. In general, we may state that this type of routing scheme is best suited for large networks where other types of routing, such as strict vector-length routing, would generate large amounts of overhead and thus capture large portions of overall network bandwidth. In the context of large, linked cluster packet radio networks, we wish to investigate how hierarchic routing schemes can be laid over the linked cluster architecture to yield a functional and effective

routing scheme. This investigation must necessarily begin with a basic understanding of hierarchical routing and hierarchical routing algorithms.

The basic objective of hierarchical routing is to reduce routing overhead by essentially hiding details of the network's topology. This is achieved by grouping nodes into clusters, clusters into superclusters, and so on.[1] Each node in the network is given a hierarchical address composed of its individual address and the address of the cluster to which it belongs. Routing decisions are then executed on the basis of this hierarchical address. Specifically, if the communicating nodes belong to different clusters, then routing decisions are taken on the basis of intercluster routes. In the case where both nodes are in the same cluster, then routing is performed on the basis of an intracluster routing mechanism. Hierarchic routing schemes thus require that each node be able to determine the hierarchical address of each possible destination [23].

Therein lies the fundamental problem of hierarchical routing in packet radio networks. In static networks, the hierarchy of the network may be administratively defined. Clearly, this approach cannot be taken for packet radio networks where, as we have seen in chapter 2, the hierarchy may be highly dynamic. In this case, address determination also becomes a highly dynamic process requiring a significant exchange of information between nodes and the associated overhead. This requirement is clearly in opposition to our initial objective of reducing routing overhead by hiding configuration information inside the cluster structure. Hence, an efficient hierarchical routing scheme is characterized by a lower overall overhead requirement than an equivalent flat routing scheme.

Hierarchical routing schemes can be divided in two types: quasi-hierarchic and strict hierarchic routing. Quasi-hierarchic routing constitutes an extension of vector-length routing where at some point in time, each node transmits its cost to other nodes in its cluster and the cost to get to another cluster. Using this information, each node may then compute the minimum-cost route to other nodes in its cluster and the minimum-cost route to other clusters in the network. Thus, quasi-hierarchic routing directs packets along the least-cost intercluster route and, once in the destination cluster, along the least-cost internodal path.

[1] We shall limit ourselves to a two-level hierarchy consisting of nodes and clusters.

In strict-hierarchic routing, the clusterheads of each cluster cooperate to compute hierarchic routing tables (HRT) which specify the intermediate clusters that a packet should traverse in getting from the source cluster to the destination cluster. The clusterheads then distribute this routing information to their respective nodes which use the information to packets from source to destination. Packets are delivered by forwarding it first toward the destination cluster as specified in the HRT and then forwarding it to the destination node, once it has reached the destination cluster, by using some nonhierarchic routing scheme.

Both types of hierarchical routing have disadvantages which must be underlined. The biggest disadvantage with the quasi-hierarchical approach is its poor performance in the context of dynamic topologies. This poor performance can be explained by the inherent tendency of small changes to overpropagate; lengthy propagation delays for faraway changes; and long convergence times in adapting to topology changes. In the case of strict hierarchic routing, the most important problem lies in the central role played by the clusterhead in route computation and distribution, which raises concerns about the routing scheme robustness under clusterhead failure [23]. It is important to point out, however, that clusterheads are not central to the packet forwarding function and that the robustness problem may be minimized by conceiving a mechanism for quickly replacing the clusterheads should they fail at any specific point in time.

4.3 A Routing Scheme for the Linked Cluster Architecture

Vector-length and hierarchical routing may both constitute the basis for a routing scheme customized for the linked cluster algorithm. Vector-length routing was found to be particularly appealing because it may easily be implemented in a distributed manner, whereas hierarchical routing coincides well with the naturally occurring hierarchy of the linked cluster architecture. We are interested in defining and describing the general framework of a routing scheme which incorporates these two approaches for our packet radio network design.

The basis of our routing scheme is the cluster. We have chosen the lowest-ID clustering algorithm for the purposes of our network design. This algorithm produces clusters with a radius of at most one hop and where every node is

connected to the clusterhead. This architecture yields a convenient local broadcast configuration which allows the clusterhead to broadcast important control information to all of the nodes in its cluster in a single transmission.

In the context of routing, this characteristic can be of much benefit. If we assume that the clusterhead has full connectivity information of the nodes in its cluster[2] — it not only knows which nodes are in its cluster but also how these nodes are interconnected — then it is possible for each clusterhead to compute the optimal routes between nodes inside the cluster and then distribute this information to all the nodes in the cluster. We suggest that this approach would undoubtedly be more efficient than having each node broadcast the least-cost route metric value for each node in its cluster, as required by a flat implementation of the Bellman-Ford algorithm inside each cluster. We further suggest that this approach would also be effective in eliminating the formation of routing loops in any given cluster, in contrast with the flat implementation of the Bellman-Ford algorithm where the formation of routing loops is a well-known problem [26].

The question of how the clusterhead is to compute routes between nodes in its cluster is relatively simple. We have mentioned that the radius of the clusters produced by the lowest-ID clustering algorithm is of at most one hop. In other words, any two nodes in any given cluster are at most two-hop neighbors. Route computation is trivial for the case where the source and destination nodes are neighbors.[3] For the case where the source and destination nodes are two-hop neighbors, more interesting alternatives can be considered.

The first alternative is to have each source node transmit its packet to the clusterhead who then forwards it to the destination node. This scheme is possible because we know the clusterhead is connected to every node in its cluster. Though appealing in its simplicity, this scheme is not necessarily desirable because the clusterheads then have the potential of becoming bottlenecks in both the intracluster and intercluster traffic flow [10, 11]. A more homogeneous approach can be taken where other two-hop routes, when they exist, are preferred

[2]We describe in the next chapter how this may be achieved.

[3]We assume that the cost of the one-hop route is less than the cost of any other less direct routes.

over the clusterhead route. Indeed, the fact that any two nodes which belong to the same cluster are, through the clusterhead, necessarily two-hop neighbors does not preclude them from being two-hop neighbors through some other intermediate, ordinary node in the cluster. Thus using these routes, when they exist, may take a significant fraction of the total two-hop traffic away from the clusterhead.

At this point, we have defined a scheme for determining and distributing routes within each cluster. We have thus enabled node pairs to establish end-to-end communications as long as both the source and destination nodes are within the same cluster. We are now interested in defining how two nodes may communicate if the source and destination nodes are in different clusters.

Any hierarchic routing scheme requires that each node know the hierarchical address of every other node in the network. Thus, the first problem we face is to somehow flood this cluster membership information to every node in the network. We know that each clusterhead has complete knowledge of its cluster, both in terms of which nodes it contains and how these nodes are interconnected. It is therefore sufficient to have clusterheads exchange membership information and then have clusterheads broadcast this consolidated information to their nodes for every node to know the hierarchical address of every other node in the network. The problem then becomes a problem of defining how the cluster membership information may be exchanged between clusterheads.

We have seen that the lowest-ID clustering algorithm yields a network configuration where no two clusterheads are directly connected. Cluster membership information must thus necessarily be forwarded to each clusterhead through the gateways which interconnect the different clusters. Furthermore, the clustering algorithm tends to yield clusters which are well spread out in space, meaning that for cases where the network is composed of more than two clusters, then clusters tend to be laid out in tandem. For example, for the case of a three-cluster network, the first cluster would be adjacent to the second which, in turn, would be adjacent to the third. In general, the algorithm would not yield a configuration where the third cluster would be adjacent to the first, for example. This characteristic is important in the context of distributing cluster membership information because we see that it is not sufficient for any given clusterhead to distribute its membership information to adjacent clusters only. Somehow, this

membership information must be distributed to all other clusterheads. In the case of clusterheads of non-adjacent clusters, this distribution scheme must clearly involve the forwarding of information through a series of adjacent and non-adjacent clusters.

Working from these observations, we now proceed to describe how the cluster membership information may be exchanged between clusterheads. We assume that each node knows of its role assignment and that each clusterhead is aware of which nodes constitute its cluster. We further assume that the gateways are aware of which clusters they interconnect, i.e. they not only know that they are members of cluster c_1 but they are also aware that they are connected to members of cluster c_2, and that each clusterhead is aware of adjacent clusters. At some point in time, each clusterhead broadcasts to each of its gateways the following information: its own cluster membership, which we shall call *internal* membership; or any cluster membership information that must be forwarded, which we term *external* membership. Gateways receive this information; process it; and retransmit it such that internal membership information is passed on to all adjacent clusters while external membership is forwarded to only those clusters from which it did not originate. Gateways can make this determination because they are aware of which clusters they are connected to and can thus choose not to transmit external membership information back to the original cluster.

At this point, a simple example may be helpful in demonstrating the dynamics of this scheme. To this effect, we consider the three-cluster network pictured in Figure 4.1. In this figure, clusterheads are labeled with numbers whereas gateways are labeled with letters. Each clusterhead is aware of adjacent clusters. Thus, clusterhead 1 is aware that cluster 2 is adjacent to its cluster; clusterhead 2 is aware that clusters 1 and 3 are adjacent to its cluster; and clusterhead 3 knows that cluster 2 is an adjacent cluster. Similarly, each gateway knows which clusters they bridge: gateways A and B know they bridge clusters 2 and 1, respectively; and gateway C knows that it bridges clusters 2 and 3.

Without loss of generality, we consider a sequence of transmissions starting with clusterhead 1, followed by gateway A, gateway B clusterhead 2, gateway C, and clusterhead 3. For any given clusterhead, the queue is initialized with that clusterhead's internal membership information, whereas each gateway's queue is initially empty. At some point, clusterhead 1 broadcasts the contents of its queue

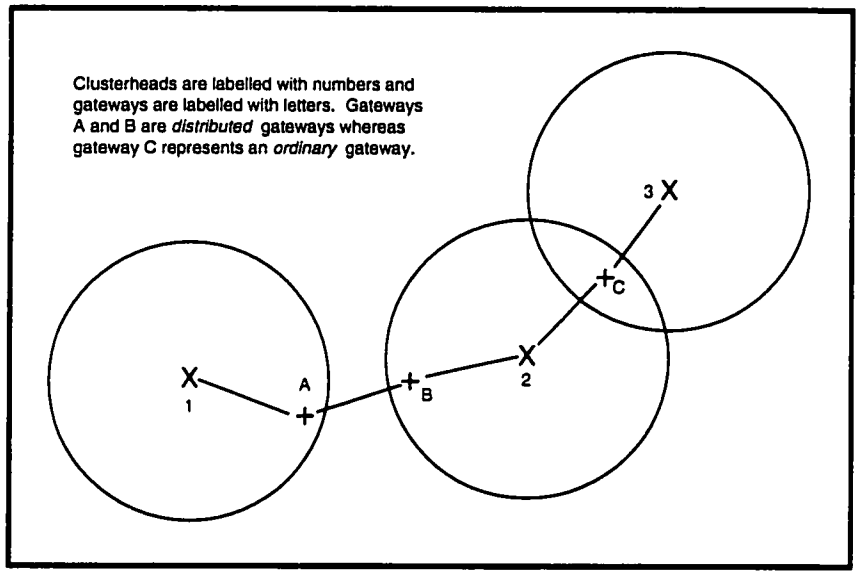


Figure 4.1: A linked cluster network with three clusters.

to all of its gateways, in this case gateway A. Gateway A receives this information and places it at the bottom of its queue. It is now its time to transmit, and it transmits the top message in its queue, which, in this case, is the only message in its queue, representing cluster 1's membership. Gateway B receives A's transmission and similarly processes it. Thus, after gateway B has transmitted, clusterhead 2 is aware of cluster 1's membership as well as its own.

Clusterhead 2 must now determine if it is responsible for forwarding cluster 1's membership information. It does so by considering if it is adjacent to clusters other than the originating / forwarding cluster. In this example, clusterhead 2 determines that it must forward cluster 1's membership information because it is adjacent to cluster 3. It thus places cluster 1's membership information at the bottom of its transmit queue and transmits the top message to all of its gateways. Since this is clusterhead 2's first transmission, this top message is cluster 2's membership.

Gateways B and C receive clusterhead 2's transmission and both determine that they must place these messages at the end of their transmit queue since the information is internal membership information with respect to the transmitting clusterhead. In the case that the transmission represented external membership information with respect to the transmitting clusterhead, then any gateways which link the current cluster to either the originating cluster or the forwarding cluster do not queue the transmission. This measure is taken to prevent external membership information from being uselessly transmitted back to the originating cluster or any cluster which has already seen the information and passed it on.

It is now time for gateway C to transmit, and it transmits cluster 2's membership, which represents the only message in its queue. Clusterhead 3 receives the information and is now aware of cluster 2's membership as well as its own. Clusterhead 2 determines that it is not responsible for forwarding this information — no sense in forwarding cluster 2's membership information back to cluster 2 — and thus does not place this message in its queue. It then transmits the message at the top of its queue to all of its gateways. In this case, clusterhead 3 transmits its cluster membership to gateway C which places this transmission at the bottom of its transmit queue.

We have now gone through an entire transmission cycle. At this point, clusterhead 1 knows only of its membership; clusterhead 2 knows of cluster 1's membership on top of its own; and clusterhead 3, in addition to its own membership, knows of cluster 2's. We do not wish to go through the description of another entire transmission cycle. We suggest, however, that successive transmission cycles will allow each clusterhead to eventually associate every node in the network to a specific cluster, as we could see if we extended this example by two more transmission cycles.

We have described a mechanism for clusterheads to exchange membership information. Membership information alone, however, does not suffice to route a packet from a source cluster to a destination cluster. It is also fundamental that each clusterhead determine a HRT which defines which intermediate clusters a packet must traverse in getting from the source cluster to the destination.

We propose that clusterheads compute their HRT on the basis of the Bellman-Ford algorithm where each cluster is treated as a node and where a link

exists between nodes if the clusters are adjacent. An initial measure of link cost could simply be to assign a cost of 1 to any link, when it exists. In this case, the optimal route between clusters would be that route which minimizes the number of intermediate clusters between the source and destination clusters. Another possible link metric could be the number of gateways — 1 or 2, depending on whether the gateways are *ordinary* or *distributed*[4] — involved in establishing the connection between clusters. In this case, the optimal route between clusters would then be that route which minimizes the number of gateways between the source and destination clusters. We propose that this last metric represents a better choice over the first.

Each clusterhead transmits, each time that its HRT changes, the list of clusters which can be reached from its cluster as well as the associated minimum path metric to each destination cluster. This information is distributed amongst clusterheads by using the same mechanism as that proposed for distributing the membership information. In this case, however, HRT information from adjacent clusters is not forwarded but is simply used to amend any given clusterhead's HRT. In this way, information about adjacent nodes HRT is incorporated into the current clusterhead's HRT and is retransmitted. Finally, it is envisaged that any given clusterhead's HRT simply lists all the clusters which can be reached from its cluster; the minimum intercluster route metric to that cluster; the adjacent cluster through which packets must travel en-route to the destination cluster; and the gateway which will be responsible for forwarding any packets to that adjacent cluster.

At some points in time, the current membership information and the HRT[5] is broadcast by the clusterhead to all of its nodes. Assuming that the membership and HRT information is complete, each node then knows how to route

[4]Ordinary gateways are those gateways which are within radio range of two clusterheads, i.e. adjacent clusters overlap, while distributed gateways bridge clusters which are adjacent but which do not overlap.

[5]We see no requirement for transmitting the minimum metric and the adjacent cluster number to the nodes since the nodes may make a routing decision based strictly on knowledge of the appropriate gateway identification.

intracluster traffic; can properly assign a hierarchical address to any node in the network; and knows to which gateway intercluster traffic must be sent in transit to a destination cluster. In this manner, both intracluster and intercluster traffic may be properly routed.

We propose the general data frame structure illustrated in Figure 4.2 to support the requirements of this routing scheme. If we consider the case of intracluster traffic, we have no requirement for transmitting the hierarchical address of the destination node because nodes are all in the same cluster. Furthermore, nodes have been given explicit instructions by the clusterhead on how to route this type of traffic. In the case where the source and destination nodes are two hops away, there exists a requirement for an intermediate node, neighbor of the source and destination nodes, to forward traffic from the source to the destination. The intracluster / transit frame type addresses this requirement by specifying both intermediate and destination nodes. By using this frame type, the sending node is in effect instructing the intermediate node specified in the transit node subfield to forward the current packet to the specified destination node.

In the case where the source and destination nodes are neighbors, or in the case where the current transmission represents the last hop in an end-to-end route, then there exists no requirement to specify any intermediate, forwarding node. In this case, all that the sending or forwarding node needs to specify is the final destination of the current packet. This requirement is addressed by the intracluster / end frame type. For both cases of intracluster frame types, it is easy to see how routing is achieved. The picture is not so clear for the case of intercluster traffic.

For the case of intercluster traffic, the intercluster frame type is used until the packet reaches the destination cluster, at which time the appropriate gateway converts the packet to one of the two intracluster frame types. Consider any intercluster source-destination pair. The first task of the source node is to identify the cluster ID of the destination node. The sending node may do this because it has been provided with the membership information of the network's nodes by its clusterhead. Associated with this destination cluster ID is the gateway to which the packet must be forwarded. The second task of the sending node is to somehow forward the packet to this gateway.

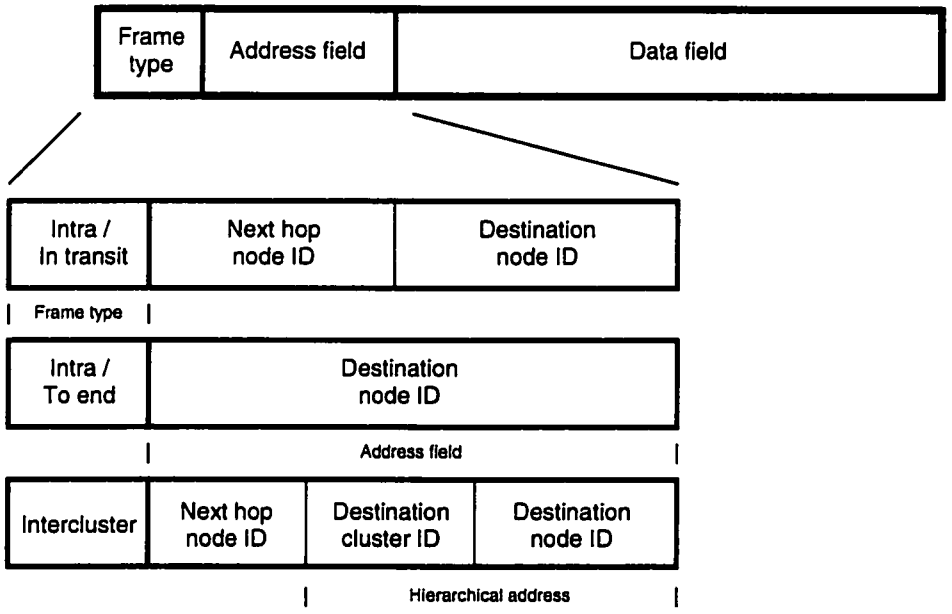


Figure 4.2: The data frame structure for intracluster and intercluster traffic.

Given that the gateway is in the same cluster as the source node, then it is possible to reach the gateway by applying the intracluster routing instructions. In other words, the source node must be one or two hops away from the gateway; it thus knows to which node its packet must be transmitted in order to reach the gateway. It specifies this next hop information in the corresponding subfield of the intercluster frame address field.

The gateway receives the packet and routes it to the the next appropriate gateway, clusterhead, or ordinary node, until it reaches the gateway of the destination cluster. This gateway then converts the intercluster frame into one of the two intracluster frame types by stripping the cluster ID off of the destination node's hierarchical address. It then uses one of the two intracluster frame types to forward the information to the appropriate destination node, depending on whether it is one or two hops away.

To better demonstrate this intercluster mechanism, we consider the example depicted in Figure 4.3. In this example, source and destination node have been

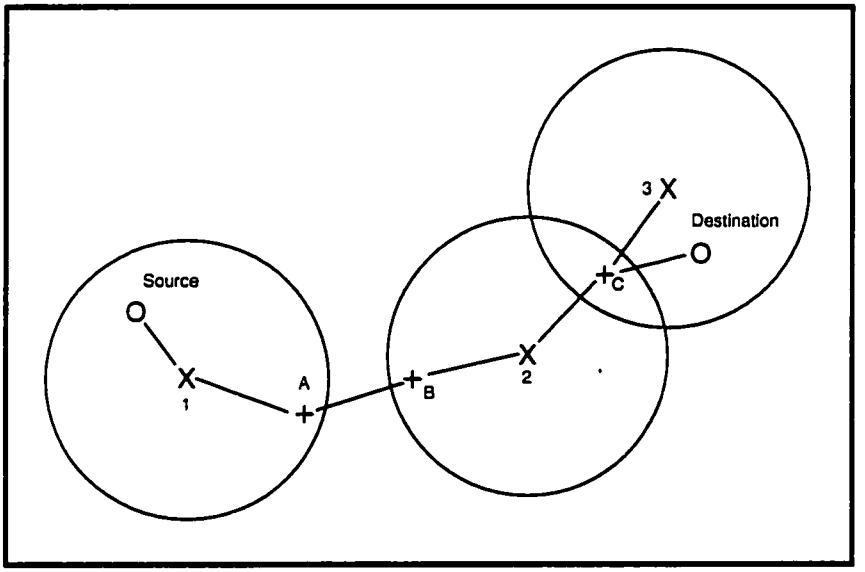


Figure 4.3: An example of intercluster routing.

added to the backbone network of Figure 4.1. We are interested in describing how a single packet is routed from the source to the destination.

To begin, we assume that all of the routing control information has been properly computed and distributed. For conciseness purposes, we shall label the source node S and the destination node D. At some point, node S determines that it must send a packet to node D. It looks up its membership information and sees that node D belongs to cluster 3. It then generates an intercluster frame whose destination cluster ID is 3 and whose destination node ID is D. The clusterhead has previously instructed node S to forward packets to gateway A if it wishes to send them to cluster 3. Based on its intracluster routing instructions, node S determines that the next hop to reach gateway A is clusterhead 1 and inserts this value in the appropriate subfield of the frame. The frame is then broadcast.

Clusterhead 1 receives the frame; sees that it is the intended next-hop destination; and processes it. It determines that the frame is an intercluster frame intended for cluster 3. Like node S, it then chooses to forward the packet to gateway A; inserts gateway A as the next hop, leaving the hierarchical values untouched; and broadcasts the frame.

Gateway A will receive the frame and will process it in exactly the same way as the clusterhead. In its case, however, it knows that packets intended for cluster 3 must be forwarded to gateway B. This is thus the value that it inserts into the next hop subfield. Once transmitted by cluster A and received by cluster B, the packet has left cluster 1 and entered cluster 2. It will cross cluster 2 in exactly the same manner as has been described for cluster 1 until it reaches gateway C. We continue our description from this point on.

Gateway C has just received the intercluster frame. By looking at the destination cluster ID, it determines that the packet has reached the final cluster and that the destination node can now be reached by using the intracluster routing instructions provided by clusterhead 3. These instructions state that gateway C may directly communicate with node D. Thus, gateway C constructs an Intra/End frame whose destination node is D and transmits. Node D receives the packet; determines that it is the end node in the route; and extracts the data in the data field. Hence, the data, originally generated by node S, has successfully reached node D.

This example concludes Chapter 4. In this chapter, we have laid the groundwork for a hierarchical routing scheme which is customized for the principal characteristics of the linked cluster architecture. Though we recognize that much work needs to be done in order to have a complete routing scheme defined for our packet radio network, we believe that the proposed scheme constitutes a solid foundation on which we can build. We now continue with the next chapter in which the configuration and radio channel elements discussed in the previous two chapters are integrated and detailed in a concrete packet radio network design.

Chapter 5

A Packet Radio Network Design

The previous three chapters of this thesis have been dedicated to defining the general framework of our packet radio network design. In Chapter 2, we identified a clustering algorithm for giving a consistent structure to a set of randomly distributed nodes: the lowest ID clustering algorithm. In Chapter 3, we examined the problem of multiple access to the radio channel and selected the GRAND broadcast scheduling algorithm for incorporation into our network design. In Chapter 4, we broadly described a hierarchical routing scheme for the linked cluster architecture. We are now ready to integrate most of these elements into a concrete packet radio network design.

We begin our integration process by defining the *data* and *control* channels of our network. We choose to separate these two channels in frequency, as depicted in Figure 5.1. The data channel essentially serves to carry all of the data traffic between nodes, whereas the control channel carries messages related to connectivity probing; network configuration; broadcast schedule establishment; and routing algorithm execution. The data channel is a single TDMA channel of bandwidth b_d and broadcast schedule length L_b. The control channel consists of a single TDMA channel of bandwidth b_c which is sequentially and entirely available to each node in N broadcast slots.

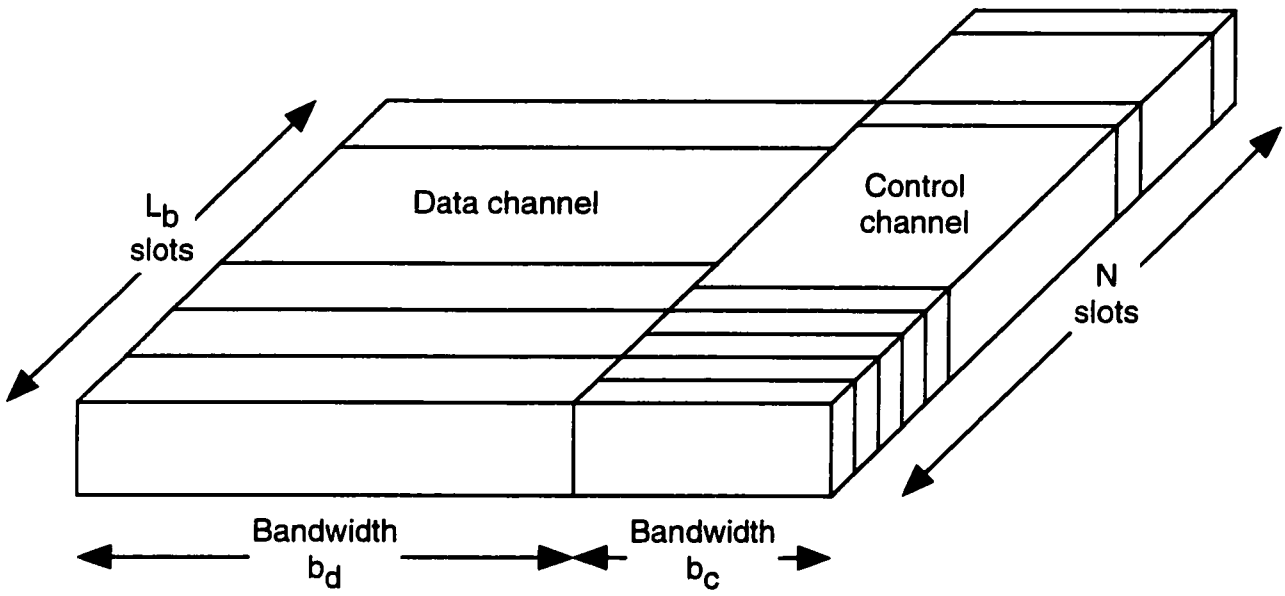


Figure 5.1: The control and data channels.

5.1 The Control Channel

5.1.1 Lowest ID Algorithm Implementation

Determining Connectivity

In Chapter 2, we presented a centralized version of the lowest ID clustering algorithm and determined that it represented the better choice of three possible algorithms for incorporation into our network design. We now wish to describe how this algorithm can be implemented in a distributed fashion. As we shall see, implementation of the distributed algorithm serves as the basis of the control channel's structure.

The Linked Cluster Algorithm described in [7, 8] constitutes a distributed implementation of the lowest ID algorithm. It is conceived to address three

principal functions: determining the connectivity map of the network; forming clusters by electing clusterheads and associating nodes to specific clusters; and electing gateways to serve as bridges between clusters. This distributed algorithm serves as the basis of our distributed implementation, which we here modify to suit our packet radio network design.

We briefly pause at this point to discuss the nature of multimedia communications and its impact on how we have so far defined network connectivity. Multimedia communications imply the simultaneous use of different *communication modes* to relay information. Today, these modes are essentially grouped into three important categories: voice, video, and data. Each of these modes have different requirements in terms of acceptable bit error rates such that good performance may be achieved.

Bit error rates are determined by $\left(\frac{E_b}{N_0}\right)$ at the receiver. Thus, different multimedia communication modes will require different signal powers at the receiver to each achieve acceptable performance. We have thus far said that a link exists between nodes when the transmitter's signal arrives at the receiver with power greater than or equal to a predetermined threshold. We can easily see, then, that different thresholds will translate into different connectivities, depending on which threshold is used to define a link between nodes. For example, consider two communication modes which require received signal power thresholds of γ_1 and γ_2. We let $\gamma_1 < \gamma_2$, and we assume that propagation losses are such that a signal transmitted from a sending node arrives at the receiving node with a power $\gamma_r, \gamma_1 < \gamma_r < \gamma_2$. In this example, we would say that a link exists between the sending and receiving nodes for the first communication mode only. Thus, the existence of k communication modes, each associated with a given threshold γ_k, translates into as much as k different connectivity maps of the same node distribution.

We choose to set our number of thresholds at three, based on the current number of multimedia communication modes. Thus, each node maintains three distinct connectivity matrices Y_i^j of size NxN, where i represents the node's number, j represents the communication mode ($j \in \{1,2,3\}$), and each row k of the matrices represents the *k-th* node's connectivity to all other nodes in the network, i.e. entry (k, l) of some connectivity matrix Y_i^j is flagged as true if node i somehow knows that node l is bidirectionally connected to node k according to

the threshold criteria for the *j-th* communication mode. The *i-th* row of the connectivity matrix Y_i^j represents node *i*'s *connectivity vector* for the *j-th* communication mode.

With these initial definitions and comments, we have laid the foundation for a distributed implementation of the lowest ID clustering algorithm. We now proceed to describing it in detail. The initial state of our network is simply a random geographic distribution of a maximum of N nodes with each node identified with a predetermined number $i, i \in \{1, 2, 3, \ldots, N\}$. The probing vectors, which we describe below, and connectivity matrices at each node are initialized at zero.[1] Building from this initial state, we seek to establish and maintain three distinct connectivity maps, one for each of the three possible communication modes. The frame structure which we have developed to achieve this objective is depicted in Figure 5.2.

We consider a TDMA cycle of N slots, where each node is assigned a broadcast permission in the slot equal to its identification number, i.e. node 1 transmits in slot 1, node 2 in slot 2, and so on. In slot 1 of the control channel TDMA cycle, node 1 transmits its probing signal with a power P_c to anyone who can receive it. All other nodes are simultaneously listening for it. Similarly, node 2, 3, ..., N will transmit their probing signal in their respective slots. The probing signal is not a simple tone but contains information which will allow each node to partially fill in their connectivity matrix.

The first purpose of the probing signal is not to distribute connectivity information throughout the network but rather to determine this connectivity. We assume that each node is capable of measuring received signal strength and that each node transmits its probing signal with power P_c. We further assume that each node transmits over the data channel with a unique power P_d. In any given control channel slot i, a subset of all nodes will receive node *i*'s probing signal. Let $P_{i,j}^p$ represent node *i*'s probing signal strength as measured at node j. Since node j knows with what power the probing signal was transmitted, it can calculate a propagation loss factor $L_{i,j} = \frac{P_{i,j}^p}{P_c}$. With this loss factor calculated, it

[1] Elements of the diagonal of each connectivity matrix are set equal to one. We may think of this characteristic as each node being necessarily connected to itself.

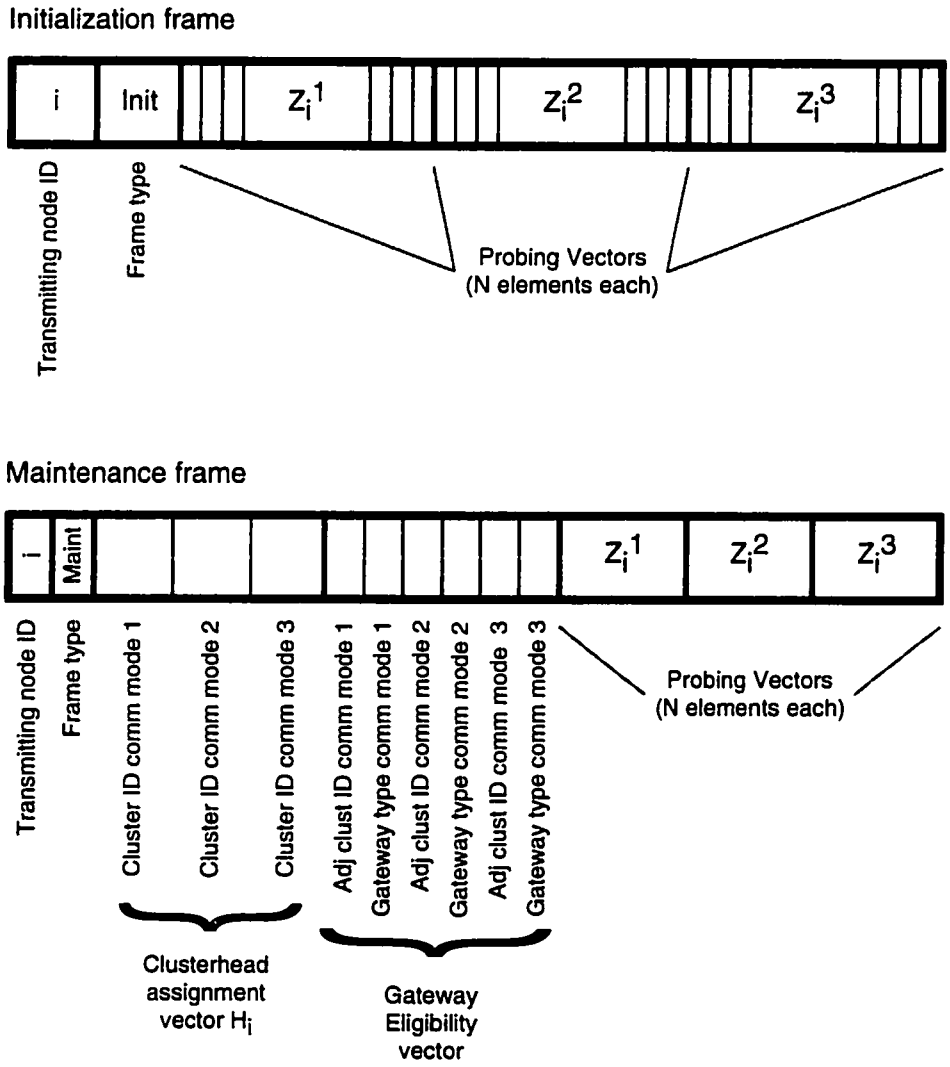


Figure 5.2: Control channel frame structure.

may then predict with the power $P^d_{i,j} = L_{i,j}P_d$ with which any data signal would be received from node[2] i.

We let γ_1, γ_2, and γ_3 represent the required thresholds of the voice, video, and data communication modes, respectively. Node j can decide that a unidirectional link exists from node i to node j for communication mode k if $P^d_{i,j} \geq \gamma_k$. If it also knows whether the reverse link exists, then it can decide that a bidirectional link exists with node i and can thus amend its appropriate connectivity matrix.

We characterize a given TDMA frame of the control channel as either an *initialization* or a *maintenance* frame. Frame type determines how the information transmitted by each node in their probing messages are processed by the receiving nodes. The connectivity determination process begins with a single initialization frame and is followed by a series of maintenance frames of some length. In any given slot i of any frame, node i will be transmitting its probing signal while all other nodes will be listening for it. Independently of the current frame type, the probing signal of each node consists, for now, of its identification number, the current frame type, and three probing vectors Z^k_i of length N, one vector for each of the three communication modes, whose contents we specify below.

Any node j which receives the probing signal first performs the required power measurements and calculations; identifies the sending node; and identifies the frame type. If the frame is an initialization frame, it then proceeds to do the following:

If $j > i$ then
    $Z^k_j(i) = (P_{i,j} \geq \gamma_k)$ (false=0, true=1)
else if $j < i$ then
    $Y^k_j(j,i) = (P_{i,j} \geq \gamma_k)$ AND $(Z^k_i(j) = 1)$
    $Z^k_j(i) = Y^k_j(j,i)$
endif

[2]A possible adjustment factor can be added to this calculation to account for the different carrier frequencies of the control and data channels.

If the frame is a maintenance frame, then the receiving node j performs the following:

$Y_j^k(j,i) = (P_{i,j} \geq \gamma_k)$ AND $(Z_i^k(j) = 1)$
$Z_j^k(i) = Y_j^k(j,i)$
If $Y_j^k(j,i) = 1$ then
$Y_j^k(i,m) = Z_i^k(m), m = 1,2,3,...,N$
else
$Y_j^k(i,m) = 0, i \neq m$ and $Y_j^k(i,i) = 1$
endif

We consider an initialization frame followed by a maintenance frame. In frame j of the initialization frame, node j in effect broadcasts the identities of the nodes which it has heard from and which satisfy the threshold criteria for a given communication mode. This information is contained in its three probing vectors which have been updated for every node which broadcast before it. Hence, by the end of the initialization frame, node j has filled in the all elements $i > j$ in its connectivity vectors.

In the maintenance frame, each node broadcasts its full connectivity vectors. This is possible because node j has completely filled in the j-th row of its connectivity matrix by the time it transmits its probing vectors in the maintenance frame. Thus, by the end of the maintenance frame, each node knows who are its one-hop and two-hop neighbors for each of the three communication modes. The global connectivity matrices are only partially available to any given node. However, if we assume that the information exchange between nodes is error-free, then all individual versions of the connectivity matrices will be consistent with the global connectivity matrices [9].

Any subsequent maintenance frames only serve to determine if the previous connectivity maps still hold. If a link fails for some reason, this implementation will detect failure. Once a link has failed, however, it is not possible to determine that it has been reestablished until the advent of an initialization frame. Thus, we ask ourselves how many maintenance frames should be placed between initialization frames. Though we will not answer this specific question, we

propose three approaches which can be studied to determine if any one of them represents a better choice.

The first approach that can be taken simply consists of a succession of initialization - maintenance frame pairs. This represents the best approach that can be taken for maintaining an accurate picture of the network connectivity. However, it has serious disadvantages that might not make it necessarily desirable. If this approach is taken, then half of all probing frames will consist of initialization frames. As we have seen above, initialization frames are more complex to process than maintenance frames. Thus, the accuracy of the network connectivity is obtained at the expense of computational complexity. Secondly, this strategy has the potential of translating into rapid and systematic changes in the network connectivity map. This poses a serious constraint in that if the connectivity map changes faster than the time it takes for routing algorithms to converge, routes may never be determined between source - destination node pairs.

A second approach that can be taken consists in placing a fixed, predetermined number of maintenance frames after the initialization frame. This number could be determined on the basis of possible values of network connectivity *coherence time*. Here, coherence time, which needs to be precisely defined, would serve as a measure of the time interval over which the network connectivity is strongly correlated. A third approach which builds on the previous strategy could consist in dynamically adjusting the number of maintenance frames between initialization frames. Specifically, an initialization frame could be sent once the network configuration has changed by more than some preset acceptable cost.

Our discussion has thus far focused on a distributed scheme for determining connectivity between nodes for each of the three communication modes. We have seen that implementation of this scheme allows any given node to determine who are its one-hop and two-hop neighbors for any specific communication mode. We now proceed to describe how clusters are formed and how clusterheads are elected such that a linked cluster network can be constituted from these low-level connectivity maps.

Determining Clusters and Clusterheads

As we shall see, the process for selecting clusters and clusterheads is readily implemented. The cluster and clusterhead selection rule can be formulated as follows:

- Node i becomes a clusterhead unless it is already connected to one; and
- If node i does not become a clusterhead, then it selects as its clusterhead the lowest ID clusterhead to which it is connected.

In the distributed implementation, this rule can be easily implemented by having each node broadcast the identity of its own clusterhead during any maintenance frame transmission. Because each node maintains three distinct connectivity maps, the selection rule must be applied to each map, yielding three distinct, possibly different, cluster and clusterhead assignments. Thus, we implement this announcement by having each node i transmit a *clusterhead assignment* vector H_i of length 3, where each element of the vector represents the clusterhead ID of node i for one of the three communication modes. We insert this vector immediately after the frame type identification field of each node's maintenance probing signal.

Each node i maintains three *role assignment* vectors R_i^k of length N in which it stores the role assignments of each node to which it is connected, for each communication mode k.[3] Each node i similarly maintains three *cluster assignment* vectors A_i^k in which it stores the ID of each node's clusterhead for each communication mode k. Each node initializes these six vectors to zero prior to transmitting the probing signal of the initialization frame.

Immediately prior to transmitting in any given maintenance frame, each node i is aware of its connectivity for each of the three communication modes. It can then perform a logical function to determine if it must become a clusterhead and to which cluster it is associated. We consider any transmitting node i and any receiving node j. In slot i of any maintenance frame, node i broadcasts the list of its three clusterheads to any node j who can receive this information. Just prior to transmitting its probing message, but immediately after amending its

[3] An ordinary node is represented by 0; a clusterhead by 1; and a gateway by 2.

connectivity matrices, node i determines its role and cluster assignments as follows:

If $A_i^k(i) = 0$ then
    $R_i^k(i) = 1$
    $A_i^k(i) = i$
    $H_i(k) = i$
endif

In slot i, each receiving node j processes the information received to both determine its cluster assignment and update its role and cluster assignment vectors. Specifically, each receiving node j performs the following computations, after amending its connectivity matrices:

If $(Y_j^k(j,i) = 0)$ then
    $R_j^k(i) = 0$
    $A_j^k(i) = 0$
    If $A_j^k(j) = i$ then
        $A_j^k(j) = 0$
    endif
else if $(Y_j^k(j,i) = 1)$ AND $(H_i(k) = i)$ then
    $R_j^k(i) = 1$
    $A_j^k(i) = i$
    If $(A_j^k(j) = 0)$ OR $(i < A_j^k(j))$ then
        $R_j^k(j) = 0$
        $A_j^k(j) = i$
        $H_j(k) = i$
    endif
else if $(Y_j^k(j,i) = 1)$ AND $(H_i(k) \neq i)$ then
    $R_j^k(i) = 0$
    $A_j^k(i) = H_i(k)$
    If $A_j^k(j) = i$ then
        $A_j^k(j) = 0$
    endif
endif

We pause at this point to briefly review the exact significance of this pseudocode. The entire process is best understood by first considering how each receiving node j determines its cluster association. Essentially, these nodes elect as their clusterheads, and thus select their cluster, the lowest ID node to which they are connected. Specifically, the distributed implementation for each receiving node j begins by considering the case where j is not connected to node i. For such a case, the receiving node registers that the role and cluster assignments of i are unknown and then further considers if its clusterhead is currently selected to be node i. If it is, it resets its cluster assignment since it knows that it is no longer connected to this clusterhead and waits to either be associated to another cluster or become a clusterhead itself.

In the event that node j is connected to node i, then node j considers if node i can be its clusterhead. First, though, it must determine if node i is a clusterhead. Node j determines that node i is a clusterhead if node i is associated to cluster i, i.e. node i's clusterhead is itself. With this determination made, node j ammends its role and cluster assignment vectors to reflect the assignments of node i. It further chooses i as its clusterhead if it either currently does not have a cluster assignment or i has a lower ID than its current clusterhead. In the case where i and j are connected but i is not a clusterhead, then j ammends its assignment vectors to show that i is an ordinary node and, since it is possible that i reverted from being a clusterhead to being an ordinary node, it considers if i was previously its clusterhead. In the event that i is j's current clusterhead, then j resets its cluster assignment and waits to either be associated to another cluster or become a clusterhead itself.

It is possible that node i has not selected a cluster by the time it must transmit its clusterhead ID. This would occur if it was not connected to any clusterhead with an ID lower than i and would be reflected in i's clusterhead ID being set to 0. We recall that any given node must become a clusterhead if it is not connected to one. Thus, if node i is not associated to a cluster by the time it must broadcast its clusterhead ID, then it becomes a clusterhead. This function is that function which is implemented immediately prior to transmission and was the first to be described.

This distributed implementation results in exactly the same structure as that produced by the centralized lowest ID algorithm described in Chapter 2. By the

end of each maintenance frame, each node knows of its role and cluster assignments as well as the role and cluster assignments of each of its neighbors. Although no node has overall knowledge of the network structure, each of its calculated vector entries are consistent with the global role and assignment vectors which would be calculated by a central controller, if it existed. We are now left with the problem of determining gateways to bridge communications, if possible, between clusters.

Determining Gateways

We wish to conclude this section by detailing how gateways are determined. We saw in the previous chapter that the gateway is critical to both the route establishment and packet forwarding functions. We begin our short description of how gateways should be selected by defining two classes of gateways: general and restricted gateways.

We say in Chapter 4 how a backbone network may be used to exchange membership information between clusterheads. In this instance, single gateways, which may be either ordinary or distributed, bridge the communication path between clusters. The important characteristic to retain in this case is that, for the purposes of backbone control information, only a single gateway is designated for intercluster traffic. We categorize this gateway, which may forward control traffic as well as ordinary user traffic, as a *general* gateway. Here, general is used in the sense that both control and user information may be forwarded by the gateway.

In reality, more than a single node may be capable of communicating with a node belonging to an adjacent cluster. Though only a single gateway need be designated for the purposes of control information exchange, we believe that an advantage exists in retaining as many gateways as possible. Indeed, the linked cluster architecture, as presented in the literature, calls for the election of a single gateway between clusters. It is even suggested in some papers that redundant gateways should be eliminated. We do not agree with this approach to gateway selection. By relying on only one gateway for each intercluster route, a large potential exists for bottlenecks to occur at the gateways, which could result in network failure. Thus, if more than a single node may bridge communications

between clusters, then we believe that as many of them as possible should be retained. Only one of them will be a general gateway. All the others will be *restricted* gateways. Here, restricted is used in the sense that these nodes are designated as gateways for the purposes of forwarding only ordinary user traffic.

Having distinguished between the two broad types of gateways, we proceed to describe how these gateways may be chosen. Consider any node u. By the end of the first maintenance frame, node u knows its role and cluster assignment as well as the role and cluster assignments of each of its one hop neighbors. If node u is not a clusterhead, then it can be a gateway if it is connected to a node which belongs to a cluster different than its own. For example, node u, which belongs to cluster U, may become a gateway if it is connected to some node v which belongs to a cluster other than cluster U.

Each node u can determine that it can become a gateway by scanning its cluster assignment vector A_u^k to determine if any of its neighbors belong to a cluster other than its own. Indeed, we have seen how each node becomes aware, by the end of the first maintenance frame, of the cluster membership of each of its neighbors, storing the results in its cluster assignment vector. Thus, a simple scan of A_u^k, after the end of the first maintenance frame, will allow node u to determine which clusters it can bridge.

We may go a step further and say that each u may determine if it is an ordinary or distributed gateway. Node u can make this determination on the basis of the contents of its role assignment vector R_u^k. In general, a gateway is an ordinary gateway if it is connected to two different clusterheads; otherwise, the gateway is a distributed gateway. Node u's role assignment vector contains the role assignment of every neighbor of u. Now consider the case where u has determined, from scanning its cluster assignment vector, that it is connected to node v, which belongs to cluster V. By looking at the v-th entry of R_u^k, node u will determine whether node v is an ordinary node or clusterhead. If node v is a clusterhead, then node u will know that it is an ordinary gateway. Similarly, if node v is an ordinary node, than node u will know that it is a distributed gateway.

Thus, each node which has not been elected to be a clusterhead can, after the first maintenance frame, determine its own eligibility to be a gateway. It can further determine whether it can be an ordinary or a distributed gateway. In the

case where the gateway is distributed, it can determine to which node intercluster packets must be transmitted. In this manner, the gateway eligibility determination process is wholly distributed. We now face the problem of forwarding this information to the clusterheads which require it to compute their HRTs. Furthermore, in those cases where multiple gateways exist, the clusterhead appears as the natural choice for determining which gateway will be the general gateway and will carry control information, in addition to general user data.

This problem can be simply resolved. We propose that each node simply announce its eligibility to be a gateway during its second maintenance frame transmission, and every maintenance frame thereafter. Specifically, we propose that this gateway announcement consist of a field of six slots, where each pair of slots is used to broadcast the cluster that the node bridges to and its gateway type, for each of the three communication modes. Thus, slots 1 and 2 of this field would represent the adjacent cluster ID of a given gateway and the gateway type, respectively; slots 3 and 4 would represent this same information for the second communication mode; and so on. We propose that this field follow the transmission of the clusterhead assignment vectors by all nodes in any maintenance frame transmission. In the first maintenance frame, this field will be empty because it can only be filled with the appropriate information after the first maintenance cycle has been completed. Likewise, the field will contain the appropriate information for the second maintenance frame, and every maintenance frame thereafter.

The clusterhead is thus made aware of all gateways, and it knows which clusters are bridged by these gateways. Furthermore, we have determined that it is preferable to maintain all of these gateways so that intercluster traffic may be distributed amongst them instead of being concentrated over a single intercluster connection. Thus, there is no requirement for the clusterhead to "enable" the gateways in the sense that the clusterhead does not have to tell the nodes if their gateway eligibility has been maintained. If a gateway is eligible to be a gateway, then it de facto becomes a restricted gateway. This decision can be made without any intervention from the clusterhead. Hence, the transmission up to the clusterhead is made strictly to allow the clusterhead to compute its HRT and, possibly, to allow it to select its general gateway.

With respect to the problem of selecting a general gateway from a set of eligible gateways, two distinct approaches may be taken. In the first case, the clusterhead can select the general gateway and then announce its choice. The clusterheads may make that choice because it knows of all eligible gateway. This approach's main disadvantage is the requirement for the clusterhead to somehow broadcast its choices back to the gateways, yielding additional overhead. An alternative could be to have the eligible gateways locally decide amongst themselves and then forward their decision to the clusterhead. This approach is based on the fact that if the gateways can all communicate with nodes of the same adjacent cluster, then they are probably all within communication range of each other, i.e. the subset of nodes which are gateways to the same adjacent cluster probably form a fully connected subnet. They can thus benefit from each other's gateway announcements to locally elect one of them to be the general gateway. This election would then be forwarded to the clusterhead during subsequent gateway announcements. Although it is not clear at this point which of these two approaches would be more advantageous, we suspect that the second approach is probably more easily implemented and that it yields lesser overhead.

In any case, the gateway identification process is complete in the sense that the clusterhead can now inform its nodes of HRTs which detail which gateways may be used to forward a packet to some destination. Because multiple gateways may be listed as possible gateways en route to some final destination, then it is the nodes themselves which will decide which gateway to forward the packets to. We will not enter into how any given node will select a specific gateway from its list of possible alternatives because that is a routing protocol design issue. Suffice it to say that we have defined a distributed mechanism for gateways to be identified to each node in the network. We have thus wholly defined a mechanism for constructing linked cluster networks for each of the three communication modes and defined the structure of the control channel for our packet radio network design. We now continue by considering the data channel and the GRAND broadcast scheduling algorithm implementation.

5.2 The Data Channel

We have seen in Chapter 4 how we propose to structure data channel frames, and, at this point, we wish to go no further than this broad description. Our discussion on the data channel will focus on providing the details of the GRAND broadcast scheduling algorithm and of how we propose global network parameters should be determined. We begin by fully describing the GRAND algorithm.

5.2.1 GRAND Algorithm Implementation

We discussed the general GRAND algorithm implementation principles in Chapter 4 and shall not repeat them here. We are now interested in providing a formal description of the algorithm. All details have been obtained from the original paper published by Chlamtac and Farago in [20]. To begin, we let $GF(q)$ represent the Galois field of order q. We further let $\beta_0, \beta_1, \ldots, \beta_{(q-1)}$ represent the q elements of $GF(q)$.

We let vectors $(a_0, a_1, \ldots, a_k)$ represent polynomials of order k over $GF(q)$, where a_i is an element of $GF(q)$. We term these polynomials the *vector identifiers* of each node. We let q_1, q_2 represent the sequence of prime powers in increasing order. The slots of the broadcast schedule are simply labeled as $1, 2, 3, \ldots, L_b$. Finally, we let D represent the maximum degree of all nodes in the network.

The GRAND algorithm implementation can be broken up into two parts. In the first part, appropriate values of k and q are computed based on the values of N and D. In the second, the broadcast schedule is determined. We recall that the algorithm is wholly based on the intersection properties of the graphs of the vector identifiers[4]. A one-to-one correspondence between these graphs and the slots of the broadcast schedule will yield a schedule in which each node is allocated q slots, no two of which will have more than k slots in common. If the values of k and q are chosen such that there are enough vector identifiers for each node and that $q \geq kD + 1$, then interference-free transmission is guaranteed for

[4]Refer to the last paragraph of page 61.

every node in at least one slot per TDMA cycle. Thus, determining the appropriate values of k and q is central to the algorithm's performance.

Specifically, the proper values of k and q are determined as follows:

$m = 0$
While $((k \geq 1)$ AND $(q^{k+1} \geq N)$ =FALSE
    $m = m + 1$
    $q = q_i$
    $k = \lfloor \frac{q-1}{D} \rfloor$
endwhile
$L_b = q^2$

This part of the algorithm thus allows the system designer to determine the appropriate values of k and q. It is with this iterative routine that the cycle lengths of Figure 3.12 were generated. Because this first algorithm section is self-explanatory, we shall not discuss it any further. We now continue our discussion by defining how the broadcast schedule is determined.

Determining the Broadcast Schedule

The GRAND algorithm constructs its broadcast schedule before network deployment from the graphs of the N distinct vector identifiers associated with each node. These vector identifiers represent polynomials over $GF(q)$ of order k, values for q and k having been previously determined, as detailed previously. We are now interested in defining how these vector identifiers translate into a comprehensive broadcast schedule. We begin by formally listing the algorithm and then continue by commenting on its exact significance.

We let $S(v_i)$ represent the subset of all slots in which node v_i may transmit. Furthermore, for any α in $GF(q)$, let ind(α) be that integer j for which $\alpha = \beta_j$.

We then say that $\text{ind}(\alpha) = j$. For each node v_i in the network, transmission slots are determined as follows:

$S(v_i) = \{\}$
Assign a unique vector identifier $(a_0, a_1, ..., a_k)$ to v_i
For $m = 0$ to $q - 1$
    $\alpha = \sum_{l=0}^{k} a_l \beta_m^l$ (over $GF(q)$)
    $S(v_i) = S(v_i) \bigcup \{mq + \text{ind}(\alpha) + 1)$
endfor

The complete broadcast schedule is the union of the individual schedules of each node. These individual schedules are those schedules computed in that part of the GRAND algorithm which we just listed. This section yields a subset of q slot labels ranging from 1 to L_b in which the node may transmit. Perhaps the easiest way of conceptualizing the slot assignment is to view the TDMA cycle as a series of q frames, each frame being divided into q slots. Each node is assigned one slot per frame. Thus each node is assigned a total of q slots in the entire TDMA cycle.

Specifically, the assignment algorithm begins by initializing any given node's schedule to an empty set. That node is then tagged with a unique vector identifier. The slot allocation process actually takes place in the third step of the algorithm. In this section, the graphs of each vector identifier is evaluated for every element of $GF(q)$, as described in the first line of the *for* loop. Each of these graph points is then translated into a TDMA slot label, which is then added to the node's individual schedule, as described in the second line of the *for* loop.

To show the mechanics of this half of the GRAND algorithm, consider the case where $k = 2$ and $q = 3$. We immediately wish to underline that these numbers do not represent realistic values but have been chosen to simplify our example. In this case, a maximum number of $N = q^{k+1} = 27$ distinct vector identifiers could be generated. The schedule will have a cycle length $L_b = q^2 = 9$ slots long.

Consider a node tagged with the vector identifier (1,1,0) representing the polynomial $p(x) = 1 + 1x + 0x^2 = 1 + x$. We note at this point that each of the

three elements of the vector could be elements of $GF(3) = \{0, 1, 2\}$. The graph of the polynomial is obtained by evaluating the polynomial for each value of element in $GF(3)$. Thus, it can be calculated that $p(0) = 1$, $p(1) = 2$, and [5] $p(2) = 0$. These are the results we expect to see from the third step of the algorithm execution. For illustration purposes, this step is executed as follows:

$$
\begin{aligned}
&m = 0 \\
&\qquad l = 0, \alpha = 1 \\
&\qquad l = 1, \alpha = 1 + 1(0) = 1 \\
&\qquad l = 2, \alpha = 1 + 0 = 1 \\
&S(v_i) = \{2\}
\end{aligned}
$$

$$
\begin{aligned}
&m = 1 \\
&\qquad l = 0, \alpha = 1 \\
&\qquad l = 1, \alpha = 1 + 1 = 2 \\
&\qquad l = 2, \alpha = 2 + 0 = 2 \\
&S(v_i) = \{2, 6\}
\end{aligned}
$$

$$
\begin{aligned}
&m = 2 \\
&\qquad l = 0, \alpha = 1 \\
&\qquad l = 1, \alpha = 1 + 2 = 0 \\
&\qquad l = 2, \alpha = 0 + 0 = 0 \\
&S(v_i) = \{2, 6, 7\}
\end{aligned}
$$

Algorithm execution confirms our expectations. Furthermore, the algorithm yields a schedule where the node is assigned $q = 3$ slots per cycle in a TDMA broadcast schedule of nine slots. In this specific case, the node which has been tagged with the vector identifier $(1, 1, 0)$ is allowed to transmit in slots {2,6,7} of the cycle.

We wish to go a step further and attempt to illustrate how the throughput of the network may be substantially higher than the minimum lower bound. Consider

[5] Operations are done over $GF(3)$. Addition is thus performed *modulus* 3 (mod 3). Thus, $1 + 1(2) \bmod 3 = 3 \bmod 3 = 0$.

the case of a pair of two-hop neighbors which must both communicate with their common neighbor. The first sending node is tagged with the vector identifier discussed in our example and transmits in slots {2,6,7}, while the second sending node is tagged such that it transmits in slots {1,5,8}. In this case, the distribution of the nodes is such that the individual schedules do not overlap. Thus, every transmission sent by each of the two nodes will be successful in the sense that the transmission is interference-free. In this case, the contribution of these two nodes to the overall throughput would be of six successful transmissions, in contrast to the lower bound guaranteed contribution of two successful transmissions.

Determining Global Operating Parameters

In our analysis of broadcast scheduling algorithms, we discussed how the use of the GRAND algorithm can be justified in terms of cycle length if the maximum connectivity values are maintained below a certain maximum value d_{max}. We then pursued our arguments on the assumption that these connectivity conditions can be achieved. We wish to conclude our present discussion by considering the validity of this assumption and its implications in terms of determining global operating parameters, principally the distribution area of the network and the proper communication range of each node.

Preliminary simulations performed to test the low-connectivity assumption indicate that the conditions required of the GRAND algorithm may be achieved. To this effect, consider the results plotted in Figures 5.3 and 5.4. Both figures illustrate the behavior of average maximum degree as a function of radio and distribution range for 25 nodes.

Referring to Figure 3.12, we see that for 25 nodes, the use of the GRAND algorithm is justified for maximum connectivity conditions less than or equal to $d_{max} = 4$. Referring to the plots in Figures 5.3 and 5.4, we see that a validity region exists where this maximum connectivity requirement can be satisfied. Specifically, if we let r represent radio range and d represent distribution range, then we see that the average maximum connectivity does not exceed a value of $d_{max} = 4$ for that area ranging approximately from $(r, d) = (25, 100)$ to $(r, d) = (45, 250)$. In other words, if the combination of (r, d), for 25 nodes specifically, is such that it falls in this region, then, on average, the maximum degree of any given node will be less than, or equal to, d_{max}.

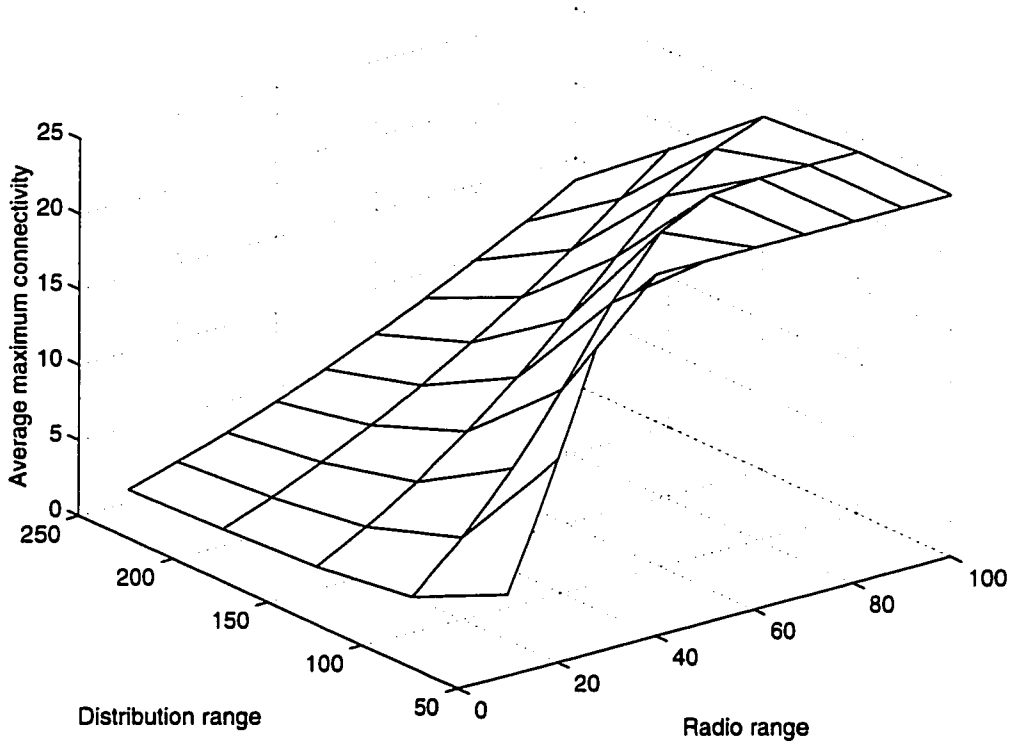


Figure 5.3: Average maximum degree for 25 nodes as a function of radio and distribution range, mesh plot.

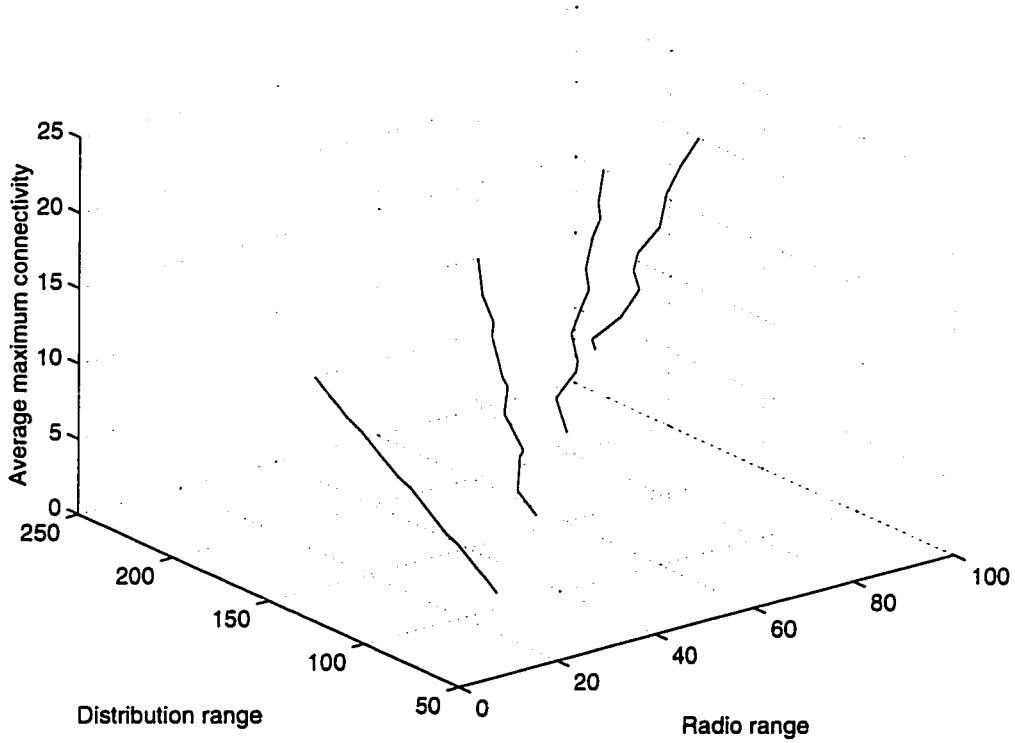


Figure 5.4: Average maximum degree for 25 nodes as a function of radio and distribution range, contour plot.

These results indicate that, for a given number of nodes, it is possible to identify *validity regions* where connectivity regions justify the use of the GRAND algorithm. It is clear that these validity regions are largely determined by nodal density and radio range. Looking at the problem from another perspective, we can say that, for any given number of nodes, the requirement to operate in a given validity region will dictate acceptable values of nodal density and radio range. This forces us to define a process for determining global operating parameters.

The problem of determining global operating parameters can be stated as such: given only the number of nodes in the network, in what area should these nodes be distributed, and what should be the radio range of each node, such that the network efficiently operates in a GRAND algorithm validity region. To resolve this problem, we propose the following:

- For a given N, apply the first half of the GRAND algorithm to determine d_{max}. This value can be obtained by calculating the cycle length of the TDMA cycle for every possible value of maximum connectivity, as was done in Figure 3.12.
- Determine validity region from a plot of $\overline{D} = f(r, \sigma)$, where σ represents nodal density, and r represents radio range. It is possible to generate such a plot by averaging, for any given value of σ and r, maximum degree over a large number of network distributions. The resulting plot would resemble our plot of Figure 5.3 with the exception that nodal density values would replace the distribution range values of this figure. It is important to also note that simulations leading to the appropriate plots should be carried out over a realistic channel model for a given operating frequency band. We suggest that a good starting point might be a channel model for the UHF frequency band with an r^{-4} propagation loss factor and log-normal shadowing. At this point, then, we have determined a range of values of nodal density and radio range over which the average maximum degree will not exceed d_{max}.

- Of the range of possible values (r, σ), select that pair for which some combined *throughput-connectivity* function is optimum. It is not clear at this point what the form of this function should be, but it is clear that the best operating point (r, σ) would be that point where throughput is maximized while end-to-end connectivity is maintained above an acceptable threshold. Clearly, we wish to prevent a low choice of r which might represent the optimal choice in terms of throughput alone, but which would yield a highly disconnected network. In any case, determining a value of $(r, \sigma)_{opt}$ will necessarily involve further simulation, over the validity region of interest, where the value of the combined throughput-connectivity function is averaged over a large number of network distributions. Data thus generated can then be analyzed to determine $(r, \sigma)_{opt}$. The appropriate distribution area A_{opt} can then be determined from σ_{opt} to be $A_{opt} = \frac{N}{\sigma_{opt}}$ square units.

In conclusion, this chapter has focussed on providing the specific implementation details of the lowest ID clustering algorithm and the GRAND broadcast scheduling algorithm. Specifically, we have seen how, working only from a set of peer radios randomly distributed in some fixed geographic area, we are able to determine the network connectivity for different multimedia communication modes and then configure the network according to the linked cluster architecture. We have further seen how an a priori application of the GRAND algorithm will yield a collection of individual transmission schedules which guarantee that each node will transmit at least once without interference in the TDMA cycle. Finally, we defined a process for determining the optimal radio range and distribution area of the nodes. Thus, we have defined a solid framework for a packet radio network design capable of supporting field, mobile multimedia communications.

Chapter 6

Conclusion

The problem of designing a packet radio network for field mobile multimedia communications is a very complex task which touches on virtually all aspects of digital communications. For each of these areas, many possible design options exist, and thus, a packet radio network design is the sum of complementary design choices. This thesis has been concerned with making some of these choices in the areas of network configuration; radio channel access; and network management.

6.1 Thesis Summary and Contributions

The problem of defining a network configuration was considered in Chapter 2. Working from an initial set of design constraints and objectives, we identified the Linked Cluster Architecture as an appropriate architecture for our packet radio network. We then continued to describe three clustering algorithms which produced linked cluster networks: the highest ID, lowest ID, and highest connectivity clustering algorithms. Of these three algorithms, we selected the lowest ID clustering algorithm for incorporation into our network design based on its inherent structural advantage over the highest ID algorithm and on its clear robustness advantage over the highest connectivity clustering algorithm.

Contributions found in Chapter 2 can be listed as follows:

- Simulated comparison of the three algorithms to identify the advantage of the highest and lowest ID clustering algorithms over the highest connectivity algorithms in terms of structural robustness;
- Elaboration of initial design objectives and constraints; and
- Identification and description of three clustering algorithms.

Chapter 3 focussed on the problem of multiple access to the common radio channel. We began this chapter by reviewing the principal common types of multiple access schemes and justified our choice of TDMA for our packet radio network. Working from this choice, we characterized the interference environment for each node of the network. We then considered the two approaches which may be taken to schedule the TDMA transmissions: broadcast and link scheduling. By complementing these scheduling approaches with our previous description of interference, we fully described the broadcast and link scheduling problems. Of the two scheduling approaches, we chose to implement broadcast scheduling based on its advantages over link scheduling in terms of cycle length, delay, and throughput. Having chosen broadcast scheduling, we then discussed the advantages and disadvantages of two specific broadcast scheduling algorithms, the topology-dependent ETBSA, and the topology-independent GRAND algorithm, and justified the choice of the GRAND algorithm for incorporation into our network design.

Chapter 3 contains the following contributions:

- Comprehensive comparison of link and broadcast scheduling, including a simulation to demonstrate the throughput characteristics of each;
- Comparison of the ETBSA and the GRAND algorithm, including a simulation which demonstrates that the advantage of the GRAND algorithm over the ETBSA in terms of cycle length is limited to certain low connectivity conditions; and
- Complete characterization of the link and broadcast scheduling problems, including a full analysis of the interference environment proper to each.

Network management issues were addressed in Chapter 4. Specifically, this chapter concentrates on defining the framework of a routing scheme customized for the linked cluster architecture. This is achieved by first reviewing the characteristics of routing in packet radio networks and by briefly describing the general routing types which currently exist. We then continued to describe in more detail two specific routing schemes, vector-length and hierarchical routing, which were found to be well suited for our network architecture. Working from these two routing schemes, we discussed how, in the context of routing, some properties of the linked cluster architecture can be of benefit, and we then proposed a general routing scheme for the linked cluster architecture. The contribution of this chapter is thus the elaboration of a distributed, hierarchical routing scheme for the linked cluster architecture, including a basic definition of the data frame structure.

Finally, Chapter 5 was dedicated to providing the implementation details of the network. We began the chapter by defining how the total available bandwidth is divided into the data and control channels and by defining the exact structure of each of these channels. With respect to the control channel, we saw how it can be used to support a distributed implementation of the lowest ID clustering algorithm. Specifically, we saw how this control channel can be used to determine connectivity; elect clusterheads; and designate gateways for each of the three communication modes. With respect to the data channel, we described in full detail how the broadcast schedule is constructed by applying the GRAND algorithm, and we provided a method for determining network global operating parameters. The contributions of this chapter thus include the following:

- A complete description of the lowest ID distributed implementation, including procedures for determining network connectivity; electing clusterheads; and designating gateways;
- Definition of the control channel frame structure;
- Full description of the GRAND algorithm;
- Simulation results indicating the existence of some GRAND algorithm validity regions and elaboration of a method for determining global network operating parameters.

6.2 Future Directions

From the beginning of this thesis, it has been our objective to develop a general and flexible framework which may be customized to fit specific operating parameters, both in terms of a specific radio environment and precise user requirements. We believe that we have met this objective. That is not to say that our packet radio network design is complete. We wish to conclude this thesis by proposing future directions which, in our opinion, offer interesting prospects for further research and development:

- **Radio channel modeling** - The results that we have presented in this thesis make no assumptions about the nature of the radio channel. This has allowed us to make broad and general design choices which are independent of this radio channel. We envisage that the next step in the design of our packet radio network will be to customize the framework presented in this thesis to fit a specific operating environment. This necessarily involves the choice of operating frequency and physical environment and thus implies relatively precise radio channel modeling. We believe that a channel model for the UHF band in a semi-urban environment, with a $r^{-\alpha}$ propagation loss factor and log-normal shadowing constitutes an interesting starting point.

- **Interference modeling** - Throughout this thesis, we have said that a link exists between two nodes if the signal power at the receiver is above a given threshold. Consider the case where two sending nodes s_1 and s_2 are allowed to simultaneously transmit to destination nodes d_1 and d_2 respectively. In this case, simultaneous transmission is possible because no link exists between s_1 and d_2, and vice-versa. That is not to say that some of the energy from s_1's transmission will not reach d_2, adding to an already noisy channel. In the generalized case where some subset of all nodes are allowed to simultaneously transmit, we ask ourselves how this interference can be modeled. Clearly, this interference will be some stochastic process dependent on both the radio channel properties and the statistical nature of the network in terms, for example, of the number of simultaneous transmitters and their physical layout. We believe that this interference may constrain the values of transmitter power to certain ranges and may even justify the requirement for an effective power control mechanism.

- **Routing protocol definition** - In Chapter 4, we developed a general description of a routing scheme appropriate for the linked cluster architecture. Though we provide a high-level description of this routing scheme, implementation inescapably requires a complete, low-level definition of the protocol. This definition will surely require additions to the current control channel frame structure to support the exchange of membership information between clusterheads; application of the Bellman-Ford algorithm to define intercluster routes; and the distribution of HRTs to the nodes of each cluster.

- **Control channel simulation** - With the above three tasks completed, one can undertake to implement the clustering and routing algorithms over the control channel. The main concern in this case would be to both demonstrate their validity — show that they yield clusters and routes which are consistent with the clusters and routes determined by a fictitious, central controller — and to gauge their robustness.

- **General network simulation** - Having proven the effectiveness of the control channel, the next step would consist in simulating a general network with real user traffic being generated and carried over the data channel. The first step in this simulation would be to determine global operating parameters, as described in Chapter 5. With these parameters determined, simulation could first be carried out for each of the communication modes alone and then be followed with a general, multimedia traffic model. In all cases, one would be interested in measuring the throughput and delay characteristics of the network under varying traffic loads.

- **Alternative multiple access schemes** - Finally, alternative multiple access schemes should be investigated for both the data and control channels. In the case of the control channel TDMA cycle, large networks will translate into an unduly long control channel TDMA cycle. In this case, it might be interesting to investigate the possibility of a contention-based control channel design. In the case of the data channel, using a fixed TDMA

schedule may prove to be an inefficient approach under light, bursty traffic conditions, though we believe that the GRAND algorithm largely contributes to alleviating this problem. To address this issue, one might consider implementing some type of demand assignment multiple access scheme such as PRMA. An interesting aspect of this problem would be to see how such a scheme, which traditionally requires the allocation of resources from a central location, might be implemented in a distributed fashion.

Bibliography

[1] Wing C Quan and E. Robert Sive. POST-2000 Tactical communications systems for NATO. *IEEE Communications Magazine*, pages 113–118, October 1995.

[2] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, mobile, multimedia radio network. *Wireless Networks*, 1(3):255–265, 1995.

[3] Jianhua Lu and Justin C-I Chuang. Wireless networking architecture and signaling for rapid deployment of rural communications. In *Proceedings of the 1996 International Conference on Communication Technologies*, volume 2, pages 702–705, Beijing, China, 1996.

[4] Barry Leiner, Donald Nielson, and Fouad Tobagi. Issues in packet radio network design. *Proceedings of the IEEE*, 75(1):6–20, January 1987.

[5] Kaveh Pahlavan and Allen H. Levesque. *Wireless Information Networks*, chapter 11. John Wiley and Sons Incorporated, 1995.

[6] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*, chapter 8. Prentice Hall Incorporated, 1996.

[7] Dennis Baker, Anthony Ephremides, and Julia Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE Journal on Selected Areas in Communications*, SAC-2(1), January 1984.

[8] Anthony Ephremides, Dennis Baker, and Jeffrey E. Wieselthier. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 75(1):56–73, January 1987.

[9] Anthony Ephremides. Distributed protocols for mobile radio networks. In *NATO ASI Series*, number 91 in Series E: Applied Sciences, pages 287–306. Martinus Nijhoff Publishers, 1983.

[10] Chunghung Richard Lin and Mario Gerla. A distributed architecture for multimedia in dynamic wireless networks. In *Proceedings of the 1995 IEEE Global Telecommunications Conference*, volume 2, pages 1468–1472, Singapore, Singapore, 1995.

[11] Chunghung Richard Lin and Mario Gerla. Multimedia transport in multihop dynamic packet radio networks. In *Proceedings of the 1995 International Conference on Network Protocols*, pages 209–216, Tokyo, Japan, 1995.

[12] Subramanian Ramanathan and Errol L. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking*, 1(2):166–177, 1993.

[13] Israel Cidon and Moshe Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Transactions on Computers*, 38(10):1353–1361, 1989.

[14] David S. Stevens and Mostafa H. Ammar. Evaluation of slot allocation strategies for TDMA protocols in packet radio networks. In *Proceedings of the 1990 IEEE Conference on Military Communications*, volume 2, pages 835–839, Monterey, California, 1990.

[15] Imrich Chlamtac and Shlomit S. Pinter. Distributed nodes organization algorithm for channel access in a multihop dynamic radio networks. *IEEE Transactions on Computers*, C-36(6):728–737, 1987.

[16] A. Ephremides and T Truong. Distributed algorithm for efficient and interference-free broadcasting in radio networks. In *Proceedings of the 1988 IEEE Conference on Computer Communications*, pages 1119–1124, 1988.

[17] Anthony Ephremides and Truan V. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, 38(4):456–460, 1990.

[18] I. Chlamtac and A. Lerner. A link allocation protocol for mobile multi-hop radio networks. In *Proceedings of the 1985 IEEE Global Telecommunications Conference*, pages 238–242, New Orleans, Louisiana, 1985.

[19] D. Baker, J Wieselthier, and A Ephremides. A channel access protocol for survivable radio networks with frequency hopping spread spectrum signaling. In *Proceedings of the 22nd Annual Allerton Conference on Communication, Control, and Computing*, pages 50–59, 1984.

[20] Imrich Chlamtac and Andrad Farago. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, 1994.

[21] Gangsheng Wang and Nirwan Ansari. A neural network approach to bradcast scheduling in multi-hop radio networks. In *Proceedings of the 1994 IEEE International Conference on Neural Networks.*, volume 7, pages 4699–4703, Orlando, Florida, 1994.

[22] Julie Shor and Thomas G. Robertazzi. Traffic sensitive algorithms for generating self-organizing network schedules. In *Proceedings of the 1990 IEEE Conference on Military Communications*, volume 2, pages 840–844, Monterey, California, 1990.

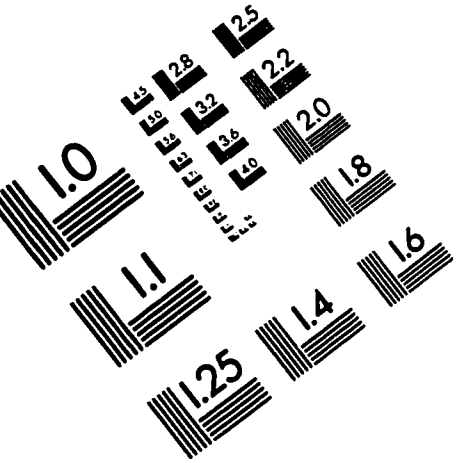
[23] Martha Steenstrup. *Routing in Communications Networks*, chapter 11. Prentice-Hall Incorporated, 1995.

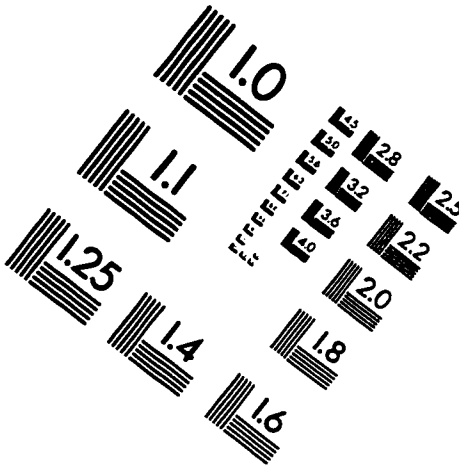
[24] Martha Steenstrup. *Routing in Communications Networks*, chapter 3. Prentice-Hall Incorporated, 1995.

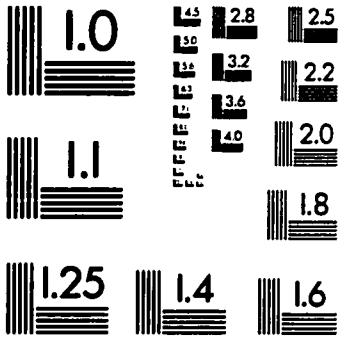
[25] William Stallings. *Data and Computer Communications*, chapter 8. Macmillan Publishing Company, 1994.

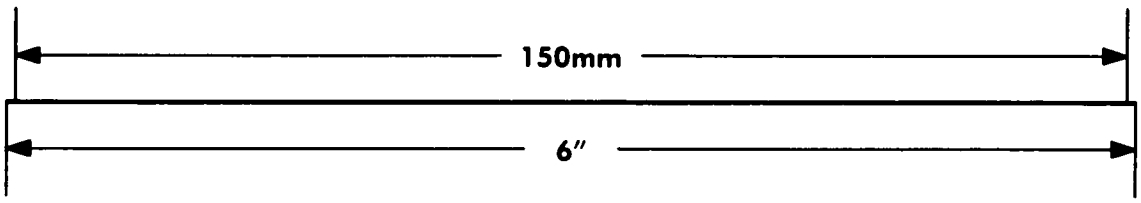
[26] Charles E. Perkins and Pravin Bhagwhat. Highly dynamic destination-sequenced distance-vector routing for mobile computers. In *Proceedings of SIGCOMM 94*, pages 234–244, 1994.

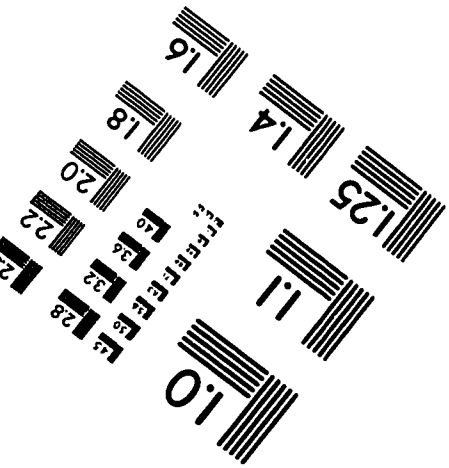
IMAGE EVALUATION
TEST TARGET (QA-3)

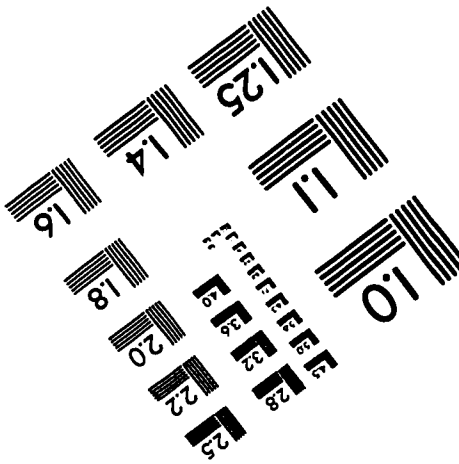












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989