An Incremental Algorithm for Broadcast Scheduling in Packet Radio Networks

Xiaopeng Ma and Errol L. Lloyd
Department of Computer and Information Sciences
University of Delaware,
Newark, DE 19716

I. ABSTRACT

Packet radio networks hold great promise for providing easy to use, mobile military communications services. However, supporting mobility with packet radio networks poses many technical challenges. One of these challenges, to dynamically and quickly adjust schedules for fixed channel access by way of TDMA/FDMA/CDMA in a large dynamic network, is the focus of this paper. In such broadcast scheduling it is required that the transmission of a given station be received collision free by all of the stations that are within its transmission range. The goal in broadcast scheduling is to produce a conflict free static schedule of minimum length in which each station is given a chance, a time slot in TDMA for example, to broadcast its packets. A number of heuristic broadcast scheduling algorithms have been proposed. Unfortunately, all of the existing algorithms are off-line. As such, whenever a change occurs in the topology of the network, these algorithms recompute the schedule for the entire network. As radio networks evolve in the direction of thousands of stations spread over a broad geographical area and operating in an unpredictable dynamic environment (imagine a battlefield) the cost of full schedule recomputation whenever the network topology changes (by even a single node) is unacceptable. In this situation, incremental algorithms are needed. Such algorithms accommodate topological changes by adapting the existing schedule, rather than rebuilding the schedule from scratch. The twin objectives of incremental algorithms are much faster execution (than an off-line algorithm) and the production of a high quality schedule. In this paper we establish the feasibility of simultaneously meeting these twin objectives, by presenting an incremental algorithm for broadcast scheduling.

II. INTRODUCTION

A *radio network* is a network of *stations* that communicate via wireless links using radio signals. The stations share a common channel and each station acts both as a host and as a switching unit. In this scheme, no station is more critical than any other in performing network functionalities. As a result, there is a greatly reduced possibility that a single point of failure will crash the network. Furthermore, because PRNs do not contain fixed base stations, they avoid the time intensive process of deploying those base stations. Thus, a packet radio network is especially useful in military applications where the construction of fixed base stations is impossible or where the network must be constructed rapidly from scratch.

Sharing a common channel introduces the question of how the channel is accessed by the stations in the network. *Fixed access* protocols, using the common channel by way of TDMA/FDMA/CDMA, are applicable in situations where the network is heavily loaded or the maximum delay must be bounded or where the network is multi-hopped. *Broadcast scheduling* the focus of this paper, is a fixed channel access technique.

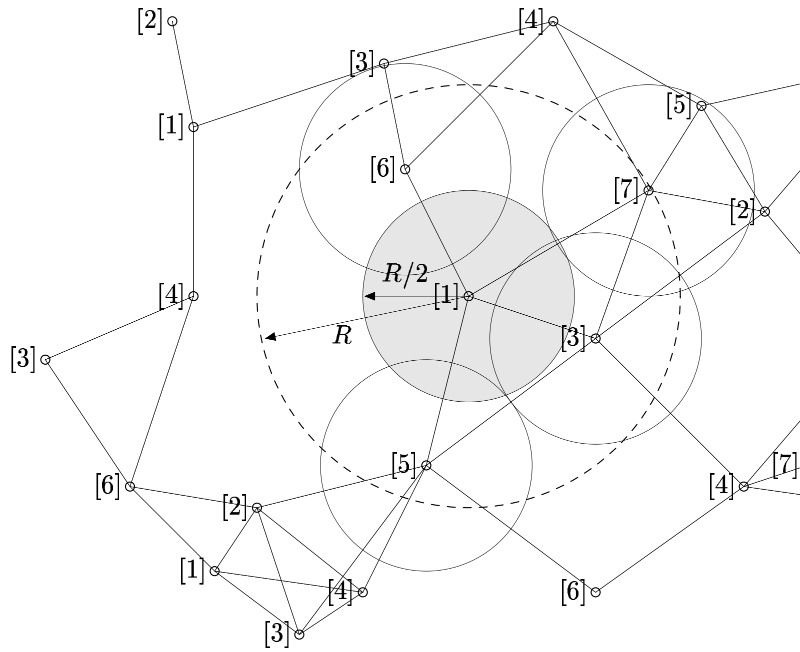
A. A Model of the Problem

In a broadcast environment, using a fixed channel access method (such as TDMA), it is required that the transmission of a given station be received collision free by all of the stations that are within its transmission range. Here, collisions caused by simultaneous sending and receiving are *primary conflicts*, and collisions caused by a single station receiving packets simultaneously from multiple sources are *secondary conflicts*. The goal for *broadcast scheduling* in a PRN is to produce a conflict free broadcast schedule of minimum length in which each station is given a chance (e.g. a time slot in TDMA) to broadcast its packets.

As in past works, we model packet radio networks by graphs, with nodes corresponding to stations, and neighbor relationships modeled by links. Since the transmission powers of various stations are often adjustable, it is standard to assume that all of the stations have the same transmission range and that all of the links are bidirectional. We will likewise make these assumptions in this paper. With such assumptions, the graphs derived from PRNs are modeled by a class of undirected graphs termed *unit disk graphs*[3]. A *unit disk graph* $G = (V, L, d)$ is an undirected graph for which there exists a planar embedding of the nodes such that there is a link between nodes u and v if and only if the Euclidean distance between u and v does not exceed d.

The problem of *broadcast scheduling* can itself be abstracted as follows: Since all links are bidirectional, a schedule that is free of primary conflicts requires that none of the neighbors of a station can transmit in the time slot in which the station is transmitting. Similarly, a schedule that is free of secondary conflicts requires that at any given time at most one of the neighbors of a station is transmitting. Hence, the problem of broadcast scheduling in a packet radio network is naturally modeled by *distance-2 graph coloring*[1]. Here, different time slots in TDMA that must be assigned to different stations, are translated into different *colors* that are assigned to the nodes representing those stations. A schedule being free of primary conflicts can be modeled by the constraint that no two nodes can be assigned the same color if there is a link between them. And, a schedule being free of secondary conflicts can be converted into the constraint that no two nodes can be assigned the same color if they have links to a common node. Figure 1 shows a PRN modeled in this way, and one coloring (broadcast schedule) of that PRN.

[1]Given a graph $G = (V, L)$, *distance-2 graph coloring* is to produce an assignment of colors $C : V \longrightarrow \{c_1, c_2, \ldots\}$ (where $c_1, c_2, \ldots$ denote different colors) such that no two nodes are assigned the same color if the two nodes are distance-2 neighbors. The *distance-2 neighbors* of a node include all of its direct neighbors and the direct neighbors of its direct neighbors.



There is a link between a pair of nodes if and only if the circles
nodes intersect (including being tangent). This is equivalent to
nodes if and only if the Euclidean distance between the two corr
The colors assigned to each node are the numbers in the bracket

Fig. 1. Broadcast scheduling and unit disk graphs

B. Off-line Algorithms

The problem of broadcast scheduling in a packet radio network has been extensively studied [4], [5], [6], [7], [8], [9], [10] in the context of off-line algorithms. Both theoretical and experimental results have been established. The reader is referred to the cited papers for descriptions of a full range of methods. In this paper we mention only two algorithms: 1) *The Pure Greedy Algorithm*: This algorithm is both advocated for use in practice due to its simplicity, and is used[6], [?] as a base line method against which the performances of other algorithms may be compared; and 2) *max_cont_color*[9]: A simulation study[9] showed that this algorithm is the strongest of all broadcast scheduling methods.

C. Issues Associated with Incremental Scheduling

In the early stage of the development of packet radio networks, many PRNs were one hop networks performing *real* broadcast. For such PRNs, off-line algorithms were entirely appropriate. At present however, radio networks are evolving towards thousands of stations spread over a broad geographical area and operating in an unpredicatable dynamic environment. This is particularly so in military situations. Such *global personal communication* defeats the usefulness of off-line algorithms since it is absolutely unaffordable to halt all communication whenever there is a change in the network, so as to produce a new schedule from scratch. In such circumstances, incremental approaches are required. That is, given a schedule for a PRN, if that PRN changes, then the schedule should be appropriately updated to a schedule for the modified PRN. While there are a variety of ways in which a PRN can change, in this paper we consider only topological changes introduced by the joining or leaving of a station.

What are the effects on an existing schedule of the joining or leaving of a station? The primary effect of the joining of a station is that conflicts may be introduced among the direct neighbors of the joining station. It is obvious that the leaving of a station never causes conflicts in the existing schedule. Hence, the *basic functionalities* of an incremental scheduling algorithm are to add/remove the affected station to/from the network, to allocate a color to a joining station, and to recolor nodes as appropriate so as to resolve conflicts introduced by a joining station.

An *incremental algorithm for broadcast scheduling* is one that, given a PRN, a broadcast schedule for that PRN, and a change in the PRN (i.e. either a station joining or leaving the network), produces a broadcast schedule for the new PRN. The twin objectives of incremental algorithms are:

- much faster execution per change than off-line methods (which would be required to compute a complete new schedule in response to each change);
- the production of a *high quality* schedule.

The major question answered by this paper is: **Are incremental algorithms feasible?**

That is, do there exist incremental algorithms for broadcast scheduling that meet the twin objectives? In the sections below we establish that the answer is *yes*.

III. AN INCREMENTAL SCHEDULING ALGORITHM

In this section, we present an incremental broadcast scheduling algorithm, *COMPRESS_COLOR*. This algorithm both performs the basic functionalities of incremental scheduling, and attempts periodically to reduce the schedule length. *COMPRESS_COLOR* includes three primary procedures: *INSERTION* which handles the joining of a station to the PRN; *DELETION* which handles the leaving of a station from the PRN; and, *MAINTAINENCE* which is called by the other two procedures, and which will monitor the schedule length in relation to a known lower bound on the optimal length. Only when the schedule length exceeds that lower bound on optimal by a specified amount will *MAINTAINENCE* actively attempt to reduce the schedule length. The details of the three procedures follow.

A. Procedures INSERTION and DELETION

In these procedures, *MAX_COLOR* is a global variable indicating the maximum color currently assigned to nodes. Also, for any node v, $COLOR(v)$ is the color that has assigned to that node.

proc *INSERTION*(*A*)
  begin
    increase the degrees of the direct neighbors of A (and i
    RECOLOR_LIST ← *min_conflict_set*(*A*);
    uncolor the nodes in RECOLOR_LIST;
    add A to RECOLOR_LIST;
    while *RECOLOR_LIST* ≠ *NULL* **do**
      v ← *node in RECOLOR_LIST most constrained*
      COLOR(v)← *least color unused by its distance* −
      delete v from RECOLOR_LIST **od**
    call *MAINTAINENCE*;
  end.

The function *min_conflict_set*(*A*) invoked by *INSERTION* constructs the set of nodes that will have to recolored as a result of A joining the network. It does this in the following fashion: Initially this "minimum conflicting set" is empty. Then, for each color c, the procedure identifies all of those nodes of color c that will conflict once A joins the network. These nodes are the direct neighbors of A that are colored c. These nodes are all placed into the minimum conflicting set, except for the one that is most constrained by its distance-2 neighbors (the node not placed into the minimum conflicting set is allowed to retain its color of c).

Next we have the procedure that handles station A leaving the network.

```
proc DELETION(A)
  begin
    decrease degrees of neighbors of A (and if necessary ρ and MAX_COLOR) by 1;
    call MAINTAINENCE;
  end.
```

B. Procedure MAINTAINENCE

```
proc MAINTAINENCE()
  begin
    if (MAX_COLOR/(1 + ρ)) > τ)
      then v ← a node with color MAX_COLOR;
             local_max ← 0;
             no_iteration ← 1;
             while (no_iteration ≤ 4) do
                    RECOLOR_LIST ←{v};
                    add all of v's direct neighbors to RECOL
                    uncolor each node in RECOLOR_LIST;
                    while RECOLOR_LIST ≠ NULL do
                           u ← node in RECOLOR_LIST mo
                           COLOR(u)← least color unused by
                           delete u from RECOLOR_LIST;
                           if (local_max < COLOR(u))
                             then local_max ← COLOR(u);
                                    v ← u;  fi od
                    if (local_max ≥ MAX_COLOR)
                      then no_iteration ← 1 + no_iteration
                       else no_iteration ← 5;  fi od
             if (local_max < Max_COLOR)
               then Reset MAX_COLOR fi fi
  end.
```

As noted above, *MAINTAINENCE* is called by the other two procedures, and monitors the schedule length in relation to ρ, the maximum degree of any node in the network. It is obvious that $\rho+1$ is a lower bound on the optimal schedule length. In monitoring the schedule length, *MAINTAINENCE* also utilizes a user defined parameter τ, called the *threshold*. Only when the *approximation factor* of the schedule (the ratio of the length of the present schedule divided by $(\rho + 1)$) exceeds the threshold, does the algorithm attempt to reduce the schedule length. Limiting the attempts to reduce the schedule length helps the algorithm maintain a good practical running time.

How does the procedure attempt to reduce the schedule length? The basic idea is to select a node of maximum color, and then to recolor that node and its direct neighbors, in the hope of reducing the number of colors that are utilized. The order in which those nodes are recolored depends upon the color constraints imposed by their already colored distance-2 neighbors with the node that is most constrained being recolored first. This "local" recoloring process may be repeated up to four times or until the local maximum color has been reduced.

C. Running Time

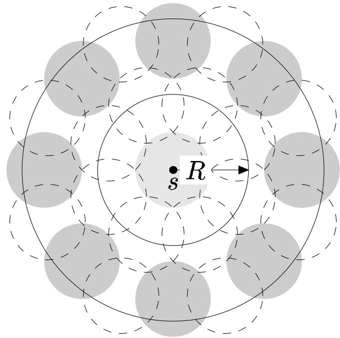
In this section we consider the running time of *COMPRESS_COLOR*. We begin by bounding the worst case running time, and follow with some remarks on the running time in practice.

Theorem 1: The worst case running time of *COMPRESS_COLOR* is $O(\rho^2)$ per joining or leaving operation.

Proof: The three critical parameters in the analysis are: ρ, the maximum node degree; D, the maximum distance-2 degree; and, R, the uniform transmission range of the nodes in the PRN.

It is easy to see that the running time for a single *INSERTION* or *DELETION* is $O(\rho D)$ or $O(\rho)$ if the *then of the outermost if* of *MAINTAINENCE* is not executed.

In *MAINTAINENCE*, if the *then of the outermost if* of *MAINTAINENCE* is executed, then there are at most ρ nodes in *RECOLOR_LIST*. Since $O(D)$ time is required to color a node, executing the entire *inner loop* requires $O(\rho D)$ time. Since the number of



$S-$cycles are shown here as both shaded and unshaded only to make them easier to visualize.

Fig. 2. $D \leq 25\rho$

iterations for recoloring a local area (the *outer loop*) is upperbounded by four, it follows that the time for recoloring a local area is also $O(\rho D)$ and the worst case cost of *INSERTION* and *DELETION* is $O(\rho D)$.

What is the relationship between ρ and D? For general graphs, D is bounded by ρ^2. However, in the case of unit disk graphs, we claim that D is $O(\rho)$. This is most easily seen by considering Figure 2. In that figure, s is an arbitrary node, and all of the distance-2 neighbors of s must lie within the circle of radius $2R$. Also in that figure, a number of circles of radius $R/2$ are shown (some are shaded and some are not). We call these S circles. Since all of the nodes within a given S circle must always be direct neighbors of each other, the maximum number of nodes within any single S circle is bounded by ρ (note that it does not matter if the circle is centered on a node or not). Further, the number of distance-2 neighbors of the node s is upperbounded by the number of nodes within its circle of radius $2R$. But, the region inside the circle of radius $2R$ can be completely covered by 25 S circles as shown in the figure. It follows that the number of nodes inside the circle of radius $2R$, and hence D, is bounded by 25ρ. Thus, D is $O(\rho)$.

Since D is $O(\rho)$, the total running time of *COMPRESS_COLOR* for a sequence of operations which includes N *INSERTION*s is $O(N\rho^2)$. Likewise, the worst case running time for *COMPRESS_COLOR* on a per operation basis is $O(\rho^2)$. *QED*

How good is this running time? We provide two measures. First, we note that any algorithm that performs the basic functionality of incremental scheduling requires time $O(\rho^2)$. In this sense, the running time of *COMPRESS_COLOR* is best possible. Second, by way of comparison, the running time of *max_cont_color* on a network with N stations is $O(N\rho^2)$. Thus, to produce a schedule for N nodes, *COMPRESS_COLOR* has an identical running time to the best off-line algorithm, *max_cont_color*, even while creating the schedule in a completely incremental fashion, and being able to also handle *DELETION*s in the same running time.

D. An Experimental Study

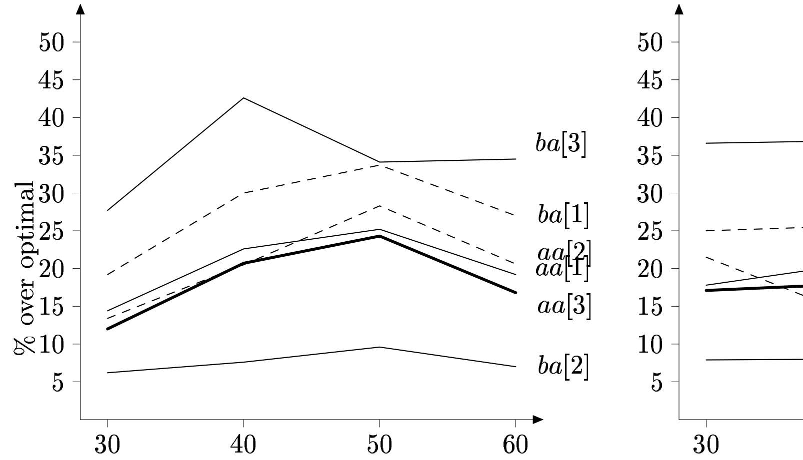
In our experiments, we studied the quality of the schedules produced incrementally by *COMPRESS_COLOR* against *Pure Greedy*, the most popular off-line method, and *max_cont_color*[9], the best of the off-line algorithms. In these experiments, networks were generated and maintained on a 400 by 400 grid. The transmission radius was varied between 30, 40, 50 and 60 grid units, and the number of nodes was varied between 300, 600, 900 and 1200. This resulted in a range of maximum node degrees from 11 to 115. There were 10 trials for each combination of transmission range and number of nodes, resulting in a total of 160 trials. The threshold τ of *MAINTAINENCE* was uniformly set at 1.06.

In each trial, the operations were performed in two stages. In the first stage, beginning with an empty network, nodes were joined one at a time to the network. Each node was uniformly randomly placed on the 400 by 400 grid. In this stage the *DELETION* operation of *COMPRESS_COLOR* was not involved. This stage represents the initialization phase of a PRN where stations are coming on line and forming the network. In the second stage, 600 operations of nodes joining or leaving the network were randomly performed with each type of operation being performed with a 50% probability. This stage represents the scenario of a mature PRN with nodes joining and leaving the network on a relatively steady state basis.

The results of the study are presented in Figure 3 and Figure 4. The graph in Figure 3 show the average results for 1200 nodes for each of the four transmission ranges. Figure 4 plots the combined results from all of the experiments and explores the relationship between the maximum node degree and the simulation performance for the second stage. In the figures, ba[1] labels the performance line for *pure greedy*; ba[2] labels the performance line for *max_cont_color*; and aa[1] labels the performance line for *COMPRESS_COLOR*. The y-coordinates in all of the figures denote performance: an algorithm with a value y on the y-coordinate indicates that the algorithm used y percent more colors than the estimated optimal (the estimated optimal is $1 + \rho$ where ρ is the maximum node degree).

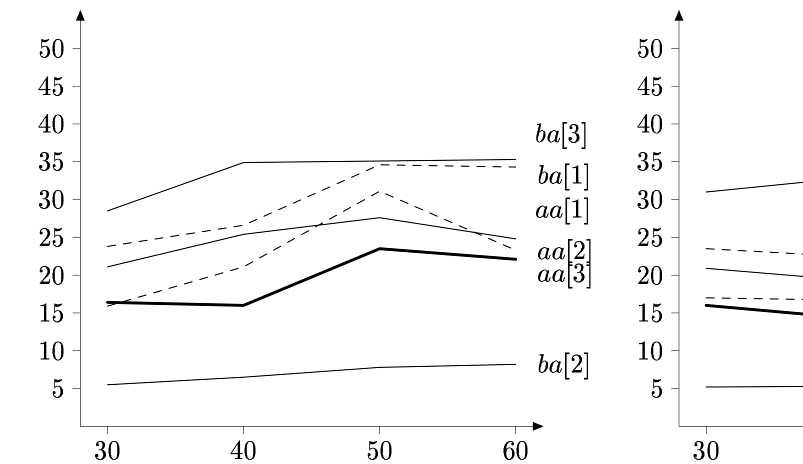
The basic conclusions that we draw are:

- The performance of *COMPRESS_COLOR* is uniformly superior to that of *Pure Greedy* (the most popular off-line method).
- The performance of *COMPRESS_COLOR* is not as strong as that of *max_cont_color*, the best off-line method.
- There is little difference in the performance of *COMPRESS_COLOR* between the first and second stages, indicating that the *INSERTION* and *DELETION* routines are well matched.
- The relative performances of all three algorithms (*COMPRESS_COLOR*, *Pure Greedy*, and *max_cont_color*) are very consistent in both stages relative to the maximum node degree.



(900a) first stage results for 900 nodes

(900b) secon



(1200a) first stage results for 1200 nodes

(1200b) secon

Fig. 3. Simulation Performance

IV. OPEN QUESTIONS

There are two primary open questions that we would like to settle: one practical and one theoretical. The practical question is to develop a distributed incremental scheduling algorithm. The hope is that when a station joins or leaves the network, the schedule can be adjusted by local computations of the nodes in some neighborhood of the affected node. The primary theoretical problem is to develop incremental broadcast scheduling methods whose *competitive ratio* is bounded by some constant. The ideal in this regard would be to have the incremental competitive ratio match the best off-line competitive ratio (presently, 7).

Disclaimer: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

REFERENCES

[1] Clifford A. Lynch. *Packet radio networks: architectures,protocols, technologies,and applications.* Oxford, England, NewYork, Perfamon-Press, 1987.

[2] Martha Steenstrup. *Routing in Communications Networks.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1995.

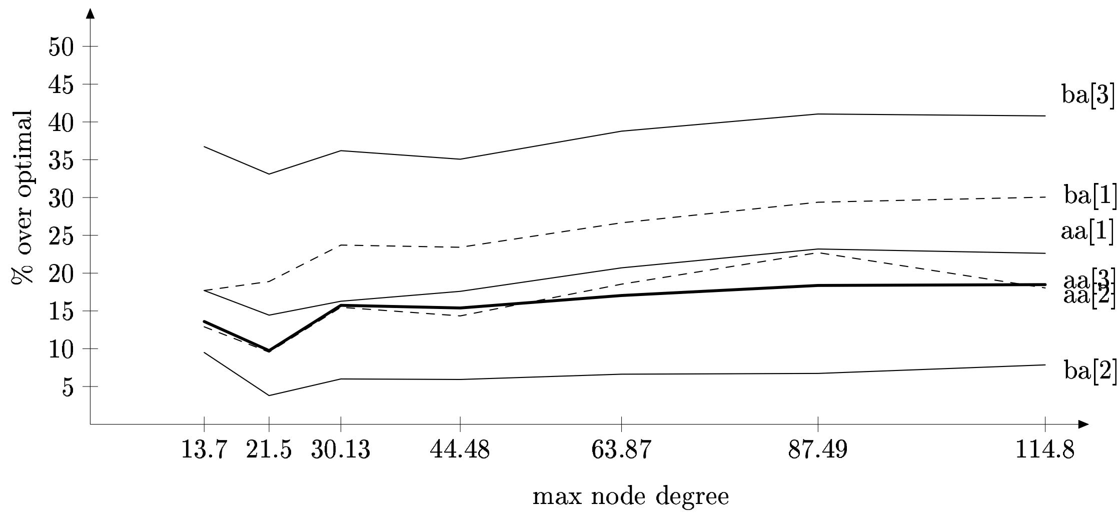


Fig. 4. Max Degree vs Performance for Second Stage

[3] Brent N. Clark, Charles J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, pages 165–167, 86(1990).

[4] Errol L. Lloyd and Ram Ramanathan. Efficient distributed algorithms for channel assignment in multihop radio networks. *IEEE Transactions on communications*, 2:405 – 423, 1993.

[5] Ram Ramanathan and Errol L. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Trans. on Networking*, 1:166–177, 1993.

[6] Arunabha Sen and Mark L. Huson. A new model for scheduling packet radio networks. *IEEE INFOCOM*, pages 1116–1124, 1996.

[7] Mark L. Huson and Arunabha Sen. Broadcast scheduling algorithms for radio networks. *IEEE MILCOM*, pages 647–651, 1995.

[8] Rajiv Ramaswami and Keshab K. Parhi. Distributed scheduling of broadcasts in radio network. *IEEE INFOCOM*, 2:497–504, 1989.

[9] Errol L. Lloyd and Xiaopeng Ma. Experimental results on broadcast scheduling in radio networks. *ATIRP Conference*, 1996.

[10] Ram Ramanathan. *Scheduling algorithms for multi-hop radio networks.* Department of Computer and Information Sciences,University of Delaware, Newark, DE, 19716, 1992.