A DISTRIBUTED PROTOCOL FOR ADAPTIVE BROADCAST SCHEDULING IN PACKET RADIO NETWORKS
Extended Abstract *

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

Abstract

A *fully distributed* protocol for adaptive broadcast scheduling in multi-hop packet radio networks is presented. This method requires neither a central computing site nor individual stations to maintain global information other than that of a global clock. A station determines its actions solely on information that concerns the transmission status of its one-hop and two-hop neighbors. This fact allows a fully distributed implementation (the first such for broadcast scheduling), and makes the method highly adaptive. The protocol that we describe is comprehensive in: having each station determine its own slot(s) in the schedule; allowing multiple stations to simultaneously run the scheduler portion of the protocol; allowing stations in non local portions of the network to remain operating even while other stations are joining or leaving the network; allowing stations to utilize otherwise empty slots; and, from a theoretical viewpoint, having a constant competitive ratio.

1 Introduction

A *packet radio network* (PRN) is a network of stations (also known as *transceivers*) that communicate by sharing a common channel. By way of this common channel, the transmission of a station can be received by all of the stations within its transmission range.

PRNs provide robust communication, can be rapidly deployed in any type of terrain, and hold great promise for providing easy to use, mobile military communications services. However, sharing a common channel poses many technical difficulties in the aspect of channel access control. This paper discusses those challenges and presents a fully distributed broadcast scheduling method with many distinguishing qualities.

1.1 Basic broadcast scheduling concepts

In a broadcast packet radio network, it is required that the transmission of a station be received collision free by all of its one-hop neighbors. To be collision free it is required that no station transmits and receives simultaneously and that no station receives from more than one station simultaneously. A collision caused by transmitting and receiving simultaneously is a *primary conflict*. A collision caused by receiving from different stations simultaneously is a *secondary conflict*.

*Prepared through participation in the Advanced Telecommunications-Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002.

The task of a *broadcast scheduling algorithm* is to produce and/or maintain an infinite *schedule* of slots[1] that each station in the PRN is periodically assigned a slot for transmission and so that all transmissions are received collision free. In this framework, all of the existing broadcast scheduling algorithms operate by producing a *nominal schedule* in which each station is assigned exactly one slot for transmission and then indefinitely replicating that nominal schedule to produce an actual broadcast schedule.

As is common in prior works, we assume that all of the radio links in a PRN are bi-directional, that stations utilize a uniform transmission range R, and that the transmission of a station will be received by all stations within a Euclidean distance R. Thus, the topology of a PRN is naturally modeled by a *unit disk graph*[1] $G = (V, L, R)$. In this model, each station in the PRN is denoted by a node in a graph. Bi-directional radio links in the PRN are denoted by graph edges, with an edge existing between nodes u and v if and only if the Euclidean distance between stations u and v does not exceed R. Relevant to broadcast scheduling and such graphs is the problem of *distance-2 coloring*:

- Given a graph $G = (V, L)$, produce an assignment of colors $C : V \longrightarrow \{c_1, c_2, \ldots\}$ (where $c_1, c_2, \ldots$ denote symbolic names for different colors) such that no two nodes are assigned the same color if the two nodes are distance-2 neighbors (i.e. the distance between the nodes is 1 or 2).
- An *optimal solution* is a coloring utilizing a minimum number of colors.

Broadcast scheduling is directly abstracted to distance-2 coloring. In this abstraction, slots that are assigned to stations are translated into colors that are assigned to nodes. It can be seen that a schedule free of collisions is a schedule in which no two stations are assigned the same slot/color if the two stations are one-hop or two-hops from each other (one or two away neighbors). In this paper, we will interchangeably use the terms PRN and graph, station and node, slot and color.

1.2 What is adaptive broadcast scheduling?

Historically, broadcast scheduling algorithms have assumed that the network topology is provided in its entirety to the scheduling algorithm, and the algorithms have been designed with the goal of producing a schedule for all stations in the network. The presumption has been that in the event that the network topology changes, then the algorithm is rerun for the entire network. However, as radio networks are evolving towards the direction of thousands of stations spread over a very broad geographical area and operating in an unpredictable dynamic environment, the use of such *off-line* scheduling algorithms is not realistic. In practice, 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, *adaptive* algorithms are required. That is, given a schedule for a PRN, if that PRN changes (by the joining or leaving of a station), then the schedule should be appropriately updated to correspond to the modified PRN. Thus, an *adaptive 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 adaptive algorithms are much faster execution (than an off-line algorithm that computes a complete new schedule) and the production of a provably *high quality* schedule.

[1] Term *slot* have different meanings in different contexts. For example, in the context of FDMA, a slot means a band of frequency in the context of TDMA, a slot means a time slot. In this paper, TDMA is assumed. Although we have this assumption, the adaptive scheduling method can well be applied to the contexts of FDMA and CDMA

1.3 An overview of earlier results

Fixed channel access has been an active research area for a decade as a promising technique for channel access control in multi-hop packet radio networks. A number of methods have been proposed, all of which rely on the production of a nominal schedule that is then replicated indefinitely. The natural goal in these earlier works has been to minimize the length of the nominal schedule.

1.3.1 Competitive ratios in earlier algorithms

Virtually all of the work on broadcast scheduling has dealt with the off-line case [2, 3, 4, 5, 6, 7]. Together these works considered both theoretical and experimental evaluations of schedule quality.

From the theoretical side, all of the existing work is based on the nominal schedule approach mentioned above. In that context it is known that for PRNs modeled by arbitrary graphs, finding an optimal nominal schedule is NP-complete [8, 4, 9]. Thus, the primary focus of earlier work has been on producing good, though non-optimal schedules. In the context of such *approximation algorithms for broadcast scheduling*, the standard measure of the goodness of the algorithm is it's *competitive ratio*. That is, the length of the nominal schedule produced by the approximation algorithm to the length of an optimal nominal schedule. The earliest results on competitive ratios for broadcast scheduling appeared in [4], where a centralized greedy algorithm is presented having a competitive ratio proportional to the maximum node degree. In [10] it was shown that for arbitrary PRN topologies (i.e. the topology need not be a unit disk graph) then the competitive ratio of the algorithm in [10] is $O(\rho)$, where ρ is the thickness[2] of the graph. It follows that the competitive ratio is a constant when the PRN topology is planar. The first modeling of PRN's by unit disk graphs appeared in [2]. In later work focusing on unit disk graphs [7] it was shown that a number of the off-line algorithms have competitive ratios of 13. The strongest results to date present algorithms with competitive ratios of 7 [7]. Finally, in the only existing work on *adaptive* methods, a family of three adaptive algorithms is presented in [11]. Each of these methods has a competitive ratio of 13.

From the experimental side, there are a number of studies on various broadcast scheduling algorithms [3, 7, 10, 6]. These studies have shown that the strongest of the existing broadcast scheduling algorithms in the off-line case for PRNs modeled by unit disk graphs are those of [7] and [3], each of which typically produces nominal schedules within 12% of the optimal length nominal schedule. Likewise, the strongest of the adaptive algorithms presented in [11], typically produces nominal schedules within 20% of optimal.

1.3.2 Deficiencies of earlier methods

Although existing research has been successful in elucidating the complexity of broadcast scheduling based on the production of a nominal schedule, all of the earlier work exhibits major deficiencies of one sort or another:

- All of the existing algorithms require either that a priori knowledge of the topology of the entire network be maintained at a central computing site [4, 3, 10, 7] or require that some global information (such as an existing schedule length) be maintained at each station [5, 11]. These requirements severely restrict the ability to implement the method in a highly distributed fashion.
- All of the existing methods operate under the "one station at a time" principle in computing the nominal schedule. Most of the methods are centralized, in that the nominal schedule is computed at a single site and then distributed to the other nodes in the network. When computing the

[2]The *thickness* of a graph G is the minimum number of edge-disjoint planar subgraphs into which G can be partitioned

nominal schedule at the central site, the algorithm necessarily operates sequentially in determining the slot for each node in turn. In practice, these methods are infeasible due to the long delay times required to distribute the schedule. Existing non centralized methods (such as [5]) allow computation to occur at each node in the network to compute the slot for that node in the nominal schedule. Unfortunately, these computations are done sequentially, for example the method presented in [5] requires that a token be passed around the network, with a node computing only when it holds the token.

- With one exception [11], the methods are not *adaptive*. That is, given a nominal schedule for a PRN, and given a change in that PRN (for instance, a new station joins the network), earlier algorithms completely recompute the nominal schedule, rather than trying to adjust the existing schedule.
- The methods make no attempt to utilize unused slots in the broadcast schedule (this will be discussed further in section 2.4).
- The scheduled networks have poor performance in situations where traffic is unbalanced and/or bursty. In such situations, some of the stations with heavy transmission requirements during a period may be forced to drop packets for lack of transmission time while other stations may let their scheduled slots go unused.

As a result of the above, all of the existing broadcast scheduling methods lack scalability and/or adaptiveness. Thus these methods are impractical for large scale multi-hop packet radio networks where thousands of stations are spread over a broad area and topologies change constantly and unpredictably.

2 FDAS—A Fully Distributed Method For Adaptive Scheduling

In this section, we present a fully distributed method for broadcast scheduling. We assume only that stations have the capacity of global slot synchronization.

The protocol that we present is distinguished in the following ways:

- No global information is required, either in central or individual sites. Rather, a station schedules itself after collecting information from it's one-hop and two-hop neighbors.
- The method is fully distributed. Thus, multiple stations can simultaneously run the scheduler portion of the protocol, and stations in non local portions of the network can transmit normally even while other stations are joining or leaving the network.
- The relative overhead in the network bandwidth is minimal.
- The method has a constant competitive ratio (though admittedly large: 26).
- Stations have the capacity to utilize otherwise unused slots.
- Stations can temporarily adjust the schedule locally to provide good utilization of the network bandwidth when the traffic is unbalanced and/or bursty.

In summary, the protocol presented in this paper not only can be implemented in a highly distributed fashion, but also has the flexibility to adjust the schedule in response to traffic changes and various traffic patterns. We term this protocol *Fully Distributed Adaptive Scheduling (FDAS).*

There are three aspects to the protocol that we present: the basic scheduling method, the adaptive aspect and the distributed aspect. In the three sections that follow we address each of these in a cumulative fashion.

2.1 The basic scheduling method (non-adaptive)

In this section we describe the basic scheduling method. Our description here focuses on the method itself and not on the distributed implementation aspect, which we will be dealt with in a later subsection. It *is* the case that the method very much facilitates the distributed implementation, particularly in comparison with earlier scheduling methods.

Diverging from the "nominal schedule" approach taken by existing algorithms for broadcast scheduling, we consider a scheduling framework that focuses on determining two essential components for each station: its *transmission slot* and its *transmission cycle*. Here, the station transmits for the first time in its transmission slot, and then every transmission cycle number of slots thereafter. Clearly, the transmission cycle is a fixed number of slots between two consecutive transmissions.

Note that the schedules produced using the nominal schedule approach can be also be framed in this context. There, although the term transmission cycle is not explicitly mentioned in any of the existing protocols for broadcast scheduling, each of these protocols implicitly utilizes the maximum number [3] for indexing the slots in the nominal schedule as the uniform transmission cycle for each station. Hence, each station needs to know not only which slot it is assigned, but also the maximum indexing number in the nominal schedule.

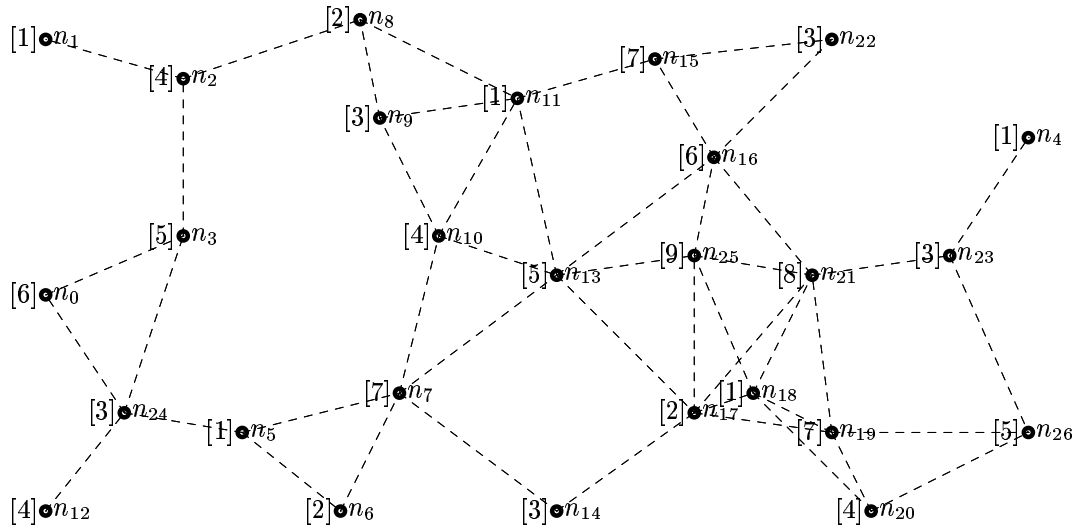
The approach we take in this paper is that a priori there is no reason why each station must utilize the same transmission cycle as every other. All that is required is that transmissions be received collision free and that each station have a periodic opportunity to transmit. Our algorithm operates using nonuniform transmission cycles. By so doing we enable each station to produce its transmission slot and transmission cycle locally, without the need for global information. Obviously these selections need to be made carefully so that collisions are never present.

Transmission Slot Since the transmission slot specifies the first time that a station transmits, the assignment of a transmission slot to a station is virtually identical to the problem faced in algorithms using the nominal schedule approach. Specifically, a transmission slot must be assigned to a station so as not to conflict with the transmission slots of the station's one-hop and two-hop neighbors. This is identical to the problem of distance-2 coloring, and we use the coloring terminology in what follows.

A slot assignment protocol *T_SLOT_ASSIGN* is given below. There, *min_conflict_set(A)* 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: This "minimum conflicting set" is initially empty. Then, for each color (transmission slot) 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).

```
proc T_SLOT_ASSIGN(A)
  begin
    RECOLOR_LIST ← min_conflict_set(A);
    decolor the nodes in RECOLOR_LIST;
    add A to RECOLOR_LIST;
    while RECOLOR_LIST ≠ NULL do
        v ← node in RECOLOR_LIST most
          constrained by distance-2 neighbors;
        T_SLOT(v) ← least color unused
          by its distance-2 neighbors;
        delete v from RECOLOR_LIST od
  end.
```

[3] For clarity, we establish a one-one correspondence between the slots in a schedule with natural numbers. That is, we index the slots in a schedule with positive numbers 1, 2, 3, ...



The number in the bracket on the left side of each node indicates the transmission slot of that node. Nodes n_0, n_1. n_2, n_3, n_4, n_5, n_6, n_8, n_9, n_{12}, n_{20},n_{24} and n_{26} have transmission cycles of 8, while all other nodes have transmission cycles of 16. An initial segment of the schedule induced by these transmission slots and cycles is shown below.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	...
n_1	n_6	n_9	n_2	n_3	n_0	n_7	n_{21}	n_{25}								n_1	...
n_4	n_8	n_{14}	n_{10}	n_{13}	n_{16}	n_{15}										n_4	...
n_5	n_{17}	n_{22}	n_{12}	n_{26}		n_{19}		n_1	n_6	n_9	n_2	n_3	n_0			n_5	...
n_{11}		n_{23}	n_{20}					n_4	n_8	n_{24}	n_{12}	n_{26}				n_{11}	...
n_{18}		n_{24}						n_5			n_{20}					n_{18}	...

Figure 1: Schedule derived from non-uniform transmission cycles

Transmission Cycle Before describing how each station's transmission cycle is determined, we require one piece of terminology: Given a station n, the *unit subset controlled by* n or the *unit subset of* n consists of the station n and all of its one-hop and two-hop neighbors.

Now, suppose that all stations have been assigned transmission slots. Consider a station n, and let C_n denote the maximum transmission slot of any station in the unit subset controlled by n. Then, the transmission cycle for station n is set to the least power of 2 that is greater than or equal to C_n. We term this method STP — Simple Two's Power. Figure 12 provides an example of a schedule derived from transmission slots and cycles.

In utilizing, the transmission slot and cycle approach, the critical element is that there are never any conflicts in the resulting schedule. The following theorem establishes that this is indeed the case:

Theorem 1 *For any two stations A and B that are one-hop or two-hop neighbors, if A and B are assigned transmission slots and cycles as specified above, then A and B never transmit in the same slot.*

[proof] Let station A and station B are two arbitrary stations that are one-hop or two-hop away. Suppose the slot numbers in a conflict free nominal schedule for A and B are S_A and S_B separately. Since A and B are within two-hops with each other and S_A and S_B are the slot numbers for A and B separately in a conflict free nominal schedule, so, we have:

$$S_A \neq S_B \tag{1}$$

Suppose the maximum number for indexing the slots for stations in the unit subset controlled by A is C_A and the maximum number for indexing the slots for stations in the unit subset controlled by B is C_B. Then, clearly,

$$1 \leq S_A \leq C_A \tag{2}$$

$$1 \leq S_B \leq C_B \tag{3}$$

Since we assume A and B are one-hop or two-hop neighbors, clearly station A is in the unit subset controlled by B and station B is in the unit subset controlled by A. Thus, we have:

$$1 \leq S_A \leq C_B \tag{4}$$

$$1 \leq S_B \leq C_A \tag{5}$$

By STP, the transmission cycle T_A for station A is set to the number such that:

$$T_A = 2^p \geq C_A \tag{6}$$

where p is the least positive integer such that $2^p \geq C_A$. Similarly, the transmission cycle T_B for station B is set to the number such that:

$$T_B = 2^q \geq C_B \tag{7}$$

where q is the least positive integer such that $2^q \geq C_B$. Thus, by (2), (5) and (6) we get:

$$\mid S_A - S_B \mid < 2^p - 1 \tag{8}$$

Since the transmission cycle for station A is T_A and the slot number for A in the nominal schedule is S_A, so station A's actual transmission schedule is expressed as:

$$S_A + sT_A = S_A + 2^p s \qquad s = 0, 1, 2, \ldots \tag{9}$$

Similarly, station B's actual transmission schedule is expressed as:

$$S_B + tT_B = S_B + 2^q t \qquad t = 0, 1, 2, \ldots \tag{10}$$

Suppose *by way of contradiction* that station A and station B might collide by STP, then there must be a $s_1 \in \{0, 1, 2, \ldots\}$ and a $t_1 \in \{0, 1, 2, \ldots\}$ such that

$$S_A + 2^p s_1 = S_B + 2^q t_1 \tag{11}$$

Thus, by the equation above, we have:

$$S_A - S_B = 2^q t_1 - 2^p s_1 \tag{12}$$

For (12), we have the following two cases:
Case (I): $p = q$.
For this case, we have:

$$S_A - S_B = 2^p (t_1 - s_1)$$

So, if $t_1 = s_1$, then $S_A = S_B$ which is a contradiction to (1). If $t_1 \neq s_1$, because $t_1 - s_1$ is an integer, so $\mid S_A - S_B \mid \geq 2^p$ which is a contradiction to (8). Thus, it is impossible that A and B would transmit at the same slot if $p = q$.

Case (II): $p \neq q$. Without loss of generality, we assume $q > p$. That is, $q = p + r$ where r is a positive integer. Then, we have:

$$S_A - S_B = 2^p (2^r t_1 - s_1)$$

So, if $2^r t_1 - s_1 = 0$, then $S_A = S_B$ which is a contradiction to (1). But if $2^r t_1 - s_1 \neq 0$, because $2^r t_1 - s_1$ is an integer, then $\mid S_A - S_B \mid \geq 2^p$ which is a contradiction to (8). Thus, it is also impossible that A and B would transmit at the same slot if $p \neq q$. □

2.2 Making the method adaptive

As noted earlier, it can be expected that in a large scale packet radio network, some of the stations may expectedly or unexpectedly join or leave the network. In this case, managing the joining and leaving of stations to ensure smooth operation of the network is among the most important functionalities for a protocol for scheduling a broadcast radio network. In this section we describe how to extend the basic scheme outlined above to this situation. As above, we specify the algorithm in a centralized fashion, leaving the distributed aspects for the following section.

2.2.1 Adaptive method: Joining of Stations

Joining of stations is easy since an existing schedule can be updated to include the joining station simply by running the procedure T_SLOT_ASSIGN (which may modify the transmission slots for all one-hop neighbors of the joining station) and then recomputing the transmission cycle for each station in the unit subsets of any station whose slot was modified. The only additional issue is that some coordination is required to assure that all stations, old and new, are operating under an identical point of reference relative to the start of the schedule. The details of this coordination are described in subsection 2.3.1.

2.2.2 Adaptive method: Leaving of Stations

```
proc DELETION(A)
  begin
    for each one-hop neighbor u of A do
        decrease degree of u by 1;
        recompute D(u);
        if T_SLOT(u) > D(u)
          then greedily recolor u; fi od
    for each two-hop neighbor v of A do
        decrease D(v) by 1;
        if T_SLOT(v) > D(v)
          then greedily recolor v; fi od
  end.
```

The leaving of a station from the network cannot introduce conflicts into the schedule. The only scheduling effect produced by the leaving of a station is that the transmission cycles of some of the stations may now be longer than necessary. Thus, when a station leaves the network, some adjustments of the schedule may be appropriate. These adjustments concern the transmission slot assignment of nodes in the unit subset of the station that has left. The details are provided on the left.

2.2.3 The competitive ratio

As noted earlier, the competitive ratio of an algorithm is a standard measure of the performance of an approximation algorithm. In the context of broadcast scheduling based on transmission slots and cycles, we define the competitive ratio to be the ratio of the maximum transmission cycle produced by the algorithm to the optimal maximum transmission cycle. In this section we establish a competitive ratio of 26 for FDAS (we doubt that the bound is tight).

We begin by stating a key lemma. The proof of the lemma is a somewhat involved analysis of the geometry related to unit disk graphs and is omitted from this extended abstract.

Lemma 1 *Letting R be the uniform communication range of the PRN, and letting p be an arbitrary station, consider a circle of radius $2R$ centered at p, with two rays emanating from p that intersect the circle at points b_1 and b_2. If $\mid b_1b_2 \mid \leq R$, then any two stations within the section pb_1b_2 are necessarily one or two-hop neighbors.*

Theorem 2 *FDAS has a competitive ratio of 26.*

Proof: Given a node v, let $SLOT(v)$ denote its transmission slot and let $D(v)$ denote its number of distance-2 neighbors. An analysis (omitted here) of the procedures T_SLOT_ASSIGN and $DELETION$ establishes the invariant that for any node v $SLOT(v) \leq 1 + D(v)$.

Let $C = Max\{D(v_1), D(v_2), \ldots, D(v_n)\}$. It follows that the maximum transmission slot assigned by FDAS does not exceed $C + 1$. By the above lemma it can be concluded that the ratio of the maximum transmission *slot* to the optimal maximum transmission cycle is bounded above by 13. Since the maximum transmission *cycle* is less than twice the maximum transmission slot, the competitive ratio of 26 follows. □

2.3 A distributed implementation

The key to producing a distributed implementation of the protocol described above is to note that the scheduling of a station requires NO global information. Rather, information is required only from one and two-hop neighbors of the station. In this respect, implementing the algorithm in a distributed fashion is straight-forward — from the perspective of an individual station, its actions in regard to scheduling precisely follow the method outlined above. The only complications arise in the communication aspects, including station registration when joining the network, coordination of geographically close stations attempting to run the protocol in an overlapping fashion, and coordination of station actions when a neighboring station leaves the network. Below we discuss these issues in detail.

2.3.1 Distributed method: Joining of Stations

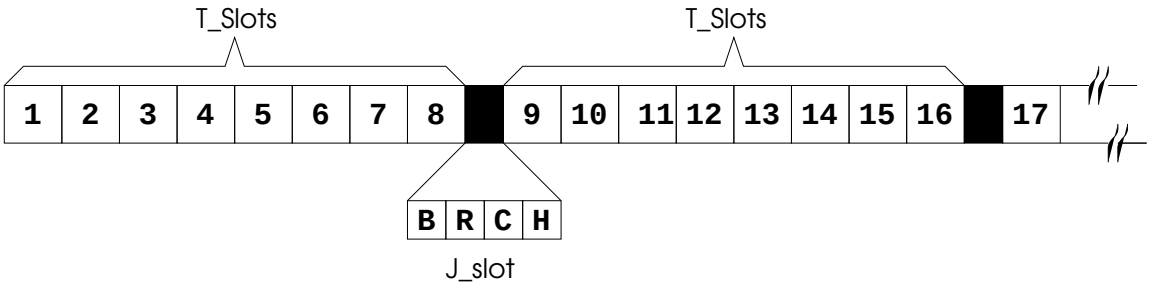
There are two primary issues associated with the joining of a station. First, is the identification of the station to its one-hop neighbors in the network. This situation becomes complicated if more than one station appears in the network (almost) simultaneously and are at locations so that the newly appeared stations will be one-hop or two-hop neighbors. We term this task *registration*. The second is running the actual joining protocol, which includes collecting information concerning local configuration and schedule and handling transmission slot conflicts that arise among existing nodes as a result of the joining of this station. We term this task *resolution*. We address each of these in turn.

Registration

To facilitate joining of new stations, we insert a special slot between every k slots. We will refer to this type of special slot *J_slot — Joining slot*.

Each *J_slot* consists of 4 sub-slots which will be referred to as *B_subslot, R_subslot, C_subslot* and *H_subslot* separately and are arranged in that order. In a sub-slot of a *J_slot*, a mini-packet which is at most a few bytes long might be transmitted. As with the case for a normal time slot, a sub-slot of a *J_slot* has a duration that is just long enough to transmit a mini-packet and for the mini-packet to reach at the most distant one-hop neighbor of the transmitting station.

The determination of the integer k depends on the particular convergence of factors, including the frequency of joining of stations in a local area, the ratio of the transmission delay, the maximum propagation delay of the packets, etc. The goals here are to minimize both the overhead on the network bandwidth and the time required for a station to join the network. It is clear that the above two goals conflict with each other. Thus, k is required to be chosen by balancing the trade-offs between overhead and joining time. Figure 2 presents an example of the arrangement of the time slots in which k is set at 8. Suppose in this example, a normal/data packet has a length of 1024 bytes, the ratio of the transmission time of a normal/data packet over the maximum propagation delay is 20 and the mini-packets that are transmitted in the sub-slots of a *J_slot* have a length of 4 bytes, then the overhead by



A *J_slot* in inserted between every 8 *T_slots*. A *J_slot* contains 4 sub-slots that is named by B, R, C, H separately.

Figure 2: Arrangement of Time Slots

Figure 3: procedure for registration

having registration slots is only about 2.5 percent.

The purpose of the registration procedure is to make the joining station known to its one-hop neighbors. In case more than one stations appear at locations within a range of two-hops, the protocol forces that only one station will be able to make a successful registration and continue joining process. The joining station that has been successful in registration is called a *head* to its one-hop neighbors. All other joining stations that are within two-hops of a head are necessarily refrain themselves from joining temporarily until the station that has succeeded in registration finishes its joining process. The procedure that is implemented by a joining station for registration is outlined in Figure 3. The procedure that is implemented by an on-line station in a *J_slot* is outlined in Figure 4.

The first action to take by a station, say station A for joining the network is to synchronize itself with the global time slot. After that, it makes effort to establish itself as a head.

As is illustrated in Figure 3, when a *J_slot* comes, A listens to the network in the *B_subslot*. If it receives something (a *Control_B* or collided *Control_B*s) in the *B_subslot*, it means that there is already a head within two-hops with it. So, in this case, station A will continue to wait until an empty *B_subslot* is encountered.

If A has detected an empty *B_subslot*, it broadcasts a mini-packet (called *Control_R*) in the following *R_subslot* to make a request for registration. If A did not detect a collision when it was broadcasting its *Control_R* and it receives nothing in the following *C_subslot*, then it can determine that there was no other station within one-hop or two-hop of it that was also making a request for registration at the same time as it did. Since in this case A was the only one within a range of two-hops around it that made a request for registration, it will establish itself as a head by broadcasting a mini-packet (called *Control_H*) in the *H_subslot*. Since when A broadcasts its *Control_H*, no other station within two-hop with it also claims itself as a head and till that time all of A's one-hop on-line stations have no head, the *Control_H* broadcasted by A will be received collision free by all of its one-hop neighbors. For a on-line

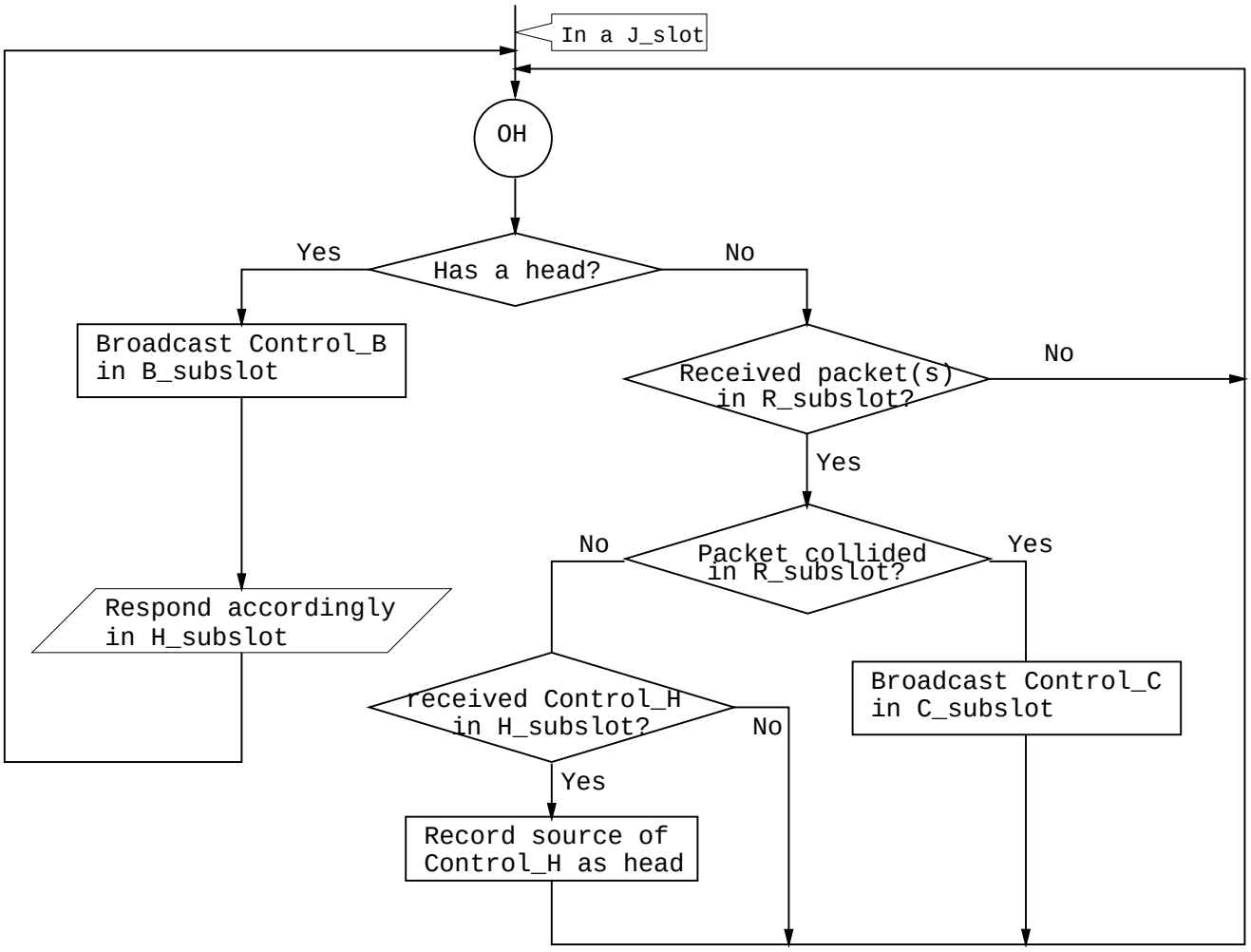


Figure 4: procedure for registration

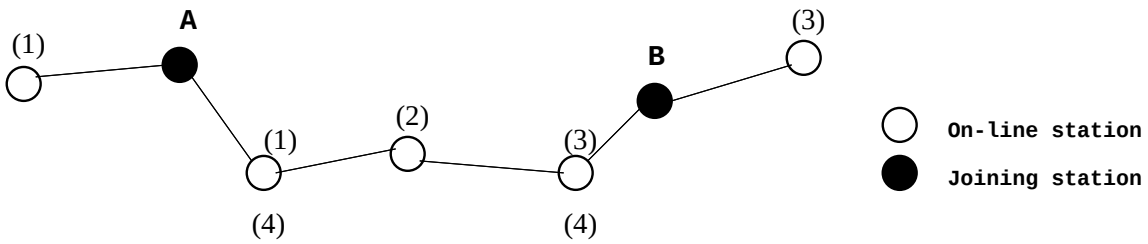
station, say station B, if it currently has no head and receives a *Control_H* in a *H_subslot*, it will record the source of the *Control_H* as its head. After an on-line station has a head, the on-line station will broadcast a mini-packet (called *Control_B*) in each *B_subslot* that follows till its head finishes joining the network.

Suppose station A has claimed itself as a head. By having each of its one-hop neighbors broadcasts a *Control_B* in the following *B_subslots* before A finishes its process of coming up on-line, it can be sure that no other joining station within two-hop of A sends out a request for registration during this period.

If otherwise after A has encountered an empty *B_subslot*, when it was transmitting its *Control_R* in the *R_subslot*, it detected a collision or after it transmitted its *Control_R*, it received a *Control_C* in the *C_subslot*, it can determine that there is other station within two-hop of it that was also trying to register. In this case, A will continue its effort for registration by contending with those stations. The contending method might be one such that A transmits its *Control_R* in a certain probability in each of the following *R_subslots* until it succeeds or some other station(s) within two-hop of it succeed.

Resolution After a station establishes itself as a head, it needs to collect information concerning the configuration and current schedule of its one-hop and two-hop neighbors. Then, it can run the scheduler to work out a schedule for itself and modify the schedule for some of its one-hop neighbors when there is a necessity.

It can be seen in Figure 5 that simultaneous modification of a schedule (by running *T_SLOT_ASSING*) by two different stations that are one-hop or two-hop or three-hop or four-hop away may result in conflicts in the nominal schedule. It is also clear that the procedure for registration only guarantees that no two heads are within two hops. So if two heads are three-hop or four-hop away, it is required for the procedure for resolution to enforce that only one head runs the scheduler at a particular time. This is done by the coordination of the on-line stations. The procedure is as follows (See Figure 6). Suppose station A is currently a head of B. Before station B sends out its local information for A to run the scheduler, it contacts its on-line neighbors to get a sense whether or not it is an appropriate time for its head to run the scheduler. That is, in the next packet that B sends out in the regular time slot, it includes in the packet the information on when its head claimed to be a head and the ID (identification) of its head. For convenience of description, we term such a piece of information *HR1*(Head Report to One-hop neighbors) and a *HR1* that is broadcasted by B is denoted by *HR1(B)*. Upon receiving a *HR1* by a on-line station, say F, F records the information and passes the information to the two-hop neighbor(s) of the source of the *HR1* in the form of *HR2*(Head Report to Two-hop neighbors). If it happens to be the case that before F sends out the *HR2(B)*, it has a new head that hasn't been reported, F includes both the *HR2(B)* and its *HR1* in the next packets it transmits in the regular time slot. Suppose as is illustrated in Figure 6, there are other stations that are within four hops with A are currently heads. Suppose the current maximum transmission cycle of the stations in the unit subset controlled by B is MT_B. Then after a period of $2TM\_B$, station B, will be able to get the up-to-date information on the heads that are within three-hops with it. In the example illustrated in Figure 6, B will get (or already has) the information about heads H and E. After B has collected the up-to-date information on the heads that are within three-hops with it, it can decide whether or not it can transmit the local information concerning configuration and schedule of the stations in its unit subset to its head. If in its head list, there is some other head has a priority higher than its head, it will wait until its own head become the one of highest priority. The priority of a head in a head list can be determined by two factors. First, if a head is established earlier than another head, then the head that claimed to be a head earlier has a higher priority. Second, if two heads claimed to be heads at the same *H_subslot*, then the priorities of the two heads can be determined by some other static factor, say the IDs of the



Station A and B are two joining stations. If A and B runs the scheduler *T_SLOT_ASSING* simultaneously, conflicts might arise as is illustrated in this figure. In this figure, the numbers in brackets above on-line stations denote the slots assigned before the joining of station A and B; the numbers in brackets below on-line stations denote the slots assigned after the joining of station A and B.

Figure 5: conflict arise by simultaneous modification of the schedule

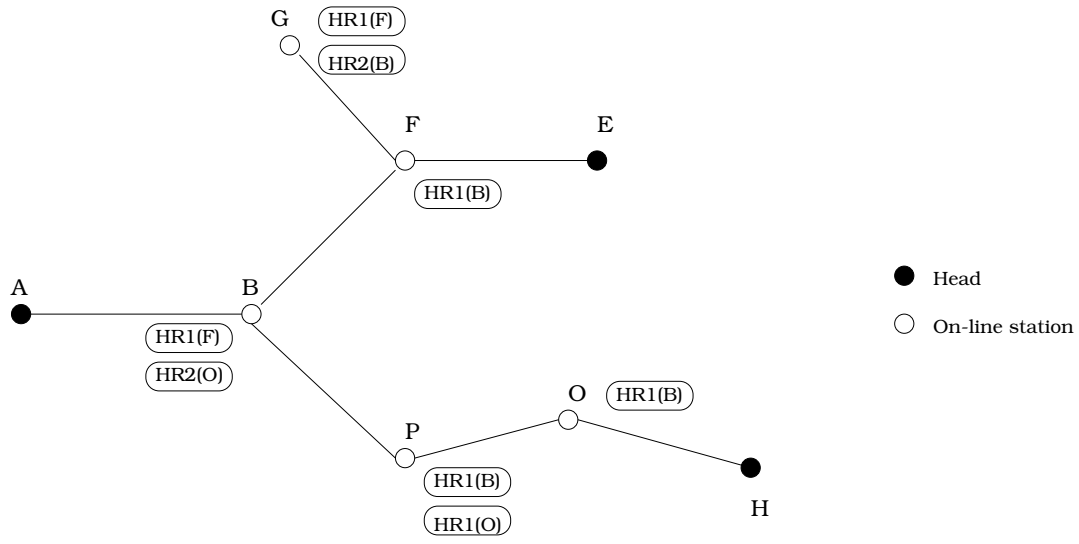


Figure 6: Information Exchange Between On-line Stations

two heads.

For a on-line station B that has its own head A, when its head becomes the one that has the highest priority, it transmits in the regular time slot the local information necessary for its head to run the scheduler. We term this information *RI*. If no other on-line station that is also in A's communication range conflicts with B's transmission schedule or there are some, but none of them transmitted in the time slot that B transmitted its *RI*, the packet containing B's *RI* will be received collision free by A. If A receives the *RI* from B, it will inform B by broadcasting a mini-packet with certain bit on in the next *R_subslot* or *C_subslot*. If B doesn't receive this confirmation, it can suspect that its broadcast schedule is conflicting with the schedule of some one-hop neighbor(s) of A. B will retransmit its *RI* on a random basis until B gets the confirmation from A. After A has claimed its head status and before B's *RI* has been correctly received by A, B needs to transmits a mini-packet called *Control_NA* in each of the *H_subslot* that it encounters. This way, A can know that it has not correctly received the *RI*s from all of its one-hop neighbors. Thus, A is prevented from running the scheduler.

When no *Control_NA* is received in a *H_subslot*, A can be sure that all of the information for running

the scheduler is available. Then, A runs the *T_SLOT_ASSIGN* to determine a schedule for itself as well as to resolve conflicts between its one-hop neighbors if there is any.

After A runs the scheduler, it needs to distribute the new local schedule to stations in its unit subset. From the procedure *T_SLOT_ASSING*, it can be seen that A chooses a slot number that is different from all of the slot numbers that are currently hold by other stations in its unit subset. Thus, the distribution of the new schedule can be done in A's own regular transmission slot. Note that, running a scheduler itself only determines a relative slot number in the nominal schedule. Station A still needs to figure out when it can transmit for the first time. The first transmission slot in execution schedule for station A can be determined as follows. Suppose the last *RI* that was correctly received by A for running the scheduler was sent by station B and C_B is the slot number for B. Suppose after running the scheduler, A determines that its slot number is C_A and its transmission cycle is T_A. Then, with the time slot when A received the last *RI* from B as the starting point, A starts transmission after a period of $rT_A + C_A - C_B$ where $r(= 0 \text{ or } 1 \text{ or } 2 \ldots)$ is determined in real time situation. First, A can not start regular time slot transmission until the new local schedule is computed. Second, after the new schedule is computed, the first transmission time is determined by the three factors: C_A, T_m^1 and T_m^2 where T_m^1 and T_m^2 are separately the minimum transmission cycles of its one and two-hop neighbors. Here we have three case.

- $C_A < T_m^1$ and $C_A < T_m^2$. In this case, A begins distribution of the new schedule immediately in its regular time slot that is determined by the formula above.

- $C_A > T_m^1 4and 4 C_A < T_m^2$. In this case, A waits till next *J_slot* and broadcasts its slot index C_A in the *R_subslot* to its one-hop neighbors. After receiving the slot index of A, a one-hop neighbor having a transmission cycle that is smaller than C_A expands its transmission cycle so A's transmissions in its regular time slots would not be collided. Then, after A broadcasted its slot index in a *J_slot*, A begins distribution of the new schedule in its regular time slot.

- $C_A > T_m^2$. In this case, A broadcasts its slot index in a *J_slot* and waits till all of its one-hop and two-hop neighbors have received its index number and extend their transmission cycles accordingly. Then A begins distribution of the new schedule in its regular time slot.

After A finishes broadcasting the new schedule and all of its one-hop neighbors that are assigned new slot numbers have broadcasted their new slot numbers, A becomes a on-line station and can proceed proper communication.

For a one-hop neighbor B of A, after it receives the new schedule and neighborhood information from A, it modifies its local information and passes the relevant part of the new schedule and configuration information to its one-hop neighbors. If B has a new slot number that is greater than the minimum transmission cycle of its two-hop neighbors, it need to halt transmission until its relevant one-hop neighbor(s) has/have a chance to broadcast the change to the two-hop neighbor(s) with a transmission cycle smaller than the new slot number of B. After that, B deletes A from its head list; proceeds communication by new schedule and stops transmitting *Control_B* in the subsequent *B_subslots* before it has a new head.

For a one-hop or two-hop neighbor G of B, if B is assigned a new slot number that is greater than the transmission cycle of G, upon receiving C_B, G expands transmission cycle accordingly. Then, G deletes A in its head list. If at the moment, G itself has a head, it can check whether not or its own head is the one of highest priority in its head list and act accordingly.

One last important note is that, during the whole process for a station coming up online, none of the on-line stations needs to stop proper communication even for stations that are one-hop or two-hop away of the joining station.

2.3.2 Distributed method: Leaving of Stations

There are two aspects to the leaving of a station. First, is how stations in the unit subset of the leaving station recognize that leaving. Second, there are the mechanics of adjusting the schedule.

Recognizing that a station is leaving There are two situations for the leaving of a station. One is that a station leaves the network in an expected fashion, and the other is that a station leaves the network unexpectedly (e.g. due to a malfunction).

In the former case, the station that is about to leave the network broadcasts a packet indicating that it is leaving. After receiving such a message, each one-hop neighbor of the leaving station makes appropriate adjustment in its local information concerning the schedule and the configuration of its one-hop and two-hop neighbors, and passes the message along to it's one-hop neighbors (they are two-hop neighbors of the leaving station). Each two-hop neighbor of the leaving station, simply adjusts its local information on the schedule and the configuration.

The situation is a more complicated for the latter case. There it may be that a station has nothing to transmit during a period. So, if a station disappears without acknowledgment, the problem is how can other stations around it tell whether the station is still on-line (but silent) or has disappeared? To resolve this situation, we introduce a network parameter α: Maximum Silent Time. A functioning station with transmission cycle T must transmit (perhaps only a control packet) within every αT slots. If a station does not transmit within αT slots, then its one-hop neighbors can safely assume that the station has left the network, and proceed as above.

Adjusting the schedule Finally, we consider how the schedule is adjusted in accordance with the deletion procedure given in the section on adaptive leaving of stations. Here, coordination is necessary among the nodes in the unit subset of station A to avoid overlapping recoloring.

Suppose station B was in A's unit subset. If, after deletion of A, it comes to a situation where B needs to change its transmission slot, it broadcasts its new slot number for confirmation. If more than one stations are making a change in their slot indexes and the new indexes of two stations that are within two-hops are accidently the same, then the one that made the request earlier get the number. Note that for two stations that are within two-hops with each other, it is certain that they can not broadcast their request at the same time. Thus, with the request time-stamped, no deadlock can occur. If a station's request for changing its index number is denied by at least one of its one-hop neighbors, it can choose another number and make another request.

2.4 Two extensions

In this section we briefly describe two extensions to the basic protocol outlined above. These extensions are aimed at improving the practicality of broadcast scheduling and fit smoothly into the distributed framework that we have described.

The first extension is to utilize the *spare slots* introduced by having the transmission cycles of some stations exceed (perhaps by a factor of 2), the maximum index relevant to those stations. For instance, a station S using transmission slot 3 and with one-hop and two-hop neighbors using slots 2, 4 and 5, would operate with a transmission cycle of 8. In this circumstance, station S could also transmit in slots $1, 6, 7$, and 8 with its transmission cycle of 8 without any collisions (assuming other stations in the unit subset of S transmit only in their single transmission slot). Thus, the first extension provides the capability for stations to claim and release spare slots as necessary.

The second extension is aimed at improving the network performance when the traffic is unbalanced and bursty. Suppose for example, that a station S requires a number of additional slots and at the same

time, most of other stations in its unit subset barely have anything to transmit. Even after employing the first extension scheme, it may be the case that S has to drop packets while a number of slots used by its one-hop and two-hop neighbors go unused. Our second extension provides a mechanism whereby stations with low utilization of their transmission slot can temporarily allocate that slot to neighbors having a need for additional slots.

Both of the above extensions can be implemented in the context of our basic scheme and in a fully distributed fashion. The protocol details are as follows.

2.4.1 Extension_A

First let's see the example illustrated in Figure 12. Suppose at a moment station n_7 sees a need of getting more transmission time, for example the buffer for storing the transmission data has filled up 4/5 and the trend still continues. In the basic scheme, only slots that are numbered 1, 2, 3, 4, 5, 6, 7, 9 on a cycle of 16 are occupied by stations in its unit subset. So, station n_7 can safely broadcast in slots 8, 10, 11, 12, 13, 14, 15, 16 without collision if all other stations stick in their basic schedule. Those slots that can be taken by a station without causing conflicts in the basic scheme are called EA_slots of that station.

If a station A in need of an extra slot and there is at least one EA_slot available,i.e., has not occupied by any other station in its unit subset, A can make a request to reserve an available EA_slot. The EA_slot is time stamped. Thus, if two stations that are two-hop away has requested the same EA_slot, the one that made the request earlier gets the grant. There is no restriction on the number of EA_slots that a station can reserve. However, a station can only request one EA_slot at one time. And, after it has successfully reserved one, the next request should be made after a period of the maximum transmission cycle of the stations in its unit subset. Thus, other stations that also in need of extra slot may get better chance to be granted.

If at a moment there is no need for a station to keep the reserved EA_slot, it can release that slot by sending out a release so that other stations in its unit subset make appropriate modification in their local information.

If at a moment a reserved EA_slot by A is assigned to a newly joined station or the EA_slot is assigned to a one-hop neighbor of a newly joined station to resolve conflicts and the station that takes the slot in the basic schedule is within two-hops with A, station A is forced to release that EA_slot at the instant it receives the notification. That is, if a slot is utilized by a station B in the basic schedule/scheme, the slot can no longer be an EA_slot to any station within two-hop with B. In case it is a forced release, the station that made the reservation still needs to announce the release so that all of the stations within two-hops of it be informed of the situation.

2.4.2 Extension_B

Suppose during a period, a station has a heavy transmission requirement and no EA_slot is available. If it noticed that most of its one-hop neighbors do not have much useful traffic at the moment, it can make a request to employ Extension_B.

Suppose station n_5 in Figure 12 is such a station at the moment. In this example, station n_5 has 8 as its transmission cycle in the basic schedule. If station n_5 makes a request to employ Extension_B, it means it tries to reserve every one of its two transmission cycles. That is, station n_5 will transmit in slots 1, 9, 10, 11, 12, 13, 14, 15, 16 on a cycle of 16 if the request is granted. Thus, you can imagine that a bunch of nodes that are colored with numbers 1, 9, 10, 11, 12, 13, 14, 15 and 16 are suddenly appeared at the location where n_5 is located. Thus, some of its one-hop and two-hop neighbors might need to yield half of their transmission time during the period that Extension_B is in effective. If at

least half of the stations in its unit subset give no objection, n_5 can inform the stations in its unit subset that Extension_B will be executed. Otherwise, it gives up the attempt.

If the traffic burden is released to a certain degree after a station executes Extension_B for a period, it stops executing Extension_B and informs all other stations in its unit subset to restore back to basic scheme.

In case Extension_A and Extension_B conflicts on some slots, Extension_A overrides Extension_B.

For a station A, in case a station in its unit subset has a head, A can not make a request to execute Extension_B. During the time A is executing Extension_B, if a station in its unit subset becomes having a head or the transmission cycle has changed, A stops executing Extension_B instantly and makes a announcement on its action.

3 Future Work

There remain a number of open problems in conjunction with the work described here. Among these, the most important is to evaluate the performance of FDAS in practice. For this purpose, we are in the process of implementing experiments on FDAS using the OPNET simulation tool. Our goal in designing these simulation experiments is to obtain practical performance results on not only schedule length, but also on: the average time for a station to come up on-line; the effectiveness of the two extension schemes; throughput; and, end-to-end delay.

References

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

[2] A. Sen and M. L. Huson. A new model for scheduling packet radio networks. *Wireless Networks*, 3:71–82, 1997.

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

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

[5] Errol L. Lloyd and Ram Ramanathan. Efficient distributed algorithms for channel assignment in multihop radio networks. *Journal of High Speed Networks*, 2:405 – 423, 1993.

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

[7] Errol L. Lloyd and Xiaopeng Ma. Experimental results on broadcast scheduling in radio networks. *ATIRP Conference*, 1997.

[8] S.Even, O. Goldreich, S. Moran, and P.Tong. On the np-completeness of certain network testing problems. *Networks*, pages 1–24, 1984.

[9] R.M.Darp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 1972.

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

[11] Xiaopeng Ma and Errol. L. Lloyd. Practical adaptive algorithms for channel assignment in multihop packet radio networks. *ATIRP Conference*, 1998.