Minimal CDMA Recoding Strategies in Power-Controlled Ad-Hoc Wireless Networks

Indranil Gupta[1]

Technical Report (January 2001)
Department of Computer Science
Cornell University
Ithaca, NY 14853

[1]This work was partially funded by DARPA/RADC grant F30602-99-1-6532 and NSF grant No. EIA 97-03470.

Minimal CDMA Recoding Strategies in Power-Controlled Ad-Hoc Wireless Networks

Indranil Gupta
Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
email: gupta@cs.cornell.edu

Abstract

The problem of Code Division Multiple Access *(CDMA)* code assignment to eliminate primary and hidden collisions in multihop packet radio networks has been widely researched in the past. However, very little work has been done on the very realistic *distributed, dynamic* version of the transmitter-oriented code assignment *(TOCA)* problem in an ad-hoc network where mobiles use CDMA technology. None of the existing dynamic TOCA CDMA algorithms in literature are efficient, in terms of maximum code index assigned in the network, or number of times a mobile has to change its code. We present a set of local and distributed *recoding* strategies for the TOCA CDMA problem in an ad-hoc network where mobiles can arbitrarily 1) connect and disconnect, 2) move about, and 3) increase or decrease their transmission power - all these may need some mobiles to be recoded, to avoid new collisions. Our strategies, unlike those proposed earlier in literature, guarantee *minimal recoding*, that is, given a current network-wide code assignment and one of the above events, our strategies change the codes of the minimum number of mobiles needed to eliminate all collisions. Minimal recoding can be very important in reducing the effect of frequent code changes on the performance and criticality of distributed applications. Further, among all possible minimal recoding strategies in a class, most of our strategies are also (provably) *optimal* in terms of the maximum code index assigned in the network. Performance results that evaluate our dynamic minimal strategies are also presented.

Keywords: *CDMA, ad-hoc networks, dynamic networks, code assignment, distributed algorithms, graph theory, minimal recoding.*

1 Introduction

Ad-hoc wireless networks are characterized by lack of an established infrastructure such as an underlying wired network or base stations. However, their potential uses range widely from scenarios where an ad-hoc network could be just convenient, such as a conference where members communicate with each other, to critical ones, such as networks formed on the fly by satellite constellations, on the battlefield etc. [1].

Transmissions in such wireless media could lead to *collisions*, where transmissions are garbled at the receiving end. This could be either a *primary* collision, where an incoming transmission is damaged by a simultaneous outgoing transmission from the receiving mobile, or a *secondary* (also *hidden*) collision, where two incoming transmissions garble each other. *C*ode *D*ivision *M*ultiple *A*ccess is a widely used technology that completely eliminates collisions by techniques such as spread spectrum and *orthogonal codes*. We consider only the case of orthogonal codes. With each mobile (computer+transceiver) modeled as a node, and each code modeled as a positive integer, codes have to be assigned to different nodes in the network, one code per node, in a 1) correct manner to eliminate all *constraints* (i.e., all primary and hidden collisions) and 2) an efficient manner, such as say, to minimize the maximum code index assigned to any network node. This is known as the *code assignment* problem [2-8]. CDMA protocols require that either receivers, or transmitters, or both, are code-agile, that is, are able to communicate over a range of codes. We are concerned only with the first kind, also called the Transmitter Oriented Code Assignment (TOCA) problem. This problem has been extensively studied for static multihop networks [2-8]. Finding an optimal TOCA code assignment in terms of the maximum code index assigned in the network has been mapped to the graph coloring problem [9], where codes are represented by colors - this has been shown to be NP-complete [5]. Several centralized and distributed heuristics have been proposed for the same [2-9].

In an ad-hoc network, nodes are free to 1) move about, 2) connect or disconnect from the network, and 3) increase or decrease their transmission ranges (such a capability is often mandatory due to the power-sensitivity of CDMA transceivers, besides other advantages such as saving power, controlling network connectivity and throughput, security etc. [10, 11, 12]). These *events* may invalidate any statically generated code assignment by introducing new *conflicts* in the assignment, that is, by causing new collisions. A *recoding*, that is, a change in the code assignment of some network nodes, is needed to eliminate these new collisions. In general, a *recoding strategy* is a set of algorithms (one for each of the above event types) for a reassignment of codes to some of the nodes in the network to maintain the correctness of the code assignment. Centralized code assignment algorithms such as those of [2, 4, 5, 6, 7, 8] are inappropriate in an ad-hoc network as they determine a new code assignment for every node on each event. The distributed heuristics of [4, 5] are also inadequate as they assume a static network. Probably the only prior works to propose distributed solutions in a dynamic ad-hoc network-like scenario are [3, 6]. None of these papers consider the recoding problem arising out of a change in transmission range.

In this paper, we give a set of efficient *recoding algorithms* for the above events. These algorithms involve communication only local to the event and are *distributed*, i.e., they require no central coordination. Our algorithms (provably) satisfy the important goal of *Minimal Recoding*, that is, a recoding strategy must try to minimize the number of nodes that are recoded (with a new code) on any network event. More concretely, given a current code assignment and one of the above events, among all possible recoding strategies, our algorithms achieve the lowest bound on the number of nodes that need to be recoded to eliminate all conflicts in the network. This could be useful, even critical, in ad-hoc networks where frequent recoding might be costly to the applications using the communication medium. Examples include hard real-time applications [13], and applications where maintaining a persistent high data rate is critical to its performance. Moreover, most of our algorithms are (provably) *optimally minimal*, that is, given a current code assignment and an event, among all possible recoding strategies that are minimal and consider recoding only nodes one hop away from the initiating node, our strategies achieve the optimal (least) increase in the maximum code index assigned to the network. We also present simulation results that verify our hypothesis that our strategies would indeed be practicable in an ad-hoc network in the long run, and perform better than previously suggested strategies. In addition, our strategies can also be used as orthogonal recoding algorithms to any global code assignment heuristic in a dynamic ad-hoc network.

The rest of the paper is organized as follows. Section 2 discusses the assumed network model, and our high-level and concrete goals in designing the dynamic recoding algorithms. Section 3 touches on the previous work on this problem. Section 4 presents our recoding algorithms. Section 5 presents performance results for our algorithms and section 6 concludes the paper.

2 Model and Problem Statement

A power controlled ad-hoc network is modeled as a dynamic directed graph (digraph) $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ the (current) set of nodes in the network. Each vertex v_i in V has a configuration defined by its current position coordinates ((x_i, y_i) in a 2-dimensional network) and a (variable) maximum transmission power range r_i which specifies the *maximum* distance from (x_i, y_i) that other nodes in the network can hear or are affected by interference from its transmissions. The set of edges $E = \{(v_i, v_j) : i \neq j \ \&\ d_{ij} \leq r_i\}$ consists of directed edges of the type $v_i \rightarrow v_j$ if and only if v_j is within v_i's transmission range, that is, if the distance d_{ij} between v_i and v_j is less than r_i. Note that this can be easily generalized for the non-free-space propagation case where, due to obstacles, although $d_{ij} \leq r_i$, $(v_i, v_j) \notin E$. Node i's assigned code is denoted by c_i and is a positive integer.

Nodes can arbitrarily join or leave the network, increase or decrease their power range r_i, and move about within the network; call each of these *events* or *reconfigurations* in the network. For simplicity, we make the following assumptions, the latter two of which are very realistic, and the first of which we relax in a subsequent theorem.

1. Network *events* or *reconfigurations* occur throughout the ad-hoc network one after the other and not simultaneously (we later relax this condition and show how to parallelize our recoding algorithms).

2. Nodes move and change their ranges in discrete (and not continuous) steps.

3. **Minimal Connectivity:** A node v_i can change its configuration if and only if there are nodes v_j, v_k $(j, k \neq i)$ in the new configuration such that v_j is within v_i's transmission range, and v_i is within v_k's transmission range.

The TOCA code assignment problem [7] is to assign a code (equivalently, a color), which is essentially a positive integer, to each node in the network so that the following constraints CA1 and CA2 are satisfied throughout the network at all times.

Condition CA1 - (Primary) Collision Avoidance 1: For every edge $(v_i, v_j) \in E$, $c_i \neq c_j$.
Condition CA2 - (Secondary) Collision Avoidance 2: For every pair of edges $(v_i, v_k), (v_j, v_k) \in E \ \&\ i \neq j$, $c_i \neq c_j$.

Fig 1(a) shows a snapshot of an ad-hoc network containing 4 nodes $\{1, 2, 3, 4\}$, with their maximum transmission ranges, as well as a new node 5 attempting to connect to the network.

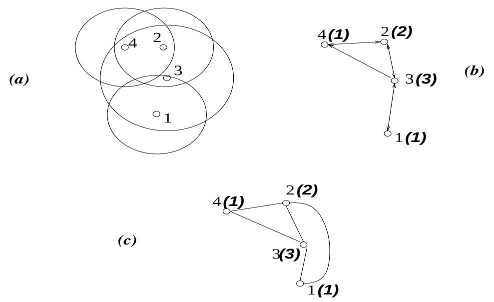


Figure 1: Example Ad-hoc Network with (a) nodes and transmission ranges, (b) induced digraph and (c) coloring constraints (two nodes connected by an edge cannot have the same code) and a correct assignment.

Fig 1(b) shows the directed graph model of this ad-hoc network and Fig 1(c) shows the constraints in the TOCA problem for the above network. The indices against the nodes show the optimal color assignment to satisfy CA1 and CA2.

Ideally, for each event, an efficient recoding strategy should attempt to (1) minimize the maximum code index used by any node in the network, (2) (*Minimal Recoding*) minimize the number of nodes that change their code, (3) minimize the overhead of recoding, and (4) keep the recoding strategy distributed and local. Goal 1 is needed because the hardware of a node can be designed to transmit on only some maximum number of codes. Goal 2 is very important for an ad-hoc network since recoding can be very costly, as mentioned in section 1. These two goals are contradictory - global coloring algorithms satisfy only goal 1 (ex. [5]) while local coloring algorithms may satisfy only goal 2. Our minimal approaches to recoding obtain a tradeoff between these two goals, while achieving the other goals. They give the best performance in terms of minimum number of nodes recoded among all possible strategies while using almost as few colors as a global coloring heuristic, in fact differing only by a few colors.

Henceforth, during the recoding for an event, for each network node, the set of colors that it cannot be assigned since it would violate either CA1 or CA2 with some other node, will be called its *constraints*. *Conflicts* will refer to the violation of CA1 or CA2 somewhere in the code assignment due to the event. In the rest of the paper, the terms "color" and "code"'are used interchangeably, as are "recoloring" and "recoding".

3 Previous Work

[3, 6] consider the TOCA problem in the dynamic and distributed context. Both works are similar in spirit and give recoding strategies for nodes joining and leaving a dynamic network with symmetric links, but they can be extended to the asymmetric case. The *CP* Recoding Strategy of [3] works as follows. When a new node joins the network, it contacts all its neighbors. The new node and its 1-hop neighbors exchange information about their current assigned colors and the constraints induced on each by the colors of nodes 1 and 2 hops away. All pairs of nodes 1 hop away from the new node which have the same colors violate CA2 and have to select new colors. In addition, the new node has to select a color that does not violate the constraints induced by nodes 1 and 2 hops away from it. This is achieved by having each nodes continuously check if it is the highest (or lowest)-identity node in its vicinity (defined by itself and nodes up to 2 hops away from it) that has not yet been assigned a color. The node selects the lowest available color (not yet taken by any of its 1 hop and 2 hop neighbors) when this condition is satisfied. The ordering by identities and respect for constraints ensures that no conflicts arise among nodes choosing new colors and with those whose colors will not change. When a node leaves the network, its neighbors update their lists, if any, about the constraints placed on them for future color selection. No recoding is required in this case. Node movement is handled as a pair of events consisting of a node disconnection and connection from/to all its neighbors.

The *CP* algorithms of [3] are proved to be correct but not optimal in any way with respect to the maximum color index used in the network, number of recodings etc. As [3] is the work that comes closest the approaches presented in this paper, we compare our approaches to theirs by simulation.

4 New Recoding Strategies

This section presents our recoding strategies. Sections 4.1, 4.2, 4.3, 4.4 respectively discuss the recoding strategies for node join, power range increase, power range decrease or leave, and movement.

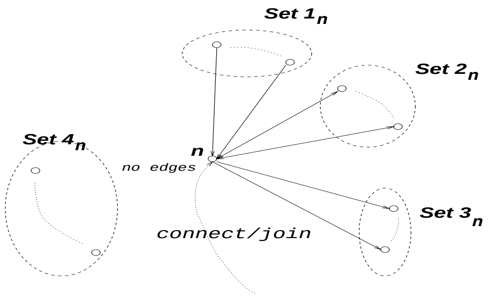


Figure 2: Node n joins the network

4.1 Handling a Node Join

We first consider the problem of recoding when a node n joins the ad-hoc network. Fig 2 shows a new node n joining the ad-hoc network, thus creating incoming edges from nodes in the sets 1_n and 2_n, outgoing edges to nodes in the sets 2_n and 3_n and no edges to nodes in the set 4_n. The sets 1_n through 4_n are thus partitions introduced on the old vertex set by the new node n. We call the colors assigned to the nodes in the network just prior to n joining as their *old* colors and those assigned to them after the *RecodeOnJoin* operation finishes as *new* colors (which, for a node, may be the same as its old color). Also, we call the constraints to be taken into account for the new coloring (recoding) as *new* constraints.

Consider the new constraints/conflicts created by this join. From CA1 and CA2, observe that all nodes in 1_n, 2_n and $\{n\}$ each need to have colors different from each other. However, nodes in 3_n need not change their color since n will be assigned a new color anyway and this will need to be different from any of the colors in 3_n. Recollect the goal of *Minimal recoding* we set down in section 2. Keeping this in mind, instead of recoding any nodes more than 1 hop away from n, we will attempt to minimize the total number of codes changed in the set $1_n \cup 2_n \cup \{n\}$.

Now, note that if a K-sized subset of nodes in $1_n \cup 2_n$ have the same old colors, only $K-1$ need to change their color, and one of them can maintain the same color in the new code assignment. More generally, if the set of old colors of the nodes in $1_n \cup 2_n$ is $\{C_1, C_2, \ldots, C_m\}$ and the associated number of nodes in $1_n \cup 2_n$ with these corresponding colors is $\{K_1, K_2, \ldots, K_m\}$, apart from recoding n, at least $\sum_{i=1}^{m}(K_i - 1) = \sum_{i=1}^{m} K_i - m$ of the nodes in $1_n \cup 2_n$ need to be recoded with different new colors to avoid conflicts in the new code assignment after node n joins. This is the minimal recoding bound.

Why ? Clearly CA2 tell us that all nodes in $1_n \cup 2_n$ have to have different new colors after the recoding. Suppose, by contradiction, that less than $\sum_{i=1}^{m}(K_i - 1) = \sum_{i=1}^{m} K_i - m$ of the nodes in $1_n \cup 2_n$ are recoded with new colors when node n joins. This means that at least $|1_n \cup 2_n| - (\sum_{i=1}^{m}(K_i - 1) - 1) = |1_n \cup 2_n| - \sum_{i=1}^{m} K_i + m + 1 = m + 1$ nodes in $1_n \cup 2_n$ retain their old colors after the recoding. However, since the old code assignment $\{C_1, C_2, \ldots, C_m\}$ to $1_n \cup 2_n$ had just m colors, this means that at least two nodes in $1_n \cup 2_n$ will have the same color after the recoding, a contradiction to CA2.

Now, the questions that arise are, which $\sum_{i=1}^{m}(K_i - 1)$ nodes from $1_n \cup 2_n$ do we chose for recoding, and what colors do we assign them ? Our solution is the following. Consider the undirected graph $G' = (V_1 \cup V_2, E')$ where $V_1 = 1_n \cup 2_n \cup \{n\}$, $V_2 = \{i : i \in Z^+ \;\&\; i \le$ maximum color constraint in the vicinity of $1_n \cup 2_n \cup n\}$, $E = \{(u, v) : u \in V_1, v \in V_2$, u is not constrained to be colored newly with v$\}$. By "maximum color constraint in the vicinity of $1_n \cup 2_n \cup n$", we mean the maximum of all old colors in $1_n \cup 2_n$ and all constraints (due to CA1 and CA2) for the new coloring on nodes in $1_n \cup 2_n \cup \{n\}$. The edges in E are weighted; edges of the kind (u, v), $u \in 1_n \cup 2_n, v \in V_2$ where v is the old color assigned to node u are assigned weight 3; all other edges have weight 1. Note that G' is an instance of a *bipartite graph*.

Next, consider a subset of edges $M = \{(u_1, v_1), (u_2, v_2), \ldots\}$ from E so that (1) no two u_i's in M are the same (assign each node in V_1 at most one color), and (2) no two v_i's in M are the same (no two nodes in V_1 have the same color), and (3) the sum of the weights of the edges in M is the largest among all such M that satisfy conditions (1) & (2). Our recoding strategy is to assign a node u in V_1 to the color in V_2 that M matches it to, and for all $u \in V_1$ not matched by M, assign them consecutive colors one by one starting from $|V_2| + 1$ onwards. Such an M defined by properties (1) and (2) is called a *matching* on the graph G' [14], a set of edges which have no end-vertices in common. A matching that satisfies condition (3) is called a *maximum matching* on weighted bipartite graph G'. Efficient algorithms exist to find a maximum matching on any weighted bipartite graph[14]. We shall not enumerate further on these algorithms but treat them as a black box in our algorithms and ensuing discussion.

The pseudo-code for the *RecodeOnJoin* algorithm executed by a new node n, which uses the maximum weighted matching on G', is shown in Fig 3 and is self-explanatory. Note that this is a local recoding strategy since the onus of recoding

RecodeOnJoin(Node n**)**
1 Obtain the constraints $(u, oldcolor(v))$ of the from-neighbors u of n, $u \in 1_n \cup 2_n, v \notin 1_n \cup 2_n \cup \{n\}$.
2 Obtain the constraints $(n, oldcolor(v))$ for n, $v \notin 1_n \cup 2_n$.
3 Let max = the maximum color seen in these constraints or old colors in $1_n \cup 2_n$.
4 Let $V_1 = 1_n \cup 2_n \cup \{n\}$, $V_2 = \{1, \ldots, max\}$.
  Draw the bipartite graph G' by joining edges from each vertex v in V_1 to each color k in V_2 that it can be assigned without conflicting with the constraints with any of the nodes not in $1_n \cup 2_n \cup \{n\}$.
    Assign this edge weight 3 if this is the old color assigned to v, otherwise assign it a weight of 1.
5 Run the bipartite matching algorithm on G. For each edge in v that is matched to some edge (v, k), assign it k as the new color.
  For all nodes in V_1 not assigned a color above, say m of them, randomly assign them colors $max + 1, \ldots, max + m$
6 Dissipate this information to all concerned nodes, agreeing on when to change color.

Figure 3: *RecodeOnJoin*

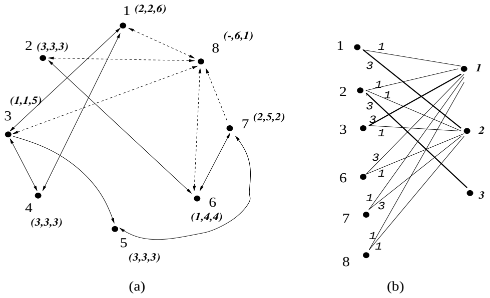


Figure 4: (a) Node 8 joins the network. Dotted edges added by the join. Against each node is shown its (old color, new color by *RecodeOnJoin*, new color by *CP*) (b) Weighted bipartite graph used by *RecodeOnJoin*; dark edges show matching constructed by *RecodeOnJoin*.

the nodes in $1_n \cup 2_n \cup \{n\}$ is *locally centralized* at node n, and uses only local information. The theorems stating the termination, correctness and claimed minimality and optimality among minimal algorithms properties of our algorithm appear in Appendix A.

An example for the recoding by the *CP* strategy and *RecodeOnJoin* when a new node (8) joins an ad-hoc network is shown in Fig 4. The reader is encouraged to work through this simple example to understand the presented algorithm. Note that *RecodeOnJoin* causes only 3 recodings while the *CP* join strategy (which uses a highest-first node ordering) causes 4 of them. Both end up using the same maximum color index after the join event (6).

Ignoring the latency of dissipation of color information in steps 1, 2 and 6, the complexity of $RecodeOnJoin(n)$ is dominated by the bipartite matching step 4. If the maximum in- and out-degree of any node in the network is k, *RecodeOnJoin* has a complexity of $O(k^9 ln(k))$. In a planar ad-hoc network, k would be expected to be a constant (as in planar graphs), thus giving us a constant expected time complexity for *RecodeOnJoin*.

4.2 Handling Node Power Increase

Let us look at Fig 2 again and envision what happens to the constraints on the nodes in $1_n, 2_n, 3_n, 4_n$ when n increases its maximum power range r_i by some amount. Nodes which were earlier in set 4_n might now be included in 3_n and nodes earlier in 1_n might jump into 2_n. However, note that *no new constraints are induced among the nodes in* $1_n \cup 2_n \cup 3_n \cup 4_n$ *due to this*. In other words, *all* constraints due to CA1 *and* CA2 added by the new edges involve node n !! If n's old color can no longer be assigned to it because of a new constraint, then the minimal recoding would need at least one node to change its color - we chose n to be this node, thus achieving the minimal bound. If n's color has no conflict with the new constraints on n, then the minimal recoding changes no colors. This is exactly what algorithm *RecodeOnPowIncrease* does (Fig 5).

We now extend the *CP* strategy to account for recoding on power range increase. When a node n increases its power range, all nodes up to two

RecodeOnPowIncrease(Node n)

```
Obtain the (new) constraints (n, oldcolor(v)), v ≠ n for self (n).
If current color does not violate new constraints, stop.
Recode n with the lowest available color that does not violate
  any of the constraints.
```

Figure 5: *RecodeOnPowIncrease*

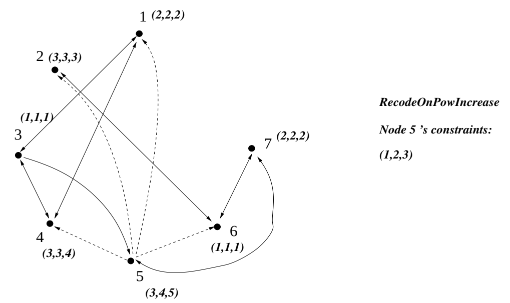


Figure 6: Node 5 increases its range. Dotted edges added to network. Against each node is shown its (old color, new color by *RecodeOnPowIncrease*, new color by *CP*)

hops away from n that now have a new constraint (due to either CA1 or CA2) with n and the same old color as n (and thus have a conflict with n), consider themselves for recoding. These nodes, along with n, do so in a distributed fashion in increasing or decreasing order of their identities, in a manner similar to the algorithm presented in section 3.

An example comparing the performance of the *CP* and *RecodeOnPowIncrease* strategies is shown in Fig 6, where node 5 increases its maximum transmission range to now include node $1, 2, 4, 6$ within its receiving range, thus setting up new constraints. *RecodeOnPowIncrease* causes only 1 new recoding while the *CP* strategy causes 2 nodes to be assigned different new colors. *RecodeOnPowIncrease* ends up with a lower maximum color index in the network (4) as against *CP*'s 5.

The termination, correctness and minimality properties of this algorithm are stated with proofs in Appendix B. However, note that *RecodeOnPowIncrease* may not always achieve the optimal bound among all minimal recoding strategies for recoding when a node increases its r_n. Consider the example of n having only one

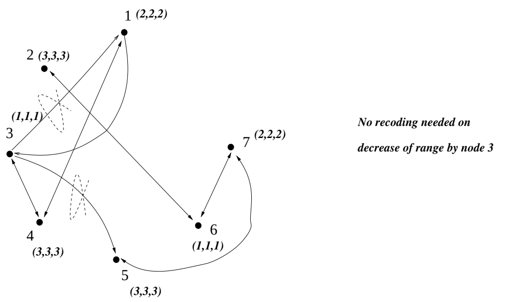


Figure 7: Node 3 decreases its power range. Crossed out edges deleted and old code assignment is valid. Against each node is shown its (old color, new color by *RecodeDecreasePowOrLeave*, new color by *CP*)

RecodeOnMove(Node n)

0 Define $1_n, 2_n, 3_n, 4_n$ for the node n in its new position as in section 4.1.
1 Obtain the constraints $(u, oldcolor(v))$ of the from-neighbors u of n, $u \in 1_n \cup 2_n, v \notin 1_n \cup 2_n \cup \{n\}$.
2 Obtain the constraints $(n, oldcolor(v))$ for n, $v \notin 1_n \cup 2_n$.
3 Let $max =$ the maximum color seen in these constraints and in the old colors of nodes in $1_n \cup 2_n$.
4 Let $V_1 = 1_n \cup 2_n \cup \{n\}$, $V_2 = \{1, \ldots, max\}$.
  Draw the bipartite graph G' by joining edges from each vertex v in V_1 to each color k in V_2 that it can be assigned without conflicting with the constraints with any of the nodes not in $1_n \cup 2_n \cup \{n\}$.
    Assign this edge weight 3 if this is the old color assigned to v, otherwise assign it a weight of 1.
5 Run the bipartite matching algorithm on G. For each edge in v that is matched to some edge (v, k), assign it k as the new color.
  For all unmatched vertices in V_1, say m of them, randomly assign them colors $max + 1, \ldots, max + m$
6 Dissipate this information to all concerned nodes, agreeing on when to change color.

Figure 8: *RecodeOnMove*

new constraint with another node m. If n has lots of old constraints (which still hold) and m very few, recoding only m might be more optimal in terms of maximum color index assigned to the network, while achieving the minimal recoding bound.

RecodeOnPowIncrease's time complexity, ignoring the constraint collection step 1, is $O(k^2)$, where $k =$ maximum in- and out-degree of any node in the network.

4.3 Handling Node Leaves and Power Decreases

As in [3], our approach for *RecodeDecreasePowOrLeave* adopts the passive strategy of no recoding when a node leaves the network or decreases its power, since no new conflicts are introduced by these events and thus the minimum number of codes to be changed in the network to maintain no conflicts is zero. An example is shown in Fig 7. The termination, correctness, minimality, and optimality among minimality properties of this algorithm follow from the above discussion and are stated in Appendix C for completeness.

4.4 Handling Node Movement

As mentioned in section 3, the *CP* strategy for handling recoding on node movement is to treat it as a pair of consecutive events where the moving node n leaves and joins the network. Such an approach can be very costly as mobility is inherent to ad-hoc networks.

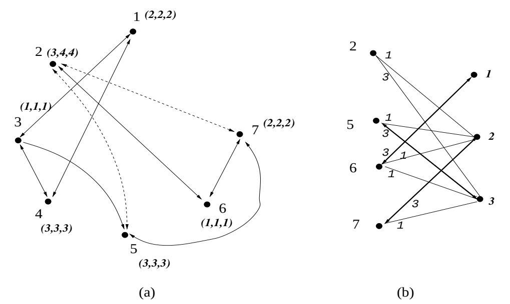


Figure 9: (a) Node 2 moves. Dotted edges added by the move. Against each node is shown its (old color, new color by *RecodeOnMove*, new color by *CP*) (b) Weighted bipartite graph used by *RecodeOnMove*; dark edges show matching constructed by *RecodeOnMove*.

Our algorithm for recoding on a node move *RecodeOnMove* is shown in Fig 8 and is very similar to *RecodeOnJoin*. In fact, were the moving node n to leave the network and then join it immediately, this would be the exact sequence of steps executed for the recoding (Appendix D). Notice, however, that our strategy is more advantageous than the *CP* strategy since a moving node disconnects from and connects to only some nodes, not the entire network.

The termination, correctness, minimality and optimality among minimality properties of this algorithm are stated in Appendix D. Their proofs are analogous to those in Appendix A. An example for the working of *RecodeOnMove* is shown in Fig 9. Both *RecodeOnMove* and the *CP* (which uses a highest-first node ordering) strategies cause 1 new recoding and end up with 4 as the maximum color index in the network after this event.

5 Simulation Results

The recoding algorithms presented in the last section are provably minimal only for the individual events considered. In this section, we address the question: how well do the set of our minimal recoding strategies, call them *Minim*, of section 4 perform for a long sequence of events in an ad-hoc network ? We evaluate the performance of the *Minim* strategies against (1) a strategy that uses a centralized coloring heuristic: the *BBB* algorithm of [7], to recolor the entire network at every event, and (2) the *CP* recoloring strategies. The performance metrics we are concerned with are 1) the maximum color index assigned in the network (the lower, the better is the code reuse) and 2) the number of nodes recolored (recoded with a new color different from its old one). This section handles this issue via discrete event simulations.

These experiments were carried out on random ad-hoc networks generated on a 2 dimensional space 100 units × 100 units square, with nodes arbitrarily joining, leaving, moving about, increasing and decreasing their maximum transmission ranges. We investigated the algorithms for recoding on a node join (section 5.1), node power increase (section 5.2) and node movement (section 5.3). The other two strategies of section 4.3 are trivial and the same as [3], hence we did not evaluate them. In order to maintain the generality of our simulation results, we have widely varied the different parameters and shown plots for the same. All points on all plots are the average of the metric measured over 100 runs of randomly generated ad-hoc networks for that particular setting of parameters.

5.1 Node Join

N nodes were consecutively allowed to join the ad-hoc network. Their position was determined by choosing their x and y coordinates independently and uniformly from the interval $[0, 100]$. Their transmission ranges were chosen uniformly in the interval $(minr, maxr)$. For this experiment, Fig 10 shows the two metrics for the three different algorithms with increasing N (Figs 10(a-c) with $minr = 20.5, maxr = 30.5$), and $\frac{(minr+maxr)}{2}$ (Figs 10(d-f) with $N = 100$ and $maxr - minr = 5.0$).

In general, the distributed algorithms - *CP* and *Minim*, achieve an almost linear variation (in N) of the total number of recodings compared to the global approach *BBB* (Fig 10(b)), with *Minim* outperforming *CP* (Fig 10(c)). *BBB* performs badly since it recolors the entire network at each event. In terms of maximum color index used, *Minim* comes closer to the (near-optimal) *BBB* heuristic than the *CP* approach (Fig 10(a)). These observations hold for a variation of transmission range too (Fig 10(d)).

Evidently, the *Minim* approach is preferable to the *CP* and the *BBB* global approaches as the best distributed recoding heuristic in the Ad-hoc network scenario.

5.2 Node Transmission Range Increase

To measure the effectiveness of $RecodeOnPowIncrease$, we started with the ad-hoc networks and the code assignment thereof generated in the last section (with $N = 100, minr = 20.5, maxr = 30.5$). A parameter $raisefactor$ was introduced, and half of the N nodes in the ad-hoc network were randomly chosen and their power ranges increased by a factor of $raisefactor$. The change (Δ's) in maximum color index assigned in the network and the total number of recodings were measured for this sequence of range increase events and are shown respectively in Fig 11(a) and Figs 11(b,c).

The *CP* approach performs better than the *Minim* minimal approach in terms of maximum color index assigned to the network (Fig 11(a)). This is because unlike the modified *CP* strategy to handle range increases, the minimal $RecodeOnPowIncrease$ strategy is very simple and does not care about minimizing the number of maximum color index after reassignment. However, this disadvantage is completely offset by the advantage gained by the much lesser number of recodings in the *Minim* case compared to the *CP* (and the *BBB*) strategy (Figs 11(b,c)).

For example, at $raisefactor = 4$, *Minim* performs worse than *CP* in the maximum color index metric by only 6 colors but outperforms it by around 50 recodings.

We conclude that while the *Minim* approach for recoding on power range increase performs only slightly worse than the *CP* strategy on the maximum color index assigned, it outperforms it by a huge margin in the total number of recodings. It is of course up to the system designer to choose between these two approaches depending on which she prefers more - a lesser number of codes or a lesser number of recodings.

5.3 Node Movement

To evaluate the performance of $RecodeOnMove$, we started with the networks generated in section 5.1. We introduced two new parameters; *maxdisp*, the maximum displacement of a node in any of the directions, and *RoundNo*, which we will explain shortly. In this experiment, there were several rounds, in each of which all the N nodes in the network were moved, one by one, in a random direction in the x-y plane by a displacement chosen uniformly in the interval $[0, maxdisp]$. *RoundNo* such rounds were carried out for different values of *maxdisp* and *RoundNo*. The two metrics - change in maximum color index in the network, and total number of recodings, were measured for this sequence of node move events (Figs 12). The values of $N = 40, minr = 20.5, maxr = 30.5$ were used for these experiments.

Fig 12(a) shows the total number of recodings for different values of *maxdisp*, with $RoundNo = 1$. The *Minim* strategy evidently outperforms the *CP* strategy. Figs 12(b-d) show the effect of varying *RoundNo*, with $maxdisp = 40$. Fig 12(b) shows the change in the maximum color index assigned in the network as we go through more and more rounds of movement. This metric obviously does not change too much even for as many as 10 rounds, and the *Minim* strategy performs worse than the *CP* strategy by at most a couple of colors. Again, Figs 12(c,d) show us the vast advantage to be gained in the number of recodings for node movement by using the *Minim* strategies. The *Minim* strategy improves vastly upon the *CP* strategy as rounds progress (eg., for $RoundNo = 10$, the *Minim* achieves 400 fewer recodings than *CP* !).

The conclusions here are similar to the previous sections - the tradeoff achieved by using the *Minim* strategies is the use of a few more extra colors (codes) for a vast gain in the number of recodings.

6 Conclusions and Future Work

The problem of CDMA code assignment to eliminate collisions in packet radio networks has been widely researched in the past, but none of the algorithms proposed for code assignment in a dynamic scenario guarantee any strong performance bounds. In this paper, we have presented a set of *recoding* strategies *Minim* for TOCA CDMA recoding in an ad-hoc network where mobiles can arbitrarily 1) connect and disconnect, 2) move about, and 3) increase or decrease their transmission power. Our strategies, unlike those proposed earlier in literature, have been proved to guarantee *minimal recoding*, that is, given a current code assignment and one of the above events, our strategies change the codes of the minimum number of mobiles needed to eliminate all collisions in the network after the event. We have further proved that, among all possible minimal recoding strategies that consider only some subset of nodes for recoding (one-hop neighbors of the node executing the event), most of our strategies are optimal in terms of number of codes assigned in the network. Simulation results reveal that our *Minim* approaches trade off a relatively small loss in terms of maximum color index assigned in the network to obtain a significant gain in terms of the total number of instances where a node has to change its code. The proposed *Minim* strategies can be very practical in scenarios such as hard real-time systems and high data rate applications running on an ad-hoc network, where it is much more preferable to use a few more codes in the network than to suffer the (possibly) critical loss incurred by changing the codes of several mobiles.

Future work will focus on a recoding strategy that seeks to maximize the network-wide code reuse by using a local gossiping strategy. This gossiping strategy would work during the (possibly significantly long) periods when no nodes connect to, move about or increase their power within the ad-hoc network.

References

[1] C.E. Perkins, "Mobile networking in the internet", *Mobile Networks and Applications*, vol. 3, no. 4, January 1999, pp. 319-334.

[2] T. Makansi, "Transmitter-oriented code assignment for multihop packet radio", *IEEE Trans. Communications*, vol. com-35, no. 12, December 1987, pp. 1379-1384.

[3] I. Chlamtac, S.S. Pinter, "Distributed nodes organization algorithm for channel access in a multihop dynamic radio network", *IEEE Trans. Computers*, vol. c-36, no. 6, June 1987, pp. 728-737.

[4] L. Hu, "Distributed code assignments for CDMA packet radio networks", *IEEE/ACM Trans. Networking*, vol. 1, no. 6, December 1993, pp. 668-677.

[5] A.A. Bertossi, M.A. Bonuccelli, "Code assignment for hidden terminal interference avoidance in multihop packet radio networks", *IEEE/ACM Trans. Networking*, vol. 3, no. 4, August 1995, pp. 441-449.

[6] S. Khuri, W.M. Moh, F. Chung, "Code assignments in CDMA networks: distributed algorithms and genetic algorithm heuristics", *Proc. 6th IEEE Singapore Intl. Conf. Networks 98*, 1998, pp. 91-105.

[7] R. Battiti, A.A. Bertossi, M.A. Bonucceli, "Assigning code in wireless networks: bounds and scaling properties", *Wireless Networks*, vol. 5, no. 3, August 1999, pp. 195-209.

[8] S. Ramanatham, "A unified framework and algorithm for channel assignment in wireless networks", *Wireless Networks*, vol. 5, no. 2, March 1999, pp. 81-94.

[9] D. Brelaz, "New methods to color the vertices of a graph", *Commun. ACM*, vol. 22, 1979, pp. 251-256.

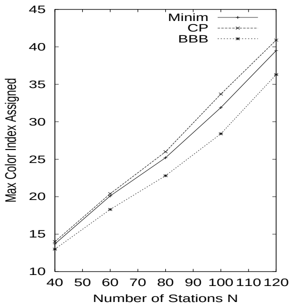
[10] L.M. Kirousis, E. Kranakis, D. Krizanc, A. Pelc, "Power consumption in packet radio networks", *Proc. STACS 97: 14th Annual Symp. Theoretical Aspects of Computer Science*, 1997, pp. 363-374.

[11] M. Adler, C. Scheideler, "Efficient communication strategies for ad-hoc wireless networks", *Proc. SPAA 98: 10th Annual ACM Symp. Parallel Algorithms and Architectures*, 1998, pp. 259-268.

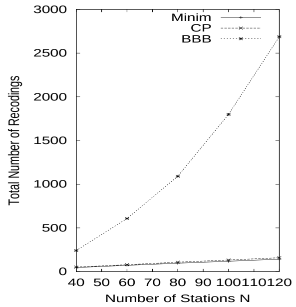
[12] L. Hu, "A novel topology control for multihop packet radio networks", *Proc. INFOCOM 1991*, vol. 3, 1991, pp. 1084-1093.

[13] Kang G. Shin and P.Ramanathan, "Real-time computing: A new discipline of computer science and engineering", *Proc. IEEE,* vol. 82, no. 1, pp. 6-24, Jan 1994.

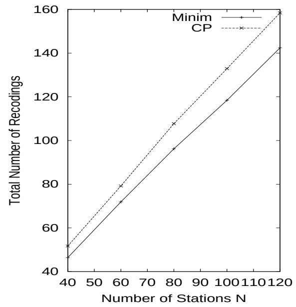
[14] Z. Galil, "Efficient algorithms for finding maximum matching in graphs", *ACM Computing Surveys*, vol. 18, no. 1, March 1998, pp. 23-38.



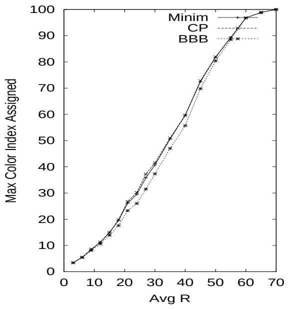
(a) Total # Colors vs N



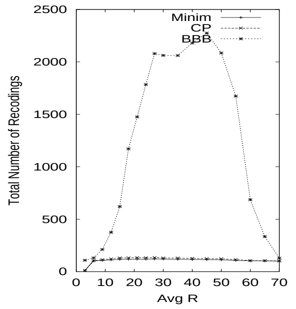
(b) # Recodings vs N



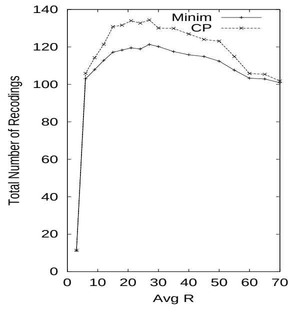
(c) # Recodings vs N



(d) # Colors vs $\frac{(minr+maxr)}{2}$

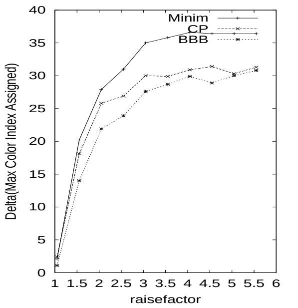


(e) # Recodings vs $\frac{(minr+maxr)}{2}$

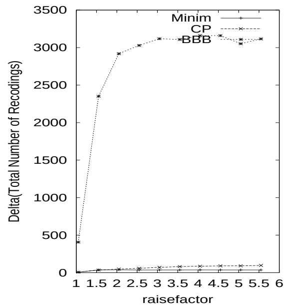


(f) # Recodings vs $\frac{(minr+maxr)}{2}$

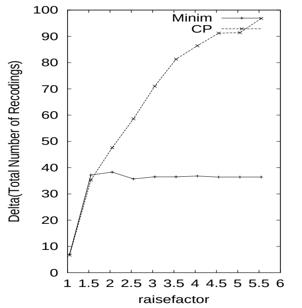
Figure 10: Simulation Results - Node Join



(a) Δ(max color used) vs raisefactor

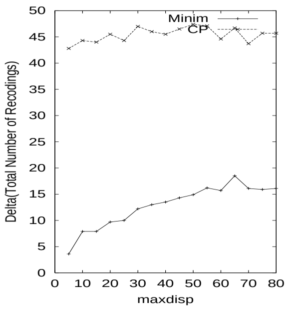


(b) Δ(# Recodings) vs raisefactor

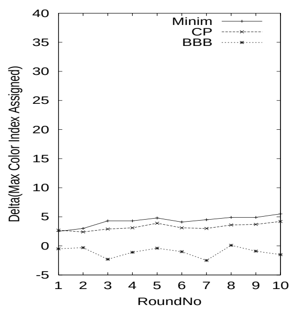


(c) Δ(# Recodings) vs raisefactor

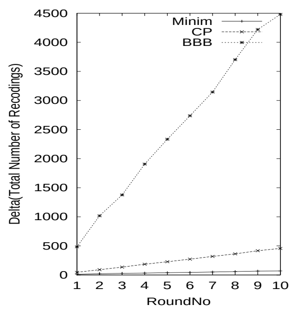
Figure 11: Simulation Results - Node Power Increase



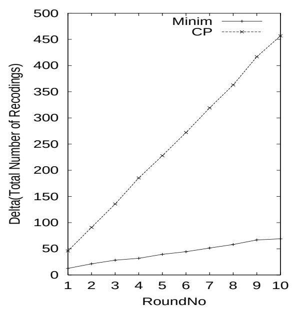
(a) Δ(# Recodings) vs $maxdisp$



(b) Δ(max color used) vs $RoundNo$



(c) Δ(max color used) vs $RoundNo$



(d) Δ(# Recodings) vs $RoundNo$

Figure 12: Simulation Results - Node Movement

Appendix A: Properties of *RecodeOnJoin*

Lemma 4.1.1: Apart from recoding n, at least $\sum_{i=1}^{m}(K_i - 1) = \sum_{i=1}^{m} K_i - m$ of the nodes in $1_n \cup 2_n$ need to be recoded with different new colors to avoid conflicts in the new code assignment after node n joins. This is the minimal recoding bound.
Proof: See section 4.1. □

Theorem 4.1.2 (Termination): If no nodes crash, messages are eventually delivered, the Minimal Connectivity assumption is maintained, and events/reconfigurations sequenced, *RecodeOnJoin* terminates.
Proof: By the Minimal connectivity assumption, no-failure assumption and eventual message delivery assumption, steps 1, 2 and 6 all terminate. The rest of the steps are carried out by the node n and terminate as n does not crash and the steps are deterministic. □

Fact 4.1.3: No node in $1_n \cup 2_n \cup \{n\}$ is reassigned the same new color as another node in the same set. This follows from the definition of a matching and step 5 of *RecodeOnJoin*.

Theorem 4.1.4 (Correctness): The new code assignment by *RecodeOnJoin* avoids primary and hidden collisions.
Proof: In this proof, we shall refer to the constraints induced by the joining of n as *new* constraints and those that existed prior to n executing *RecodeOnJoin* as *old* constraints. First, note that all nodes in $1_n \cup 2_n$ assigned a color by the second part of step 5 in *RecodeOnJoin* have no collisions with any node in the network (by the definition of *max* in step 3). Next, we show the same for nodes reassigned by the first part of step 5. The new constraints that violate CA1 are the edges between n and nodes in $1_n \cup 2_n$, 3_n. However, *RecodeOnJoin* will not violate any of the constraints generated between n and the nodes in 3_n as these edges will not be added to G' (step 4). *RecodeOnJoin* will not violate any of the constraints between n and $1_n \cup 2_n$ since all nodes in $1_n \cup 2_n \cup \{n\}$ end up with different new colors from each other (Fact 4.1.3). The new constraints that violate CA2 are that each pair of nodes in $1_n \cup 2_n \cup \{n\}$ should have different colors and each node that is a from-neighbor of a node in $2_n \cup 3_n$ should have a different color from n. Again, as earlier, the second requirement is met by the edges not added by step 2, and the first is met by the fact that all nodes in $1_n \cup 2_n \cup \{n\}$ are reassigned with different new colors from each other (Fact 4.1.3).

The old constraints on pairs of nodes neither of which are in $1_n \cup 2_n \cup \{n\}$ are not violated as these pairs of nodes have the same new and old colors. The old constraints on pairs of nodes, one of which, say u, is in $1_n \cup 2_n \cup \{n\}$ and the other is not, are not violated by the edges not added to u in step 4 of *RecodeOnJoin*. □

Fact 4.1.5: Each node in V_1 (step 4 of *RecodeOnJoin*) is adjacent to at most one edge with weight 3. This follows since a node $u \in V_1$ can be associated with only one old color before this event.

Lemma 4.1.6: Let $u \in 1_n \cup 2_n$, and C_u be its old color. Then G' as constructed in step 4 of *RecodeOnJoin* has the edge (u, C_u) and this has a weight 3.
Proof: Consider the new constraints induced on node u in $1_n \cup 2_n$ by the new node n joining the network. From steps 1 and 4 of *RecodeOnJoin*, such a constraint will affect the presence of an edge in G' iff they are between u and a node $v \notin 1_n \cup 2_n \cup \{n\}$. However, it is easy to note from Fig 2 that no *new* constraints that violate either CA1 or CA2 are induced on any node $u \in 1_n \cup 2_n$ and another node $v \notin 1_n \cup 2_n \cup \{n\}$. In other words, a constraint between u and a node $v \notin 1_n \cup 2_n \cup \{n\}$ were already satisfied by the old code assignment. However, since no node not in $1_n \cup 2_n \cup \{n\}$ is recoded by *RecodeOnJoin* and the old color C_u assigned to u does not violate any of the constraints between u and nodes not in $1_n \cup 2_n \cup \{n\}$, step 4 of *RecodeOnJoin* will add the edge (u, C_u) to G', and with a weight 3. □

Lemma 4.1.7: From the above fact and lemma, it follows that in G', each node $u \in 1_n \cup 2_n$ is associated with exactly one edge of weight 3 and that too to a node $C_u \in V_2$, which is its old color. Moreover, all edges incident with n in G' have weight 1.

Theorem 4.1.8 (Minimality): At the event where n joins the network, *RecodeOnJoin* achieves the minimal recoding bound among all possible recoding strategies.

Proof: Clearly, *RecodeOnJoin* will achieve the minimal recoding bound if every color in $\{C_1, \ldots, C_m\}$, the old set of colors assigned to nodes in $1_n \cup 2_n$, is present in the matching found in step 5. This is true since then $\sum_{i=1}^{m}(K_i - 1) = \sum_{i=1}^{m} K_i - m$ nodes in $1_n \cup 2_n$ will then be recoded with new colors (n will have to be recoded anyway).

Suppose, by contradiction, there is a color C_i in $\{C_1, \ldots, C_m\}$ associated with a vertex $v \in V_1$ by the matching found by step 5 (call this matching M) and that (v, C_i) has a weight ≤ 1 (0 if C_i is not present in the matching at all). There is at least one node $u \in V_1$ such that (u, C_i) is in G' and has weight 3 (lemma 4.1.6). Again, let u be matched with some color $C_j \neq C_i$ by step 4. The weight of this matched edge is ≤ 1 (lemma 4.1.7). Now consider the matching M' obtained by removing these two edges $(u, C_j), (v, C_i)$ (if they exist) from M and adding the edge (u, C_i). Clearly, M' is also a matching. Moreover, M' has a weight at least $3-1-1=1$ more than the weight of matching M. This is a contradiction to the maximality of M. □

Theorem 4.1.9 (Optimality among Minimality): At the event where n joins the network, among all correct recodings of the network that achieve the minimal recoding bound and consider only nodes in $1_n \cup 2_n \cup \{n\}$ for recoding, *RecodeOnJoin* gives us one which (re-)assigns the least maximum color to any node.

Proof: The theorem is true if the matching M found by step 5 of *RecodeOnJoin* has $|V_1|$ edges. Therefore, let us assume that the second part of step 5 assigns at least one color greater than max. Now assume, by way of contradiction, the existence of another adversary correct recoding of nodes in $1_n \cup 2_n \cup \{n\}$ that uses less than the colors used by *RecodeOnJoin*, and is further the best among all such adversaries. Consider the matching M'' induced on graph G' (defined in step 4 of *RecodeOnJoin*) by such a recoding. Since at least one more node in V_1 is matched in M'' than in M and both M'' and M have m edges of weight 3 (otherwise an argument similar to the proof of Theorem 4.1.8 can be used to improve the adversary recoding), this means that M'' has more edges of weight 1 than M and is hence a matching on G' with a larger weight than M. This is a contradiction to the maximality of M. □.

Theorem 4.1.10: The algorithm supports simultaneous additions of new nodes when any two of them are at least 5 hops apart.

Proof: We omit this proof as it is similar to that of Theorem 5.1 in [3] p. 735. □

Appendix B: Properties of *RecodeOnPowIncrease*

Theorem 4.2.1 (Termination): If no nodes crash, messages are eventually delivered, the Minimal Connectivity assumption is maintained, and events/reconfigurations happen only one after the other, *RecodeOnPowIncrease* terminates.

Proof: Step 1 terminates as all messages are delivered and no processes crash. Steps 2 and 3 complete as n itself does not crash. □

Theorem 4.2.2 (Correctness): The new code assignment by *RecodeOnPowIncrease* avoids primary and hidden collisions.

Proof: From prior discussion in section 4.4. □

Theorem 4.2.3 (Minimality): *RecodeOnPowIncrease* achieves the minimal recoding bound for power increase.

Proof: From prior discussion in section 4.4. □

Appendix C: Properties of *RecodeDecreasePowOrLeave*

Theorem 4.3.1 (Termination): If no nodes crash, messages are eventually delivered, the Minimal Connectivity assumption is maintained, and events/reconfigurations happen only one after the other, *RecodeDecreasePowOrLeave* terminates.

Theorem 4.3.2 (Correctness): The new code assignment by *RecodeOnPowIncrease* avoids primary and hidden collisions.

Theorem 4.3.3 (Minimality): *RecodeDecreasePowOrLeave* achieves the minimal recoding bound for the events where a node n leaves or decreases r_n.

Theorem 4.3.4 (Optimality among Minimality): $RecodeDecreasePowOrLeave$ achieves the minimal optimal recoding for the events where a node n leaves or decreases r_n.

Appendix D: Properties of $RecodeOnMove$

Theorem 4.4.1: A $RecodeOnMove(n)$ is equivalent to a $RecodeDecreasePowOrLeave(n)$ at n's old position, followed by a $RecodeOnJoin(n)$ at the new position of n.

Proof: This follows directly by looking at the pseudo-codes for $RecodeDecreasePowOrLeave(n)$, $RecodeOnJoin(n)$ and $RecodeOnMove(n)$. □

Theorem 4.4.2 (Termination): If no nodes crash, messages are eventually delivered, the Minimal Connectivity assumption is maintained, and events/reconfigurations happen only one after the other, $RecodeOnMove$ terminates.

Theorem 4.4.3 (Correctness): The new code assignment by $RecodeOnPowIncrease$ avoids primary and hidden collisions.

Theorem 4.4.4 (Minimality): $RecodeOnMove$ achieves the minimal recoding bound for the event in which a node moves within the network.

Theorem 4.4.5 (Optimality among Minimality): $RecodeOnMove$ achieves the minimal optimal recoding for the event in which a node moves within the network.