Lightweight Coloring and Desynchronization for Networks

Arik Motskin, Tim Roughgarden, Primoz Skraba and Leonidas Guibas
Stanford University, Stanford, CA 94305
Email: {amotskin, primoz}@stanford.edu, {tim, guibas}@cs.stanford.edu

***Abstract*—We study the distributed desynchronization problem for graphs with arbitrary topology. Motivated by the severe computational limitations of sensor networks, we present a randomized algorithm for network desynchronization that uses an extremely lightweight model of computation, while being robust to link volatility and node failure. These techniques also provide novel, ultra-lightweight randomized algorithms for quickly computing distributed vertex colorings using an asymptotically optimal number of colors.**

I. Introduction

As inherently distributed computational systems, sensor networks rely critically on coordination between nodes to effectively sense, communicate and interpret environmental data. Individual nodes face severe battery and computational limitations, so a notion of coordinated task-sharing and duty-cycling is critical to maintaining the longevity and efficient operation of the network. In this sense, *desynchronizing* the actions of nodes is desirable. Efficient desynchronization protocols can be applied to a variety of sensor network applications, including periodic resource sharing, coordinated sleep schedules, and evenly shared sensing burden across nearby nodes [3].

A. The Problem: Desynchronization

We consider the following desynchronization problem: given an undirected graph $G = (V, E)$, a target interval length l_v for every node $v \in V$, and a circle of circumference T, we wish to *feasibly* assign to each node v a contiguous interval of the circle of length l_v. The assignment is feasible if for each $(u, v) \in E$, the assigned intervals of u and v do not overlap. Observe that the vertex coloring problem is a special case of desynchronization, where given a set of s available colors, each node has the same target interval length $\frac{T}{s}$ and the set of possible intervals on the circle is restricted to s non-overlapping intervals of length $\frac{T}{s}$.

Our aim is to solve this problem in a decentralized manner: nodes are modeled as processors that may select intervals based only on local information. Time proceeds in rounds, and in each round nodes can interact only with their 1-hop neighbors. In the vertex coloring context, we think of time as proceeding in *synchronous* rounds. For the more general desynchronization problem we think of time as proceeding continuously in phases of length T, with intervals of the circle corresponding to intervals of time.

While minimum graph-coloring is NP-hard [6] (and hard to approximate better than $\Omega(|V|^{1-\varepsilon})$ [4]), it is easy to color any graph with $\Delta + 1$ colors – corresponding to target intervals of length $\frac{T}{\Delta+1}$ – in n rounds by the centralized greedy algorithm, where n is the total number of nodes and Δ is the maximum degree of the graph. Our goal is to achieve convergence time exponentially faster than the sequential greedy algorithm, while using target intervals of comparable length.

B. Computational Model

The key consideration when designing sensor network protocols is to keep them lightweight. As nodes are limited in processing power, simple computations are crucial. More importantly, communication between nodes is expensive [17], so all but the most critical message-passing should be avoided.

Motivated by these concerns, we consider an extremely lightweight, novel computational model. We define a *(c,d)-bit local algorithm* on the graph G, such that each node maintains a legal output (e.g. a color, or an interval), d bits of additional state, and bases its actions in a given round only on its current state and some c-bit feedback from previous rounds. In the distributed vertex coloring context, we define the feedback as whether or not a node's color conflicts with any of its neighbor's colors. In the sensor network context, the notion of feedback is adjusted to suit the continuous nature of the problem. It is defined as whether any of a node v's neighbors are broadcasting when v is listening.

C. Our Results

We present a randomized $(2, 1)$-bit local desynchronization algorithm with $l_v = T/2(\hat{d}_v+1)$, that converges in $\mathcal{O}(\Delta \log n)$ periods with high probability. A period has length T, so each node v is feasibly assigned a subinterval of that period of length l_v. Parameter $\hat{d}_v$ denotes the maximum degree in v's 1-hop neighborhood. At convergence, this means that each node has claimed a subinterval (of length $T/2(\hat{d}_v + 1)$) of the repeating period (of length T), such that the intervals of any two neighbors do not overlap. This is achieved with each node requiring only a single bit of state beyond what is required to maintain an interval[1], and at most two bits of feedback from the previous period.

The techniques we use are based on a novel lightweight way to compute a distributed vertex coloring. In particular, we give a randomized $(1, 1)$-bit local coloring algorithm that uses $\Delta + 1$ colors and converges in $\mathcal{O}(\Delta \log n)$ rounds with

[1]An interval is fully specified by two parameters: the interval length and the phase of the repeating period at which the interval begins.

high probability, and a randomized $(1,0)$-bit local coloring algorithm that uses $\mathcal{O}(\Delta)$ colors and converges in $\mathcal{O}(\log n)$ rounds with high probability. We also prove a lower bound of $\Omega(\log n/\log\Delta)$ rounds for $(1,\beta)$-bit local algorithms that use $\mathcal{O}(\Delta)$ colors, for any $\beta \geq 0$.

Our general desynchronization solution has several properties that are particularly appealing in the sensor networks context:

1) Our algorithm is very robust to network volatility, easily handling changes in network topology.
2) Nodes need not synchronize to a global clock (assuming no clock drift).
3) Nodes in sparse neighborhoods of G are assigned larger intervals than those in dense neighborhoods.
4) There is a simple trade-off characterization between convergence time and interval length values.

D. Related Work

To the best of our knowledge, the only past work on desynchronization is [3] and [14], where the problem is solved for the complete graph using target values of $\frac{T}{n}$. On the other hand, distributed vertex coloring is a well-studied problem. The standard model of communication is the powerful *message-passing* paradigm [12], where in each round nodes can trade messages of arbitrary size with their neighbors, and engage in arbitrary computation. The best upper bound for arbitrary graphs in this model is given by [12] and [13], which colors a graph using $\Delta+1$ colors in expected $\mathcal{O}(\log n)$ time, by means of a reduction to the maximal independent set (MIS) problem. Linial [12] also gives a deterministic algorithm that colors an arbitrary graph with $\mathcal{O}(\Delta^2)$ colors in $\mathcal{O}(\log^* n)$ rounds. Such bounds do not carry over to our computational model due to the complexity of the messages traded between nodes; Luby's MIS algorithm [13], for instance, requires neighbors to compare degree values if they discover a conflict with respect to the independent set property.

For lower bounds, there is a time lower bound of $\Omega(\Delta/\log^2\Delta + \log^* m)$ rounds when using $\mathcal{O}(\Delta)$ colors ($m = \#$ of edges) [10]. For the related (and potentially easier) problem of computing a MIS (there is an easy distributed way to go from a coloring to a MIS [9]) the best lower bounds are $\Omega(\sqrt{\log n/\log\log n})$ and $\Omega(\log\Delta/\log\log\Delta)$ rounds [8]. Separately, work on the *bit complexity* model [1] gives lower bounds on the number of communication bits required to complete various network tasks like leader election.

The plan for the paper is as follows. In Section II we discuss lightweight procedures for decentralized coloring and prove the lower bound on convergence time. In Section III we present the desynchronization algorithm, prove its convergence time and study several extensions. In Section IV, we present simulation results for the desynchronization algorithm. We conclude in Section V.

II. Decentralized Vertex Coloring

For our study of the decentralized vertex coloring problem, we assume that time proceeds in synchronous rounds, where in each round a node receives a single bit of feedback – whether or not it has a conflict with any of its neighbors – and then can select a new color if it wishes. No message passing of any kind is permitted. We also assume that each node v knows its own degree, denoted by d_v.

A. Conflict Detection Model

In the most restrictive model that we study – Conflict Detection – nodes must select a color for round t based *only* on their color and conflict status from round $t-1$. In particular, each node v is endowed with a color selection function f_v that is either deterministic or randomized, and selects v's next color based on feedback from last round. Formally,

$$f_v : C_v \text{ x } \{\text{conflict, no conflict}\} \rightarrow C_v$$

where C_v is the set of colors available to v. If $c_i \in C_v$ is the color selected by v in round i and $I_i \in \{0,1\}$ the conflict status in round i, node v generates a sequence of colors $\{c_1, c_2, c_3, ...\}$ where $c_{i+1} = f_v(c_i, I_i)$.

We present a randomized algorithm in the Conflict Detection Model that converges to a proper coloring in $\mathcal{O}(\log n)$ rounds with high probability using only $\mathcal{O}(\Delta)$ colors. That is, each node v has a set of colors C from which to choose, where $|C| = k\Delta$ for a constant $k \geq 1$. In fact, a stronger result holds: individual nodes need not know the maximum degree Δ of the entire graph, but instead only the maximum degree of its 1-hop neighborhood, denoted $\hat{d_v}$. Each node v can instead select only from a potentially much smaller set of colors $C_v \subset C$, where $|C_v| = k\hat{d_v}$ (for the same k as above). For purposes of technical and notational clarity, we prove only the result where nodes draw colors from the entire list C, but the stronger result follows by the same proof with only minor adjustments to the algebra.

Algorithm 1 (Conflict Detection)

```
1) Each node $v$ initializes with an arbitrary color $c$ drawn from $C_v$. $t \leftarrow 1$.
2) In round $t$, node $v$ checks whether or not it has a conflict with any of its neighbors
3) If no neighbor of $v$ is also colored $c$, $v$ selects the same color $c$ for round $t+1$; otherwise, $v$ selects a color for round $t+1$ uniformly at random from $C_v$. In either case, return to step 2. $t \leftarrow t+1$.
```

It is not immediately clear why the algorithm should achieve $\mathcal{O}(\log n)$ round convergence. Since a node that is without a conflict in some round t may very well become conflicted in a later round, it is not sufficient to show only that a constant fraction of conflicted nodes become unconflicted in any round. In Section II-C we show that by adding a single bit of memory – in effect turning nodes into two-state machines – we avoid the problem of nodes moving back and forth between being conflicted and being unconflicted. Nevertheless, observe that once *all* nodes in the graph are simultaneously unconflicted, the graph has converged to a proper coloring and no node wishes to switch its color; the global solution is verified only after it passes a local test at every node. Observe also that the algorithm is *self-maintaining*, in the sense that adjusting the coloring for newly formed or dropped edges is handled on-the-fly by the algorithm at a local level.

Suppose each node v has a set C of available colors, common to all nodes in the graph, where $|C| = k\Delta$ for constant $k \geq 1$. We say that v is *good* in round t if each of its neighbors is colored differently from v during that round; otherwise the node is *bad*. Let the random variable X_t denote the number of good nodes (out of n total) at round t. Since the algorithm halts at a proper coloring – when all nodes are good – we wish to characterize the time it takes for X_t to reach n. The following key proposition establishes that in the course of a round, in expectation at least a constant fraction of bad nodes become good.

Proposition 1 *If X_t is the number of good nodes in round t, then $\mathbb{E}[X_{t+1}|X_t] \geq X_t + 0.1(n - X_t)$ when $k \geq 5$.*

See Appendix A for the proof. The next proposition follows from a simple application of the law of iterated expectations.

Proposition 2 *Let $n > 0$ be an integer, and X_t an integer random variable in $[0, n]$ for $t \in \{0, 1, 2, \ldots\}$. Define variable R to be the smallest t such that $X_t = n$. Suppose there exists value $0 < c < 1$ such that $\forall t$, $\mathbb{E}[X_{t+1}|X_t] \geq X_t + c(n - X_t)$. Then for any integer $m \geq 0$,*

$$\mathbb{P}(R \geq \lceil \log_b n \rceil + m) \leq (1 - c)^m$$

where $b = \frac{1}{1-c}$.

We can now establish the convergence time of Algorithm 1.

Theorem 1 *Algorithm 1 converges to a proper coloring in $\mathcal{O}(\log n)$ rounds with high probability, when using at least 5Δ colors.*

Proof: If we let R be the number of rounds until the algorithm converges, then we interpret R as the number of rounds until all n nodes become good. Therefore by Propositions 1 and 2, for integer $m \geq 0$,

$$\mathbb{P}(R \geq \lceil \log_b n \rceil + m) \leq (1 - c)^m$$

where $c \geq 0.1$, and so $b \geq \frac{10}{9}$. Note that $\ln b > \frac{1}{10}$, so for concreteness it follows that $\mathbb{P}(R \geq 11 \ln n) = \mathcal{O}(\frac{1}{n})$. ∎

Note that the large constant is due both to the relatively loose analysis and the value $k = 5$, which can be increased in order to accelerate convergence time.

B. Lower Bound

Despite the simplicity of Algorithm 1, it is in a sense the only effective $(1, 0)$-bit local coloring algorithm. Moreover, it turns out that increasing each node's state size (but keeping the feedback fixed) does not lead to faster convergence. We prove a lower bound of $\Omega(\log n / \log \Delta)$ rounds for all $(1, \beta)$-bit local coloring algorithms using $\mathcal{O}(\Delta)$ colors, where β is *any* number of state bits. Recall that for coloring we defined the single bit of feedback to be whether or not a node has a conflict with any of its neighbors.

It is a lower bound in the following sense: given a node set V and a pre-determined degree d_v for each $v \in V$, then for any $(1, \beta)$-bit local algorithm there exists a graph satisfying these degree constraints for which the algorithm requires $\Omega(\log n / \log \Delta)$ rounds to converge with high probability. This means in particular that Algorithm 1 is optimal for small Δ.

Theorem 2 *For every $(1, \beta)$-bit local algorithm using $\mathcal{O}(\Delta)$ colors, where $\beta \geq 0$, the lower bound on convergence time is $\Omega(\log n / \log \Delta)$ rounds with high probability.*

See Appendix B for the proof.

C. Conflict Detection with Memory

In Section II-A, we achieved an upper bound of $\mathcal{O}(\log n)$ time using $\mathcal{O}(\Delta)$ colors in the very restrictive Conflict Detection model, asymptotically matching the best known coloring algorithm in the message-passing model for arbitrary graphs [12], [13]. In this section, however, we seek to match the upper bound of $\Delta + 1$ colors *exactly* while still using a restrictive model. For this purpose, we consider a more powerful version of the Conflict Detection Model, where nodes use an extra bit of memory to act as a switch between two algorithmic modes: a *search* mode and a *permanent* mode. As a two state machine, nodes must select a color and a state for round t based only on their color in round $t-1$, conflict status in round $t-1$, and state in round $t-1$. Each node v uses a color selection function g_v that is either deterministic or randomized, and selects v's next color based on feedback from last round. Formally,

$$g_v : C_v \text{ x } \{\text{conflict, no conflict}\} \text{ x } \{\textit{search, permanent}\} \\ \rightarrow C_v \text{ x } \{\textit{search, permanent}\}$$

where C_v is the set of colors available to v.

In this model, we present a randomized algorithm that converges to a proper coloring in $\mathcal{O}(\Delta \log n)$ rounds with high probability using only $\Delta + 1$ colors. Note that while the two-state machine is more frugal with the number of colors, the convergence time has increased for dense graphs. The algorithm uses Δ+1 colors in total, but as in the last algorithm, individual nodes need not know the maximum degree Δ. Rather, it is sufficient for a node v to select only from a subset of size $d_v + 1$ from the entire set of possible colors. One can thus think of colors as the positive integers, such that nodes u and v have the integers $\{1, 2, ..., \min\{d_u, d_v\} + 1\}$ common to both of them. The algorithm proceeds as follows:

Algorithm 2 (Conflict Detection with Memory)

1) *Each node v initializes in **search** mode, and selects an arbitrary color c_v from its set of $d_v + 1$ available colors. $t \leftarrow 1$.*
2) *In round t, given all color updates from the previous round, node v checks whether it has a conflict with any neighbors.*
3) *If v has no conflicts, it enters **permanent** mode and halts, having chosen c_v as its permanent color.*
4) *Otherwise, v selects a new color c uniformly at random from its set of $d_v + 1$ available colors, remains in **search** mode, and returns to step 2. $t \leftarrow t + 1$. $c_v \leftarrow c$.*

We remark that the algorithm ends once each node has entered *permanent* mode, and thus has settled on a permanent color. Moreover, if the algorithm does indeed halt, it does so at a proper vertex coloring.

Proposition 3 *If node v has not halted before round i, then $\mathbb{P}(v$ halts in round $i) \geq \frac{1}{d_v+1}$.*

Proof: In round i, at most d_v of node v's d_v+1 available colors will be selected by v's neighbors. So with probability $\geq \frac{1}{d_v+1}$, the node v will not have a conflict. ■

This establishes that in the course of a round, in expectation at least a $\frac{1}{\Delta+1}$ fraction of non-permanent nodes will become permanent. We use this result to establish the convergence time of the algorithm.

Theorem 3 *Algorithm 2 converges to a proper coloring in $\mathcal{O}(\Delta \log n)$ rounds with high probability.*

Proof: If we let R be the number of rounds until the algorithm converges, then we interpret R as the number of rounds until all n nodes have selected a permanent color. Therefore by Propositions 2 and 3, for an integer $m \geq 0$,

$$\mathbb{P}(R \geq \lceil \log_b n \rceil + m) \leq (1-c)^m$$

where $c \geq \frac{1}{\Delta+1}$, and so $b \geq \frac{\Delta+1}{\Delta}$. We remark that $\ln b \geq \frac{1}{2\Delta}$, so by setting $m = \Delta \ln n$ we get that

$$\mathbb{P}(R \geq 3\Delta \ln n) = \mathcal{O}(\tfrac{1}{n}).$$

■

Algorithm 2 also lends itself well to characterizing the tradeoff between the number of colors used and the speed of convergence. Suppose instead that for a fixed parameter $\varepsilon > 0$, we allow each node v to select from $\lceil (d_v+1)(\varepsilon+1) \rceil$ possible colors (from an entire set of $\lceil (\Delta+1)(\varepsilon+1) \rceil$ total colors). Then the following corollary, whose straightforward proof we omit, characterizes the convergence time[2]:

Corollary 4 *Algorithm 2 converges to a proper coloring in $\mathcal{O}(\frac{1}{\varepsilon} \log n)$ rounds with high probability, when each node v selects from $\lceil (d_v+1)(\varepsilon+1) \rceil$ possible colors, for any $\varepsilon > 0$.*

III. Desynchronization

In this section we present a randomized desynchronization algorithm for sensor networks with arbitrary topology. We adapt the ideas of Section II into a continuous analog for vertex coloring, where rather than selecting between discrete pre-determined time slots, sensor nodes choose intervals from a continuous range of possible start times. As in Algorithm 2, nodes behave as two-state machines. In the *search* state, nodes attempt to identify a suitable interval during which to fire. Once such an interval has been identified, nodes enter the *permanent* state, in which they behave as oscillators with frequency $\omega = \frac{1}{T}$ (we assume that the parameter T is common to all nodes in the network). In this state, nodes fire for the duration of their selected interval in every period of length T. To keep the protocol as lightweight as possible, nodes store minimal information while in the *search* state, instead relying on randomization to avoid conflicts and select good intervals.

[2]The proof follows exactly that of Theorem 3, except that each round a non-permanent node becomes permanent with probability at least $\frac{\varepsilon}{\varepsilon+1}$.

In contrast with vertex coloring where time proceeds in synchronous rounds, in this setting time proceeds continuously. While in the *search* state, nodes act during phases whose length is bounded by T, whereas in the *permanent* state nodes cycle in phases of length exactly T.

Consider a network on n sensor nodes. For a node v, let d_v denote its degree, and $\hat{d}_v$ denote the maximum degree among v and its 1-hop neighbors, where a neighbor of v refers to a node within communication distance of v. If node v *fires*, then all of its neighbors who are *listening* can detect that *someone* is transmitting, but cannot identify who it is. Practically, a sensor node can *fire* by broadcasting a carrier wave, while a node can *listen* by checking whether the RSSI value is above a noise threshold. This approach is the predominant indicator of link and channel quality [5], [15]. Crucially, we assume that nodes cannot both fire and listen simultaneously, since most radio transceivers cannot do both actions at the same time. Our algorithm never requires nodes to synchronize to a global clock (assuming no clock drift). For clarity, the algorithm will be described from the perspective of a particular node v, meaning all references to time (except for the common parameter T) are according to the frame of reference of v's internal clock.

Algorithm 3 (Desynchronization)
Initalization:

*Node v informs each of its neighbors of its degree, d_v. After receiving each neighbor's degree value, compute $\hat{d}_v = \max\{d_u |\ u$ is a neighbor of v, or $u = v\}$ and compute the interval length $b_v = \frac{T}{2(\hat{d}_v+1)}$. Initialize in the **search** state, and select time t to begin search. Set $\hat{t} \leftarrow t$.*

Main Loop:

```
1) Select value $\alpha \in [0, T]$ uniformly at random. Set $\hat{t}_\alpha \leftarrow \hat{t} + \alpha$.
2) Listen whether any neighbor fires at any point during the interval $[\hat{t}_\alpha, \hat{t}_\alpha + b_v]$.
   If so, set $\hat{t} \leftarrow \hat{t}_\alpha + b_v$ and return to step 1. Else, continue.
3) Listen whether any neighbor is firing at time $\hat{t}_\alpha + T$. If so, set $\hat{t} \leftarrow \hat{t}_\alpha + T$ and return to step 1. Else, continue.
4) Enter the permanent state, and fire for the duration of the interval $[\hat{t}_\alpha + kT, \hat{t}_\alpha + kT + b_v]$ for all integers $k \geq 1$.
```

In step 1, the node selects at random a trial interval, whose length b_v is specified to guarantee the convergence time of the algorithm. This trial interval must pass two tests in order to be accepted by the node: first, in step 2, no neighbor can fire at any time during the interval; then, in step 3, after another period has passed, no neighbor can be firing at the very beginning of the trial interval. Only then does the node accept the trial interval and enter the permanent state, in which the interval is claimed for all subsequent

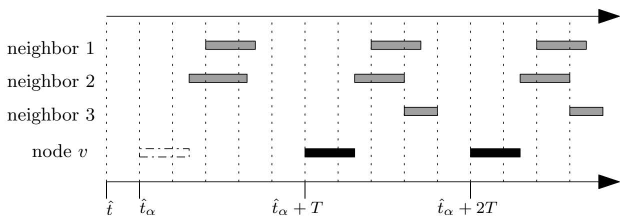

FIGURE 1. Node v proposes a trial interval at time $\hat{t}_\alpha$, which passes the checks of steps 2 and 3 to become permanent.

periods. The motivation behind step 3 is that two searching neighbors whose trial intervals overlap may both pass the first test in step 2, but we do not want both to claim their intervals permanently. Eschewing excess communication, step 3 resolves the contention by ensuring that only the node whose start time is *earliest* will pass the second test and thus claim the interval permanently. Observe that neighbors pick the *exact* same start time with probability 0. We note that a more complicated chain of overlapping trial intervals could all potentially pass the first test at the same time, but step 3 ensures that only non-overlapping trial intervals will be accepted.

Observe that each node maintains a single bit of state that determines whether it is in *search* or *permanent* mode, in addition to maintaining a legal output (ie. an interval), which is given by the start time and length of its trial or permanent interval. Moreover, in each iteration of the Main Loop, a node receives at most 2 bits of feedback (the listening checks of steps 2 and 3) on which it bases its actions. Therefore by our definition in Section I-B, we have a $(2, 1)$-bit local algorithm.

A. Convergence Properties

In this section we prove that Algorithm 3 converges after $\mathcal{O}(\Delta \log n)$ periods of length T with high probability.

Proposition 4 *If nodes u and v are neighbors that are both in the* permanent *state, then their selected intervals do not overlap.*

Proof: Suppose without loss of generality that node u entered the permanent state before node v (according to the global notion of time), and suppose for sake of contradiction that their selected intervals overlap. *Case 1: Node v's interval begins in the middle of node u's interval.* Then since u entered the permanent state before v, v would have heard u firing at the beginning of its trial interval (in steps 2 or 3), so it would not have accepted this interval. *Case 2: Node u's interval begins in the middle of node v's interval.* Then v would have heard u firing in the middle of its trial interval during step 2 of the algorithm, so it would not have accepted this interval. ■

Proposition 4 implies that it is sufficient to compute the time required for all n nodes to enter the *permanent* state.

Proposition 5 *Suppose a node v is at the beginning of the* **search** *state loop, in step* 1. *Then*

$$\mathbb{P}(v \textit{ enters } \textbf{permanent} \textit{ state within time } 2T) \geq \frac{1}{d_v + 1}$$

Proof: Consider a node v beginning a loop in the *search* state, at (internal) time $\hat{t}$. When node v selects the parameter $\alpha \in [0, T]$ for the start of its trial interval, we call α *acceptable* if the chosen trial interval (beginning at $\hat{t}_\alpha$) does not overlap with any of v's neighbors' intervals. We remark that for any neighbor u of v, $\hat{d_u} \geq d_v$, meaning that the interval length $b_u \leq T/2(d_v + 1)$. Moreover, v's own interval length $b_v \leq T/2(d_v + 1)$ since $\hat{d_v} \geq d_v$. Thus, a neighbor u causes an interval (or two disjoint intervals) from $[0, T]$ of total length $(b_v + b_u) \leq T/(d_v + 1)$ to not be acceptable.

In the worst case, all d_v of v's neighbors will have already selected their permanent intervals, blocking off a subset of $[0, T]$ of measure $\leq d_v T/(d_v + 1)$ from being acceptable choices of α. Therefore,

$$\mathbb{P}(\text{selected } \alpha \text{ is acceptable}) \geq \frac{1}{d_v + 1}$$

As the check in step 3 of the algorithm occurs by time $\hat{t} + 2T$ (and a node enters the *permanent* state if that check is passed), the result holds. Note that this analysis holds even taking into account any neighbors of v that become permanent between the checks of steps 2 and 3. ■

As a consequence of this result, since $b_v \leq \frac{T}{4}$, a node that is in the *search* state (not just at the beginning of the loop) enters the permanent state within time $9T/4$ with probability $\geq \frac{1}{d_v+1}$. This directly implies the following proposition:

Proposition 6 *Suppose at (global) time t that Y_t of the nodes are permanent. Then $\mathbb{E}[Y_{t+9T/4} | Y_t] \geq Y_t + \frac{n - Y_t}{\Delta+1}$.*

Finally, we establish the number of periods (of length T) required for the algorithm to converge.

Theorem 5 *The Desynchronization Algorithm converges in $\mathcal{O}(\Delta \log n)$ periods with high probability.*

Proof: Defining a round as an interval of length $9T/4$, this theorem follows directly from Proposition 6, as well as Proposition 2 and Theorem 3, where parameters b and c are as in Theorem 3. ■

B. Algorithm Speed-Up

The interval length $b_v = T/2(\hat{d_v} + 1)$ was chosen just large enough so as to ensure convergence in a reasonable amount of time, and to give a continuous analog to $\Delta + 1$ coloring of graphs. Here we consider how the convergence time may be improved if nodes claim shorter interval lengths.

Consider a parameter $\varepsilon > 0$, common to all nodes, such that a node v selects its interval length according to $b_v^\varepsilon = T/2(\hat{d_v} + 1)(1 + \varepsilon)$. Following the proof of Proposition 5, when a node v is selecting the parameter α in the search loop,

$$\mathbb{P}(\text{selected } \alpha \text{ is acceptable}) \geq \frac{\varepsilon}{\varepsilon + 1}$$

By the proof steps of Theorem 3, this establishes the following counterpart to Theorem 5:

Theorem 6 *The Desynchronization Algorithm, where nodes select interval lengths according to $b_v^\varepsilon = T/2(\hat{d_v}+1)(1+\varepsilon)$ for $\varepsilon > 0$, converges in $\mathcal{O}(\frac{1}{\varepsilon}\log n)$ periods with high probability.*

C. *Self-Maintenance Subroutine*

In this section, we discuss a simple way to manage the desynchronized solution in the context of link volatility and changes in network topology, a critical consideration when modeling wireless sensor networks as wireless connections are volatile over time [2].

Changes in network topology may create several problems. If a new communication link is suddenly formed between two permanent nodes, an overlap between the two claimed intervals may be created; even if the new link appears a hop away from a permanent node v but causes v's one-hop max degree $\hat{d_v}$ to increase, there may no longer be enough bandwidth (ie. intervals of the period T) to accommodate the nodes still in search mode. If instead a link is destroyed, we want nodes to take advantage of the newly available bandwidth to claim larger intervals.

A natural solution is for nodes to store the degree values of all of their neighbors, requiring total storage of $\mathcal{O}(d_v \log \hat{d_v})$ at each node[3]. If a node v detects that its degree has changed, it reports its new degree to each of its neighbors. Each such neighbor u in turn recomputes the value $\hat{d_u}$; if this value has changed, u recomputes its new interval length b_u, reverts to the *search* state (if it had already been in the *permanent* state), selects a time $\hat{t}$ to begin the search, and returns to step 1 of Algorithm 3. Nodes outside v's neighborhood are unaffected. Following the convergence results of Section III-A, any local network change causing m nodes to restart takes $\mathcal{O}(\Delta \log m)$ rounds to repair with high probability.

IV. Simulation

To confirm the correctness of the desynchronization algorithm and examine its performance under the stress of network effects, we simulated the algorithm in the TOSSIM environment. As TOSSIM runs the same code as sensor motes, we show that the algorithm not only has strong theoretical bounds, but is also practically implementable.

A. *Setup*

We implemented the algorithm on a network topology derived from the communication graph of a 54-node deployment of sensors in the Intel Berkeley Research Lab [16]. For the purposes of this simulation, we included an edge between two nodes in our topology if and only if the aggregate connectivity data in [16] showed $> 40\%$ probability of successful transmission in *both* directions, resulting in a 54-node undirected graph with a wide variety of node degrees d_v and maximum one-hop neighborhood degrees $\hat{d_v}$. Note that three nodes remain disconnected due to weak connectivity data.

[3]While it requires less memory to store only the largest neighborhood degree $\hat{d_v}$, in a network with volatile links, only knowing the current value $\hat{d_v}$ is not sufficient to knowing *future* values of $\hat{d_v}$.

Degree Distribution

d_v	0	1	2	3	4	5	6	7	8	9
No. of Nodes	3	1	6	8	13	12	7	2	1	1

Max 1-hop Neighborhood Degree Distribution

$\hat{d_v}$	0	1	2	3	4	5	6	7	8	9
No. of Nodes	3	0	0	0	9	18	5	0	9	10

We implemented the *firing* of a node by having it broadcast noise for the appropriate interval. For concreteness, we specified the transmission power to be 0dBm and each link to have a gain of -54dBm. *Listening* was implemented by having nodes check whether the RSSI values were above a threshold relative to the noise floor. There was therefore no need to consider issues of packet loss, as no message reception is required. The most important network effects we had to consider are the false alarm rate – the probability that a node hears a signal above the threshold even though no neighbor is actually broadcasting – and the miss probability – the chance that no signal above the threshold is recorded even though a neighbor is indeed broadcasting. A high false alarm rate would have the effect of scaling up the convergence time of the algorithm, since a node in *search* mode that hears a signal above the threshold during its trial interval must reset and start again, irrespective of whether the signal is due to a neighbor's firing or a spike in ambient noise. The higher the false alarm rate, the more times the node must reset. We measured this effect by simulating the algorithm over a range of thresholds, from -84dBm (many false positives) to -72dBm (virtually no false positives)[4].

On the other hand, a large miss probability is problematic since it causes a node to fail to hear a neighbor firing during the listening checks, which can lead the desynchronization algorithm to converge to a state where neighbors' intervals overlap. However, while signal strength drop-offs can and do occur in practice, the desynchronization algorithm is extremely robust to such signal strength fluctuations, since it will only register a false negative if the drop-off is steep (below the noise threshold) and lasts for a node's entire listening period. In reasonable physical settings, this probability is negligible and can be further reduced by scaling up the period and interval lengths, so it is ignored for the purposes of this simulation [7].

Finally, we mention that the algorithm implemented in this simulation is a slightly sped-up version of Algorithm 3 – in step 2, after hearing a neighbor, instead of waiting for the end of the trial interval to reset, we have nodes immediately return back to step 1. It is trivial to show that the convergence results of Section III still hold for this version.

B. *Results and Discussion*

Figure 2 shows the average simulation time, in periods, that the desynchronization algorithm takes before all nodes in the sensor deployment reach the permanent state, with the sensors' noise threshold varying between -84dBm (frequent

[4]Noise was modeled in TOSSIM with the `meyer-light` noise trace [11].

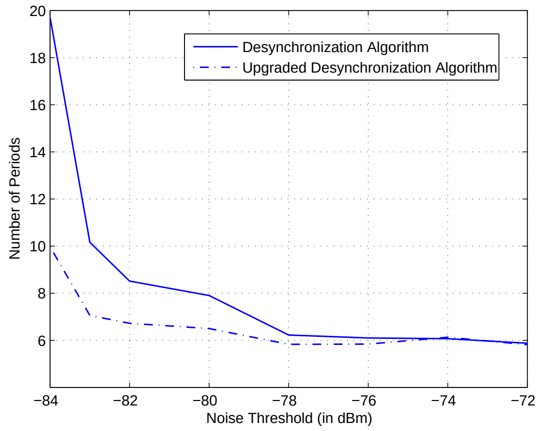

FIGURE 2. Average time, in periods, before all nodes reach the permanent state. Each data point an average over 200 simulation runs. The period T was set to be $5.04\mathtt{s}$, with interval lengths ranging from $252\mathtt{ms}$ (for nodes with $\hat{d_v} = 9$), to $504\mathtt{ms}$ (for nodes with $\hat{d_v} = 4$).

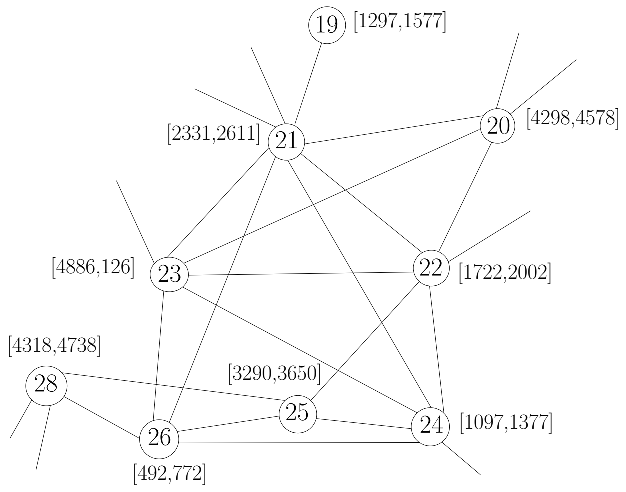

FIGURE 3. A local view of the 54-node communication network after convergence in a simulation run. Numbers inside the circles indicate node ID, while edges between pairs of nodes indicate that the pair is within communication distance. The numbers $[x, y]$ denote the node's claimed interval: the period has phase $T = 5040\mathtt{ms}$, so x and y denote (in terms of phase) the start and end times respectively (in $\mathtt{ms}$) of the repeating interval. All phases are according to a global clock, so a phase of $0\mathtt{ms}$ indicates the beginning of the global period of length T. Node 23's interval begins at the end of a global period and ends just after the start of the next global period.

false positives) and $-72\mathtt{dBm}$ (virtually no false positives). Observe that, as expected, the average time to convergence is reduced as the noise threshold is raised and the algorithm becomes less sensitive to spikes in the noise power.

We also simulated an upgraded version of the desynchronization algorithm to improve convergence time in noisy environments: in the first listening check of Step 2, instead of having nodes reset after *any* RSSI reading above the noise threshold during the trial interval, we required two distinct high readings. As seen in Figure 2, this greatly improves robustness to noise by avoiding spurious false positives caused by isolated spikes in ambient noise, while still correctly detecting the broadcasts of neighbors. Note that convergence time by this technique is significantly reduced when the noise threshold is low. This technique can be generalized by requiring k high readings for any $k > 1$, though it increases the number of bits used in the algorithm's (c,d)-bit characterization.

Each simulation run produced a correct solution at convergence, in that each node's permanent interval did not overlap with its neighbors' intervals. The only exception occurred in cases where an overlap of length $\leq 1\mathtt{ms}$ between neighbors' intervals could exist. This was caused by our implementation in TOSSIM of Step 3 of the algorithm: instead of having nodes send a continuous broadcast signal, we had nodes send rapid-fire packets for the duration of their chosen interval (TOSSIM is better geared towards packet-level broadcast, a limitation in simulating the underlying hardware). As a result, to avoid false negatives, we extended the instantaneous listening check of Step 3 (which might have the misfortune of listening *in between* packets) to a $1\mathtt{ms}$-long check. The tradeoff is that the tail $1\mathtt{ms}$ end of a node's interval might overlap with the leading $1\mathtt{ms}$ end of another node's interval. However, this problem can be mitigated by scaling up the period and interval lengths.

V. Conclusion

We show that even when placing severe limits on allowable computation and communication on nodes in a network, a global desynchronization or proper vertex coloring solution can still be reached quickly. Our algorithms succeed in these problems because a globally feasible desynchronization can be confirmed by checking local feasibility at every node. We expect that variants of our algorithms' randomized approach that minimizes feedback and state may find success in other important network problems such as the design of efficient communication protocols – where nodes within two hops of one another should not have overlapping intervals – as well as graph problems such as distributed Minimum Dominating Set or Maximum Independent Set where confirming local feasibility at every node is sufficient.

Acknowledgment: Arik Motskin, Primoz Skraba and Leonidas Guibas gratefully acknowledge the support of Army AHPCRC grant W911NF-07-2-0027-1 and NSF grants CNS-0626151, CCF-0634803. Tim Roughgarden is supported in part by NSF CAREER Award CCF-0448664, an AFOSR MURI grant, an ONR Young Investigator Award, and an Alfred P. Sloan Fellowship.

References

[1] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*. John Wiley Interscience, March 2004.
[2] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proc. 4th int. symp. on Information processing in sensor networks (IPSN)*, page 11, 2005.
[3] J. Degesys, I. Rose, A. Patel, and R. Nagpal. Desync: self-organizing desynchronization and tdma on wireless sensor networks. In *Proc. 6th int. conf. on Information processing in sensor networks (IPSN)*, pages 11–20, 2007.
[4] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187:199, 1998.
[5] K. Jamieson, B. Hull, A. Miu, and H. Balakrishnan. Understanding the real-world performance of carrier sense. In *ACM E-WIND*, pages 52–57, 2005.

[6] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[7] S. M. Kay. *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*. Prentice Hall PTR, January 1998.

[8] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proc. 23rd annual ACM symp. on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

[9] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In *Proc. 24th annual ACM symp. on Principles of Distributed Computing (PODC)*, pages 60–68, 2005.

[10] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In *Proc. 25th annual ACM symp. on Principles of Distributed Computing (PODC)*, pages 7–15, 2006.

[11] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *Proc. 6th int. conf. on Information processing in sensor networks (IPSN)*, pages 21–30, 2007.

[12] N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[13] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1055, 1986.

[14] A. Patel, J. Degesys, and R. Nagpal. Desynchronization: The theory of self-organizing algorithms for round-robin scheduling. In *Proc. 1st int. conf. on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 87–96, 2007.

[15] K. Srinivasan and P. Levis. Rssi is under-appreciated. In *Proc. 3rd Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[16] `http://db.csail.mit.edu:80/labdata/labdata.html`. Intel lab data. accessed 07/2008.

[17] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Elsevier/Morgan-Kaufmann, 2004.

APPENDIX A
PROOF OF PROPOSITION 1

We prove this proposition by establishing a lower bound on the probability that a good node *remains* good from round to round. The key is to show that this probability converges to 1 as the number of good nodes approaches n. Fix a round t. Suppose that good node v, currently colored c, has Δ bad neighbors.

$$\begin{aligned}
&\mathbb{P}(\text{node } v \text{ remains good next round})\\
&\qquad = \mathbb{P}(\text{none of } v\text{'s neighbors switch to } c)\\
&\qquad \geq \left[1-\left(\frac{1}{k\Delta}\right)\right]^{\Delta} \qquad (1)\\
&\qquad \geq \left[e^{-1}\left(1-\frac{1}{k\Delta}\right)\right]^{\frac{1}{k}} = p^*
\end{aligned}$$

Clearly, nodes that have fewer than Δ bad neighbors also remain good with probability at least p^*. By choosing k high enough, we can make p^* as close to 1 as we wish, say $p^* > 0.78$ if $k \geq 5$. We now lower bound the probability that a node u that is currently *bad* will become good next round. As u is bad, it will select a new color next round:

$$\begin{aligned}
&\mathbb{P}(\text{node } u \text{ becomes good next round})\\
&\geq \mathbb{P}(u \text{ switches to a color that is currently unclaimed, and}\\
&\qquad \text{which no neighboring node will also claim next round})\\
&\geq \left(\frac{k\Delta-\Delta}{k\Delta}\right)\mathbb{P}(\text{no neighbor of } u \text{ switches to } u\text{'s color})\\
&\geq \left(\frac{k-1}{k}\right)p^* = q^*
\end{aligned} \qquad (2)$$

By setting $k \geq 5$, we get that $q^* > 0.6$.

Lemma 1 *If* $X_t \leq \frac{2n}{3}$, *then* $\mathbb{E}[X_{t+1}|X_t] \geq X_t + 0.1(n-X_t)$.

Proof: By Equations 1 and 2, $\mathbb{E}[X_{t+1}|X_t] > 0.78X_t + 0.6(n-X_t) > (0.75X_t + 0.5(n-X_t)) + 0.1(n-X_t)$. Since $X_t \leq \frac{2n}{3}$, we have that $X_t \leq 2(n-X_t)$. Therefore, $\mathbb{E}[X_{t+1}|X_t] > (0.75X_t + 0.25X_t) + 0.1(n-X_t) = X_t + 0.1(n-X_t)$. ■

Next we will prove a similar result for $\frac{n(\Delta-1)}{\Delta} \geq X_t > \frac{2n}{3}$. Suppose that there are currently $n\frac{\Delta-1}{\Delta} \geq X_t = n\frac{w}{w+1}$ good nodes, for $w > 2$. Since each bad node can be a neighbor to at most Δ good nodes (in fact, to at most $\Delta-1$), there are at most $n\frac{\Delta}{w+1}$ bad neighbors, distributed among $n\frac{w}{w+1}$ good nodes. Therefore, the average number of bad neighbors each good node has is at most $\frac{\Delta}{w}$. Since we wish to lower bound in expectation the number of good nodes that will remain good next round, we note that by the convexity of Equation 1 (with respect to number of bad neighbors), in the worst case all good nodes have the same number of bad neighbors as each other. That is, in the worst case, each good node has $\lceil\frac{\Delta}{w}\rceil$ bad neighbors:

$$\begin{aligned}
&\mathbb{P}(\text{good node remains good next round})\\
&\quad = \mathbb{P}(\text{none of } v\text{'s } \lceil\Delta/w\rceil \text{ bad neighbors switch to } v\text{'s color})\\
&\quad \geq \left[1-\left(\frac{1}{k\Delta}\right)\right]^{\lceil\frac{\Delta}{w}\rceil}\\
&\quad \geq (p^*)^{\frac{\lceil\frac{\Delta}{w}\rceil}{\Delta}} \geq (p^*)^{\frac{2}{w}}
\end{aligned} \qquad (3)$$

since we know that $\frac{2}{w} \geq \frac{\lceil\frac{\Delta}{w}\rceil}{\Delta}$ when $\Delta > w$, as it is here.

Lemma 2 *If* $n\frac{\Delta-1}{\Delta} \geq X_t > \frac{2n}{3}$, *then* $\mathbb{E}[X_{t+1}|X_t] \geq X_t + 0.1(n-X_t)$.

Proof: Suppose $X_t = n\frac{w}{w+1}$ where $(\Delta-1) \geq w > 2$. Then by Equations 2 and 3,

$$\begin{aligned}
&\mathbb{E}[X_{t+1}|X_t]\\
&\quad \geq (0.78)^{\frac{2}{w}}\left(\frac{w}{w+1}\right)n + (0.6)\left(\frac{1}{w+1}\right)n\\
&\quad \geq \left(1+\frac{2\ln .78}{w}\right)\left(\frac{w}{w+1}\right)n + (0.6)\left(\frac{1}{w+1}\right)n\\
&\quad = \left(\frac{w}{w+1}\right)n + \frac{(2\ln .78)+0.6}{w+1}n\\
&\quad > X_t + 0.1(n-X_t)
\end{aligned}$$

■

Finally, we prove a similar result for $n \geq X_t > \frac{n(\Delta-1)}{\Delta}$.

Lemma 3 *If* $n \geq X_t > n\frac{\Delta-1}{\Delta}$, *then* $\mathbb{E}[X_{t+1}|X_t] \geq X_t + 0.1(n-X_t)$.

Proof: We recall that a bad node can be a neighbor to at most $\Delta-1$ good nodes, and that in the worst case each good node has an equal number of bad neighbors. Therefore if $X_t > n\frac{\Delta-1}{\Delta}$, then in the worst case $(\Delta-1)(n-X_t)$ good nodes have a single bad neighbor, while the remaining good nodes have no bad neighbors.

$$\begin{aligned}
&\mathbb{E}[X_{t+1}|X_t] \\
&\geq [(\Delta-1)(n-X_t)]\left(1-\frac{1}{\Delta k}\right)+ \\
&\qquad [X_t-(\Delta-1)(n-X_t)]+0.6(n-X_t) \\
&=(\Delta-1)(n-X_t)\left(-\frac{1}{\Delta k}\right)+X_t+0.6(n-X_t) \\
&=[X_t+0.1(n-X_t)]+(n-X_t)\left(\frac{1}{\Delta k}-\frac{1}{k}+0.5\right) \\
&\geq X_t+0.1(n-X_t) \qquad \text{(as long as } k\geq 2)
\end{aligned}$$

■

Lemmas 1, 2 and 3 imply the desired proposition.

APPENDIX B PROOF OF THEOREM 2

In this paper, we prove the theorem for $\Delta = 1$ and give intuition as to why it extends to larger Δ.

Consider two random processes on a set of m nodes, $\frac{m}{k}$ of which are colored red for some fixed $k > 2$, with the rest colored blue. In process 1, m nodes are sampled uniformly at random with replacement, with two successively sampled nodes defining an edge. In process 2, m nodes are sampled uniformly at random *without* replacement, with two successively sampled nodes defining an edge. Note that process 2 selects a matching on the node set uniformly at random. Observe that in expectation these processes place the same number of edges between two red nodes. Let $P_1(s)$ and $P_2(s)$ denote the probabilities that s edges between two red nodes are selected by processes 1 and 2 respectively.

Lemma 4 *For large enough m, we have that for any $s < \frac{m}{k^4}$, $P_2(s) < P_1(s)$.*

We omit the proof of this lemma for brevity. Returning to coloring algorithms, assume for the remainder of the proof that each node selects from $k\Delta$ colors for a constant $k \geq 2$.

Lemma 5 *Consider a node set V such that $|V| = m = \Omega(\log n)$. Suppose the input graph $G = (V,E)$ to these nodes is a random matching, and each node v picks a color according to some color selection function. Then there exists a monochromatic submatching $G' = (V',E')$, where $G' \subset G$, such that $\mathbb{E}|V'| \geq m/\alpha$, for a constant $\alpha > 1$. Moreover, $\mathbb{P}(|V'| < \frac{4m}{\alpha k^2}) = \mathcal{O}(\frac{1}{n})$.*

Proof: By the pigeonhole principle, there is some color c selected by at least m/k nodes. Any such node is matched to another node colored c with probability $\geq (\frac{m}{k}-1)/m > 1/2k$. Set $\alpha = 2k^2$. By linearity of expectation there is a c-colored submatching $G' = (V',E')$, where $G' \subset G$, with $\mathbb{E}|V'| \geq m/\alpha$. Observe that $|V'|$ is the number of c-colored nodes matched to another c-colored node by Process 1 of Lemma 4. Therefore, using Chernoff bounds on Process 2 (where all c-colored node's probabilities of being connected to another c-colored node are independent), and combining with Lemma 4, we have that when $k > 2$, $\mathbb{P}(|V'| < \frac{4m}{\alpha k^2}) < \exp(-m\delta^2/2\alpha) = \mathcal{O}(\frac{1}{n})$ since $m = \Omega(\log n)$ and $\delta = 1-\frac{4}{k^2}$. For $k = 2$, note that the probability of having an s-sized monochromatic submatching is only higher than for $k = 3$, so that we also have $\mathbb{P}(|V'| < \frac{m}{\alpha d}) = \mathcal{O}(\frac{1}{n})$ for some constant $d > 1$. ■

Proof of Theorem: Consider a set V_0 of n nodes. For each $v \in V_0$, fix its color selection function. Suppose the input graph $G_0 = (V_0,E_0)$ is a random matching. Consider what happens in the first round: by Lemma 5, there is a monochromatic submatching $G_1 = (V_1,E_1)$ with $\mathbb{E}|V_1| \geq n/\alpha$, and $\mathbb{P}(|V_1| < n/d\alpha) = \mathcal{O}(\frac{1}{n})$ (since $n = \Omega(\log n)$), for some constant $d > 1$ (which depends on k as in Lemma 5). Conditioning on this large deviation not occuring (ie. assuming $|V_1| \geq n/d\alpha$), we proceed to the next round and consider only the nodes in subgraph G_1.

Notice that since each node $v \in V_1$ knows *only* that it had a conflict in the previous round (not the identity of the node to whom it was matched), we can assume for the next round that by the principle of deferred decision, G_1 is itself a random matching on the node set V_1. Therefore we can recursively apply the above argument to subgraph G_1. In particular, for $i > 1$, given a random matching on the node set V_{i-1} where $|V_{i-1}| \geq n/(d\alpha)^{i-1}$, by Lemma 5, as long as $|V_{i-1}| = \Omega(\log n)$, after round i there is a monochromatic submatching $G_i = (V_i,E_i)$ with $\mathbb{E}|V_i| \geq n/\alpha(d\alpha)^{i-1}$, and $\mathbb{P}(|V_i| < n/(d\alpha)^i) = \mathcal{O}(\frac{1}{n})$. Conditioning on this large deviation not occuring (ie. assuming $|V_i| \geq n/(d\alpha)^i$), proceed to the next round, treating subgraph G_i as a random matching on the nodes V_i.

Observe that as long as there is still a nonempty monochromatic submatching, the coloring algorithm has not yet converged. We run the above procedure for j total rounds; we can choose $j = \Omega(\log n)$ such that after j rounds (conditioning after each round on the large deviation not occurring), we have $|V_j| \geq n/(d\alpha)^j = \Omega(\log n)$. By the union bound, $\mathbb{P}$(large deviation never occurs in the j rounds) $\geq 1 - j\mathcal{O}(1/n) = 1 - \mathcal{O}(\log n/n)$.

Therefore, with high probability after $\Omega(\log n)$ rounds there is still a monochromatic submatching (of size $\Omega(\log n)$), so the coloring algorithm has not yet converged.

■

If $\Delta > 1$, then we consider the input graph to be a set of Δ random matchings $M_1, M_2, ..., M_\Delta$ on n nodes (allowing parallel edges). After the first round, we consider the largest monochromatic submatching involving *only* edges from the first matching M_1, leaving a subgraph $G_1 = (V_1,E_1)$ with $|V_1| \geq n/d\Delta\alpha$ with high probability. Thereafter, the argument reduces to that for $\Delta = 1$, except that instead, in each round the cardinality of the monochromatic submatching reduces by a $1/(k\Delta)^2$ factor, since each node can choose from a larger set of $k\Delta$ colors. This results in a lower bound of $\Omega(\log n/\log\Delta)$ rounds to convergence with high probability.