Deploying Wireless Networks with Beeps

Alejandro Cornejo[1] and Fabian Kuhn[2]

[1] Massachusetts Institute of Technology
[2] University of Lugano

Abstract. We present the *discrete beeping* communication model, which assumes nodes have minimal knowledge about their environment and severely limited communication capabilities. Specifically, nodes have no information regarding the local or global structure of the network, do not have access to synchronized clocks and are woken up by an adversary. Moreover, instead on communicating through messages they rely solely on carrier sensing to exchange information. This model is interesting from a practical point of view, because it is possible to implement it (or emulate it) even in extremely restricted radio network environments. From a theory point of view, it shows that complex problems (such as vertex coloring) can be solved efficiently even without strong assumptions on properties of the communication model.

We study the problem of *interval coloring*, a variant of vertex coloring specially suited for the studied beeping model. Given a set of resources, the goal of interval coloring is to assign every node a large contiguous fraction of the resources, such that neighboring nodes have disjoint resources. A k-interval coloring is one where every node gets at least a $1/k$ fraction of the resources.

To highlight the importance of the discreteness of the model, we contrast it against a continuous variant described in [17]. We present an $\mathcal{O}(1)$ time algorithm that with probability 1 produces a $\mathcal{O}(\Delta)$-interval coloring. This improves an $\mathcal{O}(\log n)$ time algorithm with the same guarantees presented in [17], and accentuates the unrealistic assumptions of the continuous model. Under the more realistic discrete model, we present a Las Vegas algorithm that solves $\mathcal{O}(\Delta)$-interval coloring in $\mathcal{O}(\log n)$ time with high probability and describe how to adapt the algorithm for dynamic networks where nodes may join or leave. For constant degree graphs we prove a lower bound of $\Omega(\log n)$ on the time required to solve interval coloring for this model against randomized algorithms. This lower bound implies that our algorithm is asymptotically optimal for constant degree graphs.

1 Introduction

Communication models face the unavoidable tension between their practicality and their potential for designing interesting yet provably correct algorithms. With enough assumptions concerning the knowledge of the deployment environment and the communication capabilities of the devices used, it is not difficult to design efficient and elegant distributed algorithms. However, it is often difficult (if not impossible) to translate these algorithms to the real world. On the other hand, communication models which are cluttered with physical details encumber designing algorithms, and makes it significantly more complicated to prove correctness or efficiency.

This motivates the study of models such as the *discrete beeping* model considered in the present paper. This model makes little demands on the communication devices, nodes need only be able to do *carrier sensing* and differentiate between silence and the presence of a jamming signal. Carrier-sensing can typically be done much more reliably and requires significantly less energy and other resources than transmitting and receiving actual messages , see e.g. [7]. Besides requiring reliable carrier sensing, we make almost no assumptions. In particular, we do not assume knowledge of the local or global structure of the network or synchronized clocks. Further, we assume that an adversary controls when processors are woken up.

We show that even such a "weak" model allows for interesting algorithms for non-trivial tasks. In particular we focus on the problem of *interval coloring*, a variant of classic vertex coloring. Given a set of resources, the goal of interval coloring is to assign each node a large contiguous fraction of the resources such that neighboring nodes have disjoint resources. A k-interval coloring is one where every node gets at least a $1/k$ fraction of the resources. Similar to vertex coloring, interval coloring is a useful building block to establish a reliable Medium Access Layer (MAC), as it can be used to e.g. compute time or frequency division multiple access (TDMA or FDMA) schedules that avoid conflict between potentially interfering nodes. In some sense, interval coloring is even better suited for these tasks than standard graph coloring. While in a standard coloring, every node gets assigned a single color (a single slot or frequency), in an interval coloring, we can assign larger intervals to certain nodes (e.g. to nodes with a small degrees). An interval then corresponds to multiple consecutive colors in a standard coloring context.

Moreover, by relying exclusively on carrier sensing, the beeping model becomes specially well-suited for coordination tasks in wireless networks for various reasons, for example: ◇ Most prior work [1, 3, 4, 9, 11, 14, 18, 23, 25] on coloring assumes some existing infrastructure to reliably exchange messages. If used as a building block to e.g. compute a TDMA schedule, these algorithms suffer from a chicken-and-egg problem; such colorings cannot be computed without a reliable MAC layer, however to achieve a reliable MAC layer one first needs to compute a coloring. A coloring algorithm for the beeping model would not suffer from this problem, since the model makes almost no assumptions on the communication infrastructure. ◇ The presence of a signal can be reliably detected by carrier sensing at lower receiving power than would be required to correctly decode a message. Hence, carrier sensing can be used to communicate more energy efficiently and over larger distances than when transmitting regular messages. For example, by default the NS2 [26] simulator uses a carrier sensing range that is more than twice as large as the transmission range. Therefore, the beeping model (carrier sensing) can directly be used to compute a 2-hop interval coloring of the communication graph (for regular transmission), a necessity when using the coloring for a MAC layer that avoids hidden terminal collisions. ◇ Although IEEE 802.11 and Bluetooth share the same frequency spectrum, they use incompatible modulation and encoding schemes. However since carrier sensing only detects the presence of a signal, it is potentially possible for a IEEE 802.11 radio to detect the presence of a Bluetooth jamming signal and vice versa. Therefore, algorithms for the beeping model could be used to allow these two seemingly incompatible devices to agree on a non-conflict transmission schedule thereby allowing them to coexist in a non-destructive fashion.

Contributions. We assume that there is a common globally known period length T. This is a parameter of the algorithms which captures the number of resources to be shared (e.g. the number of available frequencies in FDMA). The paper has three main contributions.

First, we significantly improve a result from [17] for a continuous variant of the beeping model. The authors of [17] describe an algorithm that solves $\mathcal{O}(\Delta)$-interval coloring in $\mathcal{O}(\log n)$ periods is described in [17]. Specifically they assign every node v a $\Omega(1/d^{\max}(v))$ fraction of the resources, where $d^{\max}(v)$ is the largest degree in the 1-neighborhood of v. We describe a simpler algorithm that improves the results of [17] by computing an interval coloring with the same properties in a constant number of periods. Our result highlights the unrealistic assumptions behind the continuous model.

Second, we give a discrete variant of the beeping model and describe a Las Vegas randomized interval coloring algorithm for the discrete model. The algorithm computes a $\mathcal{O}(\Delta)$-interval coloring in $\mathcal{O}(\log n)$ periods with probability $1 - \frac{1}{n}$. Furthermore, we describe how to adapt the algorithm to work in a dynamic graph setting where nodes can join and leave arbitrarily. A new node obtains an interval at most $\mathcal{O}(\log n)$ periods after joining the network, and a node only recomputes its interval if the size of its neighborhood becomes drastically smaller. The correctness proof of both the static and dynamic versions of the algorithm rely on a balls and bins analysis.

Finally, for a local broadcast model with constant size messages, we prove a lower bound of $\Omega(\log n)$ time against randomized algorithms that solve $\mathcal{O}(\Delta)$-vertex coloring (or $\mathcal{O}(\Delta)$-interval coloring). For the discrete beeping model this implies a lower bound of $\Omega(\log n)$ periods for constant-degree graphs and $\Omega(\log n/\Delta)$ for general graphs. Moreover, if we restrict the number of beeps per period to $\mathcal{O}(1)$ it yields a lower bound of $\Omega(\log n/\log \Delta)$ for general graphs.

Related Work. Using carrier sensing for distributed computation is not novel. Scheideler et al. [21] considered a model where in addition to sending and receiving messages, nodes can perform physical carrier sensing, and described how to approximate the minimum dominating set problem under this model. Flury and Wattenhofer [7] demonstrate how to use carrier sensing as an elegant and efficient way for coordination in practice.

Our beeping model is a discretized variant of the desynchronization model first introduced by [6]. Degesys et al. [6] considered only complete graphs, and proved the eventual convergence of a biologically inspired algorithm DESYNC to a *'desynchronized state'* and conjectured a running time of $\mathcal{O}(n^2)$. Degesys and Nagpal [5] experimentally studied the performance of DESYNC in multi-hop topologies. They proved that a desynchronized state exists for 2-colorable graphs and Hamiltonian graphs, and posed the open problem of proving that a desynchronized state exists for all graphs. Later Motskin et al. [17] studied interval coloring under the same desynchronization model. In addition to assuming the continuous variant of the model, [17] assumes that nodes have knowledge of their own degree and that they are able to exchange this information to compute the maximum neighbor degree over their 1-hop neighbors. It is not clear how nodes should obtain the maximum degree among their neighbors without reliably transmitting messages. Further, as we show in Section 4, their assumptions are too strong and allow for constant time solutions. This motivates studying the strictly weaker discrete beeping model.

Coloring the nodes of a graph is one of the most fundamental combinatorial optimization problems in computer science and has therefore been widely studied, also in a distributed context. The work on distributed coloring algorithms started with the seminal work of Linial [14] and includes a large number of papers (see e.g. [1, 3, 4, 9, 11, 13, 18, 23, 25]). The best bounds are known for randomized algorithm and they are $\mathcal{O}(\sqrt{\log n} + \log \Delta)$ for $(\Delta + 1)$-colorings (i.e., the number of colors needed by the sequential greedy algorithm) and $\mathcal{O}(\sqrt{\log n})$ for $\mathcal{O}(\Delta)$-colorings [11, 25]. Interesting in the context of TDMA schemes for wireless networks might be [12] where it is shown how to compute a coloring where each node with degree d obtains an $\Omega(1/d)$-fraction of the colors in a single communication round (i.e., nodes just need to learn the identifiers of all neighbors). Coloring in unstructured radio networks (with collisions) was considered by [16], where a randomized algorithm to compute $\mathcal{O}(\Delta)$-colorings in $\mathcal{O}(\Delta \log n)$ rounds is described (later improved in [24] to $\mathcal{O}(\Delta + \log \Delta \log n)$ rounds). In addition to the theoretical work on distributed coloring, there are many papers that describe some variant of coloring in order to compute TDMA schedules or similar MAC schemes (see e.g. [2, 8, 10, 15, 19, 20, 27]).

2 Model and Definitions

We consider a wireless network model that is as primitive as possible. In contrast to standard communication models, nodes cannot exchange messages reliably (message passing) or unreliably (unstructured radio networks), instead nodes rely entirely on carrier sensing. At any particular time, a node can be in beeping or listening mode. When a node is listening, it can only distinguish between silence or the presence of one or more beeps. This model is weaker than collision detection since nodes cannot distinguish between a single beep and a collision of two or more beeps. Moreover, a beep conveys less information than a bit, and although one could conceive coding schemes to encode bit messages using beeps, this would require additional overhead and be susceptible to collisions, thus we focus on different techniques.

We assume that nodes wake up asynchronously and the wake-up pattern is determined by an adversary. Upon waking up, a node does not know anything about the structure of the communication network, not even an estimate of its size. Similarly, nodes do not know their neighbors in the communication network or have an estimate of the size of this set. Furthermore, nodes do not have unique identifiers and the structure of the communication network is not restricted in any way (e.g. by requiring it to be a unit disk graph, a bounded independence graph, or any other special type of graph considered in the wireless networks literature [22]). Every node has access to a local clock, where the local clock of every node advances at the same rate and has no drift, however we do not assume clocks to be synchronized.

The communication network is modeled as an undirected graph $G = (V, E)$, $|V| = n$, where the set V of nodes of G represents the set of wireless devices. There is an edge $\{u, v\} \in E$ if and only if u can listen to a beep emitted by v and viceversa. For a node $u \in V$, let $N(u) := \big\{v \in V \big| \{u, v\} \in E\big\}$ be the set of neighbors of u, and let $d(u) = |N(u)|$ be its degree. We denote by $\Delta = \max_{v \in V} d(v)$ the maximum degree of the graph. A *phase* refers to a time point (in the continuous model) or a time slot (in the discrete model) measured relative to the beginning of the last period. We will use phases to capture the time at which different beeps are heard with respect to the local clock of each node. Given a set S of phases, we define $S[a, b]$ to be the subset of phases in the range $[a, b]$ in S. To correctly account for ranges that cross the period boundary, we give a formal definition. Let τ be the period length (in the continuos model the period length is T time units, while in the discrete model the period length is Q time slots), and let $x = a \mod \tau$ and $y = b \mod \tau$. If $x \leq y$, $S[a, b] = \{p \in S \mid x \leq p \leq y\}$, otherwise $S[a, b] = \{p \in S \mid p \geq x \vee y \geq p\}$.

If t_u represents the time of occurrence of some event with respect to node u we use $\mathring{t}_u$ to represent the time of occurrence of the event in a global reference frame. For example, consider neighboring nodes u and v, and suppose that node u executes some event e_u at local time t_u which is instantaneously observed by node v at local time t_v. Since we do not assume synchronized clocks, then in general $t_u \neq t_v$, however $\mathring{t}_u = \mathring{t}_v$.

We say that an event happens almost surely if it happens with probability one, an event happens with high probability if it occurs with probability at least $1 - \frac{1}{n}$. Let $\mathfrak{U}(a, b)$ denote the continuous uniform distribution in the range $[a, b]$ and $\mathfrak{U}[a..b]$ denote the discrete uniform distribution in the range $[a..b]$.

We believe the model described is simple enough to be implemented or simulated in real hardware. However it is still complex enough to allow for the design of interesting algorithms with strong theoretical guarantees. We consider two variants of the basic model, a continuous version and a discrete version.

Discrete Model. Time is divided into slots of length μ, where μ depends on the physical characteristics of the wireless devices and of the communication medium. There is a known integer $Q > 0$ that denotes the number of slots per period, and is related to the number of resources available. Hence, the period length is $T = Q\mu$. Although we do not assume synchronized clocks, we assume that slots boundaries are synchronized, i.e., all nodes start new slots at the same time. Note that at the cost of small constant factors and more technical arguments, all results obtained in this paper can also be achieved in a model with unsynchronized slot boundaries.

In each slot s, each node v can either listen or beep for the whole duration of s. If a beep is emitted by node u at slot s, it is heard by any neighboring node $v \in N(u)$ that is in listening mode in slot s. In particular the operation `listen`$[m]$ puts the node in listening mode for the next m slots and returns the set of slots where it detected a beep. The operation `beep` emits a beep for the duration of the current slot.

Continuous Model. All nodes share some period length T and a beep can be infinitely short (i.e., a unit impulse function). If a beep is emitted by node u at time t, it is heard by any neighboring node $v \in N(u)$ that is in listening mode at time t. In particular the operation `listen`(δ) puts the node in listening mode for the next δ units of time and returns the set of time points where it detected beeps. The operation `beep` emits an infinitely short beep. We discuss the shortcomings of this variant in Section 4.

3 Interval Coloring

One of the central motivations behind vertex coloring in distributed environments is to use it as a building block for MAC protocols. In this setting the number of colors used translates to the number of communication channels used, and thus fewer colors imply higher throughput. In general we are interested in efficient (polylog or better) algorithms that produce vertex colorings with $\mathcal{O}(\Delta)$ colors, where Δ is the maximum degree. However, most known distributed algorithms for coloring are based on the assumption that there is already an infrastructure to reliably transmit messages with neighboring nodes, which makes them unsuitable for MAC protocols. This motivates studying coloring in the beeping model. We focus on interval coloring, a variant of vertex coloring specially well suited for the beeping model.

Given an ordered set of resources, an interval coloring assigns each node an interval (contiguous fraction) of resources such that neighboring nodes do not share resources. A k interval coloring is one where every node gets at least a $1/k$ fraction of the resources.

In particular, we focus on the case where the set of resources to be shared is time (i.e. computing a TDMA schedule). The discrete beeping model assumes all nodes agree on a period of length T, which is composed of Q slots of length μ where slot boundaries are synchronized. However, the lack of synchronized clocks implies the periods of different nodes are not aligned, and hence the first slot of a period for node u could be in the middle of the period for node v. Therefore, although nodes agree on the set of resources to be shared (Q time slots), they do not agree on an ordering of these resources. To sidestep this problem we will require interval coloring to output a tuple $\langle p_v, I_v \rangle$ for each node v, where p_v is the offset with respect to the period start of node v, and I_v is the interval length. These tuples should be such that for every pair of neighbors $\{u, v\} \in E$, the intervals $[\mathring{p}_v - I_v, \mathring{p}_v]$ and $[\mathring{p}_u - I_u, \mathring{p}_u]$ are disjoint for every period. Analogous to $\mathcal{O}(\Delta)$-vertex colorings, we are interested in $\mathcal{O}(\Delta)$-interval colorings, where each node gets assigned at least a $\Omega(1/\Delta)$ fraction of the resources.

Hardness of Interval Coloring. Discrete interval coloring is strongly related to vertex coloring. For each node v let $\langle p_v, I_v \rangle$ be the tuple output by an interval coloring at node v. The definition of interval coloring implies that for any two neighbors u and v it holds that $\mathring{p}_v \neq \mathring{p}_u$. Therefore, we can define a valid vertex coloring by assigning to each node v the color $c_v = \mathring{p}_v \pmod{Q}$. Observe that if $Q \in \Theta(\Delta)$, this is a $\mathcal{O}(\Delta)$-vertex coloring. Hence, even in executions where all nodes have either synchronized clocks or wakeup at the same time, a $\mathcal{O}(\Delta)$-interval coloring is at least as hard as $\mathcal{O}(\Delta)$-vertex coloring.

4 Continuous Interval Coloring

We essentially use the same model as Motskin et al. [17], and adhering to it we also assume each node v knows its own degree $d(v)$ and the maximum degree of its 1-hop neighbors $d^{\max}(v)$. Motskin et al. [17] described a randomized algorithm that solves continuous interval coloring and terminates with high probability in a logarithmic number of periods. In contrast, we present a randomized algorithm that solves the same problem but terminates almost surely in a constant number of periods. While describing the algorithm we expose the flaws of this model that make such an algorithm possible.

Algorithm Description. Since nodes can emit an infinitely short beep at any point in time, then if two nodes choose to beep at random times in the interval $[0, T]$, their beeps will collide with probability zero (i.e. the probability that two samples from a continuous uniform distribution are equal is zero). We will exploit this property with the greedy algorithm BEEPFIRST, described in detail in Algorithm 1. Informally speaking, the BEEPFIRST algorithm searches for the first available time where a node can beep while respecting a buffer of size b_v around existing beeps. To ensure that no two nodes choose the same time to beep, the buffer size and starting time are randomized with a continuous variable.

More precisely, the algorithm has a parameter $\varepsilon \in (0, 1)$ which affects the size of the resulting intervals. In the initialization state, each node v sets its interval length to $I_v = (1-\varepsilon)T/2(d^{\max}(v)+1)$ and chooses $\varepsilon_v \in \mathfrak{U}[0, \varepsilon]$ to randomize its start time and set its buffer length to $b_v = (1 - \varepsilon_v)T/2(d(v) + 1)$.

In the searching state, nodes listen for one full time period T recording the phases at which beeps are heard. If a node hears no beeps in this first period it sets $p_v = 0$ and goes to the stable state. Otherwise nodes search for the first phase p_v such that (i) in the previous period no other node beeped in the interval $[p_v - b_v, p_v + b_v]$, and (ii) in this period no other node beeps on the interval $[p_v - b_v, p_v]$. Once such a phase is found, nodes beep to reserve it and listen for whatever remains of the period, switching to the stable state. Once a node becomes stable, it remains stable thereafter, beeping at the same phase every period.

Algorithm 1 BEEPFIRST running at node v

```
$\varepsilon_v \leftarrow \mathfrak{U}(0, \varepsilon)$                                        ▷ Initialize
$I_v \leftarrow (1-\varepsilon)\frac{T}{2(d^{\max}(v)+1)}$, $b_v \leftarrow (1-\varepsilon_v)\frac{T}{2(d(v)+1)}$
listen($\varepsilon_v$) (* randomized start time *)
$S \leftarrow$ listen($T$)                                            ▷ Search
$p_v \leftarrow 0$
while $\exists$ beep in $S[p_v - b_v, p_v + b_v]$ do
    $t_v \leftarrow p_v$
    $p_v \leftarrow b_v +$ time of last beep in $S[p_v - b_v, p_v + b_v]$
    $S \leftarrow S \cup$ listen($p_v - t_v$)
end while
beep, listen($T - p_v$)                                               ▷ Stable
loop
    listen($p_v$), beep, listen($T - p_v$)
end loop
```

For each node v in the searching state, the separation between beeps heard by v is at most $2b_v$, otherwise it would have exited the search state. Assume in a period node v hears at most one beep from each neighbor (the same result can be proved without this assumption with a slightly more technical argument). Therefore node v hears at most $d(v)$ beeps in one period, which means that after time $d(v)2b_v < T$ in the searching state node v finds a proper phase to beep and enters the stable state.

Lemma 1. *The searching state of* BEEPFIRST *lasts less than one period.*

By construction node v will select $p_v = 0$ or $p_v = p_u + b_v$ where p_u is the phase of node u. However, recall that both the starting time and the buffer length are randomized using a continuous probability distribution. Therefore, with probability one, no two nodes will ever select the same phase. (The same argument is used by Motskin et al. [17] to prove that neighbors "pick the *exact* same start time with probability 0".) Which is captured by the following proposition.

Proposition 2. *Given a pair of nodes u and v (where $u \neq v$) at any point during the execution of* BEEPFIRST *almost surely $\mathring{p}_u \neq \mathring{p}_v$.*

From Proposition 2 it follows that given two neighboring nodes, one selects an earlier phase than the other. This fact can be leveraged to show that the intervals produced by BEEPFIRST do not overlap.

Lemma 3. *Let u and v be two neighboring nodes in a stable state of* BEEPFIRST, *then their intervals do not overlap ($\mathring{p}_u \notin [\mathring{p}_v - I_v, \mathring{p}_v + I_v]$).*

We can tie Lemmas 1 and 3 together in the following theorem.

Theorem 4. *The* BEEPFIRST *algorithm computes a $\mathcal{O}(\Delta)$-interval coloring almost surely in $O(1)$ time.*

Observe that if instead of setting the interval length in the initialization phase, we delayed it until the stable phase by setting it to the largest value such that $[p_v - I_v, p_v + I_v]$ does not contain any beeps, we would get a slightly stronger result which does not require knowledge of $d^{\max}(v)$. The BEEPFIRST algorithm hints at two flaws in this model (i) It assumes knowledge of $d(v)$ and $d^{\max}(v)$, where neither is trivial to compute. (ii) The algorithm's correctness relies on computation with arbitrary real numbers and sampling from continuous probability distributions.

5 Discrete Interval Coloring

We now turn our attention to a more realistic model where beeps occur at discrete times and have a minimum length, thus the probability distributions involved are discrete and finite. We present a Las Vegas randomized algorithm for $\mathcal{O}(\Delta)$-interval coloring that terminates with high probability in $\mathcal{O}(\log n)$ periods. This requires $Q \geq \Delta$ and in particular we assume $Q = \kappa\Delta$ where κ is a large enough constant ($\kappa \geq 3/\eta$ suffices, for η to be fixed later).

Algorithm Description. The JITTERANDJUMP algorithm relies on three key insights: (i) The number of beeps heard by a node is a good estimate of its degree. (ii) By adding a small random jitter to every beep, neighboring nodes which beep at the same slot can detect the collision with constant probability. (iii) If a node jumps into a random slot which is surrounded by "enough" empty slots it finds a non-overlapping interval assignment with constant probability.

The detail pseudo-code is presented in Algorithm 2, in the following paragraphs we give an informal description. All nodes executing are initially uncolored, and they become colored when they believe to have found a non-overlapping interval. Except for the first period (where nodes listen without beeping), all nodes beep once per period. Therefore in a single period a node can hear at most two beeps per neighbor, and it follows that if $\tilde{d}_v$ is the number of beeps observed by node v during a period, then $1 \leq \tilde{d}_v \leq 2d(v)$.

To resolve collisions, if node v has decided to beep at the slot p_v, it chooses at random $jitter_v \in \mathfrak{U}[0..1]$, and beeps at $p_v + jitter_v$ instead. If a colored node detects a beep one slot before, or two slots after its own beep, it becomes uncolored.

Each node v sets the buffer length $b_v = \eta \frac{Q}{\tilde{d}_v+1}$ to a fraction of the period proportional to its degree estimate, where η is a sufficiently small constant (we will show that any $\eta \leq 1/16$ suffices). Using the information collected in the previous period, node v computes a set of *free slots* F_v. A free slot $s \in F_v$ is one where no beep was heard in the $b_v + 2$ slots preceeding it, and the $b_v + 1$ following it. An uncolored node v selects a slot p_v to beep uniformly at random from the set of free slots F_v. If after beeping node v determines no other node is in the interval $[p_v - b_v, p_v]$ it becomes colored.

Algorithm 2 JITTERANDJUMP running at node v

```
$colored_v \leftarrow false$
$S \leftarrow \texttt{listen}[Q]$
$\tilde{d}_v \leftarrow \max(|S|, 1)$
$b_v \leftarrow \eta \frac{Q}{\tilde{d}_v+1}$
loop
    if not $colored_v$ then
        $F_v \leftarrow \{p \mid S \cup \{p_v\}\,[p - b_v - 2, p + b_v + 1] = \varnothing\}$
        $p_v \leftarrow \mathfrak{U}_{F_v}$
    end if
    $jitter_v \leftarrow \mathfrak{U}[0..1]$
    $S \leftarrow \texttt{listen}[p_v + jitter_v - 1] \cup \texttt{beep} \cup \texttt{listen}[Q - p_v - jitter_v]$
    $I_v \leftarrow \max s$ s.t. $S[p_v - s, p_v] = \varnothing$
    $\tilde{d}_v \leftarrow \max(|S|, 1)$
    $b_v \leftarrow \eta \frac{Q}{\tilde{d}_v}$
    if $S[p_v - b_v, p_v + b_v] = \varnothing$ then
        $colored_v \leftarrow true$
    else if $S[p_v - 1, p_v + 2] \neq \varnothing$ then
        $colored_v \leftarrow false$
    end if
end loop
```

Two neighboring nodes are *colliding* if they beep at the same slot. Every period, each nodes selects independently at random a jitter which affects where they beep. It is possible to show that two collided nodes will detect the collision and become uncolored with constant probability (proof omitted).

Lemma 5. *If neighboring nodes u and v collide in* JitterAndJump*, they become uncolored in the next period with probability at least $\frac{1}{2}$.*

By adjusting κ and η appropriately, it is possible to guarantee that the number of free slots observed by each node is a constant fraction of the number of slots.

Proposition 6. *If $\kappa \geq 4/\eta$ and $\eta \leq 1/3$ then $|F_v| \geq (1 - 3\eta)Q$ for every node v.*

We have already established that the degree estimate is an upper bound on the real degree; we also show that with constant probability it is a lower bound on the number of uncolored nodes. To do so we use the following result concerning the classical balls and bins problem (proof omitted).

Theorem 7. *When placing m balls randomly into n bins, if $n \geq m \geq 12$ then with probability more than $\frac{1}{2}$ the number of occupied bins is at least $\frac{m}{4}$.*

Using this result we can prove the following lemma.

Lemma 8. *With probability $\frac{1}{2}$ the number of beeps observed by a node is at least a quarter of the number of its uncolored neighbors.*

Proof. Fix node v and let $P \subseteq N(v)$ be its uncolored neighbors. We want to show $\mathbb{P}\left[\tilde{d}_v > |P|/4\right] \geq \frac{1}{2}$.

Each node $u \in P$ beeps at random in F_u and if $\kappa \geq 4/\eta$ then from Proposition 6 $|F_u| \geq (1 - 3\eta)Q = (1 - 3\eta)\kappa\Delta$. If we let $\eta \leq 1/16$ then $\kappa \geq 1/(1 - 3\eta)$ and thus $|F_u| \geq \Delta$.

On the other hand, the probability of collisions (and a lower degree estimate $\tilde{d}_v$) is increased if $\forall u, w \in P\ F_u = F_v$. In other words, if $|P| \leq \Delta$ beeps are randomly distributed in $|F_v| \geq \Delta$ slots, and assuming enough beeps theorem 7 implies that with probability $\frac{1}{2}$ the number of occupied slots is $|P|/4$. $\square$

To argue termination we partition nodes into `good` and `bad` nodes. Informally, a `good` node is one which, modulo the jitter, continues to beep at the same slot in the rest of the execution.

Definition 1. Node v is `good` if it is colored and there does not exist a neighboring node $u \in N(v)$ with a phase p_u such that $|\mathring{p}_u - \mathring{p}_v| \leq 1$; otherwise v is `bad`.

By definition, once a node becomes `good` no neighboring node is colliding with it. Since nodes listen before beeping and always beep at slots which were previously unoccupied, it is not surprising that once a node becomes `good` it remains `good` thereafter (proof omitted).

Lemma 9. *Once a node is* `good`*, it remains* `good` *for the rest of the execution.*

We classify `bad` nodes further as colored and uncolored. First we consider the easier case of colored `bad` nodes.

Lemma 10. *A colored* `bad` *node becomes* `good` *or uncolored with probability* $\geq \frac{1}{2}$.

Proof. Fix a colored `bad` node v. Since it is bad and uncolored, then by definition a nonempty set of its neighbors $P \subseteq N(v)$ beep at the same slot as u.

If all nodes in P are uncolored, then they all jump to a random slot and node v becomes `good`. Otherwise there exists a colored node $u \in P$. However by Lemma 5 with probability $\frac{1}{2}$ in the next period both nodes detect the collision and become uncolored. $\square$

Now we consider uncolored `bad` nodes.

Lemma 11. *An uncolored* `bad` *node becomes* `good` *with probability* $\geq \frac{1}{2}e^{-\frac{16\eta}{1-3\eta}}$.

Proof. Fix an uncolored `bad` node v. Let B_u be the event that node u choses to beep in the interval $[p_v - b_v, p_v + b_v]$. In other words, B_u is the event that node u interferes with the beep of node v. By definition $\mathbb{P}[B_u] \leq \frac{2b_v}{|F_u|}$, and from Proposition 6 $|F_u| \geq Q(1-3\eta)$ and thus $\mathbb{P}[B_u] \leq \frac{2b_v}{Q(1-3\eta)} \leq \frac{2\eta}{\tilde{d}_v(1-3\eta)}$.

Let G_v be the event that node v becomes good. Node v becomes `good` unless a nonempty subset of its (uncolored) neighbors pick a slot that interferes with its beep. Hence $\mathbb{P}[G_v] = \prod_{u \in P} \mathbb{P}[\neg B_u]$ where $P \subseteq N(v)$ are the uncolored neighbors of v.

Let P_v be the event that the number of beeps observed by v is at least one quarter of the number of its uncolored neighbors, that is $\tilde{d}_v \geq |P|/4$. We show that conditioned on P_v, node v becomes good with constant probability.

$$\mathbb{P}[G_v|P_v] = \prod_{u \in P} \mathbb{P}[\neg B_u|P_v] = \prod_{u \in P}(1 - \mathbb{P}[B_u|P_v]) \geq \left(1 - \frac{8\eta}{|P|(1-3\eta)}\right)^{|P|} \geq e^{-\frac{16\eta}{1-3\eta}}$$

Where the last inequality holds for sufficiently small $\eta \leq \frac{1}{16}$. Finally from Lemma 8 we have $\mathbb{P}[P_v] \geq \frac{1}{2}$, hence $\mathbb{P}[G_v] \geq \mathbb{P}[G_v|P_v]\,\mathbb{P}[P_v] \geq \frac{1}{2}e^{-\frac{16\eta}{1-3\eta}}$. $\square$

From Lemmas 10 and 11, after two periods a `bad` node becomes `good` with constant probability. Therefore the probability that a node remains `bad` drops off exponentially with the number of periods. Using standard arguments one can show that a `bad` node becomes `good` with high probability after $\frac{6}{e^{-\frac{16\eta}{1-3\eta}}}\log n \in \mathcal{O}(\log n)$ rounds.

We now show that each node is assigned a "large" fraction of the slots.

Lemma 12. *Let v be a* `good` *node, then* $I_v \geq \eta\frac{Q}{2d^{\max}(v)+1}$.

Proof. Consider the period when v became colored. By construction in that period node v observed no beeps in the interval $[p_v - b_v, p_v]$, thus $I_v \geq \eta\frac{Q}{\tilde{d}_v+1}$ in that period.

Fix a node $u \in N(v)$. Node u will only select to beep in phases that respect a buffer of size $b_u + 2 = \eta\frac{Q}{\tilde{d}_u+1} + 2$ before the beep of node v. So independent of the jitter, node v will never observe a beep of u within within b_u of its phase. Finally, since $\forall u \in V$ it holds that $\tilde{d}_u \leq 2d^{\max}(v)$, and hence $I_v \geq \eta\frac{Q}{2d^{\max}(v)+1}$ in all subsequent periods. $\square$

This leads to our main theorem.

Theorem 13. *The* JITTERANDJUMP *algorithm computes a $\mathcal{O}(\Delta)$-interval coloring with high probability in $\mathcal{O}(\log n)$ periods.*

5.1 Dynamic Graphs

We turn our attention to dynamic graphs, where nodes and edges are added and removed throughout the execution. Adding nodes or edges is analogous to waking up, which is already handled gracefully by JITTERANDJUMP. However this is not the case for node or edge removals. In particular, once the algorithm has stabilized to an $\mathcal{O}(\Delta)$-interval coloring, the interval of each node is not guaranteed to increase, even if sufficiently many nodes leave and the new maximum degree becomes $\Delta' \ll \Delta$.

A natural solution would be to go back to an uncolored state when the degree estimate falls below a certain threshold. However, colliding nodes can cause the degree estimate to drop artificially, even when no nodes or edges are removed. Moreover in some cases, the colliding nodes are not aware of each other and can remain collided forever despite jittering. For example in a star graph, from the center's perspective the spokes may be colliding, but the spokes have no means of detecting the collision.

Algorithm description (modifications to JITTERANDJUMP*).* Regardless of the state, each node v picks a second phase p'_v at random from the free slots F_v. As before, node v will beep at $p_v + jitter_v$, but additionally also beep at p'_v. Let $S_v(i)$ be the set of slots where node v heard a beep in period i. We define $d^*_v(i) = \max_{j\in[i-r,i])} |S_v(j)|$ as the maximum number of beeps over a moving window of the last r periods. At period i we update the degree estimate by taking the maximum of the current beep count and $d^*_v(i)$ ($\tilde{d}_v = \max(\tilde{d}_v, d^*_v(i))$). Finally, if $d^*_v(i) < \frac{\tilde{d}_v}{16}$ we set $\tilde{d}_v = d^*_v(i)$ and uncolor node v.

Since nodes beep twice at every period then for every period i, $S_v(i) \leq 4d(v)$. In executions where the degree estimate doesn't decrease, the analysis of Section 5 still holds, albeit with slightly different constants. To prove correctness we need to show that with sufficiently high probability the degree estimate will decreases if and only if the degree drops by a large enough factor.

From proposition 6 the number of free slots is $|F_v| \geq (1-3\eta)Q = (1-3\eta)\kappa\Delta$, and since $\kappa \geq \frac{1}{1-3\eta}$ then $|F_v| \geq \Delta$. Given that a node v has $d(v)$ neighbors, and each neighbor beeps at least once per period in a random slot (at most twice), we are interested in the probability that the beeps observed account for a constant fraction of the neighbors. This is essentially the same scenario described by lemma 8 which can be viewed as an occupancy problem. Using theorem 7 we can show that with probability at least $\frac{1}{2}$ the number of beeps observed is at least $d(v)/4$.

Hence, at every period i we have $|S_v(i)| \geq d(v)/4$ with probability $\geq \frac{1}{2}$. Since the degree estimate is computed using the information of the last r periods, the degree estimate decreases only if in the last r periods the beep count observed was below $\tilde{d}_v/16$. However, unless the real degree has decreased by a constant factor, this happens with probability less than $\frac{1}{2}^r$. On the other hand, if the real degree decreases by a large enough factor, the degree count observed for the next r periods will be at most four times the real degree, which will cause the degree estimate to decrease with certainty after r periods.

Finally,y setting $r \in \mathcal{O}(\log 1/\varepsilon)$ the same argument used before can be used to prove the algorithm described computes an $\Omega(T/\Delta)$-interval coloring in $\mathcal{O}(\log 1/\varepsilon)$ periods with probability $1-\varepsilon$.

6 Lower Bound

We consider a stronger model, namely standard synchronous local broadcast with messages of constant size. During each slot a node sends a message of constant size and receives the set of messages sent by its neighbors. Assume every node v knows its own degree $d(v)$, the maximum degree Δ and the size of the network n, but does not have unique IDs. All nodes start the execution (wakeup) simultaneously.

The rest of this section is devoted to proving the following theorem.

Theorem 14. *Under the model described, in less than $\mathcal{O}(\log n)$ slots it is impossible to compute a $\mathcal{O}(\Delta)$-interval coloring or a $\mathcal{O}(\Delta)$-vertex coloring with high probability.*

Proof. Let $G_i = (B_i, E_i)$ be a graph on four vertices, with vertex set $B_i = \{a_i, b_i, c_i, d_i\}$ and edge set $E_i = \{(a_i, b_i), (b_i, c_i), (c_i, d_i), (a_i, c_i), (b_i, d_i)\}$. Define G as the cycle graph generated by pasting together $n/4$ copies of G_i, where $\forall i \in [0, \frac{n}{4} - 1]$ we connect component G_i with component $G_{i+1 \mod \frac{n}{4}}$ by adding the edge $(d_i, a_{(i+1 \mod \frac{n}{4})})$. G is a 4-regular graph of size n and inside every component G_i the vertices b_i and c_i have the same closed neighborhood.

Let s_u^k be the state of node u at slot k, and let m_u^k be the message sent by node u in slot k. Regardless of its state, a node can only choose to send a message amongst a set of constant size of possible messages, let c be the size of this set.

Consider a component B_i, and assume the states of b_i and c_i are identical at slot k. Since their closed neighborhood is identical, if they send the same message at slot k, they will receive the same set of messages and remain in identical states at slot $k + 1$. Formally, if $s_{b_i}^k = s_{c_i}^k$ and $m_{b_i}^k = m_{c_i}^k$ then $s_{b_i}^{k+1} = s_{c_i}^{k+1}$.

Moreover, if b_i and c_i are in the same state at slot k, they choose what to send according to the same probability distribution, in particular let p_i (where $i \in [1, c]$) be the probability of sending the i^{th} message. By definition $\sum_{i=1}^{c} p_i = 1$, and thus by Cauchy-Schwarz we have $\sum_{i=1}^{c} p_i^2 \geq \frac{1}{c}$

We prove a lower bound on the probability that b_i and c_i remain in the same state in the next slot:

$$\mathbb{P}\left[s_{b_i}^{k+1} = s_{c_i}^{k+1} \mid s_{b_i}^k = s_{c_i}^k\right] \geq \mathbb{P}\left[m_{b_i}^k = m_{c_i}^k \mid s_{b_i}^k = s_{c_i}^k\right] = \sum_{i=1}^{c} p_i^2 \geq \frac{1}{c}$$

Therefore, if nodes b_i and c_i start at the same state ($s_{b_i}^0 = s_{c_i}^0$) the probability that they remain in the same state after ℓ slots is $\mathbb{P}\left[s_{b_i}^\ell = s_{c_i}^\ell \mid s_{b_i}^0 = s_{c_i}^0\right] \geq \frac{1}{c}^\ell$. If we let $\ell = \log_c \frac{n}{4}$ then $\mathbb{P}\left[s_{b_i}^\ell = s_{c_i}^\ell \mid s_{b_i}^0 = s_{c_i}^0\right] \geq \frac{4}{n}$, and thus $\mathbb{P}\left[s_{b_i}^\ell \neq s_{c_i}^\ell \mid s_{b_i}^0 = s_{c_i}^0\right] \leq 1 - \frac{4}{n}$.

Since there are no unique identifiers, initially all nodes have the same state ($\forall u, v \in V$, $s_u^0 = s_v^0$), and the probability that after ℓ slots every component B_i has $s_{b_i}^\ell \neq s_{c_i}^\ell$ is:

$$\mathbb{P}\left[\forall B_i, s_{b_i}^\ell \neq s_{c_i}^\ell\right] = \prod_{i=1}^{n/4} \mathbb{P}\left[s_{b_i}^\ell \neq s_{c_i}^\ell\right] \leq \left(1 - \frac{4}{n}\right)^{\frac{n}{4}} \leq \frac{1}{e}$$

Therefore there exists a pair of neighboring nodes that remain in the same state after ℓ slots with constant probability.

$$\mathbb{P}\left[\exists (u,v) \in E \text{ s.t. } s_u^\ell = s_v^\ell\right] \geq \mathbb{P}\left[\exists B_i \text{ s.t. } s_{b_i}^\ell = s_{c_i}^\ell\right] = 1 - \mathbb{P}\left[\forall B_i, s_{b_i}^\ell \neq s_{c_i}^\ell\right] \geq 1 - \frac{1}{e}$$

Moreover, since G is a 4-regular graph, it should ensure interval lengths of size $\Omega(Q/4) \in \Omega(Q)$. Finally, if two nodes in the same state select intervals of size $\Omega(Q)$ slots out of a total of Q slots, the probability that they select overlapping intervals is greater than a constant. Therefore with constant probability after $\Omega(\log n)$ slots there is a pair of neighboring nodes which do not have an $\mathcal{O}(\Delta)$-interval coloring.

Observe that if instead of solving interval coloring we were considering vertex coloring, the probability that two neighboring nodes select the same color out of Δ available colors is also a constant, and thus with constant probability a pair of neighboring nodes select the same color. Which concludes the proof. $\square$

In light of the upper bound of $\mathcal{O}(\log n)$ periods presented in Section 5, the previous bound is asymptotically tight for constant degree graphs. Since each period has $Q \geq \Delta$ slots this implies a lower bound of $\Omega(\log n/\Delta)$ periods for general graphs. If we additionally assume each node beeps at most $\mathcal{O}(1)$ times per period, the same argument yields a lower bound of $\Omega(\log n/\log \Delta)$ periods for general graphs, since for each node the probability of beeping in the same slot as a neighbor is $1/\kappa\Delta$.

References

1. B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. of 30th Symposium on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.
2. B. Balasundaram and S. Butenko. Graph domination, coloring and cliques in telecommunications. In M.G.C. Resende and P.M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 865–890. Springer, 2006.
3. L. Barenboim and M. Elkin. Distributed $(\Delta + 1)$-coloring in linear (in Δ) time. In *Proc. of the 41st ACM Symposium on Theory of Computing (STOC)*, 2009.
4. L. Barenboim and M. Elkin. Deterministic distributed vertex coloring in polylogarithmic time. In *Proc. 29th ACM Symposium on Principles of Distributed Computing (PODC)*, 2010.
5. J. Degesys and R. Nagpal. Towards desynchronization of multi-hop topologies. In *Proc. 2nd IEEE Conference Self-Adaptive and Self-Organizing Systems (SASO)*, pages 129–138, 2008.
6. J. Degesys, I. Rose, A. Patel, and R. Nagpal. Desync: self-organizing desynchronization and TDMA on wireless sensor networks. In *Proc. 6th Conference on Information Processing in Sensor Networks (IPSN)*, page 20, 2007.
7. R. Flury and R. Wattenhofer. Slotted programming for sensor networks. In *Proc. 9th Conference on Information Processing in Sensor Networks (IPSN)*, 2010.
8. S. Gandham, M. Dawande, and R. Prakash. Link scheduling in sensor networks: Distributed edge coloring revisited. In *Proc. of 24th IEEE Conference on Computer Communications (INFOCOM)*, pages 2492–2501, 2005.
9. A.V. Goldberg, S.A. Plotkin, and G.E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988.
10. T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *Proc. of 1st Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 45–58, 2004.

11. K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer. Distributed coloring in $o(\sqrt{\log n})$ bit rounds. In *Proc. of 20th IEEE Parallel and Distributed Processing Symposium (IPDPS)*, 2006.
12. F. Kuhn. Local multicoloring algorithms: Computing a nearly-optimal TDMA schedule in constant time. In *Proc. of 26th Symp. on Theoretical Aspects of Computer Science (STACS)*, 2009.
13. F. Kuhn. Weak graph coloring: Distributed algorithms and applications. In *Proc. of 21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2009.
14. N. Linial. Locality in distributed graph algorithms. In *SIAM Journal on Computing*, pages 193–201, 1992.
15. S. Mecke. MAC layer and coloring. In D. Wagner and R. Wattenhofer, editors, *Algorithms for Sensor and Ad Hoc Networks*, pages 63–80, 2007.
16. T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. In *Proc. 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 39–48, 2005.
17. A. Motskin, T. Roughgarden, P. Skraba, and L. Guibas. Lightweight coloring and desynchronization for networks. In *Proc. 28th IEEE Conference on Computer Communications (INFOCOM)*, 2009.
18. A. Panconesi and A. Srinivasan. On the complexity of distributed network decomposition. *Journal of Algorithms*, 20(2):581–592, 1995.
19. S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5:81–94, 1999.
20. I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks. In *7th ACM Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 190–201, 2006.
21. C. Scheideler, A. Richa, and P. Santi. An o(log n) dominating set protocol for wireless ad-hoc networks under the physical interference model. In *Proc. 9th ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 91–100, 2008.
22. S. Schmid and R. Wattenhofer. Algorithmic models for sensor networks. In *Proc. 14th Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2006.
23. J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *Proc. of 27th ACM Symposium on Principles of Distributed Computing (PODC)*, 2008.
24. J. Schneider and R. Wattenhofer. Coloring unstructured wireless multi-hop networks. In *Proc. 28th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 210–219, 2009.
25. J. Schneider and R. Wattenhofer. A new technique for distributed symmetry breaking. In *Proc. 29th ACM Symposium on Principles of Distributed Computing (PODC)*, 2010.
26. USC/ISI. *Network Simulator 2 (NS2)*. URL `http://www.isi.edu/nsnam/ns/`.
27. X. Zhang, J. Hong, L. Zhang, X. Shan, and V.O.K. Li. CP-TDMA: Coloring- and probability-based TDMA scheduling for wireless ad hoc networks. *IEICE Transactions on Communication*, E91-B(1):322–326, 2008.