Coloring Unstructured Radio Networks *

Thomas Moscibroda and Roger Wattenhofer
{moscitho,wattenhofer}tik.ee.ethz.ch
Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland

Abstract

During and immediately after their deployment, ad hoc and sensor networks lack an efficient communication scheme rendering even the most basic network coordination problems difficult. Before any reasonable communication can take place, nodes must come up with an initial structure that can serve as a foundation for more sophisticated algorithms. In this paper, we consider the problem of obtaining a vertex coloring as such an initial structure. We propose an algorithm that works in the unstructured radio network model. This model captures the characteristics of newly deployed ad hoc and sensor networks, i.e. asynchronous wake-up, no collision-detection, and scarce knowledge about the network topology. When modeling the network as a graph with *bounded independence*, our algorithm produces a correct coloring with $O(\Delta)$ colors in time $O(\Delta \log n)$ with high probability, where n and Δ are the number of nodes in the network and the maximum degree, respectively. Also, the number of locally used colors depends only on the local node density. Graphs with bounded independence generalize unit disk graphs as well as many other well-known models for wireless multi-hop networks. They allow to capture aspects such as obstacles, fading, or irregular signal-propagation.

1 Introduction

Wireless multi-hop radio networks such as ad hoc or sensor networks [1] are formed of autonomous nodes communicating via radio. Typically, if two nodes are not within their mutual transmission range, they may communicate through intermediate nodes. In other words, the communication infrastructure must be organized by the nodes themselves, rather than being provided as part of a fixed built-in infrastructure as in traditional wired networks.

The lack of available a-priori infrastructure is particularly pronounced during and after the deployment, when the network is unstructured and chaotic [15, 16, 19]. Before any reasonable communication can be carried out and before the network can start performing its intended task, the nodes must establish some kind of structure that allows an efficient communication scheme. Once this *initial structure* is achieved, sophisticated and well-studied algorithms and network organization protocols may be used on top of it. Naturally, the inherent problem faced when setting up such an initial structure is that there is no existing infrastructure that could facilitate the task. In fact, coping with the absence of an *initial structure* is one of the quintessential tasks in ad hoc and sensor networks and finding efficient solutions for that purpose is of great practical importance. In existing systems such as Bluetooth, for instance, the initialization tends to be slow even for a small number of devices.

*A preliminary version of this work has been published in [20] as Coloring Unstructured Radio Networks, In *Proceedings of the 17th Symposium on Parallel Algorithms and Architectures (SPAA)*, Las Vegas, Nevada, 2005.

In this paper, we study the construction of an initial structure useful for subsequent network organization tasks. Technically, we study how the network nodes can quickly compute a good *vertex coloring* without relying on any existing infrastructure. A correct vertex coloring for a graph $G = (V, E)$ is an assignment of a color $\mathrm{color}(v)$ to each node $v \in V$, such that any two adjacent nodes have a different color. Colorings can be well-motivated as initial structures in wireless ad hoc and sensor networks: When associating different colors with different time slots in a time-division multiple access (TDMA) scheme, a correct coloring corresponds to a medium access control (MAC) layer without *direct interference*, that is, no two neighboring nodes send at the same time.

It is well known that in order to guarantee an entirely collision-free schedule in wireless networks, a correct vertex coloring is not sufficient, for what is needed is a coloring of the *square* of the graph, i.e., a valid distance 2-coloring [14, 26]. However, besides being a non-trivial first step towards obtaining a distance 2-coloring, a simple vertex-coloring ensures a schedule in which a receiver can be disturbed by at most (a small) constant number of interfering senders in a given time slot. This allows for simple randomized algorithms guaranteeing every sender a constant sending probability in each scheduled time slot. As the available bandwidth (and hence the possible throughput) of a node v in such a schedule depends on the highest color assigned in its local 2-neighborhood, only low colors should be assigned in sparse areas of the network, whereas the higher colors should only be used in dense areas. Particularly, a good coloring should have the property that the highest color assigned to a node in each neighborhood should depend only on the *local node density* of that neighborhood.

In view of our goal of setting up an initial MAC scheme in a newly deployed network, our coloring algorithm must not rely on any previously established MAC layer. Instead, we are interested in a simple algorithm that quickly computes a coloring entirely from *scratch*. Note that this precludes algorithms working under any sort of *message passing model* in which nodes know their neighbors a-priori, and in which messages can be sent to neighbors without fearing collision, e.g. [3, 8, 24]. Studying classic network coordination problems such as coloring in absence of an established MAC layer highlights the *chicken-and-egg* problem of the initialization phase [15]. A MAC layer ("chicken") helps achieving a coloring ("egg"), and vice versa. The problem is that in a newly deployed ad-hoc/sensor network, there is typically no built-in structure, i.e. there are neither "chickens" nor "eggs."

Clearly, one important aspect when studying the initialization phase of ad hoc/sensor networks is to use an appropriate model. On the one hand, the model should be realistic enough to actually capture the particularly harsh characteristics of the deployment phase. But on the other hand, it ought to be concise enough to allow for stringent reasoning and proofs. Recently, the *unstructured radio network model* has been proposed as a model that attempts to combine both of these contradictory aims [15]. It makes the following assumptions.

- We consider *multi-hop* networks, that is, there exist nodes that are not within their mutual transmission range. Therefore, it may occur that some neighbors of a sending node receive a message, while others experience interference from other senders and do not receive the message.

- Nodes can wake up *asynchronously*. In a wireless, multi-hop environment, it is realistic to assume that some nodes wake up (e.g. become deployed, or switched on) later than others. Thus, nodes do not have access to a global clock. Contrary to work on the so-called *wake-up problem* [6, 13], nodes are *not* woken up by incoming messages, that is, sleeping nodes do neither send nor receive any messages. Finally, the node's wake-up pattern can be

completely arbitrary.

- Nodes do not feature a reliable *collision detection* mechanism. This assumption is often realistic, considering that nodes may be tiny sensors [1] with equipment restricted to the minimum due to limitations in energy consumption, weight, or cost. Moreover, the sending node itself does not have a collision detection mechanism either. Hence, a sender does not know how many (if any at all!) neighbors have received its transmission correctly.

- At the time of their waking-up, nodes have only limited knowledge about the total number of nodes in the network and no knowledge about the nodes' distribution or wake-up pattern. Particularly, they have no a-priori information about the number of neighbors and when waking up, they do not know how many neighbors have already started executing the algorithm.

Naturally, algorithms for such uninitialized, chaotic networks have a different flavor compared to "traditional" algorithms that operate on a given network graph that is static and well-known to all nodes.

In this paper, we show that even in this restricted model, a good vertex coloring can be computed efficiently. Specifically, we propose a randomized algorithm that computes a correct vertex coloring using $O(\Delta)$ colors in time $O(\Delta \log n)$ with high probability in any network graph as long as the maximal number of mutually independent nodes in the 2-hop neighborhood of any node is bounded by some arbitrary constant. This *bounded independence model* generalizes the frequently studied models for wireless networks, such as the unit disk graph. Unlike the unit disk graph or other explicit geometric graph models, however, our *bounded independence* model can capture obstacles as well as physical signal-propagation aspects such as fading, reflection, or shielding. Finally, our algorithm features the property that the highest color assigned to any node in a certain area of the network depends only on the *local density* of that area.

The remainder of the paper is organized as follows. An overview of related work is given in Section 2. Section 3 describes our model of computation and particularly the bounded independence model studied in this paper. The coloring algorithm is subsequently presented and analyzed in Sections 4 and 5. Finally, Section 6 concludes the paper.

2 Related Work

Coloring graphs belongs to the most fundamental NP-hard problems in theoretical computer science and has been thoroughly studied. In distributed computing, the study of vertex coloring has lead to several seminal contributions. Cole and Vishkin gave a deterministic distributed algorithm for computing a correct coloring on a ring using three colors in time $O(\log^* n)$ [3]. A generalization of the same technique can be used to color trees and arbitrary bounded-degree graphs with 3 and $\Delta + 1$ colors in time $O(\log^* n)$, respectively [8]. Recently, it has been shown in [17] that a running time of $O(\log^* n)$ also suffices to obtain a $\Delta + 1$ coloring in unit disk graphs and its generalization, the unit ball graph with constant doubling dimension.

All these upper bounds are tight due to the seminal lower bound by Linial [18], even for the case of randomized algorithms. For arbitrary graphs, a $\Delta + 1$-coloring can be computed in time $O(\log^* n + \Delta^2)$ [24] or $O(\Delta \log n)$ [7]. The authors of [9] present distributed approaches for finding colorings in graphs that admit a coloring with less than $\Delta + 1$ colors. Finally, an experimental study of various vertex coloring algorithms is given in [5].

All the above algorithms are based on a *message passing model* [25] that abstracts away problems such as interference, collisions, asynchronity, or the hidden-terminal problem, which are crucial in the context of wireless ad hoc and sensor networks. Specifically, it is assumed that nodes know their neighbors at the beginning of the algorithm and that the transmission of messages is handled flawlessly by an existing, underlying MAC layer. Furthermore, all nodes wake up synchronously and start the algorithm at the same time. As motivated in the introduction, these assumptions are invalid when studying multi-hop radio networks during or immediately after their deployment.

In view of its practical importance, it is not surprising that there has recently been a lot of effort in designing efficient algorithms for setting up initial structures, i.e., [10, 28, 19, 16, 4]. The *unstructured radio network* model was first proposed in [15] and subsequently improved and generalized in [16]. It is an adaptation of the classic *radio network model* (e.g., [2]), combining various of its flavors in order to model the harsh conditions during and immediately after the deployment. In [16], an algorithm is proposed that efficiently computes a *minimum dominating set* approximation from scratch. The paper [21] goes one step further by giving an algorithm for computing a *maximal independent set* in the unstructured radio network model in time $O(\log^2 n)$. Finally, notice that the recently proposed *weak sensor model* of [4] is essentially equivalent to the *unstructured radio network model*. The authors of [4] present an algorithm that, based on [21], computes a constant degree subgraph with low stretch.

In the preliminary version of this paper [20], we presented a randomized $O(\Delta \log n)$ time coloring algorithm for unit disk graphs using at most $O(\Delta)$ colors. In comparison to [20], we have generalized our result from unit disk graphs to bounded independence graphs and the algorithm and its analysis have been significantly revised.

Coloring networks for the purpose of obtaining a channel assignments or TDMA scheme has been studied in [14, 26], among others. Moreover, coloring a network in which all nodes are within mutual transmission range of all other nodes (*single-hop* networks) reduces to the so-called *initialization problem*. This problem has been feverously studied and analyzed during the past years [22, 23]. For several reasons, the approach taken in these papers cannot be translated into efficient algorithms for the unstructured radio network model. First and foremost, the multi-hop character of our network model complicates matters. In the single-hop case, if there is a collision, no node in the network receives a message, whereas in our multi-hop scenario, it is likely that some neighbors of the sender may receive the message, while others experience a collision and do not receive the message. This difference renders it impossible for nodes to keep a coherent picture of the local situation. Secondly, most initialization papers assume *strong communication* [12], that is, a sending node can distinguish whether its message was successfully received by all nodes or whether it has caused a collision. In a multi-hop scenario, this assumption makes little sense. Finally, unlike [22, 23], we consider asynchronous wake-up where nodes can wake up at any time.

There has also been work on models containing asynchronous wake-up. In the so-called *wake-up problem* [6, 13], the goal is to wake up all nodes in the graph as quickly as possible by sending them messages. The assumption made in these papers is that a node is *woken up* by an incoming message. While the algorithmic problems resulting from this assumption are very interesting, current sensor nodes do not have such an external wake-up capability.

3 Model and Notation

In the *unstructured radio network model* [15] (subsequently also called *weak sensor model* in [4]), we consider *multi-hop* radio networks *without collision detection*. That is, nodes are unable to distinguish between the situation in which two or more neighbors are sending and the situation in which no neighbor is sending. A node receives a message if and only if exactly one of its neighbors sends a message. Nodes may wake up *asynchronously* at any time.

Upon waking up, a node has no information as to whether it is the first to wake up, or whether other nodes have been running the algorithm for a long time already. We call a node *sleeping* before its wake-up, and *awake* thereafter. Only awake nodes can send or receive messages, and sleeping nodes are *not* woken up by incoming messages. The two extreme cases of our asynchronous wake-up model are the following. First, all nodes start synchronously at the same time, or only one of the sleeping nodes wakes up while all others remain sleeping for a long time. Recall again that nodes are unaware which (if any) of the two extreme cases holds. The *time complexity* T_v of a node v is defined as the number of time slots between the node's *waking up* and the time it has made its irrevocable *final decision* on its color. The algorithm's *time complexity* is the maximum number T_v over all nodes in the network.

We model the network as a graph $G = (V, E)$, where two nodes u and v can communicate with each other if there is an edge $(u, v) \in E$. In order to capture the typical wireless characteristics, ad hoc and sensor networks have often been modeled as unit disk graphs (UDG) in which nodes are located in the Euclidean plane and there is an edge between two nodes if their Euclidean distance is at most one. In this paper, we study a more general *bounded independence model* which not only generalizes the unit disk graph, but also many other known models for wireless networks such as the quasi unit disk graph, or general disk graphs.

Two nodes v_1 and v_2 are called *independent* if there exists no communication link between them. A set of nodes S is called an *independent set*, if all nodes in S are mutually independent. In the *bounded independence model*, there can be at most κ_1 mutually independent nodes in the 1-hop neighborhood of any node. Similarly, there are at most κ_2 nodes in the 2-hop neighborhood of any node. This more general model for wireless networks captures the intuitive notion that if many nodes of a wireless network are located close from each other, many of them must be within mutual transmission range. Unlike in the unit disk graph, however, obstacles or irregular signal propagation are easily captured in the bounded independence model. Specifically, an obstacle (such as a wall) in close physical proximity to a sending node destroys the disk shape of the node's transmission range. On the other hand, the maximal number of mutually independent nodes is still bounded by a (possibly somewhat larger) constant. Finally, notice that the unit disk graph is a special case of a bounded independence graph with $\kappa_1 = 5$ and $\kappa_2 \leq 18$.

Note that due to asynchronous wake-up, some nodes may still be asleep, while others are already transmitting. Hence, at any time, there may be sleeping nodes which do *not receive* a message in spite of there being a communication link between the two nodes. When waking up, nodes have only scarce knowledge about the network graph's topology. In particular, a node has no information on the number of nodes in its neighborhood. However, every node has estimates n and Δ for the number of nodes in the network and the maximum degree, respectively. In reality, it may not be possible to foresee these global parameters precisely by the time of the deployment, but it is usually possible to pre-estimate rough bounds.

For the sake of simplicity, we assume time to be divided into discrete time slots that are synchronized between all nodes. This assumption is used merely for the purpose of facilitating the analysis, i.e., our algorithm does not rely on this assumption in any way (see the standard argument given in [27]), as long as the nodes' internal clock runs at the same speed.

In each time slot, a node can either transmit or not transmit. If a node transmits in a time slot t, it cannot receive any messages in time slot t. A node v receives a message in a time slot t only if *exactly one node* in its neighborhood transmits a message in this time slot (and if it is not sending itself). The message size in our model is limited to $O(\log n)$ bits per message. Further, notice that in contrast to previous work on the unstructured radio network model [15, 16], we do not make the simplifying assumption of having several independent communication channels. In our model, there is only one communication channel.

Every node has a unique identifier, which does not need to be in the range $1, \ldots, n$. Particularly, the algorithm does not perform explicit operations on the node's IDs. Instead, the ID is merely required to let a receiver recognize whether or not two different messages were sent by the same sender. In some papers on wireless sensor networks, it is argued that sensor nodes do not feature any kind of unique identification (such as a MAC number, for instance). In such a case, each node can randomly choose an ID uniformly from the range $[1 \ldots n^3]$ upon waking up. The probability that two nodes in the system end up having the same ID is bounded by $P_{ambIDs} \leq \binom{n}{2} \frac{1}{n^3} \in O(\frac{1}{n})$.

We denote by $\mathcal{N}_v$ the set of neighbors of node v, including v itself. Further, $\mathcal{N}_v^2$ is the two-hop neighborhood of node v, i.e., the set of all nodes within distance at most 2 from v. The *degree* $\delta_v = |\mathcal{N}_v|$ of a node is the number of its neighbors. The color assigned to node v is denoted by color_v and p_v is v's sending probability in a given time slot. Finally, we use the following well-known mathematical fact.

Fact 3.1. *For all values of n and t with $n \geq 1$ and $|t| \leq n$, it holds that*

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n \leq e^t.$$

4 Algorithm

During the algorithm, each node can be in various *states*. At any point in time, a node is in exactly one state, i.e., the sets of nodes induced by the different states form a partition of V.

- $\mathcal{Z}$: Nodes before their waking up. Sleeping nodes do not take part in the algorithm.
- $\mathcal{A}_i$: Nodes that are verifying (i.e., trying to decide on) color i.
- $\mathcal{R}$: Nodes that are requesting a *intra-cluster color* from their leader.
- $\mathcal{C}_i$: Nodes that have already irrevocably decided on color i.

The state $\mathcal{C}_0$ plays a special role and nodes in state $\mathcal{C}_0$ are called *leaders*. The algorithm itself is divided into three subroutines: Algorithm 1 for nodes in states $\mathcal{A}_i$, Algorithm 2 for nodes in state $\mathcal{R}$, and Algorithm 3 for nodes in state $\mathcal{C}_i$. The sequence of states that a node can be part of during the course of the algorithm is shown in Figure 1. A solid arrow represents a state transition a node makes when the event denoted by the arrow's label occurs. A dashed arrow between two states indicates the message type which is significant for the communication between nodes in these two states. In our model, however, *every* neighbor of a sending node—regardless of their current state—may actually receive the message or experience a collision. Upon waking up, each node starts in state $\mathcal{A}_0$, without having any knowledge whether some of its neighbors have already started the algorithm beforehand.

From a global point of view, the algorithm's main idea can be described easily: In a first stage, the nodes elect a set of mutually independent *leaders* (nodes in state $\mathcal{C}_0$) among themselves and

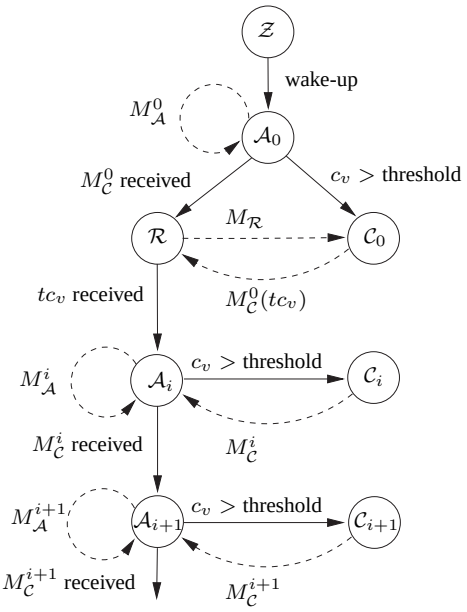

Figure 1: Sequence of states in the algorithm. Each color i is represented by a state $\mathcal{C}_i$, which a node enters at the moment it (irrevocably) selects color i. Before deciding on i, a node first has to *verify* (or compete for) i in state $\mathcal{A}_i$. If the node does not prevail in this verification, it moves from $\mathcal{A}_i$ to a new state $\mathcal{A}_{suc}$, which corresponds to either the intermediate requesting state $\mathcal{R}$ or the verification state of the next higher color $\mathcal{A}_{i+1}$ (cf Lines 3 and 15 of Algorithm 1).

each non-leader associates itself with a leader within its neighborhood. Since leaders are independent, they can safely assign themselves color 0. The set of leaders naturally induces *clusters* consisting of all nodes associated with the same leader. The task of each leader is to assign a unique *intra-cluster color* tc_v to every node v within its cluster. Unfortunately, the coloring induced by these intra-cluster colors may not form a valid coloring since two neighboring nodes in different clusters may be assigned the same intra-cluster color.

On the other hand, if the set of leaders is really independent, there can only be a small number of neighboring nodes with the same intra-cluster color. The coloring induced by these intra-cluster colors thus represents a first coarse structuring of the network that facilitates the subsequent task of actually assigning colors to nodes. Technically, upon receiving an intra-cluster color tc_v from its leader, a node goes on to verify a specific color $tc_v(\kappa_2 + 1)$ against neighboring nodes from different clusters that may have received the same intra-cluster color. This *verification procedure* must ensure that no two neighboring nodes end up selecting this specific color.

The algorithmic difficulty of the above process stems from the fact that nodes wake up asynchronously and do not have access to a global clock. Therefore, the different phases (verification, requesting intra-cluster color, etc...) of different nodes may be arbitrarily intertwined or shifted in time. While some nodes may still compete for becoming leader in state $\mathcal{A}_0$, their neighbors may already be much more advanced in their coloring process. Moreover, messages may be lost due to collisions at any time. In view of this harsh environment, the primary challenge is that the algorithm must achieve two contradictory aims: symmetries between nodes must be broken both

correctly and rapidly. That is, no two neighboring nodes ever select the same color and yet, every node can take its decision shortly after its wake-up (i.e., there is no starvation).

A crucial part of reconciling these contradictory aims takes place in the verification procedure. In order to ensure both correctness and fast progress in all parts of the network with high probability, our algorithm uses a technique of counters, critical ranges, and local competitor lists. Roughly, the idea is that every node v uses a local counter c_v which it increments in every time slot. Intuitively, this counter represents v's progress towards deciding on color i and v selects i as soon as c_v reaches a certain threshold.

In order to prevent two neighboring nodes from selecting the same color, the algorithm must make sure that as soon as a node v selects its color, all neighbors of v can be notified before their counter also reaches the threshold. In view of collisions and message losses being always possible, there must be a large enough time interval between two neighboring counters reaching the threshold. A simple idea to achieve this correctness condition is to have every node transmit its current counter with a certain sending probability. Whenever a node receives a message with higher counter, it resets its own counter. Unfortunately, this technique may lead to *chains of cascading resets*, i.e., a node's counter is reset by a more advanced node, which in turn is reset by another node and so forth. While, eventually, one node will end up selecting the color, this method does not prevent nodes from starving in certain (local) parts of the network graph.

Our algorithm therefore employs a more subtle handling of the counters. The general idea is that upon receiving a message from a neighbor, a node only resets its counter if it is within a *critical range* of the received counter. On the one hand, this critical range is large enough to ensure correctness with high probability. On the other hand, this technique allows for much more parallelism in the network because many nodes can simultaneously make progress towards deciding on the color. In order to truly avoid cascading resets and achieve the claimed running time, however, using only counters and critical ranges is insufficient. Specifically, nodes should also be prevented from resetting their counter to a value within the critical range of neighboring nodes and furthermore, all counters must remain relatively close to the verification threshold even after a reset. For this purpose, each node stores a local *competitor list* containing the current counter values of neighboring nodes.

Unfortunately, in the unstructured radio network model, it is impossible to constantly keep this competitor list and the corresponding locally stored counters complete and correctly updated. Interestingly, we can prove in Section 5 that in spite of this inevitable inconsistency, our technique of using counters and critical ranges in combination with storing local competitor lists avoids cascading resets and at the same time ensures the correctness condition. That is, whenever a node v selects a color, the counters of all neighboring competing nodes are far enough from the threshold so that v has enough time to inform its neighbors with high probability.

In more detail, the algorithm works as follows. Upon waking up, a node enters state $\mathcal{A}_0$ and tries to become a leader. Generally, whenever a node v enters a state $\mathcal{A}_i$, for $i \geq 0$, it first waits for $\lceil \alpha \Delta \log n \rceil$ time slots. As soon as it receives a messages $M_{\mathcal{C}}^i$ from a neighboring node that has already joined $\mathcal{C}_i$ (Line 7 of Algorithm 1), v joins the succeeding state $\mathcal{A}_{suc}$, which corresponds to $\mathcal{R}$ in the case $i = 0$, and $\mathcal{A}_{i+1}$, otherwise. If no such message is received, v becomes *active* and starts competing for color i (Line 10).

In order to ensure with high probability that no two neighbors enter the same state $\mathcal{C}_i$, the following process is employed: In each time slot, an active node $v \in \mathcal{A}_i$ increments its counter c_v and transmits a message $M_{\mathcal{A}}^i$ with probability $1/(\kappa_2 \Delta)$ (Lines 11 and 14, respectively). Whenever v receives a message $M_{\mathcal{C}}^i$ from a neighbor $w \in \mathcal{C}_i$, v knows that it cannot verify color i anymore and consequently moves on to state $\mathcal{A}_{suc}$.

Algorithm 1 Coloring Algorithm—Node v in state $\mathcal{A}_i$

upon entering state $\mathcal{A}_i$: (when waking up, a node is initially in state $\mathcal{A}_0$)

1: $P_v := \emptyset$; {* v is passive *}
2: $\zeta_i := \begin{cases} 1 & , i = 0 \\ \Delta & , i > 0 \end{cases}$
3: $\mathcal{A}_{suc} := \begin{cases} \mathcal{R} & , i = 0 \\ \mathcal{A}_{i+1} & , i > 0 \end{cases}$
4: **for** $\lceil \alpha \Delta \log n \rceil$ time slots **do**
5: **for each** $w \in P_v$ **do** $d_v(w) := d_v(w) + 1$;
6: **if** $M^i_{\mathcal{A}}(w, c_w)$ received **then** $P_v := P_v \cup \{w\}$; $d_v(w) := c_w$; **end if**
7: **if** $M^i_{\mathcal{C}}(w)$ received **then** state $:= \mathcal{A}_{suc}$; $L(v) := w$; **end if**
8: **end for**
9: $c_v := \chi(P_v)$, where $\chi(P_v)$ is the maximum value such that,
$\chi(P_v) \leq 0$ and $\chi(P_v) \notin [c_w - \lceil \gamma \zeta_i \log n \rceil, \ldots, c_w + \lceil \gamma \zeta_i \log n \rceil]$ for each $w \in P_v$;
10: **while** $state = \mathcal{A}_i$ **do** {* v is active *}
11: $c_v = c_v + 1$;
12: **for each** $w \in P_v$ **do** $d_v(w) := d_v(w) + 1$;
13: **if** $c_v \geq \lceil \sigma \Delta \log n \rceil$ **then** state $:= \mathcal{C}_i$; **end if**
14: **transmit** $M^i_{\mathcal{A}}(v, c_v)$ with probability $1/(\kappa_2 \Delta)$;
15: **if** $M^i_{\mathcal{C}}(w)$ received **then** state $:= \mathcal{A}_{suc}$; $L(v) := w$; **end if**
16: **if** $M^i_{\mathcal{A}}(w, c_w)$ received **then**
17: $P_v := P_v \cup \{w\}$; $d_v(w) := c_w$;
18: **if** $|c_v - c_w| \leq \lceil \gamma \zeta_i \log n \rceil$ **then** $c_v := \chi(P_v)$; **end if**
19: **end if**
20: **end while**

Algorithm 2 Coloring Algorithm—Node v in state $\mathcal{R}$

upon entering state $\mathcal{R}$:

1: **while** $state = \mathcal{R}$ **do** {* v is active *}
2: **transmit** $M_{\mathcal{R}}(v, L(v))$ with probability $1/(\kappa_2 \Delta)$;
3: **if** $M^0_{\mathcal{C}}(L(v), v, tc_v)$ received **then**
4: $state := \mathcal{A}_{tc_v \cdot (\kappa_2 + 1)}$;
5: **end if**
6: **end while**

When receiving a message $M^i_{\mathcal{A}}(w, c_w)$ from a neighboring competing node $w \in \mathcal{A}_i$, v adds neighbor w to its *competitor list* P_v and stores a *local copy* of w's counter c_w denoted by $d_v(w)$ (Line 17). In each subsequent time slot, these local copies $d_v(w)$ are incremented in order to keep track with the real current counter of w as much as possible. Moreover, in Line 18, v compares c_w to its own counter c_v. If the two counters are within the *critical range* $\lceil \gamma \zeta_i \log n \rceil$ of each other, v resets its own counter to $\chi(P_v)$. The value $\chi(P_v) < 0$ (defined in Line 9) is defined such that the new counter is not within the critical range $\lceil \gamma \zeta_i \log n \rceil$ of any locally stored copy of neighboring counters. Notice, however, that because counters may be reset in any time slot, a locally stored copy $d_v(w)$ of c_w may be outdated without v knowing it. For instance, if w has to reset its counter due to receipt of a message $M^i_{\mathcal{A}}(x, c_x)$ from a neighbor x, and if v does not receive this message (possibly due to a collision or because x and v are not neighbors), it subsequently

Algorithm 3 Coloring Algorithm—Node v in state $\mathcal{C}_i$

upon entering state $\mathcal{C}_i$:

1: $\text{color}_v := i$; {* v is active *}
2: **if** $i > 0$ **then**
3: **repeat forever transmit** $M^i_{\mathcal{C}}$ with probability $1/(\kappa_2\Delta)$;
4: **else if** $i = 0$ **then**
5: $tc := 0$;
6: $\mathcal{Q} := \emptyset$; {FIFO request queue }
7: **repeat forever**
8: **if** $M_{\mathcal{R}}(w, v)$ received **and** $w \notin \mathcal{Q}$ **then** add w to $\mathcal{Q}$; **end if**
9: **if** $\mathcal{Q}$ is empty **then**
10: **transmit** $M^0_{\mathcal{C}}(v)$ with probability $1/\kappa_2$;
11: **else**
12: $tc := tc + 1$;
13: Let w be first element in $\mathcal{Q}$;
14: **for** $\lceil \beta \log n \rceil$ time slots **do**
15: **transmit** $M^0_{\mathcal{C}}(v, w, tc)$ with probability $1/\kappa_2$;
16: **end for**
17: Remove w from $\mathcal{Q}$;
18: **end if**
19: **end repeat**
20: **end if**

holds $d_v(w) \neq c_w$. Hence, in spite of the definition of $\chi(P_v)$, a node's counter may be within the critical range of a neighboring counter after a reset.

If in the above process, a node succeeds in incrementing its counter up to the threshold of $\lceil \sigma\Delta \log n \rceil$ (Line 13), it decides on color i and joins state $\mathcal{C}_i$. As mentioned before, the technique of using counters and critical ranges guarantees that quick progress is made simultaneously in all parts of the network. Specifically, this method ensures that after a limited (constant) number of trials, at least one competing node in $\mathcal{A}_i$ can select $\mathcal{C}_i$ in *every region* of the graph. At the same time, the method also guarantees with high probability that no two neighboring nodes join $\mathcal{C}_i$, i.e., the set of nodes induces by $\mathcal{C}_i$ is independent.

A special role in the algorithm plays the state $\mathcal{C}_0$, the set of leaders. A leader's duty is to assign unique intra-cluster colors to each node in its cluster. Specifically, each non-leader v in $\mathcal{R}$ assigned to leader w sends requests $M_{\mathcal{R}}(v, w)$ for an intra-cluster color to w. Upon receiving such a request message from v, w transmits for $\lceil \beta \log n \rceil$ time slots with probability $1/\kappa_2$ a message $M^0_{\mathcal{C}}(w, v, tc)$, where tc denotes the intra-cluster color assigned to v and is incremented for each subsequent requesting node. If necessary, requests are buffered in an internal queue $\mathcal{Q}$, which helps in keeping all messages within the size of $O(\log n)$ bits.

In state $\mathcal{R}$, a non-leader node v requests an intra-cluster color from its leader. As soon as v receives a message $M^0_{\mathcal{C}}(w, v, tc_v)$ from leader w containing its intra-cluster color tc_v, v moves on to state $\mathcal{A}_{tc_v(\kappa_2+1)}$, i.e., it attempts to verify color $c = tc_v(\kappa_2 + 1)$ next. If verifying color c is unsuccessful (i.e., if a neighbor selects color c earlier), a node joins the next higher state $\mathcal{A}_{suc} = \mathcal{A}_{tc_v(\kappa_2+1)+1}$, and so forth, until it manages to verify and decide on a color. In Corollary 5.7 of Section 5, we show that every node is capable of deciding on a color from $tc_v(\kappa_2+1), tc_v(\kappa_2+1)+1, \ldots, tc_v(\kappa_2 + 1) + \kappa_2$ with high probability. Hence, the reason for a node to verify $\mathcal{A}_{tc_v(\kappa_2+1)}$

upon receiving tc_v is that by doing so, two nodes with different intra-cluster colors do never compete for the same color. This turns out to be an important ingredient when upper bounding the amount of time each node must maximally wait before deciding on its color.

The algorithm's four parameters, α, β, γ, and σ can be chosen as to trade-off the running time and the probability of correctness. The higher the parameters, the less likely the algorithm fails in producing a correct coloring. In order to obtain the high probability result in Section 5, the parameters are defined as $\alpha \geq 2\gamma\kappa_2 + \sigma + 1$, $\beta \geq \gamma$, and

$$\gamma = \frac{5\kappa_2}{\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}}, \quad \sigma = \frac{10e^2\kappa_2}{\left(1-\frac{1}{\kappa_2}\right)\left(1-\frac{1}{\kappa_2\Delta}\right)},$$

for $\Delta \geq 2$. Specifically, the constant are chosen as to guarantee a correct coloring and running time with probability at least $1-O(n^{-1})$. Simulation results show that in randomly distributed networks significantly smaller values suffice. In fact, the constants are small enough to yield a practically efficient coloring algorithm for wireless ad hoc and sensor networks that can be employed for the purpose of initializing the network.

5 Analysis

In this section, we prove that the algorithm of Section 4 is both correct and complete with high probability. *Correctness* means that no two adjacent nodes end up having the same color, *completeness* leaves no node without a color. Furthermore, we show that every node decides on a color after at most $O(\Delta \log n)$ time slots for constant κ_2. For clarity of exposition, we will omit ceiling signs in our analysis, i.e, we consider all non-integer values to be implicitly rounded to the next higher integer value. Further, let $c_v(t)$ be the value of the counter of node v at time t. We call a node in $\mathcal{A}_i$ *covered* if either itself or one of its neighbors is in $\mathcal{C}_i$.

For future reference, we begin with simple lemma that bounds the maximum number of nodes in the 2-hop neighborhood of any node.

Lemma 5.1. *Let $G = (V, E)$ be a graph with at most κ_2 independent nodes in the 2-hop neighborhood of any node. It follows that every node has at most $\kappa_2\Delta$ 2-hop neighbors.*

Proof. Every node has at most κ_2 mutually independent nodes in its 2-hop neighborhood, and each such node has at most Δ neighbors. □

We now state two lemmas that give us probabilistic bounds on the amount of time required until a message is correctly transmitted from a sender v to an intended receiver u in the algorithm. Notice that both lemmas holds only under the assumption that the set $\mathcal{C}_0$ of leaders forms a correct independent set.

Lemma 5.2. *Assume $\mathcal{C}_0$ forms an independent set. Consider two neighboring nodes u and v and let I be a time interval of length $\gamma\Delta\log n$. If v is active throughout the interval I, u receives at least one message from v during I with probability $1 - n^{-5}$.*

Proof. Let p_v denote the transmission probability of v. Recall that nodes in C_0 transmit with a probability of $1/\kappa_2$, whereas the sending probability of all other nodes is $1/(\kappa_2\Delta)$. The probability

P_s that v succeeds in sending a message to u in a time slot $t \in I$ is

$$\begin{aligned} P_s &= p_v \prod_{i \in \mathcal{N}_u \setminus \{v\}} (1-p_i) = p_v \prod_{i \in \mathcal{N}_u \cap \mathcal{C}_0} (1-p_i) \prod_{j \in \mathcal{N}_u \setminus \mathcal{C}_0} (1-p_j) \\ &\geq p_v \left(1-\frac{1}{\kappa_2}\right)^{\kappa_1} \left(1-\frac{1}{\kappa_2 \Delta}\right)^{\Delta} > p_v \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}} \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}, \quad (1) \end{aligned}$$

where the last inequality follows from Fact 3.1 and $\kappa_2 \geq \kappa_1$. Because v is assumed to be active throughout the interval I and for every active node $p_v \geq 1/(\kappa_2\Delta)$, the probability P_{no} that u does not receive a message from v during I is

$$\begin{aligned} P_{no} &= (1-P_s)^{|I|} < \left(1 - \frac{1}{\kappa_2\Delta}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}} \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}\right)^{\gamma\Delta\log n} \\ &\underset{\text{Fact 3.1}}{\leq} n^{-\frac{\gamma}{\kappa_2}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\kappa_1/\kappa_2}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{1/\kappa_2}} < n^{-5}, \end{aligned}$$

where the last inequality follows from the definition of γ. □

Lemma 5.3. *Assume $\mathcal{C}_0$ forms an independent set. Consider two neighboring nodes u and $v \in C_0$ and let I' be a time interval of length $\gamma \log n$. If $v \in C_0$ throughout the interval I', u receives at least one message from v during I' with probability $1 - n^{-5}$.*

Proof. The proof is virtually identical to the previous one. In the case $v \in \mathcal{C}_0$, it holds that $p_v = 1/\kappa_2$ and plugging this value into Inequality (1) and applying Fact 3.1 yields

$$\begin{aligned} P_{no} &= (1-P_s)^{|I'|} < \left(1 - \frac{1}{\kappa_2}\left[\frac{1}{e}\left(1-\frac{1}{\kappa_2}\right)\right]^{\frac{\kappa_1}{\kappa_2}} \left[\frac{1}{e}\left(1-\frac{1}{\kappa_2\Delta}\right)\right]^{\frac{1}{\kappa_2}}\right)^{\gamma\log n} \\ &\underset{\text{Fact 3.1}}{\leq} n^{-5}. \end{aligned}$$

□

For the next lemma, we first define the notion of a *successful transmission.* A node v transmits *successfully* in a time slot t if *all* nodes $u \in \mathcal{N}_v \setminus \{v\}$ within the transmission range of v (i.e., in v's 1-hop neighborhood) receive the message without collision. In the following lemma, we show that with high probability, at least one node in v's neighborhood can transmit *successfully* during any interval of length $O(\kappa_2\Delta\log n)$.

Lemma 5.4. *Assume $\mathcal{C}_0$ forms an independent set. Consider a node $v \in \mathcal{A}_i$ for an arbitrary i. Further, let I be a time interval of length $|I| = \frac{\sigma}{2}\Delta\log n$ during which $v \in \mathcal{A}_i$ is active. With probability $1 - n^{-5}$, there is at least one time slot $t \in I$ such that a node $u \in \mathcal{N}_v \cap \mathcal{A}_i$ transmits* successfully.

Proof. By Lemma 5.1, there are at most $\kappa_2\Delta$ nodes in the 2-neighborhood of any node. If in a time slot, a node w is the only transmitting node in $\mathcal{N}_w^2$, it is guaranteed that w transmits *successfully* because no node outside $\mathcal{N}_w^2$ can cause a collision at a neighbor of w. Define P_s to be the

probability that a node $w \in \mathcal{N}_v \cap \mathcal{A}_i$ transmits successfully in a given time slot $t \in I$. By the above argument, P_s is lower bounded by

$$\begin{aligned} P_s &\geq \sum_{w \in \mathcal{N}_v \cap \mathcal{A}_i} \left(p_w \cdot \prod_{\substack{u \in \mathcal{N}_w^2 \\ u \neq w}} (1 - p_u) \right) \\ &\geq \sum_{w \in \mathcal{N}_v \cap \mathcal{A}_i} p_w \cdot \prod_{u \in \mathcal{N}_w^2 \setminus \mathcal{C}_0} \left(1 - \frac{1}{\kappa_2 \Delta}\right) \cdot \prod_{u \in \mathcal{N}_w^2 \cap \mathcal{C}_0} \left(1 - \frac{1}{\kappa_2}\right) \\ &\geq \sum_{w \in \mathcal{N}_v \cap \mathcal{A}_i} p_w \cdot \left(1 - \frac{1}{\kappa_2 \Delta}\right)^{\kappa_2 \Delta} \left(1 - \frac{1}{\kappa_2}\right)^{\kappa_2} \underset{\text{Fact 3.1}}{\geq} \frac{1}{e^2 \kappa_2 \Delta} \left(1 - \frac{1}{\kappa_2 \Delta}\right) \left(1 - \frac{1}{\kappa_2}\right) \end{aligned}$$

because $\sum_{w \in \mathcal{N}_v \cap \mathcal{A}_i} p_w$ is at least $1/(\kappa_2 \Delta)$ for as long as v is active in $\mathcal{A}_i$. Finally, the probability P_{no} that no node in $\mathcal{N}_v \cap \mathcal{A}_i$ manages to transmit successfully within the interval I during which v is active in $\mathcal{A}_i$ is

$$\begin{aligned} P_{no} &= (1 - P_s)^{|I|} \leq \left(1 - \frac{1}{e^2 \kappa_2 \Delta} \left(1 - \frac{1}{\kappa_2 \Delta}\right) \left(1 - \frac{1}{\kappa_2}\right)\right)^{\frac{\sigma}{2} \Delta \log n} \\ &\underset{\text{Fact 3.1}}{\leq} e^{-\frac{\sigma}{2e^2 \kappa_2} \frac{\kappa_2 - 1}{\kappa_2} \frac{\kappa_2 \Delta - 1}{\kappa_2 \Delta} \log n} < n^{-5}. \end{aligned}$$

The last step follows from the definition of σ. □

Lemmas 5.2, 5.3, and 5.4 are based on the assumption that the set of leaders $\mathcal{C}_0$ forms an independent set. Therefore, in order to make full use of these lemmas, we need to prove that this assumption holds for the entire duration of the algorithm. Intuitively, the reason for our claim is the following. By the definition of the algorithm, only nodes in state $\mathcal{A}_0$ can enter state $\mathcal{C}_0$. If such a candidate node $v \in \mathcal{A}_0$ transmits *successfully*, all neighboring nodes $w \in \mathcal{N}_v \cap \mathcal{A}_0$ having a counter value within the *critical range* $\gamma\zeta_0 \log n = \gamma \log n$ of v's counter will reset their counter to $\chi(P_w)$, which is by definition outside the critical range of v. Hence, once node v was able to transmit successfully, no neighboring candidate node $w \in \mathcal{N}_v \cap \mathcal{A}_0$ can block v from incessantly incrementing its counter until it reaches the threshold $\sigma\Delta \log n$ which enables to join $\mathcal{C}_0$. The only way v can still be prevented from becoming a leader is if v receives a message $M_{\mathcal{C}}^0$ from a neighbor that has entered $\mathcal{C}_0$ before v's counter reaches the threshold. Moreover, the counters of neighboring nodes being outside the critical range, it can be shown that upon becoming leader, v has enough time to inform all neighbors of its having joined $\mathcal{C}_0$.

We formalize this intuition in Theorem 5.5 and its subsequent proof. More precisely, the following theorem proves the more general statement that every color class $\mathcal{C}_i$ (i.e., not merely $\mathcal{C}_0$) forms an independent set at all times during the algorithm's execution with high probability. Notice that the theorem establishes the algorithm's *correctness*, because if all color classes form independent sets, the resulting coloring is necessarily correct.

Theorem 5.5. *For all i, the color class $\mathcal{C}_i$ forms an independent set throughout the execution of the algorithm with probability at least $1 - 2n^{-3}$.*

Proof. At the beginning, when the first node wakes up, the claim certainly holds, because $\mathcal{C}_i = \emptyset$ for all i. We will now show that with high probability the claim continues to hold throughout the algorithm's execution. For this purpose, consider an arbitrary node $v \in \mathcal{A}_i$ and assume for

contradiction that v is the *first node* to violate the independence of $\mathcal{C}_i$ for an arbitrary $i \geq 0$. That is, we assume that v is the first node to enter $\mathcal{C}_i$ even though a neighboring node w has entered $\mathcal{C}_i$ in the same or a previous time slot. Note that if two or more nodes violate the independence of $\mathcal{C}_i$ simultaneously, we consider each of them to be the *first node*. We prove in the sequel that the probability of v being such a *first node* for a specific $\mathcal{C}_i$ is at most $2n^{-5}$. Applying the union bound, we conclude that the probability that there exists a node $v \in V$ that violates the independence of some $\mathcal{C}_i$ is bounded by $n^2 \cdot 2n^{-5} = 2n^{-3}$.

Let t_v^* be the time slot in which v enters state $\mathcal{C}_i$, $i \geq 0$. Since v is among the *first* nodes to violate the independence of any $\mathcal{C}_i$, and hence also $\mathcal{C}_0$, we know that for all time slots $t < t_v^*$, $\mathcal{C}_0$ is a correct independent set. That is, if v is among the first nodes to create a violation, Lemmas 5.2, 5.3, and 5.4 can be applied until time slot $t_v^* - 1$.

Let w be a neighbor of v that has joined $\mathcal{C}_i$ before v (or in the same time slot as v), say at time $t_w^* \leq t_v^*$. We consider two cases, $t_w^* < t_v^* - \gamma\zeta_i \log n$ and $t_w^* \geq t_v^* - \gamma\zeta_i \log n$, and start with the former.

If $t_w^* < t_v^* - \gamma\zeta_i \log n$, then w entered state $\mathcal{C}_i$ at least $\gamma\zeta_i \log n$ time slots before v. By Lemma 5.2 ($i > 0$) or Lemma 5.3 ($i = 0$), the probability that w manages to successfully send a message $M_{\mathcal{C}}^i$ to v during these $\gamma\zeta_i \log n$ time slots (during which v must be in $\mathcal{A}_i$ if it joins $\mathcal{C}_i$ at time t_v^*) is at least $1 - n^{-5}$. By Line 15 of Algorithm 1, however, v leaves state $\mathcal{A}_i$ and moves on to state $\mathcal{A}_{suc}$ upon receiving $M_{\mathcal{C}}^i$, i.e., it does not enter $\mathcal{C}_i$.

For the second case, we compute the probability that v joins $\mathcal{C}_i$ within $\gamma\zeta_i \log n$ time slots after t_w^*. Recall that by the definition of the algorithm, it holds that $c_w = \sigma\Delta \log n$ at time t_w^*. Consider the time interval I_w of length $\gamma\Delta \log n$ before t_w^*. Because in each time slot, counters of nodes in $\mathcal{A}_i$ are either incremented by one or set to $\chi(P_v) \leq 0$ and because $\sigma\Delta \log n > 2\gamma\Delta \log n$, it follows that c_w was not reset during I_w. If it was, c_w would not have reached $\sigma\Delta \log n$ by time t_w^*. Similarly, if c_v was reset during I_w, t_v^* could not be within $\gamma\zeta_i \log n$ of t_w^*. Hence, neither c_w nor c_v were reset during the interval I_w and it holds that at time t_w^*, $c_v \geq \sigma\Delta \log n - \gamma\zeta_i \log n$. More generally, it holds that

$$|c_w(t_w^* - h) - c_v(t_w^* - h)| \;\leq\; \gamma\zeta_i \log n$$

for each $h = 0, \ldots, \gamma\Delta \log n - 1$. By Lemma 5.2, the probability that v receives at least one message $M_{\mathcal{A}}^i$ from w during these $\gamma\Delta \log n$ time slots in I_w is at least $1 - n^{-5}$. If it does receive such a $M_{\mathcal{A}}^i$, v resets its counter (Line 18) and does not enter $\mathcal{C}_i$ within $\gamma\zeta_i \log n$ time slots of t_w^*.

Combining both cases, we know that with probability $1 - n^{-5}$, v does not enter $\mathcal{C}_i$ until $\gamma\zeta_i \log n$ time slots after its first neighbor has joined $\mathcal{C}_i$. And with probability $1 - n^{-5}$, v does not enter $\mathcal{C}_i$ thereafter. Consequently, the probability of v being a first node to violate the independence of a specific $\mathcal{C}_i$ is at most $2n^{-5}$. Each state $\mathcal{C}_i$ thus remains independent throughout the algorithm's execution with probability at least $1 - 2n^{-4}$. Finally, we can crudely upper bound the number of non-empty states $\mathcal{C}_i$ used in the algorithm by n, because in Lines 7 and 15, a node changes its state only if it has received a message $M_{\mathcal{C}}^i$ from a node that has already decided on $\mathcal{C}_i$. The probability that all color classes form independent sets at all times is at least $1 - 2n^{-3}$. □

Theorem 5.5 proves that with high probability, all color classes are independent and hence, the algorithm eventually produces a correct coloring. Particularly, notice that the theorem implies that the set of leaders $\mathcal{C}_0$ forms an independent set with high probability and hence, we can use Lemmas 5.2, 5.3, and 5.4 without restriction. What remains to be shown are the bounds on the running time as well as on the number of colors required. For this purpose, we first prove a helper lemma that bounds the number of nodes v that can simultaneously be in the same active state $\mathcal{A}_i$.

Lemma 5.6. *If the set of nodes in $\mathcal{C}_0$ is independent, then for any $i > 0$, the number of nodes in any 1-hop neighborhood that ever enter state $\mathcal{A}_i$ is at most κ_2.*

Proof. A node v enters a state $\mathcal{A}_i$, $i > 0$, for the first time when being in state $\mathcal{R}$ and receiving a message $M^0_{\mathcal{C}}(w, v, tc_v)$ from its leader $w = L(v)$. Consider a leader $w \in \mathcal{C}_0$ and let S_w denote the set of nodes having w as their leader, i.e., $S_w = \{v \mid L(v) = w\}$. Since the value tc is incremented for every new node v in the queue $\mathcal{Q}$, w assigns to each $v \in S_w$ a unique *intra-cluster color* tc_v. While being unique within each cluster, these intra-cluster colors do not constitute a legal coloring, because neighboring nodes belonging to different clusters may be assigned the same intra-cluster color tc_v by their respective leaders. If the set of leaders $w \in \mathcal{C}_0$ forms an independent set, the maximum number of leaders $w \in \mathcal{C}_0$ in any 2-hop neighborhood is κ_2. Therefore, every node can have at most κ_2 1-hop neighbors (including itself!) with the same intra-cluster color tc_v.

In Line 4 of Algorithm 2, a node v with tc_v enters state $\mathcal{A}_{tc_v(\kappa_2+1)}$. That is, two nodes with subsequent intra-cluster colors tc_v and $tc_v + 1$ enter states $\mathcal{A}_i$ and $\mathcal{A}_j$, where $|i - j| = \kappa_2 + 1$. By the definition of Algorithm 1, the only way a node can move from state $\mathcal{A}_i$ to state $\mathcal{A}_{i+1}$ is by receiving a message $M^i_{\mathcal{C}}$ from a neighboring node that has already entered $\mathcal{C}_i$. Without receiving $M^i_{\mathcal{C}}$ a node will eventually join state $\mathcal{C}_i$ itself. Hence, whenever a node v in $\mathcal{A}_i$ moves on to state $\mathcal{A}_{i+1}$, at least one of its neighbors must have joined $\mathcal{C}_i$. From this, it follows that each of the at most κ_2 neighbors of a node v that are assigned the same intra-cluster color tc_v will decide on a color in the range $tc_v(\kappa_2 + 1), \ldots, tc_v(\kappa_2 + 1) + \kappa_2$. Notice that this range does not overlap with the corresponding range of the next higher intra-cluster color which starts with color $(tc_v + 1)(\kappa_2 + 1) > tc_v(\kappa_2 + 1) + \kappa_2$. Consequently, nodes assigned to different intra-cluster colors are never in the same state $\mathcal{A}_i$ for any $i > 0$. And because at most κ_2 nodes are assigned the same tc_v in the 1-hop neighborhood of any node v, the lemma follows. □

The proof of Lemma 5.6 implicitly gives raise to the following corollary.

Corollary 5.7. *While executing the algorithm, every node v is at most in $\kappa_2 + 1$ different states $\mathcal{A}_i$, namely $\mathcal{A}_0, \mathcal{A}_{tc_v(\kappa_2+1)}, \ldots, \mathcal{A}_{tc_v(\kappa_2+1)+\kappa_2}$. This holds under the condition that the nodes in $\mathcal{C}_0$ are independent.*

The next lemma gives a lower bound on the counters c_v of any node $v \in \mathcal{A}_i$.

Lemma 5.8. *Let c_v be the counter of node $v \in \mathcal{A}_i$. It holds throughout the execution of the algorithm that $c_v \geq -2\gamma\Delta \log n - 1$, if $i = 0$, and $c_v \geq -2\kappa_2\gamma\Delta \log n - 1$, otherwise. This holds under the condition that the nodes in $\mathcal{C}_0$ are independent.*

Proof. Consider a node $v \in \mathcal{A}_i$. The only time v's counter c_v is set to a negative value is when (re)setting c_v to $\chi(P_v)$ in Lines 9 or 18 of Algorithm 1. $\chi(P_v)$ is defined as the largest value such that $\chi(P_v) < 0$ and $\chi(P_v) \notin [c_u - \gamma\zeta_i \log n, \ldots, c_u + \gamma\zeta_i \log n]$ for each $u \in P_v$. Because the set P_v contains only nodes that are also in state $\mathcal{A}_i$, it follows from Lemma 5.6 that $|P_v| \leq \kappa_2$ for any $i > 0$, if the nodes in $\mathcal{C}_0$ form an independent set. In the case $i = 0$, it trivially holds that $|P_v| \leq \Delta$.

The number of values that are prohibited for $\chi(P_v)$ is therefore at most $\kappa_2 \cdot 2\gamma\zeta_i \log n$ in the case $i > 0$ and $\Delta \cdot 2\gamma\zeta_0 \log n$ if $i = 0$. Plugging in the values for ζ_i, we can write

$$\chi(P_v) \;\geq\; \begin{cases} -2\gamma\Delta \log n - 1 & , i = 0 \\ -2\kappa_2\gamma\Delta \log n - 1 & , i > 0 \end{cases},$$

which concludes the proof. □

Having the last two helper lemmas, we are now ready to analyze the algorithm's running time, that is, to bound the maximum amount of time between a node's waking up and its entering a color class $\mathcal{C}_i$. We first obtain a bound on the amount of progress achieved by nodes in a state $\mathcal{A}_i$ in every part of the graph.

Lemma 5.9. *Let T_v^i denote the number of time slots a node v spends in state $\mathcal{A}_i$. With probability $1 - 3n^{-3}$, it holds for all v and i that $T_v^i \in O(\kappa^3 \Delta \log n)$.*

Proof. By Lemma 5.5, we know that with probability $1 - 2n^{-3}$, the set of nodes in state $\mathcal{C}_0$ form an independent set. In the sequel of the proof, we focus on this case and assume that all nodes in $\mathcal{C}_0$ are mutually independent.

Let t_v denote the time slot in which node $v \in \mathcal{A}_i$ executes Line 9 of Algorithm 1. Until t_v, v spends exactly $\alpha\Delta \log n$ time slots in $\mathcal{A}_i$. By Lemma 5.4, we know that at least one node $w \in \mathcal{N}_v \cap \mathcal{A}_i$ is able to transmit successfully during the interval $I = [t_v, t_v + \frac{\sigma}{2}\Delta \log n]$ with probability $1 - n^{-5}$ (unless v leaves state $\mathcal{A}_i$ during that interval in which case Lemma 5.9 clearly holds). Say this happens at time t_w^s. According to Lines 6 and 17 of Algorithm 1, all nodes $u \in \mathcal{N}_w \cap \mathcal{A}_i$ store a local copy $d_u(w)$ of w's current counter c_w upon receiving w's message $M_{\mathcal{A}}^i$ in time slot t_w^s. In Lines 5 and 12, this local copy is incremented by one in each subsequent time slot. That is, as long as w's real counter is not reset to $\chi(P_w)$, every node $u \in \mathcal{N}_w \cap \mathcal{A}_i$ has a correct local copy $d_u(w)$ of w's current counter c_w.

We now show that w's counter c_w cannot be reset by any node $u \in \mathcal{N}_w \cap \mathcal{A}_i$ after t_w^s anymore. First, in Line 18, every node $u \in \mathcal{N}_w \cap \mathcal{A}_i$ whose counter $c_u(t_w^s)$ at time t_w^s is in the range

$$[c_w(t_w^s) - \gamma\zeta_i \log n, \ldots, c_w(t_w^s) + \gamma\zeta_i \log n]$$

resets its own counter to $\chi(P_u)$ in time slot t_w^s. Recall that $\chi(P_u)$ is defined as the maximum value such that $\chi(P_u) \leq 0$ and $\chi(P_u) \notin [c_x - \gamma\zeta_i \log n, \ldots, c_x + \gamma\zeta_i \log n]$ for each $x \in P_u$. Specifically, because w transmited successfully, this means that $\chi(P_u) \notin [c_w - \gamma\zeta_i \log n, \ldots, c_w + \gamma\zeta_i \log n]$, and hence $|c_u(t_w^s + 1) - c_w(t_w^s + 1)| > \gamma\zeta_i \log n$. Clearly, the same inequality also holds for all nodes $u \in \mathcal{N}_w \cap \mathcal{A}_i$ whose counter was not in the critical range $[c_w(t_w^s) - \gamma\zeta_i \log n, \ldots, c_w(t_w^s) + \gamma\zeta_i \log n]$ in the first place.

In summary, we have that in time slot $t_w^s + 1$, every node $u \in \mathcal{N}_w \cap \mathcal{A}_i$ has a correct local copy $d_u(w)$ of c_w, and

$$|c_u(t_w^s + 1) - c_w(t_w^s + 1)| > \gamma\zeta_i \log n.$$

Because the counter of every neighbor in $\mathcal{A}_i$ thus differs by at least $\gamma\zeta_i \log n$ from c_w, none of these nodes can cause w to reset its counter in Line 18 of the algorithm. Node w can thus increment its counter in each time slot and hence, all nodes $u \in \mathcal{N}_w \cap \mathcal{A}_i$ continue to have a correct local copy of c_w after t_w^s. Consequently, even if a neighboring node u has to reset its counter to $\chi(P_u)$, this cannot cause c_u to come within $\gamma\zeta_i \log n$ of c_w by the definition of $\chi(P_u)$. Thus, it follows by induction over the subsequent time slots that no node $u \in \mathcal{A}_i$ is able to reset w's counter after its successful transmission at time t_w^s. By Lemma 5.8, we know that for all i, $c_w \geq -2\gamma\kappa_2\Delta \log n - 1$ at time t_w^s. Hence, if w stays in $\mathcal{A}_i$, it requires at most $(2\gamma\kappa_2 + \sigma)\Delta \log n + 1$ time slots in order to reach the threshold $\sigma\Delta \log n$, which enables to enter state $\mathcal{C}_i$. Also, nodes that join $\mathcal{A}_i$ *after* t_w^s do not transmit for at least $\alpha\Delta \log n$ time slots, and because $\alpha > 2\gamma\kappa_2 + \sigma + 1$, it follows that such nodes cannot interfere with w's incrementing its counter either. Hence, after a successful transmission, there remains only one way to prevent w from incessantly incrementing its counter and entering $\mathcal{C}_i$: if w receives a message $M_{\mathcal{C}}^i$ *before* its counter reaches $\sigma\Delta \log n$.

In summary, we have that after a successful transmission, either w enters $\mathcal{C}_i$ itself within $(2\gamma\kappa_2 + \sigma)\Delta \log n + 1$ time slots or there must exist a neighboring node x of w that joins $\mathcal{C}_i$

earlier (see Figure 2). In the first case, v receives a message $M_{\mathcal{C}}^i$ from w within $\gamma\zeta_i \log n$ after w's entering $\mathcal{C}_i$ with probability at least $1 - n^{-5}$ (by Lemma 5.2 if $i > 0$ and by Lemma 5.3 if $i = 0$). In the other case, the node x (which, in this case, is not a direct neighbor of v) must be a 2-hop neighbor of v. If v is not covered by x and remains in $\mathcal{A}_i$, at least one node $w_2 \in \mathcal{N}_v \cap \mathcal{A}_i$ can transmit successfully within $\frac{\sigma}{2}\Delta \log n$ time slots thereafter with high probability (Lemma 5.4), and the argument repeats itself. That is, as long as v is active in $\mathcal{A}_i$, at least one node in v's 2-hop neighborhood enters $\mathcal{C}_i$ per $\frac{\sigma}{2}\Delta \log n + (2\gamma\kappa_2 + \sigma)\Delta \log n + 1$ time slots with probability $1 - n^{-5}$.

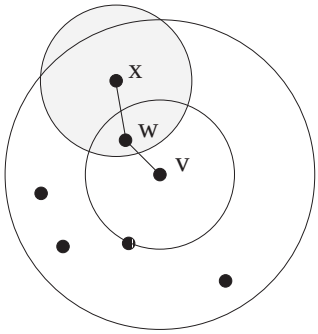

Figure 2: When v is active, a neighbor w can transmit successfully within $\frac{\sigma}{2}\Delta \log n$ time slots. This node w can only be blocked from entering $\mathcal{C}_i$ if one of its neighbors x joins $\mathcal{C}_i$ earlier.

As shown in Figure 2, the number of times a node $x \in \mathcal{N}_v^2$ can join $\mathcal{C}_i$ without covering v (and thus forcing v to leave state $\mathcal{A}_i$) is by definition at most κ_2. Finally, once v becomes covered, an additional $\gamma\zeta_i \log n$ time slots in $\mathcal{A}_i$ may be required before, with probability $1 - n^{-5}$, its first neighbor in $\mathcal{C}_i$ sends a message $M_{\mathcal{C}}^i$ to v. As stated at the beginning of the proof, our argument holds under the condition that the set of leaders $\mathcal{C}_0$ forms an independent set which is true with probability $1 - 2n^{-3}$ by Theorem 5.5. Therefore, with probability P_v, node v spends at most

$$T_v^i \;\leq\; \alpha\Delta \log n + \kappa_2 \left(\frac{\sigma}{2}\Delta \log n + (2\gamma\kappa_2 + \sigma)\Delta \log n + 1\right) + \gamma\zeta_i \log n \;\in\; O(\kappa_2^3 \Delta \log n)$$

time slots in state $\mathcal{A}_i$, where P_v is at least

$$P_v \;\geq\; 1 - (\kappa_2 \cdot n^{-5} + n^{-5} + 2n^{-3}) \;>\; 1 - 3n^{-3},$$

for large enough n because $\kappa_2 \leq n$ and $\gamma \in O(\kappa_2)$. This concludes the proof. □

Next, we bound the time until a node v in the request state $\mathcal{R}$ receives its intra-cluster color (via a message $M_{\mathcal{C}}^0(w, v, tc_v)$) from its leader w upon which it leaves state $\mathcal{R}$ (cf Line 4 of Algorithm 2). Specifically, the following lemma shows that each node v spends at most time $O(\kappa_2 \Delta \log n)$ in state $\mathcal{R}$.

Lemma 5.10. *Let $T_v^{\mathcal{R}}$ denote the number of time slots a node v spends in state $\mathcal{R}$. With probability $1 - 3n^{-3}$ it holds for each $v \in V$ that $T_v^{\mathcal{R}} \leq (\gamma + \beta)\Delta \log n$.*

Proof. The time $T_v^{\mathcal{R}}$ denotes the time between v starting to request an intra-cluster color from its leader $L(v) \in \mathcal{C}_0$ to the time this leader succeeds in assigning the intra-cluster color tc_v to v without collision. Let w be the leader of v, i.e., $w = L(v)$. We divide $T_v^{\mathcal{R}}$ into two parts. First, by

Lemma 5.2, v is able to send its request $M_{\mathcal{R}}(v, L(v))$ to w within time $\gamma\Delta \log n$ with probability $1 - n^{-5}$. Upon receipt, w queues v's request until it has served all its other, previously received requests. In Line 15 of Algorithm 3, w transmits a message $M_{\mathcal{C}}^0$ with probability $1/\kappa_2$ to the currently considered requesting node for $\beta \log n$ time slots, before moving on to the next request, if available. Because $\beta \geq \gamma$, Lemma 5.3 holds for w's response to v with probability $1 - n^{-5}$. Because w can have at most Δ requesting nodes in its queue, $T_v^{\mathcal{R}}$ is at most

$$T_v^{\mathcal{R}} \ \leq \ \gamma\Delta \log n + \Delta \cdot \beta \log n = (\gamma + \beta)\Delta \log n$$

for each node $v \in V$ with probability at least $1 - 2n^{-5}$. As the set $\mathcal{C}_0$ forms an independent set with probability $1 - 2n^{-3}$, for large enough n the claim holds with probability $1 - 2n^{-5} - 2n^{-3} \geq 1 - 3n^{-3}$. □

Lemmas 5.9 and 5.10 are the ingredients required to prove the following theorem that bounds the algorithm's running time, i.e., the amount of time every node requires after its wake-up before deciding on a color.

Theorem 5.11. *Every node decides on its color within time $O(\kappa_2^4 \Delta \log n)$ after its wake-up with probability $1 - 3n^{-1}$.*

Proof. Let $T_v^{\mathcal{Y}}$ be the number of time slots a node v spends in state $\mathcal{Y}$. For each node v, we have

$$T_v \ = \ \sum_{i \geq 0} T_v^{\mathcal{A}_i} + T_v^{\mathcal{R}}.$$

Lemma 5.10 bounds $T_v^{\mathcal{R}}$ by $(\gamma + \beta)\Delta \log n$ with probability $1 - 3n^{-3}$ for each v, and thus with probability $1 - 3n^{-2}$ for all nodes in V. Moreover, when applying the union bound to the result of Lemma 5.9, it follows that $T_v^{\mathcal{A}_i} \in O(\kappa_2^3 \Delta \log n)$ for all v and i with probability $1 - 3n^{-1}$. Finally, because every node is in at most $\kappa_2 + 1$ different states (due to Corollary 5.7) $\mathcal{A}_i$, it follows that for some constant λ,

$$T_v \ = \ (\kappa_2 + 1) \cdot \lambda\kappa_2^3 \Delta \log n + (\gamma + \beta)\Delta \log n \ \in \ O(\kappa_2^4 \Delta \log n)$$

with probability at least $1 - 4n^{-1}$, for large enough n. □

The only thing remaining is a bound on the number of different colors assigned by the algorithm. For practical purposes, the *locality* of the assignment of colors to nodes plays a crucial role. Generally, the colors assigned to each node should be as "low" as possible. If the vertex coloring in the graph is used for setting up a *time-division scheduling* in a wireless network, for instance, the bandwidth assigned to a node v is often inversely proportional to the value of the *highest color* in its neighborhood. The highest color assigned to a neighbor of a node v by the algorithm in Section 4 is dependent only on *local graph properties*. This allows nodes located in low density areas of the network to send more frequently than nodes in dense and congested parts.

Theorem 5.12. *Let $\theta_v := \max_{w \in \mathcal{N}_v^2} \delta_w$ be the maximum node degree in $\mathcal{N}_v^2$ and let ϕ_v be the highest color assigned to a node in $\mathcal{N}_v$. With probability at least $1 - 2n^{-3}$ the algorithm produces a coloring such that, for all $v \in V$, $\phi_v \leq \kappa_2 \cdot \theta_v$.*

Proof. Let $w \in \mathcal{C}_0$ be a leader and let s_w be the number of nodes $v \in \mathcal{N}_w$ having w as their leader. Leader w assigns unique intra-cluster colors $1, 2, \ldots, s_w$ to these nodes. As shown in Corollary 5.7, if the set of leaders forms a correct independent set, a non-leader node v assigned intra-cluster color tc_v ends up selecting a color from the range $tc_v(\kappa_2 + 1), \ldots, tc_v(\kappa_2 + 1) + \kappa_2$. Since $s_w \leq \delta_w$ and every node $u \in \mathcal{N}_v$ is assigned to a leader $w \in \mathcal{N}_v^2$, the theorem follows. □

The following theorem combines the results obtained in Theorems 5.5, 5.11, and 5.12.

Theorem 5.13. *The algorithm produces a correct coloring with at most $\kappa_2\Delta$ colors with probability $1 - 2n^{-3}$. Furthermore, with probability $1 - 4n^{-1}$ every node irrevocably decides on its color $O(\kappa^4\Delta \log n)$ time slots after its wake up.*

Theorem 5.13 gives raise to a number of specific results for graphs that have been frequently studied in the literature on ad hoc and sensor networks. The most frequently adopted model has been the unit disk graph in which nodes are assumed to be located in the Euclidean plane and there is a communication link between two nodes iff their mutual distance is at most 1. In unit disk graphs as well as any other family of graphs in which $\kappa_2 \in O(1)$, we have the following result.

Corollary 5.14 (Unit Disk Graph). *Let $G = (V, E)$ be a unit disk graph. With high probability, the algorithm produces a correct coloring with $O(\Delta)$ colors and every node decides on its color within $O(\Delta \log n)$ time slots after its wake up.*

In [17], the unit disk graph model has been extended to general metric spaces resulting in so-called *unit ball graphs* (UBG). The nodes of a UBG are the points of a (possibly non-Euclidean) metric space; two nodes are connected if and only if their distance is at most 1. Using this definition, we can formulate a result on coloring in general network graphs that depends on the *doubling dimension* of the underlying metric. A metric's doubling dimension is the smallest ρ such that every ball of radius d can be covered by at most 2^ρ balls of radius $d/2$.

Lemma 5.15. *Let G be a unit ball graph and let ρ be the doubling dimension of the underlying metric space. Every 2-hop neighborhood in G contains at most 4^ρ mutually independent nodes, i.e., $\kappa_2 \leq 4^\rho$.*

Proof. By the definition of a UBG, the 2-hop neighborhood of node v in G is completely covered by the ball $B_2(v)$ with radius 2 around v. By the definition of the doubling dimension ρ, $B_2(v)$ can be covered by at most $2^{2\rho}$ balls of radius $1/2$. By the triangle inequality, two nodes inside a ball of radius $1/2$ have distance at most 1, that is, the nodes inside a ball of radius $1/2$ form a clique in G. The number of independent nodes in the 2-hop neighborhood of v is therefore at most 4^ρ. □

Plugging in the result of Lemma 5.15, we obtain the following result for coloring in the unstructured radio network model of general graphs.

Corollary 5.16 (Unit Ball Graphs). *Let $G = (V, E)$ be a unit ball graph and let ρ be the doubling dimension of the underlying metric space. With high probability, the algorithm produces a correct coloring with $O(4^\rho\Delta)$ colors and every node decides on its color within $O(4^{4\rho}\Delta \log n)$ time slots after its wake-up. For metrics with constant doubling dimension, the same asymptotic bounds as in the unit disk graph are achieved.*

6 Conclusions

Setting up an initial structure in newly deployed ad hoc and sensor networks is a challenging task that is of great practical importance. In this paper, we have given a randomized algorithm that computes an initial coloring from scratch. This is a step towards the ultimate goal of establishing an efficient medium access control (MAC) scheme.

A direction for future research is to address the issue that our algorithm is based on the assumption that nodes know an estimate of n and Δ. In single-hop radio networks with synchronous wake-up, there are efficient methods enabling nodes to approximately count the number of their neighbors, e.g. [11]. If such techniques could be adapted to an asynchronous multi-hop scenario, nodes might be able to estimate the local maximum degree, which could then be used instead of Δ throughout the algorithm.

7 Acknowledgements

We are indebted to Maurice Herlihy, Lucia Draque Penso, and Dieter Rautenbach for valuable comments on the initial version of this paper.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks Journal*, 38(4):393–422, 2002.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the Time-Complexity of Broadcast in Radio Networks: an Exponential Gap between Determinism and Randomization. In *Proceedings 6^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 98–108, 1987.

[3] R. Cole and U. Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Inf. Control*, 70(1):32–53, 1986.

[4] M. Farach-Colton, R. Fernandes, and M. Mosteiro. Bootstrapping a Hop-Optimal Network in the Weak Sensor Model. In *Proceedings of 13^{th} Annual European Symposium on Algorithms (ESA)*, pages 827–838, 2005.

[5] I. Finocchi, A. Panconesi, and R. Silvestri. Experimental Analysis of Simple, Distributed Vertex Coloring Algorithms. In *Proceedings of the 13^{th} ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 606–615, 2002.

[6] L. Gasieniec, A. Pelc, and D. Peleg. The Wakeup Problem in Synchronous Broadcast Systems (Extended Abstract). In *Proceedings of the 19^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 113–121, 2000.

[7] A. Goldberg, S. Plotkin, and G. Shannon. Parallel symmetry-breaking in sparse graphs. In *Proceedings of the 19^{th} Annual ACM Conference on Theory of Computing (STOC)*, pages 315–324, 1987.

[8] A. V. Goldberg and S. A. Plotkin. Parallel $(\Delta + 1)$-Coloring of Constant-degree Graphs. *Information Processing Letters*, 25:241–245, 1987.

[9] D. A. Grable and A. Panconesi. Fast Distributed Algorithms for Brooks-Vizing Colourings. In *Proceedings of the 9^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 473–480. Society for Industrial and Applied Mathematics, 1998.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33^{rd} Hawaii International Conference on System Sciences (HICSS)*, page 8020. IEEE Computer Society, 2000.

[11] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Energy-Efficient Size Approximation of Radio Networks with No Collision Detection. In *Proceedings of the 8^{th} Annual International Conference on Computing and Combinatorics (COCOON)*, pages 279–289, 2002.

[12] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Weak Communication in Radio Networks. In *Proceedings of Euro-Par*, pages 397–408, 2002.

[13] T. Jurdzinski and G. Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *Proceedings of 13^{th} Annual International Symposium on Algorithms and Computation (ISAAC)*, volume 2518 of *Lecture Notes in Computer Science*, pages 535–549, 2002.

[14] S. O. Krumke, M. V. Marathe, and S. S. Ravi. Models and Approximation Algorithms for Channel Assignment in Radio Networks. *Wireless Networks*, 7(6):575–584, 2001.

[15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Radio Network Clustering from Scratch. In *Proceedings of 12^{th} Annual European Symposium on Algorithms (ESA)*, pages 460–472.

[16] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing Newly Deployed Ad Hoc and Sensor Networks. In *Proceedings of 10^{th} Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 260–274, 2004.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In *Proc. of the 23^{rd} ACM Symposium on Principles of Distributed Computing (PODC)*, 2005.

[18] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[19] M. J. McGlynn and S. A. Borbash. Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks. In *Proceedings of the 2^{nd} ACM International Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 137–145. ACM Press, 2001.

[20] T. Moscibroda and R. Wattenhofer. Coloring Unstructured Radio Networks. In *Proceedings of the 17^{th} ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2005.

[21] T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. In *Proc. of the 23^{rd} ACM Symp. on Principles of Distributed Computing (PODC)*, 2005.

[22] K. Nakano and S. Olariu. Energy-Efficient Initialization Protocols for Single-Hop Radio Networks with No Collision Detection. *IEEE Trans. Parallel Distributed Systems*, 11(8):851–863, 2000.

[23] K. Nakano and S. Olariu. Randomized Initialization Protocols for Radio Networks. pages 195–218, 2002.

[24] A. Panconesi and R. Rizzi. Some Simple Distributed Algorithms for Sparse Networks. *Distributed Computing*, 14(2):97–100, 2001.

[25] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[26] S. Ramanathan and E. L. Lloyd. Scheduling Algorithms for Multi-Hhop Radio Networks. In *Conference Proceedings on Communications Architectures & Protocols (SIGCOMM)*, pages 211–222. ACM Press, 1992.

[27] F. A. Tobagi and L. Kleinrock. Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution. COM-23(12):1417–1433, 1975.

[28] A. Woo and D. E. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proceedings of the 7^{th} International Conference on Mobile Computing and Networking (MOBICOM)*, pages 221–235. ACM Press, 2001.