# Coloring Unstructured Wireless Multi-Hop Networks

Johannes Schneider
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
jschneid@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

## Abstract

We present a randomized coloring algorithm for the *unstructured radio network model*, a model comprising autonomous nodes, asynchronous wake-up, no collision detection and an unknown but geometric network topology. The current state-of-the-art coloring algorithm needs with high probability $O(\Delta \cdot \log n)$ time and uses $O(\Delta)$ colors, where $n$ and $\Delta$ are the number of nodes in the network and the maximum degree, respectively; this algorithm requires knowledge of a linear bound on $n$ and $\Delta$. We improve this result in three ways: Firstly, we improve the time complexity, instead of the logarithmic *factor* we just need a polylogarithmic *additive term*; more specifically, our time complexity is $O(\Delta + \log \Delta \cdot \log n)$ given an estimate of $n$ and $\Delta$, and $O(\Delta + \log^2 n)$ without knowledge of $\Delta$. Secondly, our vertex coloring algorithm needs $\Delta + 1$ colors only. Thirdly, our algorithm manages to do a distance-$d$ coloring with asymptotically optimal $O(\Delta)$ colors for a constant $d$.

**Categories and Subject Descriptors**
F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems – *computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory –*graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory – *network problems*

**General Terms**
Algorithms, Theory
**Keywords**
Ad Hoc Networks, Sensor Networks, Unstructured Radio Networks, Unit Disk Graphs, Growth Bounded Graphs, Bounded Independence Graphs, Local Algorithms, Parallel Algorithms, Distributed Algorithms, Coloring

## 1. INTRODUCTION

Wireless networks have become omnipresent; there are different architectures available, e.g. GSM-driven mobile phones, or laptops equipped with 802.11 WLAN. A challenging network architectures are so-called wireless multi-hop networks, e.g., ad hoc, sensor, or mesh networks. In these networks, the participating nodes themselves provide the infrastructure. This lack of already available infrastructure expresses itself primarily during the start-up phase of the wireless multi-hop network, as nodes need to establish their own infrastructure before being able to communicate. For instance, upon deployment the nodes act independently, as they are unaware of each other. In this stage communication is unreliable and slow. A major problem is that messages might be lost due to collisions, i.e. if nodes $u$ and $w$ concurrently transmit a message, a node $v$ being in the transmission range of both $u$ and $w$ might not receive any message correctly, or $v$ might not even detect that there was a transmission. Coordinating the nodes in a distributed manner to achieve efficient and reliable communication is a time consuming task – in particular in multi-hop networks, where the exchange of messages among nodes requires the help of intermediate nodes.

Once a wireless network is fully operational, it runs a reliable Media Access Control (MAC) scheme, for instance Time Division Multiple Access (TDMA). In TDMA, time is slotted, and individual time slots are assigned to different nodes, such that potentially interfering nodes do not share a common time slot. Algorithmically speaking this boils down to a coloring problem: Assign colors to nodes such that "close-by" nodes do not share the same color. Depending on the application the term "close-by" may have a different meaning: direct neighbors, or also two-hop neighbors (since two-hop neighbors $u$ and $w$ share a common neighbor $v$ that will be affected by a concurrent transmission of $u$ and $w$), or more generally $k$-hop neighbors for some constant $k$.[1] To achieve a high throughput in the wireless network, we must aim for a coloring that minimizes the number of colors. As the number of colors makes a significant impact in practice, constants do matter.

The current state of the art by Moscibroda et al. [13] needs with high probability $O(\Delta \cdot \log n)$ time and uses $O(\Delta)$ colors in a unit disk graph, where $n$ and $\Delta$ are the number of nodes in the network and the maximum degree, respectively; the algorithm requires a linear bound on $n$ and $\Delta$. Similarly to the journal version of [13] we generalize the network topology from unit disk graph to bounded-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*PODC'09,* August 10–12, 2009, Calgary, Alberta, Canada.
Copyright 2009 ACM 978-1-60558-396-9/09/08 ...$5.00.

[1]Strictly speaking even that is a simplification because physical radio signals do not comply to graph-based models; however, in practice graph-based models are used, and do usually work alright.

independence graph (sometimes also called growth-bounded graphs[2]) [9]. Bounded-independence graphs restrict for any node the maximum size of a set of independent nodes in its neighborhood. In a unit disk graph nodes are points in the plane, with two nodes $u$, $v$ being neighbors if and only if their Euclidean distance is at most 1. We improve the result of [13] in three ways:

1. We reduce the time complexity, instead of the logarithmic *factor* we just need a polylogarithmic *additive term*; more specifically, our time complexity is $O(\Delta + \log\Delta \cdot \log n)$ given an estimate of $n$ and $\Delta$, and $O(\Delta + \log^2 n)$ without any knowledge of $\Delta$.

2. For a simple (one-hop) coloring, our algorithm needs $\Delta + 1$ colors only. As in prior work, the number of actual colors used depends only on the local density of the nodes.

3. Our algorithm can be adapted to compute a distance-$d$ coloring with $O(\Delta)$ colors for any constant $d$.

On the road to our main results we also encounter a special type of broadcasting problem, which we call the *distance-d broadcasting problem*: A message should be delivered from an originator to all nodes within some (hop) distance $d$ but (ideally) to no node further away. We present a solution that is time-optimal for constant distances such that only nodes at distance $d + 1$ might receive the message.

## 2. RELATED WORK

Coloring a graph with as few colors as possible is a difficult problem – even deriving an approximate solution is NP-hard [7]. In contrast, computing a $(\Delta + 1)$-coloring is straightforward using a greedy algorithm as long as one has full knowledge of the graph. Deriving a $(\Delta + 1)$-coloring in a distributed manner is more challenging, even in the LOCAL model, where each node can concurrently send a (distinct) message to each neighbor and also concurrently receive a message from each neighbor without facing a collision. For instance, for arbitrary graphs, a $(\Delta + 1)$-coloring can be computed deterministically in time $\mathrm{O}(\Delta + \log^* n)$ [3], in time $\mathrm{O}(\Delta \cdot \log n)$ [15], or in time $\mathrm{O}(\log n)$ [12]. Distributed coloring has also been studied in special graph classes. In bounded-independence graphs, for instance, coloring only needs time $\Theta(\log^* n)$ [16], deterministically and without knowledge of the network topology.

However, these fast algorithms are not applicable in our setting, as they presume an established and powerful communication framework. In this paper we deal with unstructured radio networks (interfering transmissions, asynchronous wake-up) [10]. In this harsher model algorithms without at least a little topology information, i.e. the number of nodes $n$, are inherently slow [6]. Efficient deterministic algorithms are often hard or impossible to design. For instance, for the broadcasting problem there is an exponential gap in the time complexity between any deterministic and randomized algorithm [8]. Still, an algorithm achieving an $\mathrm{O}(\Delta)$-coloring in time $\mathrm{O}(\Delta \cdot \log n)$ with probability $1 - \frac{1}{n}$ for unit disk graphs was presented by Moscibroda et al. [13]. In that algorithm knowledge about the network topology is limited to a bound on $n$ and $\Delta$. The algorithm computes a set of leaders which grant a color range for all other nodes (so called slaves). This general idea has also been followed previously, e.g. [5], however in a simpler model allowing for powerful communication.[3] Likewise, distributed coloring for TDMA has been studied several times, however, always in different settings (single-hop network, stronger communication models, etc.); for a detailed treatment we refer to the discussion of related work in [13].

There is also an ample body of work on asynchronous wake-up. In the so-called wake-up problem [6], the goal is to wake up all nodes in the graph as quickly as possible. The assumption is that a node is woken up by an incoming message. While the algorithmic problems resulting from this assumption are interesting, multi-hop wake-up radios are currently technically infeasible.

Finally, there is a connection of our work to broadcasting in radio networks. In [2] a broadcasting scheme was proposed performing a broadcast in time $O(D \cdot \log n + \log^2 n)$. Essentially each node broadcasts $O(\log n)$ times with the same probability $\frac{1}{2^i}$ for $1 \leq i \leq \log n$, starting with probability $\frac{1}{2}$, heading towards $\frac{1}{n}$, and then restarting again with $\frac{1}{2}$. This broadcasting algorithm was improved to $\Theta(D \cdot \log \frac{n}{D} + \log^2 n)$ in [4], which is optimal due to lower bounds in [1, 11]. The key idea is to choose high sending probabilities more often. Both algorithms rely on a global clock. Furthermore, these algorithms are not straightforward to adjust such that they solve the *distance-d* broadcasting problem, where a message should be broadcast to all nodes up to hop distance $d$ but not further. For instance, just adding a time-to-live variable to a message does not solve the problem. Our broadcasting algorithm in Section 4 uses the same transmission probabilities as [2, 4], but is independent of a global clock.

## 3. MODEL AND DEFINITIONS

We assume that nodes have clocks running at the same speed.[4] We assume that time is divided into discrete synchronized time intervals (also called time slots); this is not a restriction [17]. The time complexity of an algorithm corresponds to the number of intervals needed to finish its task. However, we do not assume the existence of a global clock, i.e. each node $v$ might have a different value $t_v$ for its clock. Upon wake-up $t_v := 1$, $t_v$ increases by 1 with every passed time slot.

The communication network is modeled with a graph $G = (V, E)$. For a node $v$ its neighborhood $N^d(v)$ represents all nodes within $d$ hops of $v$ (not including $v$ itself). A message transmitted by node $v$ might only be received by its direct neighbors $u \in N(v)$. Nodes do not feature a collision detection mechanism. More specifically, if nodes $u$ and $w$ concurrently transmit a message, a node $v \in N(u) \cap N(w)$ suffers from a collision and will either (i) receive nothing, (ii) $u$'s message, (iii) $v$'s message or (iv) detect a collision. The actual outcome is determined by an adversary, i.e. for each collision the outcome is chosen to maximize the time

[2]In some publications growth-bounded refers to the growth of the number of neighbors and in other publications it denotes the growth of the number of (maximum) possible independent nodes.

[3]Specifically: Synchronous start, and each node can sense the presence of a signal. If the medium is free, it has a constant chance of transmitting without collision.

[4]A TDMA schedule only makes sense given reasonably synchronized clocks since otherwise no node is able to follow the schedule for long.

complexity of the algorithm. Thus to be sure that a node $v$ correctly receives a message, there must be exactly one sender $u \in N(v)$. For a distance-$d$ coloring, it holds that no two nodes within (hop) distance $d$ have the same color or equivalently, for each node $v$ every node $u \in N^d(v)$ has a different color than $v$. A coloring simply refers to a distance-1 coloring. A set $S^d \subseteq V$ is a maximal independent set (MIS) of distance $d$, if any two nodes $u, v \in S^d$ have hop distance more than $d$ and for every node $v \in V$ there exists a node $u \in N^d(v) \cap S^d$. A MIS $S^d$ of maximum cardinality, i.e. $|S^d| \geq \max_{\text{MIS } T^d} |T^d|$, is called a maximum independent set of distance $d$ (MaxIS). We consider bounded-independence (also known as growth-bounded) graphs:

DEFINITION 1. *A graph $G = (V, E)$ is of bounded-independence if there is a polynomial bounding function $f(r)$ such that for each node $v \in V$, the size of a MaxIS in the neighborhood $N^r(v)$ is at most $f(r)$, $\forall r \geq 0$.*

In particular, this means that for a constant $c$ the value $f(c)$ is also a constant. A subclass of bounded-independence graphs are (quasi) unit disk graphs, which are often used to model wireless communication networks and have $f(r) \in O(r^2)$.

We assume that all nodes are sleeping initially. A sleeping node can neither transmit nor receive any messages. Nodes wake up asynchronously at any time. The wake-up of a node is not triggered by incoming messages but is assumed to occur spontaneously at an unpredictable point in time.

A node does not have any topology information except for a polynomial bound of the number of nodes $n$ in the network.[5] Without an estimate of $n$, every algorithm requires at least time $\Omega(\frac{n}{\log n})$ until one single message can be transmitted without collision [6].

The value $\log^* n$ describes how often one has to take the logarithm of an initial value $n$ to end up with at most 2.

Every node $v$ has an $ID_v \in \{0, 1, ..., n\}$.[6] For convenience, if the meaning is clear in the context, we use $v$ instead of $ID_v$.

# 4. ALGORITHM BROADCAST

Algorithm Broadcast is an essential part of the coloring algorithm given in the next section. It performs a broadcast of the content $msg_w$ created by node $w$, which is delivered to all nodes $u \in N^h(w)$ up to distance $h$ with probability at least $1 - \epsilon$. The message might also be received by (some) nodes $u \in N^{h+1}(w) \setminus N^h(w)$ at distance $h + 1$, but not by nodes at distance more than $h+1$. The algorithm still works correctly if up to $5 \cdot f(h)$ (see Definition 1) nodes $u \in N^h(w)$ perform a broadcast concurrently.

Nodes forward a message in a synchronized manner, meaning that all nodes having received a message by $w$, transmit (sufficiently often) with the same probabilities. To achieve a synchronized behavior of the broadcasting nodes, each forwarding node $u$ determines the same schedule $r_w$ for a message originator $w$.[7] The value $r_w[i]$ denotes the transmission probability of the $i^{th}$ time-slot in the schedule. Only a fraction $\frac{1}{c}$ with $c := 5 \cdot f(h)$ of all transmission probabilities are larger than 0. More precisely, for every interval $[c \cdot i, c \cdot (i + 1) - 1]$ of length $c$, one time slot with non-zero probability is chosen uniformly at random. For illustration consider the following example with $c = 2$: $[0, > 0, 0, > 0, > 0, 0, 0, > 0, ...]$.[8] The non-zero probabilities themselves can be chosen such as in [2] or [4]. For our purposes it suffices to stick to the simpler sequence [2], which starts with probability $\frac{1}{2}$, then $\frac{1}{4}$, followed by $\frac{1}{8}$ a.s.o. until $\frac{1}{\Delta}$.[9] Then the sequence repeats starting with probability $\frac{1}{2}$. An example of a schedule with $\Delta = 8$ and $c = 2$ is $[0, \frac{1}{2}, 0, \frac{1}{4}, \frac{1}{8}, 0, 0, \frac{1}{2}, 0, \frac{1}{4}, ...]$. For a schedule $r_w$ the sequence $\frac{1}{2}, \frac{1}{4}, .., \frac{1}{\Delta}$ is repeated $e^7 \cdot f(h) \cdot h \cdot \log \frac{1}{\epsilon}$ times. Since only one slot out of $c = 5 \cdot f(h)$ slots is non-zero the length $|r_w|$ of the schedule $r_w$ becomes $e^7 \cdot 5 \cdot f(h)^2 \cdot h \cdot \log \Delta \cdot \log \frac{1}{\epsilon}$, where e denotes the Euler constant.

A node $u$ having received a message originated at $w$ must transmit according to the schedule $r_w$. In our coloring algorithm (Section 5) a node $u$ might have to forward several messages originating from up to $5 \cdot f(h)$ different nodes $w \in N^h(u)$. Therefore it is supposed to transmit according to multiple schedules at the same time. Unfortunately, it might be that two (or more) schedules overlap, i.e. there is a time slot where at least two schedules have non-zero transmission probability. In such a case a node transmits according to an arbitrary non-zero transmission probability. Furthermore, if a node $u$ received a message from node $w$ then it cannot just start transmitting according to the schedule upon reception of a message. Due to collisions neighbors of $w$ might receive the message at different times and thus the schedule would not be followed synchronously. Thus, a node $v$ broadcasting according to $r_w$ appends the current position within the schedule, i.e. if it transmits with probability $r_w[i]$, it appends $i$. A node receiving a message waits until the sender completed the schedule (i.e. for $|r_w| - i$ time slots) before starting to transmit according to schedule $r_w$. Summing up, an actual message used in the protocol consists of three elements: a message $msg$, a time to live $h$ stating how many hops a message will still be forwarded and finally the current slot in $r$. A node $v$ keeps track of all messages it still has to broadcast using the set $M_v$.

# 5. ALGORITHM FASTCOLORING

We present the main ideas of our coloring approach in Section 5.1 assuming synchronous wake-up and a given estimate of the maximal degree $\Delta$. This simplifies the algorithm, allows to focus on the main ideas, and permits to directly relate our time complexity to previous work. In Section 5.2 we remedy these assumptions. Finally, in Section 5.3 we show how to get a distance-$d$ coloring.

## 5.1 Synchronous wake-up, $\Delta$ known

Algorithm FastColoring roughly iterates two main steps. First, a set of (independent) leaders is elected that control all

[5]The bound can be different for each node.

[6]The same asymptotic time complexity holds also for much larger intervals, e.g. for $ID_v \in \{0, 1, ..., n^n\}$ since $\log^*(n^n) = O(\log^* n)$. Thus the IDs might as well be randomly chosen out of an interval $[0, n^n]$ without altering the time complexity of the algorithm.

[7]It would also be possible to let $w$ determine the schedule $r_w$ and append it to the message. Furthermore, the schedule could be changed for every broadcast initiated by $w$.

[8]Since the schedule $r_w$ must be the same for all nodes, each node creating $r_w$ must make the same random choices. From a practical point of view, this could be achieved by initializing a pseudo random number generator with the ID of $w$.

[9]In case $\Delta$ is not known, it is approximated using $n$.

**Algorithm Broadcast(hops $h$, message $msg_w$, failure probability $\epsilon$)**

**For each** node $v \in V$

1: $|r| := \mathrm{e}^7 \cdot 5 \cdot f(h)^2 \cdot h \cdot \log \Delta \cdot \log \frac{1}{\epsilon}$
2: **if** $v = w$ **then** $M_v := \{(msg_w, h+1, 0, Started)\}$ **else** $M_v := \{\}$ **end if**
3: **for** $i = 0..h \cdot |r|$ **do**
4: $m_C := (msg_u, h-1, t)$ for some entry $(msg_u, h, t, Started) \in M_v$ with $t < |r_u|$ and transmission probability $> 0$ (i.e. $r_u[t] > 0$)
5: **if** $m_C \neq \{\}$ **then**
6: Transmit $m_C$ with probability $r_u[t]$
7: **else**
8: Listen for message $m_L$ of the form $(msg_u, h, t)$
9: **if** (received $m_L$) $\wedge$ ($\nexists (msg_u, ., .) \in M_v$) $\wedge$ ($h > 0$) **then**
10: $M_v := M_v \cup (msg_u, h, t, NotStarted)$
11: **end if**
12: **end if**
13: Replace all entries $(msg_u, ., t, NotStarted) \in M_v$ with $t = |r_u|$ by $(msg_u, ., 0, Started)$ and all others by $(msg_u, ., t+1, Started)$
14: **end for**

other nodes within distance 6. Second, these leaders assign colors to their (direct) neighbors.

In order to get the set of leaders $S^6$ we first compute a MIS $S^1$ using algorithm [14] and then extend it using [16]. After the MIS $S^1$ is computed, all nodes are synchronized, i.e. nodes in the MIS wait until all nodes within 6 hops have completed algorithm [14] and tell (by broadcasting message $InMIS$) all nodes $N^6(v) \setminus S^1$ to stop and wait until the computation of $S^6$ is over. Next, a node $v \in S^1$ gets to know all nodes $u \in N^6(v) \cap S^1 \subseteq IS^6(v)$ within 6 hops in the independent set $S^1$. The algorithm presented in [16] requires $IS^6(v)$ and computes $S^6$. All leaders in $S^6$ ensure that they are not disturbed during their color assignment by broadcasting messages $DoNotTransmit$ and $DoNotTryMIS$. Then a leader $v \in S^6$ and its neighbors $u \in N(v)$ repeatedly execute 3 synchronized time slots. In the first slot every uncolored node $u$ might apply for the permission to choose a color. If leader $v$ received a request, it grants the request. Node $u$ chooses an available color $c$. In the third slot node $u$ informs its neighbors that color $c$ is used.

Let us go through the algorithm in more detail. The broadcast of message $InMIS$ following the computation of MIS $S^1$ serves the purpose of telling nodes $v \notin S$ that the computation of the MIS $S^1$ is over and that they must wait before (re)starting algorithm TryMIS until all nodes in $S^1$ have finished the computation of $S^6$ (Lines 21 and 22). Additionally, using the $InMIS$ messages a node $v$ gets to know all neighbors $u \in (N^6(v) \cap S)$, i.e. set $IS^6(v)$, (Lines 3 to 6) that joined $S^1$ in parallel with $v$, i.e. before having received a message $InMIS$. For communication nodes in the MIS $S^1$ use algorithm Broadcast (Section 4). Unfortunately, if a node $u$ received a message $msg_v$ broadcast by $v$, then $v$ might not receive a message $msg_u$ even if both broadcast up to the same hop distance $h$, i.e. call Broadcast(.,$h$,.,.). Thus per se the set $IS^6(v)$ is not symmetric, i.e. if $u \in IS^6(v)$ then not necessarily $v \in IS^6(u)$. However, symmetry is required in [16] to get set $S^6$ and therefore guaranteed (with high probability) in lines 7 to 8. The set $S^6$ is computed on the (undirected) graph $G' = (V' = \{u | u \in S\} = S, E' = \{\{u, v\} | u, v \in V', v \in IS^6(u)\})$ using algorithm [16]. The algorithm presented in [16] assumes that a node $v \in V'$ knows all nodes $u \in V'$ with $\{v, u\} \in E'$, i.e. $v$ knows the set $IS^6(v)$ and that communication among nodes is reliable, i.e. no message is lost. Therefore, in case a node $v$ misses some information of a node $u \in IS^6(v)$, e.g. due to a collision, it continuously sends a request to $u$ asking for a retransmission. Furthermore, communication between two nodes $u, v \in V'$ is not just a simple transmission between direct neighbors in $G$, since their hop distance in $G$ is up to 7. Thus we use Algorithm Broadcast with parameters $h = 7$ and $\epsilon = \frac{1}{\sqrt{2}}$. Once a node starts executing algorithm [16], it will execute it for a fixed number of steps, i.e. $c_5 \cdot \log \Delta \cdot \log n$ for some constant $c_5$. With high probability the computation of $S^6$ has finished by then. In case a node terminates the algorithm prior to the allowed number of steps it simple waits for the remaining time slots.

In the second step every node $v \in S^6$ assigns colors to all uncolored neighbors $u \in N(v)$. To begin with, nodes $v \in S^6$ ensure that all nodes within 3 hops remain quiet, while they are busy assigning colors. A node receiving a notification $DoNotTransmit(v)$ forwards the message $DoNotTransmit(v)$ and then must not transmit until it received a message $TransmitAndTryMIS(v)$. All messages $InMIS$ or $MIS6$ are ignored by a node while it is not allowed to transmit, in particular it will also not forward them once it is allowed to transmit again. Any interrupted broadcast of a message $DoNotTryMIS$ or $TransmitAndTryMIS$ originated at some node $w$ is restarted from scratch, i.e. the schedule $r_w$ is executed from slot 0 onwards such that the forwarding nodes still transmit synchronized. More precisely, assume a node $u$ received message $DoNotTransmit(v)$ at time $t_u^{DoNot}$, when executing slot $j$ of $r_w$ and $TransmitAndTryMIS(v)$ at time $t_u^{Can}$. Node $u$ will transmit again at time $(t_u^{DoNot} - j) + i \cdot |r_w|$, such that $i$ is the smallest positive integer with $(t_u^{DoNot} - j) + i \cdot |r_w| > t_u^{Can}$. If a node $u$ received a broadcast message while not being able to transmit and was supposed to start the broadcast at time $t_u$, it will start it at time $t_u + i \cdot |r_w|$ such that $i$ is the smallest positive integer with $t_u + i \cdot |r_w| > t_u^{Can}$. The message $DoNotTryMIS(v)$ ensures that only nodes in $S^6$ within 8 hops from $v$ execute algorithm TryMIS while node $v$ is assigning colors. In particular, the broadcast of messages $DoNotTryMIS$ and $DoNotTransmit$ ensures that if a node $v$ assigns colors to its neighbors all neighbors are allowed to transmit and will get a color. A node $u$ keeps track of all received messages $DoNotTryMIS$ and $DoNotTransmit$ using the set $WaitFor(u)$.

To obtain a color, nodes $v \in S^6$ and its uncolored neighbors $u \in N(v)$ begin executing algorithm $GetColor$ concurrently. Essentially, algorithm $GetColor$ repeats a schedule consisting of 3 synchronized time slots, i.e. all nodes $N(v) \cup v$ execute the same time slot concurrently. In the first slot every uncolored node $u$ might apply for a color by sending a request to color itself. If the node $v \in S^6$ performing the assignment, received a message $Request(u)$, it grants the request by transmitting $Grant(u)$ in the second slot. Upon reception of message $Grant(u)$, node $u$ chooses an available color $c$, transmits $Taken(c)$ in the third slot to all its neighbors and exits the algorithm. Every uncolored node $u$ keeps track of the used colors in its neighbor-

**Algorithm FastColoring**

**Upon wake-up:**

1: $TakenColors(v) := WaitFor(v) := \{\}$; $color_v := -1$

**TryMIS():**

2: $s_v^{MIS} := false$
3: Compute MIS[14] $S^1$ {Out: state $s_v^{MIS}$}
4: **if** $s_v^{MIS} = true$ **then**
5: **wait until** Broadcast($v$,6, $InMIS(v)$,$\frac{1}{n^5}$) completed plus the duration of a Broadcast(.,6, .,$\frac{1}{n^5}$)
6: $IS^6(v) := \{u|$recv msg $InMIS(u)\}$
7: **wait until** Broadcast($v$,7, $MIS6(v, IS^6(v))$,$\frac{1}{n^5}$) completed plus the duration of a Broadcast(.,7, .,$\frac{1}{n^5}$)
8: $IS^6(v) := IS^6(v) \cup \{u|$recv msg with $MIS6(u, IS^6(u)) \wedge (v \in IS^6(u))\}$ {Ensure symmetry of $IS^6$}
9: Calculate MIS [16] $S^6$ on $G' = (V' = \{u|u \in S\}, E' = \{\{u, v\}|u, v \in V', v \in IS^6(u)\})$ {Out: state $s_v^{MIS6}$}
10: **if** $s_v^{MIS6} = true$ **then**
11: Broadcast($v$,8,$DoNotTryMIS(v)$,$\frac{1}{n^5}$)
12: Broadcast($v$,3,$DoNotTransmit(v)$,$\frac{1}{n^5}$)
13: Transmit $GrantingColors$
14: GetColor() {see Table 1; Out: $color_v$}
15: Broadcast($v$,9, $TransmitAndTryMIS(v)$,$\frac{1}{n^5}$)
16: **end if**
17: **end if**

**At any time:**

18: **if** recv msg $GrantingColors$ **then** GetColor(); Exit; **end if** {see Table 1; Out: $color_v$}
19: **if** $(s_v^{MIS6} = false) \wedge$(recv msg $\in \{DoNotTransmit(u), DoNotTryMIS(u)\}$) **then** Stop executing TryMIS(); $s_v^{MIS} := false$;$WaitFor(v) := WaitFor(v) \cup msg$ **endif**
20: **if** recv msg $TransmitAndTryMIS(u)$ **then** $WaitFor(v) := WaitFor(v) \setminus \{DoNotTryMIS(u), DoNotTransmit(u)\}$ **end if**
21: **if** (recv msg $InMIS$) $\wedge$ $(s_v^{MIS} = false)$ **then** Stop executing TryMIS() **end if**
22: **if** $(color_v = -1) \wedge (WaitFor(v) = \{\}) \wedge$(no msg recv for the duration $constant \cdot \log \Delta \cdot \log n$ slots) **then** TryMIS() **end if** {duration bounds time from start of broadcast $InMIS$ until $DoNotTryMIS$ finished}

hood, i.e. $TakenColors(u)$. It also maintains an estimate of the number of uncolored neighbors $\tilde{n}(u)$ (initially $\Delta$)that might concurrently apply for a color. The probability that a node transmits a request is given by $\frac{1}{\tilde{n}(u)}$. Since more and more neighbors get colored over time and the estimate might overshoot the true number of uncolored nodes, the estimate is updated from time to time, i.e. if out of a sequence of $128 \cdot e^2 \cdot \log n$ time slots less than $32 \cdot \log n$ $Grant$ messages were received the estimate $\tilde{n}(u)$ is divided by 2.

## 5.2 Arbitrary wake-up, $\Delta$ unknown

To bound $\Delta$ we simply use $n$. If all nodes are known to wake up within a time span of length $t$, any newly woken-up node listens for $t$ time slots and forwards broadcasts but does not initiate any broadcasts, i.e. algorithm TryMIS is not started. This way, a node will be fully aware of the state of the computation; as for synchronous wake-up algorithm FastColoring can be used.

Because of lack of space, for arbitrary wake-up we only sketch the algorithm. The main difficulty is that a woken-up node is unaware of the state of its neighbors. For instance, a node $u$ might wake up and have multiple leaders as neighbors that all assign colors. If $u$ transmits, it might disturb the color assignment. Thus a leader $v$ assigning colors, interrupts algorithm GetColor from time to time (say after $O(\log^2 n)$ slots) to broadcast messages $DoNotTransmit(v)$ and $DoNotTryMIS(v)$ again and all nodes that are only prevented from transmitting by $v$ forward the message. However, more needs to be done: Assume a node $u$ has received $DoNotTransmit$ from two or more different leaders. In that case, it cannot forward any message. If a node $w \in N(u)$ wakes-up and node $u$ is its only neighbor, i.e. $N(w) = u$, then it does not receive any message and has to wait very long (i.e. $O(n)$) to be sure that it can transmit safely. To remedy this problem, we introduce some empty slots during the color assignment. More precisely, out of $f(8)$ slots we only use 1 slot for the color assignment. The other slots are used for forwarding broadcasts. The leader determines which slots are used and broadcasts the chosen slots together with the message $DoNotTransmit$, such that a node is not allowed to transmit in the chosen slots. Thus, there are enough free slots to forward broadcasts. However, there are two more issues: First, a leader does not necessarily color all its uncolored neighbors. Second, a (newly) woken-up node is unaware of the previously chosen colors, but still must avoid deciding on an already chosen color.

We address these two problems by splitting algorithm GetColor into a sequence of four repeated phases, each of duration $O(\log^2 n)$. Any woken-up node can try to get a color, whenever the first phase is (re)started. If a node chose a color, already chosen by node $u$, then $u$ can place a veto. More precisely, during the first phase the number of uncolored neighbors of the leader is estimated and used to set the transmission probability. During the second phase, uncolored nodes apply for a color as before. During the third and fourth phase all nodes $u$ that faced a color collision, i.e. a node $v$ chose a color previously selected by a node $u$, transmit a veto, forcing $v$ to choose another color. During the third phase, a node $u$ estimates the number of nodes that also faced a color collision and then uses this estimate to determine the transmission probability in the veto phase.

The estimate of the number of nodes having a color collision can be computed as follows. A node $w \in S^6$ broadcasts $StartGettingDensity$ for 2 hops and all nodes in $N^2(w)$ that faced a color collision execute the following protocol synchronously for $O(\log^2 n)$ time steps: Transmit $Prob(\frac{1}{2^i})$ with probability $\frac{1}{2^i}$ and for all $i$ starting from 1 to $\log n$ for $O(\log n)$ time slots for each $i$. The estimate $|\tilde{N}(u)|$ is given by $\max\{2^i|u$ received $O(\log n)$ messages $Prob(\frac{1}{2^i})\}$. The number of uncolored nodes $u \in N(w)$ can be estimated in an analogous way.

## 5.3 Distance-$d$ coloring

We focus on synchronous wake-up. To compute a distance-$d$ coloring (for constant $d$) with $O(\Delta)$ colors, algorithm FastColoring can be used with minor modifications. It must be ensured that the distance among nodes assigning colors to their direct neighbors is at least $\min\{d+3, 6\}$, such that nodes with distance at least $d+1$ get the same color. Furthermore, leaders having assigned colors broadcast all (newly) assigned colors up to distance $d$. The

| | node $v \in S$ | uncolored node $u \in N(v)$ |
|---|---|---|
| Initialization | $color_v := 0$ | $\tilde{n}(v) := \Delta$ |
| Slot 1 | Listen | Transmit $Request(u)$ with probability $\frac{1}{\tilde{n}(v)}$ |
| Slot 2 | **if** received $Request(u)$ **then** Transmit $Grant(u)$ | Listen |
| Slot 3 | Sleep<br><br>**if not** received msg $Request$ for $384 \cdot e^2 \cdot \log n \cdot (\log \Delta + 1)$ slots **then** Exit **end if** | **if** received $Grant(u)$ **then** $color_v := \min\{c \mid c \in \{1, 2, .., \Delta\} \setminus TakenColors(v)\}$ Transmit $Taken(c)$ and exit **else** Listen **end if**<br>**if** received $Taken(c)$ **then** $TakenColors(v) := TakenColors(v) \cup c$ **end if**<br>**if** received $< 32 \cdot \log n$ $Grant$ messages and $\tilde{n}(v)$ not altered during the last $384 \cdot e^2 \log n$ time slots **then** $\tilde{n}(v) := \frac{\tilde{n}(v)}{2}$ **end if** |

**Table 1: Algorithm *GetColor*, repeats a schedule of three (synchronized) time-slots**

assigned colors can be broadcast together with message $TransmitAndTryMIS$. Finally, we have to alter the distances of the various broadcasts. The broadcast of message $InMIS$ is performed with $h = \max\{d+2, 6\}$[10], of $MIS6$ with $h = \max\{d+2, 6\}+1$, of $DoNotTryMIS$ with $h = \max\{d+2, 6\}+3$ and of $TransmitAndTryMIS$ with $h = \max\{d+2, 6\}+4$. The broadcast of $DoNotTransmit$ remains unchanged.

# 6. ANALYSIS ALGORITHM BROADCAST

Throughout the proof we assume that a broadcast is only performed up to a constant distance $d$ and for constant $f(d)$. We use the notion of a subschedule $r_w^k \subseteq r_w$. The $k^{th}$ subschedule $r_w^k$ denotes the transmission probabilities of the slots $r_w[j]$ with $k \cdot d < j \leq k \cdot (d+1)$ with $d := e^7 \cdot 5 \cdot f(h)^2 \cdot h \cdot \log \Delta$ and $0 \leq k \leq \log \frac{1}{\epsilon}$.

The first lemma states that all nodes broadcasting a message according to schedule $r_w$ follow the schedule synchronously, i.e. transmit with the same probability $r_w[t]$ for some $0 \leq t \leq |r_w| - 1$.

LEMMA 1. *If two nodes $u, v \in V$ transmit concurrently according to schedule $r_w$ then both transmit the same message $(msg_w, h, t)$ with probability $r_w[t]$ for some $0 \leq t \leq |r_w| - 1$ at the same time.*

PROOF. Assume node $u$ received the message from node $s$ and node $v$ from node $t$ and both $s$ and $t$ execute the schedule $r_w$ synchronously. More precisely, say node $v$ received message $(msg_w, h, t_v)$ at local time $t_s$ and node $u$ received message $(msg_w, h, t_u)$ $x$ slots later at time $t_s + x$. Since $s$ and $t$ transmit the schedule synchronously (i.e. are at the same position in the schedule) and $t_u$ gives the current position, we have $t_u = t_v + x$. Due to the algorithm node $v$ will start executing schedule $r_w$ from slot 0 onwards at time $t_s + |r_w| - t_v$ (of node $s$) and node $u$ at time $t_s + x + |r_w| - t_u = t_s + x + |r_w| - t_v - x = t_s + |r_w| - t_v$. Thus both nodes will start in parallel.

The case when both nodes $u, v$ received the message from the same node is analogous. □

The next lemma shows that if some neighbors of node $v$ follow (concurrently) one subschedule $r_w^k$ then the chance that node $v$ receives a message is constant.

[10]Distance $d+2$ suffices, since the distance between nodes in the MIS $S^{d+2}$ is $d+3$.

LEMMA 2. *Assume neighbors $T \subseteq N(v)$ of node $v$ transmit according to schedule $r_w$. After an execution of some subschedule $r_w^k$ with $0 \leq k \leq \log \frac{1}{\epsilon}$, node $v$ has received $msg_w$ with probability $1 - \frac{1}{e^2 \cdot f(h) \cdot h}$.*

PROOF. By definition (Section 4) within distance $h$ of $v$ at most $5 \cdot f(h)$ schedules are allowed to be executed concurrently and each schedule only contains one entry with value larger than 0 out of $5 \cdot f(h)$ entries. The chance $p_{r_w^k}$ that all nodes $u \in N(v)$ transmit either according to $r_w^k[i]$ for some entry $r_w^k[i] > 0$ or not at all, is at least the probability that the (current) entry of all other concurrently executed schedules (at most $5 \cdot f(h) - 1$) is 0:

$$p_{r_w^k} \geq (1 - \frac{1}{5 \cdot f(h)})^{5 \cdot f(h) - 1} \geq \frac{1}{e}$$

The probability $p_{exactly1}$ that exactly one node $u \in T$ transmits if all nodes in $T$ transmit with probability $\frac{1}{2 \cdot |T|} \leq r_w^k[i] \leq \frac{2}{|T|}$ is

$$p_{exactly1} \geq |T| \cdot \frac{1}{r_w^k[i]} \cdot (1 - r_w^k[i])^{|T|-1}$$

$$\geq |T| \cdot \frac{1}{2 \cdot |T|} \cdot (1 - \frac{2}{|T|})^{|T|-1} \geq \frac{1}{2} \cdot (1 - \frac{2}{|T|})^{|T|-1} \geq \frac{1}{2 \cdot e^2}$$

Thus the total probability that node $v$ receives the message within $5 \cdot f(h)$ slots is $p_{r_w^k} \cdot p_{exactly1} \geq \frac{1}{e^4}$.

Due to construction schedule $r_w^k$ with $|r_w^k| = e^7 \cdot 5 \cdot f(h)^2 \cdot h \cdot \log \Delta$ contains one entry $r_w^k[i]$ out of $5 \cdot f(h) \cdot \log \Delta$ many with $\frac{1}{2 \cdot |T|} \leq r_w^k[i] \leq \frac{2}{|T|}$ and $0 < |T| \leq \Delta$. Thus when executing $r_w^k$ a node transmits $e^7 \cdot f(h) \cdot h$ times with probability $p$ for $\frac{1}{2 \cdot |T|} \leq p \leq \frac{2}{|T|}$. The chance that all attempt fail is: $(1 - \frac{1}{e^4})^{e^7 \cdot f(h) \cdot h} \leq \frac{1}{e^2 \cdot f(h) \cdot h}$. □

THEOREM 3. *The probability that a message originated at $w$ reaches all nodes $u \in N^h(w)$ and no node $u \in N^{h+2}(w) \setminus N^{h+1}(w)$ within time $O(\log \Delta \cdot \log \frac{1}{\epsilon})$ is at least $1 - \epsilon$.*

PROOF. When a node receives a message $(msg_w, d, .)$ it will transmit $(msg_w, d-1, .)$ as long as $d > 0$. Since the originator transmits $(msg_w, h, .)$, a message will only reach nodes up to distance $h+1$.

The neighborhood $N^h(w)$ can be covered by a MIS $S^1$ with $|S^1| \leq f(h)$ (see Definition 1). Clearly, any node $v \in S^1$ is reachable by a path $P = \{w = u_0, u_1, ..., u_h\}$ of length

at most $h$ nodes. Since $|S^1| \leq f(h)$ we need at most $f(h)$ (distinct) paths to get from $w$ to all nodes $v \in S^1$. To ensure that the message reaches node $v$, a node $u_i$ with $0 \leq i < h$ must have/receive a message $(msg_w, d, .)$ with $d \geq h - i$. Clearly, this holds for $u_0 = w$. Assume that for a node $u_i$ all neighbors forward a message $(msg_w, d, .)$ with $d \geq h - i$. Then the chance $p_k$ that $u_i$ receives the message after the execution of a subschedule $r_w^k$ is $p_k := 1 - \frac{1}{\mathrm{e} \cdot f(h) \cdot h}$ (Lemma 2). The chance that a message is forwarded along a path of length $h$ (i.e. $h$ consecutive times) due to $r_w^k$ is $(p)k^h$. The chance that this happens for at most $f(h)$ paths is $(p_k)^{f(h) \cdot h}$. Once all nodes $v \in S^1$ have the message, they must transmit it to their neighbors (By definition every node $u \in N^h(w)$ is adjacent to a node $v \in S^1$). Thus the total chance that a message is forwarded to all nodes $v \in S^1$ and they transmit it to their neighbors if a single subschedule $r_w^k$ is executed is $(p_k)^{f(h) \cdot (h+1)} = (1 - \frac{1}{\mathrm{e}^2 \cdot f(h) \cdot h})^{f(h) \cdot (h+1)} \geq \frac{1}{2}$. Since a node having received a message starts the execution after its neighbors have completed all subschedules, we get that the failure probability after having executed $\log \frac{1}{\epsilon}$ subschedules is $\frac{1}{2}^{\log \frac{1}{\epsilon}} = \epsilon$ □

# 7. ANALYSIS ALGO. FASTCOLORING

## 7.1 Synchronous wake-up, $\Delta$ known

After the first three lemmas, we prove the correctness and time complexity of all steps of the algorithm one after the other. We assume that $n > \mathrm{e}^{(\log^* n)^{c_3 - 1}}$ for some constant $c_3$.[11] By $M^{Broad}$ we denote the set of all broadcasts of any message $msg \in \{InMIS, MIS6, DoNotTryMIS, DoNotTransmit, TransmitAndTryMIS\}$ performed during algorithm FastColoring.

The upcoming lemma is an extension of Lemma 1 and shows that despite a delayed start or interruption of a (broadcast) schedule due to messages $DoNotTransmit$ and $TransmitAndTryMIS$ nodes still execute any broadcast in a synchronized manner.

LEMMA 4. *Lemma 1 holds even if schedule $r_w$ got interrupted or its start got delayed.*

PROOF. Due to Lemma 9 we can assume that,initially, all nodes $u, v \in V$ execute schedule $r_w$ synchronously. Assume a node $u$ received message $(DoNotTransmit(s), ., y_u^{DoNot})$ at time $t_u^{DoNot}$ and $TransmitAndTryMIS(s)$ at time $t_u^{Can}$. Say, $x$ slots later, node $v$ got $(DoNotTransmit(z), ., y_v^{DoNot})$ and $TransmitAndTryMIS(z)$ at time $t_v^{Can}$. Since $y_v^{DoNot}$ gives the current position in the schedule, which advanced by $x$ slots since $(DoNotTransmit(w), ., y_u^{DoNot})$ has been transmitted, we have $y_v^{DoNot} = y_u^{DoNot} + x$.

Node $u$ starts $r_w$ at time $(t_u^{DoNot} - y_u^{DoNot}) + i_u \cdot |r_w|$, such that $i_u$ is the smallest positive integer with $(t_u^{DoNot} - y_u^{DoNot}) + i_u \cdot |r_w| > t_u^{Can}$. Node $v$ restarts at time $(t_u^{DoNot} + x - y_v^{DoNot}) + i_v \cdot |r_w| = (t_u^{DoNot} - y_u^{DoNot}) + i_v \cdot |r_w| > t_v^{Can}$, such that for integer $i_v$ holds $(t_u^{DoNot} - y_u^{DoNot}) + i_v \cdot |r_w| > t_v^{Can}$. Thus either $i_v$ equals $i_u$ and both nodes start synchronously or in case they are distinct they execute schedule $r_w$ sequentially.

The proof for a delayed start is analogous. □

[11]The assumption can be dropped changing the overall time complexity from $O(\Delta + \log \Delta \cdot \log n)$ to $O(\Delta + \log \Delta \cdot \log n \cdot \log^* n)$.

The next two lemmas show that all nodes assigning colors have distance at least 7. The first lemma proves the claim if all nodes are allowed to transmit. The second lemma shows that the claim holds even if some nodes are not permitted to transmit.

LEMMA 5. *Assuming that no broadcast in $M^{Broad}$ fails and all nodes are allowed to transmit, $S^6$ and $S^1$ are computed (correctly) in time $O(\log \Delta \cdot \log n)$, then no nodes with distance less than 7 will ever assign colors at the same time.*

PROOF. Assume that two nodes $u, v$ with $v \in N^6(u)$ have joined the MIS $S^6$ while all nodes are allowed to transmit (i.e. $\nexists DoNotTransmit \in WaitFor(s)$ for all $s \in V$). Assume $u$ has joined first. Node $v$ must have joined $S^1$ before the broadcast of $InMIS(u)$ could have reached $v$. If $InMIS(u)$ reached $v$ before it joined $S^1$, it would have stopped executing algorithm TryMIS upon reception of a message $InMIS$ and would have waited with the restart for at least $constant \cdot \log \Delta \cdot \log n$ slots. Since $S^6$ is computed in time $O(\log \Delta \cdot \log n)$ (Lemma 8) and a broadcast is also in $O(\log \Delta \cdot \log n)$ (Lemma 3), the *constant* can be chosen such that for any node $u$ that joined $S^6$ the message $DoNotTryMIS(u)$ will have reached all nodes within distance 8 of $u$ before they (re)start algorithm TryMIS. Clearly, after the reception of $DoNotTryMIS(u)$ a node must wait until it received a message $TransmitAndTryMIS$ and thus no node $v \in N^6(u)$ is able to join $S^6$, while $u$ assigns colors. Thus node $v$ must have joined $S^1$ before it received $DoNotTryMIS(u)$ or $InMIS(u)$. Since $u$ broadcasts $InMIS(u)$ directly after it joined, node $v$ must have joined while the broadcast was still going on. Thus, node $v$ will receive the broadcast of $InMIS(u)$ and add $u$ to its set $IS^6(v)$. But also $u$ must receive $InMIS(v)$ since $u$ waits for the duration of a broadcast $InMIS$ after it completed the broadcast $InMIS(u)$. With the same argument node $v$ will receive $IS^6(u)$ and node $u$ receives $IS^6(v)$ before it starts the computation of $S^6$. Therefore, we have that $u \in IS^6(v)$ and $v \in IS^6(u)$ and by assumption $S^6$ is computed correctly given a correct input to algorithm [16] and thus all nodes have distance 7. □

LEMMA 6. *Assuming that no broadcast in $M^{Broad}$ fails, $S^6$ and $S^1$ are computed (correctly) in time $O(\log \Delta \cdot \log n)$, then no nodes with distance less than 7 will ever assign colors at the same time.*

PROOF. Due to Lemma 5 the statement is true, if all nodes are allowed to transmit. For the proof we require that a message $DoNotTryMIS(v)$ is received by some node $u$ before (or concurrently) it received $DoNotTransmit(v)$. We refrain from the lengthy proof and remark that the two separate broadcasts of $DoNotTryMIS$ and $DoNotTransmit$ can be replaced by a single $Broadcast(v, \{DoNotTransmit(v), DoNotTryMIS(v)\}, 8, \frac{1}{n^5})$. If a node has received $(\{DoNotTransmit(v), DoNotTryMIS(v)\}, 5, \frac{1}{n^5})$, it forwards the message without $DoNotTransmit(v)$.

Consider a node $v \in S^1$ (possibly also in $S^6$) for which some nodes in $T \subseteq N^6(v)$ did not receive a broadcast initiated by $v$, i.e. the broadcast got interrupted due to nodes that are not allowed to transmit. Let set $W \subseteq S^6$ be the set of nodes which caused the nodes not to transmit, i.e. each node $w \in W$ broadcast $DoNotTransmit$ and some node $s \in N^6(v) \cap N^4(w)$ received the message.

For every node $u \in T \subseteq N^6(v)$ and every path from $v$ to $u$ (of length at most 6) there must be a node $s$ that received a message $DoNotTransmit(w)$ by a node $w \in W$. Otherwise $u$ would have received the broadcast by $v$. Let $s$ be the closest node to $u$. Since both $s$ and $u$ are on a path of length at most 6, we have that the distance from $s$ to $u$ is at most 5. Since $s$ is assumed to be the closest non-transmitting node to $u$ and node $s$ will forward $DoNotTryMIS(w)$ for 5 more hops (see above: $DoNotTryMIS$ is forwarded 5 hops more than $DoNotTransmit$), node $u$ must have received $DoNotTryMIS(w)$. Since a node $u$ will not start TryMIS for *constant* $\log\Delta \cdot \log n$ steps, node $u$ will not execute algorithm TryMIS before node $v$ completed its broadcasts $DoNotTryMIS(v)$ and $DoNotTransmit(v)$ (Same argument as in proof of Lemma 5).

As long as node $u$ has not received $DoNotTryMIS(v)$, such a non-transmitting node $s$ must exist and node $u$ will not execute TryMIS because of a message $DoNotTryMIS(w)$ with $w \in W$. □

The following lemma deals with the time complexity of algorithm [14]. In the paper the running time of the algorithm is $O(\log^2 n)$ with probability $1 - \frac{1}{n}$ without an estimate of $\Delta$. We briefly show how to increase the success probability (using more time slots though) and how to make use of the estimate $\Delta$.

LEMMA 7. *Within time $O(\log\Delta \cdot \log n)$ MIS $S^1$ is computed with probability $1 - \frac{1}{n^5}$.*

PROOF. The success probability can be increased from $1 - \frac{1}{n}$ to $1 - \frac{1}{n^5}$ by using a factor 5 more time slots. Throughout the proof in [14] bounds of the form $e^{c \cdot \log n} = \frac{1}{n^c}$ are used. Therefore a multiplying the occurring constants by a factor of 5 yields the desired result.

With an estimate of $\Delta$ the time complexity can be improved to $O(\log\Delta \cdot \log n)$ by replacing $n$ in $p_v$ by $\Delta$ in Algorithm 1 in [14], the bound in Line 3 of $4\mu\delta \log^2 n$ by $4\mu\delta \log\Delta \log n$, $\log n$ by $\log\Delta$ in Line 15 and finally $\delta \log^2 n$ by $\log\Delta \log n$ in Line 16. [12] □

The next lemma shows that the set $S^6$ is computed correctly and efficiently using algorithm [16].

LEMMA 8. *Assuming that no broadcast in $M^{Broad}$ fails, algorithm [16] computes a correct $S^6$ within time $O(\log\Delta \cdot \log n)$ with probability $1 - \frac{1}{n^4}$.*

PROOF. Due to Lemma 6 the input for algorithm [16] is correct, i.e. the set $IS^6(v)$ for a node $v$ contains all other nodes in $S^1$ that might join $S^6$ within distance 6 and is also symmetric, i.e. if $u \in IS^6(v)$ then also $v \in IS^6(u)$.

By definition set $S^6$ corresponds to a MIS in the graph $G' = (V' = \{u | u \in S\}, E' = \{\{u,v\} | u,v \in V', v \in IS^6(u)\})$ on which we run the deterministic algorithm MIS [16]. (The graph is undirected since for $v \in IS^6(u)$ also $u \in IS^6(v)$.) The proof of the correctness of algorithm [16] in the message passing model, where no collisions occur, is given in [16]. Thus we must show that algorithm [16] receives all required information despite of lost messages. To compute a MIS on $G'$, node $v \in S^1$ has to exchange messages with all neighbors $u \in IS^6(v)$ in $G$ (by the definition of graph $G'$). In case node $v$ misses some information of some neighbor $u$ in $G'$, it halts the execution of algorithm [16] and asks $u$ (via a broadcast) to retransmit the data. Thus algorithm [16] eventually receives all necessary information for the computation, progresses and finishes outputting a correct result.

The time complexity can be derived as follows. The graph $G'$ has constant degree, i.e. the number of neighbors of a node $u \in S^6$ in $G'$ is bounded by the size of a MIS in $G$ within 7 hops, which is $f(7)$ (by definition of $f$). The graph $G'$ is thus also of bounded-independence with $f'(h) \leq f(7 \cdot h)$. Opposed to the message passing model, a node $v$ cannot exchange messages with its neighbors $v$ in one round but needs to use Algorithm Broadcast for it. Algorithm [16] progresses one step in its computation once every node $v \in S^1$ performed a successful broadcast. The number of required steps is bounded by $O(f'(f'(2)+1) \cdot \log^* n) = O(\log^* n)$ due to the analysis in [16]. This implies that for a node $v$ only a node $u \in N^{O(\log^* n)}(v) \cap S^1$ can influence $v$'s computation, i.e. if such a node $u$ fails to broadcast a message, node $v$ might be delayed. The number of these neighbors is bounded by $|N^{O(\log^* n)}(v) \cap S^1| = f'(O(\log^* n)) \leq c_2 \cdot (\log^* n)^{c_1} = (\log^* n)^{c_3}$ with constants $c_1, c_2, c_3$.

Due to Theorem 3 the chance that Algorithm Broadcast with $\epsilon = \frac{1}{\sqrt{2}}$ transmits a message originated at $v$ to all nodes $u \in IS^6(v)$ within time $O(\log\Delta)$is at least $\frac{1}{\sqrt{2}}$. If a broadcast message was not delivered to some node, this node asks for a retransmission. The chance that both the request and the retransmission succeed is $\frac{1}{2}$. Let $f_i$ be the number of requests and retransmissions for $v$ until all $(\log^* n)^{c_3}$ nodes that influence the computation of $v$ have performed one successful broadcast, i.e. algorithm [16] can execute one of the $O(\log^* n) = c_0 \cdot \log^* n$ (for some constant $c_0$) steps. The chance that for one node out of $(\log^* n)^{c_3}$ the request and retransmission broadcast fail $f_i$ times in a row is $1 - (1 - \frac{1}{2^{f_i}})^{(\log^* n)^{c_1}}$, in particular if $f_i$ is larger than $g := \frac{2 \cdot \log n}{\log^* n}$ the chance becomes $1 - (1 - \frac{1}{2^{f_i}})^{(\log^* n)^{c_3}} \leq \frac{1}{2^{\frac{f_i}{2}}}$ for $g > (\log^* n)^{c_3}$ (i.e. $n > e^{(\log^* n)^{c_3 - 1}}$). We used $(1 - \frac{1}{2^{\frac{2\log n}{\log^* n}}})^{(\log^* n)^{c_3}} = (1 - \frac{1}{n^{\frac{2}{\log^* n}}})^{(\log^* n)^{c_3}} \leq \frac{1}{n^{\frac{1}{\log^* n}}}$. The probability that more than $c_4 \cdot \log n$ (with constant $c_4$) broadcasts are needed (i.e. $\sum_{i=0}^{c_0 \cdot \log^* n} f_i > c_4 \cdot \log n$) can be bounded as follows: The total time until algorithm [16] computed MIS $S_i$ is estimated from above as follows:

$$\sum_{i=0}^{c_0 \cdot \log^* n} f_i \leq \sum_{i=0, f_i \leq g}^{c_0 \cdot \log^* n} f_i + \sum_{i=0, f_i > g}^{c_0 \cdot \log^* n} f_i$$

$$\leq \sum_{i=0}^{c_0 \cdot \log^* n} g + \sum_{i=0, f_i > g}^{c_0 \cdot \log^* n} f_i \leq O(\log n) + \sum_{i=0, f_i > g}^{c_0 \cdot \log^* n} f_i$$

Assume we fix a set $e_k^{fix}$ of values $t_i > g$ with $0 \leq i \leq c_0 \cdot \log^* n$ arbitrarily, such that the sum of all $t_i$ equals $k$ ($\sum_{i=0, t_i > g}^{c_0 \cdot \log^* n} t_i = k$). The chance that an event $e_k^{fix}$ occurs, i.e. the total delay is $k$ and the number of needed requests/retransmissions $f_i$ are as given by the set $e_k^{fix}$ (i.e. $f_i = t_i$) is $prob(\text{event } e_k^{fix} \text{ occurs}) = \prod_{i=0}^{c_0 \cdot \log^* n} prob(t_i = f_i) \leq \prod_{i=0}^{c_0 \cdot \log^* n} \frac{1}{2^{\frac{f_i}{2}}} = \frac{1}{2^{\sum_{i=0}^{c_0 \cdot \log^* n} \frac{f_i}{2}}} = \frac{1}{2^{\frac{k}{2}}}$. The number of

[12] We refrain from restating the full proof from [14], since the improvement of the original bound of $O(\log^2 n)$ to $O(\log\Delta \cdot \log n)$ is not of crucial importance.

events $e_k^{fix}$, which add up to a fixed $k$, can be bounded by

$$(c_0 \cdot \log^* n)! \cdot \sum_{i_0=0}^{k} \sum_{i_1=0}^{k-i_0} \cdots \sum_{i_j=0}^{k-\sum_{s=0}^{j-1} i_s} \cdots \sum_{i_{c_0 \cdot \log^* n}=0}^{k-\sum_{s=0}^{c_0 \cdot \log^* n - 1} i_s} 1$$

$$= k^{c_0 \cdot \log^* n}$$

Therefore the chance that all fail becomes $(1 - \frac{1}{2^{\frac{k}{2}}})^{k^{c_0 \cdot \log^* n}}$. For $k \geq 32 \cdot \log n$ we get $(1 - \frac{1}{2^{\frac{k}{2}}})^{k^{c_0 \cdot \log^* n}} \geq (1 - \frac{1}{2^{\frac{k}{4}}})$. The chance none of all possible $e_k^{fix}$ for any $k \geq 32 \cdot \log n$ occurs is:

$$\prod_{k=32 \cdot \log n}^{\infty} (1 - \frac{1}{2^{\frac{k}{4}}}) = 2^{\sum_{k=32 \cdot \log n}^{\infty} \log(1 - \frac{1}{2^{\frac{k}{4}}})} \geq 2^{\sum_{k=32 \cdot \log n}^{\infty} -\frac{1}{2^{\frac{k}{4}}}}$$

$$= 2^{-\frac{2 \cdot 2^{-\frac{32 \cdot \log n}{4}}}{2 - 2^{\frac{3}{4}}}} \geq 2^{-\frac{1}{n^7}} \geq 1 - \frac{1}{n^7}$$

where we used $\log(1 - x) \geq -x$ for $0 \leq x \leq 0.1$.

Thus the chance to have more than $O(\log n)$ failures is less than $\frac{1}{n^7}$. The time to perform $O(\log n)$ broadcasts with failure probability $\frac{1}{2}$ is $O(\log \Delta \cdot \log n)$ for $n > e^{(\log^* n)^{c_3 - 1}}$. □

The next three lemmas deal mainly with algorithm GetColor. Lemma 9 guarantees a synchronous start of GetColor for a node $v \in S^6$ and its uncolored neighbors $u \in N(v)$. Lemma 10 proves the correctness of the coloring and Lemma 11 gives a bound on the time complexity of algorithm GetColor.

LEMMA 9. *Given that no broadcast in $M^{Broad}$ fails, a node $v \in S^6$ and all uncolored nodes $u \in N(v)$ start and execute algorithm GetColor concurrently and no neighbor $u \in N^3(v) \setminus N(v)$ transmits.*

PROOF. Due to Lemma 6 and 8 no nodes within distance 6 will assign colors concurrently. Since a node granting colors broadcasts *DoNotTransmit* up to 4 hops right before entering GetColor, for a node $v \in S^6$ executing GetColor will hold that all nodes $N^3(v)$ have received message $DoNotTransmit(v)$ and will not transmit after forwarding the message (except for uncolored nodes $u \in N(v)$, which must transmit to get colored). In particular no neighbor $u \in N^2(v)$ will transmit, when $v$ transmits *GrantingColors* and therefore all uncolored nodes $u \in N(v)$ call algorithm GetColor concurrently. □

LEMMA 10. *Given that no broadcast in $M^{Broad}$ fails, all uncolored nodes $u \in N(v) \cup v$ with $v \in S^6$ will obtain a (correct) color on termination of algorithm GetColor.*

PROOF. Due to Lemma 9 all uncolored nodes $u \in N(v) \cup v$ execute algorithm GetColor concurrently and no node $w \in N^3(v) \setminus (N(v) \cup v)$ transmits while $v$ executes GetColor. A color is assigned using three time slots and due to the synchronous start no collision occurs in slots 2 and 3 of the schedule of algorithm GetColor.

No node picks a color already chosen by a neighbor. A node $v \in S^6$ chooses color 0. Since no node $u \notin S^6$ chooses color 0 and node $v$ colors all its uncolored neighbors, no neighbor of node $v$ will attempt to join $S^6$ after it has executed algorithm GetColor. Thus no node $u \in N(v)$ will get color 0. Consider a node $u \in V \setminus S^6$. Whenever a neighbor $w \in N(u)$ chooses some color $c$, there must be a node $v \in S^6 \cap N^2(u)$ in its 2 hop neighborhood. Since node $v$ has transmitted $DoNotTransmit(v)$ up to at least 3 hops before starting to assign colors and only one node $w \in N(v)$ transmits in the third slot of algorithm GetColor, node $u \in N^2(v)$ will receive any message $Taken(c)$ by a neighbor $w \in N(u)$ and store it in its $TakenColors$ set. Therefore node $u$ won't choose an already chosen color.

Since a node $u$ has a neighbor $v \in S^6$ (which has color 0), once it can choose a color, we have that $|N(u) \setminus v| \leq \Delta - 1$. Thus the number of required colors for its neighbors is at most $\Delta$ and one color remains for $u$. □

LEMMA 11. *Given that no broadcast in $M^{Broad}$ fails, a node $v \in S^6$ and all uncolored nodes $u \in T \subseteq N(v)$ terminate algorithm GetColor within $O(\Delta + \log \Delta \cdot \log n)$ time slots with probability $1 - \frac{1}{n^4}$.*

PROOF. If a node $v \in S^6$ receives a request by a node $u \in N(v)$, it can transmit $Grant(u)$ without collision and node $u$ can transmit a message $Taken$ without collision due to Lemma 10. Thus some neighbor $u \in N(v)$ gets colored, if it transmits a request without collision.

For a sequence of $t_s := 384 \cdot e^2 \cdot \log n$ time slots, either at least $8 \cdot \log n$ nodes got colored (i.e. *Grant* and Taken messages were transmitted) or the transmission probability $\frac{1}{\tilde{n}(v)}$ is doubled by each uncolored neighbor $u \in T$. Assume that (directly) after a doubling of $\tilde{n}(v)$ we have that $n(v) \leq \tilde{n}(v) \leq 2 \cdot n(v)$. The chance that the transmission probability of an uncolored node gets doubled if $\frac{n(v)}{2} \leq \tilde{n}(v)$ can be computed as follows. The probability that a node $u \in T$ transmits a request without collision is $\frac{1}{\tilde{n}(v)} \cdot n(v) \cdot (1 - \frac{1}{\tilde{n}(v)})^{n(v)-1} \geq \frac{1}{2} \cdot \frac{1}{e^2} = \frac{1}{2 \cdot e^2}$. Out of the $t_s$ time slots, we use one third (i.e. $128 \cdot e^2 \log n$) for transmitting requests and thus expect at least $64 \cdot \log n$ transmissions of *Request* messages without collision. Using a Chernoff bound, the chance that there are less than $32 \cdot \log n$ (successful) transmissions is: $p$(less than $32 \cdot \log n$ *Requests* transmitted) $\leq e^{\frac{1}{8} \cdot 64 \cdot \log n} = \frac{1}{n^8}$. Therefore, either a constant fraction of the last $t_s$ time slots were used for successful transmissions or half of the remaining nodes got colored between two doublings of the transmission probability. Since $\tilde{n}(v) \leq \Delta$ after the probability is doubled at most $\log \Delta + 1$ times, all nodes have been colored with probability $1 - \frac{1}{n^4}$. If no sequence fails (i.e. $\tilde{n}(v)$ is halved despite $\tilde{n}(v) \leq n(v)$), then for every sequence of $t_s$ slots holds: Either the transmission probability is multiplied by two or $32 \cdot \log n$ nodes get colored. The number of successful sequences to color all nodes is given by $O(\Delta + \log \Delta \cdot \log n) \in O(n)$. The chance that all sequences succeed is at least $(1 - \frac{1}{n^8})^{c \cdot n} \geq 1 - \frac{1}{n^4}$ for some constant $c$.

The chance that the leader $v \in S^6$ exits the algorithm before having assigned colors to all neighbors, can be computed as follows. Due to the algorithm, the leader only exists if it has not received a request for $384 \cdot e^2 \cdot \log n \cdot (\log \Delta + 1) = t_s \cdot (\log \Delta + 1)$ time slots. Within the last $t_s$ time slots either there was at least one transmission without collision of a message *Request* or the transmission probability got doubled with probability at least $1 - \frac{1}{n^8}$ (see previous paragraph). The number of sequences as well as the overall success probability become $1 - \frac{1}{n^4}$. Therefore the time complexity becomes $O(\Delta + \log \Delta \cdot \log n)$ with probability $(1 - \frac{1}{n^4})^2$. □

THEOREM 12. *Within time $O(\Delta+\log\Delta\cdot\log n)$ every node is colored with probability $1-\frac{1}{n^3}$ using $\Delta+1$ colors.*

PROOF. Given that no broadcast in $M^{Broad}$ fails, due to Lemma 7 MIS $S^1$ is computed within time $O(\log\Delta\cdot\log n)$. Thanks to Lemma 8 and 6, algorithm FastColoring correctly computes a set $S^6$ within time $O(\log\Delta\cdot\log n)$ with probability $1-\frac{1}{n^4}$. A node $v\in S^6$ correctly colors its neighbors within time $O(\Delta+\log\Delta\cdot\log n)$ with probability $1-\frac{1}{n^4}$ due to Lemma 11.

An uncolored node $v\in V$ must have a neighbor within 15 hops that is coloring all its uncolored neighbors (i.e. executing algorithm GetColor) or will do so within time $O(\log\Delta\cdot\log n)$. If node $v$ is not executing algorithm TryMIS, then in case it is waiting due to a node $v\in S^6$ having transmitted message $DoNotTryMIS$ the statement holds. In case it is waiting due to message $InMIS$, i.e. due to a node $u\in S\cap N^7(v)$ then some neighbor $w\in N^7(u)\subseteq N^{15}(v)$ will join $S^6$ within $O(\Delta+\log\Delta\cdot\log n)$.

If a node assigns colors to its neighbors, all uncolored neighbors get colored (see Lemma 10). Therefore two nodes $u,v\in V$ that assign colors to all their neighbors are independent. Within distance 16 any node has at most $f(16)$ independent nodes (by definition of function $f$). The time until a node in $N^{16}(v)$ calls algorithm GetColor and finishes is $O(\Delta+\log\Delta\cdot\log n)$, thus after time $f(16)\cdot O(\Delta+\log\Delta\cdot\log n)$ any node must have a neighbor that assigns colors.

Next we bound the number of necessary broadcasts, i.e. $|M^{broad}|$. A node can only issue a broadcast, once it executes TryMIS. Per execution of algorithm TryMIS it issues at most 5 broadcasts. If a node executes algorithm TryMIS then using the same reasoning as above a node will get colored after at most $f(16)+1$ calls to TryMIS. Therefore overall at most $5\cdot(f(16)+1)\cdot n$ successful broadcasts are needed. The chance that no broadcast and no execution of algorithm MIS [14] fails is $(1-\frac{1}{n^5})^{6\cdot(f(16)+1)\cdot n}$. Due to Theorem 3 the time for one broadcast is $O(\log\Delta\cdot\log n)$.

The chance that all executions of algorithm GetColor work in time $O(\Delta+\log\Delta\cdot\log n)$ (see Lemma 11) is given by $(1-\frac{1}{n^5})^n$. If a node calls algorithm [16] to compute $S^6$ then after the calculation some neighbor (or itself) within distance 7 will color all its neighbors. Thus in total there are at most $f(7)\cdot n$ calls of algorithm [16]. The chance that all succeed in time $O(\log\Delta\cdot\log n)$ (see Lemma 8) is given by $(1-\frac{1}{n^5})^{f(7)\cdot n}$.

The overall probability that all algorithms succeed in the given time and no broadcasts fails is given by $(1-\frac{1}{n^4})^{c_4\cdot n}\geq 1-\frac{1}{n^3}$ with constant $c_4$. The overall time is bounded by $O(\Delta+\log\Delta\cdot\log n)$. □

## 7.2 Distance-$d$ coloring

Most of the lemmas and proofs in Section 7.1 hold with minor modifications. The fact that $O(\Delta)$ colors are sufficient, can be seen as follows: All nodes within constant distance $d$ of a node $v$ can be covered by an independent set $S^1_d$ of constant size $f(d)$ (by Definition of $f$). Thus any node $u\in N^d(v)$ has at least one neighbor in $S^1_d$, thus the number of nodes $|N^d(v)|$ can be bounded as follows: $|N^d(v)|\leq\sum_{u\in S^1_d}d(u)\leq f(d)\cdot\Delta\in O(\Delta)$.

# 8. REFERENCES


[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. In *Journal of Computer and System Sciences*, pages 290–298, 1991.

[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 *$6^{th}$ Symp. on PODC*, 1987.

[3] L. Barenboim and M. Elkin. Distributed (delta+1)-coloring in linear (in delta) time. In *41st STOC*, 2009.

[4] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology . In *44th Symp. on FOCS*, 2003.

[5] T. Herman and S. Tixeuil. A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks. In *Lecture Notes in Computer Science*, volume 3121/2004, 2004.

[6] T. Jurdziński and G. Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *$13^{th}$ ISAAC*, 2002.

[7] S. Khot. Improved Inapproximability Results for MaxClique, Chromatic Number and Approximate Graph Coloring. In *42nd Symp. on FOCS*, 2001.

[8] D. R. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. In *Distributed Computing*, volume 18, pages 43–57, 2005.

[9] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Local Approximation Schemes for Ad Hoc and Sensor Networks. In *$3^{rd}$ Workshop DIALM-POMC*, 2005.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *$10^{th}$ Conf. on MOBICOM*, 2004.

[11] E. Kushilevitz and Y. Mansour. An $\Omega(D\cdot\log(\frac{N}{D}))$ lower bound for broadcast in radio networks. In *$12^{th}$ Symp. on PODC*, 1993.

[12] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15, 1986.

[13] T. Moscibroda and R. Wattenhofer. Coloring Unstructured Radio Networks. In *17th SPAA*, 2005.

[14] T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. In *24th Symp. on PODC*, 2005.

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

[16] J. Schneider and R. Wattenhofer. A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs. In *27th Symp. on PODC*, 2008.

[17] 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. In *COM-23(12)*, 1975.