Report

How the Hidden-Terminal Problem Affects Clustering in Ad Hoc and Sensor Networks

Author(s):
Moscibroda, Thomas; Kuhn, Fabian; Wattenhofer, Roger

Publication Date:
2004

Permanent Link:
https://doi.org/10.3929/ethz-a-006742075

Rights / License:
In Copyright - Non-Commercial Use Permitted
Abstract

A newly deployed multi-hop radio network is unstructured and lacks a reliable and efficient communication scheme. In this paper, we define a model containing the characteristics of the initialization phase of such networks: asynchronous wake-up, scarce knowledge about the topology of the network graph, unreliable collision detection, and the hidden terminal problem. We show that even for this restricted model, a good clustering can be computed efficiently. We propose a new randomized algorithm for clustering in multi-hop radio networks, such as ad-hoc or sensor networks. The algorithm computes an asymptotically optimal clustering in polylogarithmic time. Additionally, our simulation results show the efficiency and practicability of the algorithm in a variety of settings.

1 Introduction

Multi-hop radio networks, for example ad-hoc and sensor networks, may be formed by autonomous nodes communicating via radio, without any additional infrastructure. Typically, if two nodes are not within their mutual transmission range, they may communicate through intermediate nodes. In other words, the communication infrastructure is provided by the nodes themselves. Setting up and organizing such a virtual infrastructure is an important common challenge.

When being deployed, the nodes initially form an unstructured radio network, which means that no reliable and efficient communication pattern has been established yet. Thus, before any reasonable communication can be carried out, the nodes must bring structure into the network. In this paper we call this self-organized transition from an unstructured to a structured multi-hop radio network the initialization. There are several ways of bringing structure into the network. We say that structure is achieved when the nodes have established an efficient media access control (MAC) scheme which provides reliable point-to-point connections to higher-layer protocols and applications. We also say that the network is structured when the nodes have partitioned themselves into clusters. Not surprisingly, clustering and MAC protocols are in close relation to each other — given a good clustering, one can derive an efficient MAC protocol, and vice versa. Thus, clustering is one prominent approach to solving the problem of bringing structure into a multi-hop radio network [3, 5, 6, 11, 14, 23].

Clustering allows the formation of virtual backbones, it enables efficient routing [24], it improves the usage of scarce resources, such as bandwidth and energy [15], and clustering helps realizing spatial multiplexing in non-overlapping clusters. Depending on the specific network organization problem at hand, various forms of clustering have been proposed. In this paper, we consider a clustering in which each node in the network is either a cluster-head or has a cluster-head within its communication range, such that cluster-heads can act as coordination points for the MAC scheme.

When we model a multi-hop radio network as a graph \( G = (V, E) \), clustering can be formulated as a classic graph theory problem: In a graph, a dominating set is a subset of nodes such that for every node \( v \), either a) \( v \) is in the dominating set or b) a direct neighbor of \( v \) is in the dominating set. As it is advantageous to compute a dominating set with few dominators (i.e. cluster heads), we study the well known minimum dominating set (MDS) problem which asks for a dominating set of minimum size.

The computation of dominating sets for the purpose of clustering networks has been studied extensively in a number of papers. A wide variety of algorithms have been proposed...
To the best of our knowledge, all these algorithms assume that the scheduling of transmissions is handled by an existing MAC layer providing point-to-point connections between neighboring nodes. In other words, it is taken for granted that a MAC layer has been established on top of which the algorithm can operate. Assuming an operational MAC layer solves a variety of problems that arise in unstructured networks, such as collision detection, asynchronous wake-up, or the hidden terminal problem. Studying clustering in absence of an established MAC layer highlights the chicken-and-egg problem of the initialization phase. As argued above, a MAC layer (“chicken”) helps achieving a clustering (“egg”), and vice versa. In a newly deployed ad-hoc/sensor network, for example, there is no structure, i.e. there are neither “chickens” nor “eggs”. Consequently, none of the various existing dominating set algorithms helps in the initialization process of such a network, since they basically focus on the problem of computing the “egg”, given the “chicken”. Instead, we are interested in a simple and practical algorithm which quickly computes a clustering from scratch. Based on this initial clustering, the MAC layer can subsequently be established.

The problem of initializing and structuring radio networks is of great importance in practice. Even in a single-hop ad-hoc network such as Bluetooth and for a small number of devices/nodes, the initialization tends to be slow. In a multi-hop scenario with large number of nodes, the time consumption for establishing a reasonable communication pattern increases even further.

In this paper, we give a new clustering algorithm which can be used for the initialization of unstructured multi-hop radio networks lacking an established MAC layer. We start by modelling the conditions of such a network after its deployment. In particular, we make the following model assumptions:

- The network is a multi-hop network, that is, there exist nodes that are not within their mutual transmission range. Being multi-hop complicates things since some of the neighbors of a sending node may receive a transmission, while others are experiencing interference from other senders and do not receive the transmission. Further, the hidden terminal problem precludes dominating set algorithms that are based on building a maximal independent set [2, 26, 27].

- The nodes do not feature a reliable collision detection mechanism. In particular, nodes are not able to distinguish between the situation in which two or more neighbors are sending and the situation in which no neighbor is sending. In many scenarios (particularly when considering the lack of an established MAC protocol) not assuming any collision detection mechanism is realistic. Nodes may be tiny sensors in a sensor network [1] where equipment is restricted to the minimum due to limitations in energy consumption, weight, or cost [28]. We would like to note that a sending node itself does not have a collision detection mechanism either, that is, a sender does not know how many (if any at all!) neighbors have received its transmission correctly. Naturally, algorithms without collision detection tend to be less efficient than algorithms with collision detection. Note that the absence of a reliable collision detection mechanism prevents us from using protocols such as Busy Tone Multiple Access (BTMA) [25] or Dual Busy Tone Multiple Access (DBTMA) [8].

- Our model allows nodes to wake-up asynchronously. In a multi-hop environment, it is realistic to assume that some nodes wake up (e.g. become deployed, or switched on) later than others. Consequently, nodes do not have access to a global clock. It is important to observe the implications of asynchronous wake-up. If all nodes started the algorithm simultaneously, we could easily assume an ALOHA kind of MAC-layer in which each node sends with probability θ(1/n). It is well known that this approach leads to a quick and simple communication scheme on top of which existing dominating set algorithms can be used. If nodes wake-up asynchronously, however, the same approach results in an expected linear runtime if only one single node wakes-up for a long time. More sophisticated algorithms are required to guarantee polylogarithmic runtime in case of asynchronous wake-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.

In this paper, we show that even in this harsh model, a good clustering can be computed efficiently. We present a randomized algorithm which computes an asymptotically optimal clustering in polylogarithmic time only. Since the algorithm even works in absence of any MAC layer, it has practical relevance particularly in scenarios in which traditional dominating set algorithms fail. We also provide simulations showing the efficiency and practicability of the algorithm.

The paper is organized as follows. Section 2 gives an overview over relevant previous work. Section 3 introduces our model as well as some well-known facts. In Sections 4 and 5 the algorithm is developed and analyzed using multiple communication channels. The subsequent Section 6 extends our analysis to the single-channel case. Section 7 presents
the results of our simulations before the paper is concluded in Section 8.

2 Related Work

The problem of finding a minimum dominating set has been proven to be NP-hard [12, 18]. Furthermore, it has been shown in [10] that the best possible approximation ratio for this problem is \( \ln \Delta \) where \( \Delta \) is the highest degree in the graph, unless \( \text{NP} \) has deterministic \( n^{O(\log \log n)} \)-time algorithms. For Unit Disk Graphs, the problem remains NP-hard, but constant factor approximations are possible.

Several distributed algorithms have been proposed, both for general graphs [16, 19, 21, 29] and the Unit Disk Graph [2, 11, 26]. Two of the most recent additions to the literature of computing dominating sets in the context of clustering networks are [19] and [11]. In [19], a distributed linear program is used to compute a dominating set of expected size \( O(k^2/\log \Delta |DS_{OPT}|) \) in \( k^2 \) rounds with \( \Delta \) being the maximum degree in an arbitrary graph. The algorithm proposed in [11] achieves constant approximation in time \( O(\log \log n) \) in Unit Disk Graphs, but relies on the ability of nodes to sense the distance to neighboring nodes. The algorithms proposed in [2, 26] make use of the fact that a maximal independent set (MIS) is a constant approximation to the minimum dominating set. Assuming that each node knows the IDs of all its neighbors and that collision-free point-to-point connections between two neighbors are established, the distributed computation of a MIS is relatively straight-forward. If MAC layer issues are considered, however, computing a MIS suddenly becomes a difficult task in view of the hidden-terminal problem. As mentioned in the introduction, all the above algorithms assume point-to-point connections between neighboring nodes, i.e. they build on top of an functioning MAC layer.

A model similar to the one used in this paper has previously been studied in the context of analyzing the complexity of broadcasting in multi-hop radio networks, e.g., [4, 9]. A striking difference to our model is that throughout the literature on broadcast in radio networks, synchronous wake-up is considered, i.e. all nodes have access to a global clock and start the algorithm simultaneously.

A model featuring asynchronous wake-up has been studied in recent papers on the so-called wake-up problem in single-hop networks [13, 17]. In comparison to our model, these papers define a much weaker notion of asynchrony. Particularly, it is assumed that sleeping nodes are woken up by a successfully transmitted message. In a single-hop network, the problem of waking up all nodes reduces to analyzing the number of time-slots until one message is successfully transmitted, because once this message is transmitted without collision, all nodes will be woken up. While this definition of asynchrony leads to theoretically interesting problems and algorithms, it does not closely reflect reality.

3 Model

In this section we introduce our model which will be used throughout the rest of this paper. We believe that the model is on the one hand strong enough to derive mathematically precise results, but on the other hand close enough to reality to actually have practical merit.

We consider multi-hop radio networks without collision detection. 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. Further, in Sections 4 and 5, we assume that nodes have three independent communication channels \( \Gamma_1, \Gamma_2, \) and \( \Gamma_3 \) which may be realized with an FDMA scheme. Having three communication channels simplifies the analysis, but it is not a fundamental necessity to obtain our results. We prove in Section 6 that we can achieve the same approximation-ratio even with a single communication channel in polylogarithmic time.

Nodes may wake up asynchronously at any time. Our model is more general than the often studied synchronous model, in which all nodes wake up and start their local algorithm at the same time. As mentioned in Section 1, asynchrony has a considerable impact on the algorithm’s design. We call a node sleeping before its wake-up, and active thereafter. Sleeping nodes can neither send nor receive any messages.

We consider the well known Unit Disk Graph (UDG) to model the network. In a UDG \( G = (V, E) \), there is an edge \((u, v) \in E \) iff the Euclidean distance between \( u \) and \( v \) is at most 1. In Section 5, we show how our algorithm can be extended to work in more general models as well. It is important to note, however, that due to asynchronous wake-up, some nodes may still be asleep, while others are already sending. Therefore, at any time, there may be sleeping nodes which do not take part in the communication in spite of their being within the transmission range of a sending node.

Nodes do not have any a-priori knowledge about the topology of the network. However, they do have an upper bound \( \bar{n} \) for the total number of nodes \( n = |V| \) in the graph. While \( n \) is unknown, all nodes have the same estimate \( \hat{n} \). It has been shown in [17] that without any estimate of \( n \), every algorithm requires at least time \( \Omega(n/\log n) \) until one single message can be transmitted without collision. In practice, the number of nodes in a network may not be known exactly, but it can roughly be estimated in advance.
For the sake of simplicity, we assume — for the analysis of the algorithm — that time is divided into time-slots. However, we attach great importance to the observation that our algorithm does not rely on synchronized time-slots in any way. Since nodes do not access to a global clock and synchronizing time-slots is an expensive task, such an assumption would be highly unrealistic. In this paper, it is solely for the purpose of analyzing the algorithm that we assume slotted channels. This simplification of the analysis is justified due to the standard trick which has been introduced in the analysis of slotted vs. unslotted ALOHA [22]. In [22], it is shown that the realistic unslotted case and the idealized slotted case differ only by a factor of 2. The basic intuition is that a single packet can cause interference in no more than two successive time-slots. Similarly, by analyzing our algorithm in an “ideal” setting with synchronized time-slots, we obtain a result which is only by a constant factor better as compared to the more realistic unslotted setting.

In each time-slot, a node can either send or not send. A node receives a message in a time-slot only if exactly one node in its neighborhood has sent a message in this time-slot. The variables $p_k$ and $q_k$ are the probabilities that node $k$ sends a message in a given time-slot on channel $\Gamma_1$ and $\Gamma_2$, respectively. Unless otherwise stated, we use the term sum of sending probabilities to refer to the sum of sending probabilities on channel $\Gamma_1$.

We conclude this section by providing two facts, the first of which has been proven in [17] and the second can be found in standard mathematical textbooks.

**Fact 3.1.** Given a set of probabilities $p_1, \ldots, p_n$ with $\forall i : p_i \in [0, \frac{1}{2}]$, the following inequalities hold:

$$\left(\frac{1}{4}\right) \sum_{k=1}^{n} p_k \leq \prod_{k=1}^{n} (1 - p_k) \leq \left(\frac{1}{e}\right) \sum_{k=1}^{n} p_k.$$

**Fact 3.2.** For all $n$, $t$, such that $n \geq 1$ and $|t| \leq n$,

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

### 4 Algorithm

A node starts executing the dominator algorithm (Algorithm 1) upon waking up. In the first phase (lines 1 to 5), nodes wait for messages (on all channels) without sending themselves. The reason is that nodes waking up late should not interfere with already existing dominators. Thus, a node first listens for existing dominators in its neighborhood for some time before actively trying to become dominator itself. In particular, we will choose the parameter $\delta$ as to ensure that a awakening node, which is already within an existing dominators transmission range, does not become dominator itself.

The main part of the algorithm (starting in line 6) works in rounds, each of which contains $\delta$ time-slots. As described in Section 3, these time-slots are not required to be synchronized between the nodes. In every time-slot, a node sends with probability $p$ on channel $\Gamma_1$. Starting from a very small

---

**Algorithm 1 Dominator Algorithm**

```
decided := false;
dominator := false;
upon wake-up do:
  1: for $j := 1$ to $\delta \cdot \lceil\log \hat{n}\rceil$ by $1$ do
  2: if message received then
  3: decided := true;
  4: fi
  5: od
  6: for $j := \lceil\log \hat{n}\rceil$ to $0$ by $-1$ do
  7: $p := 1 / (2^{j+\beta})$;
  8: for $i := 1$ to $\delta$ by $1$ do
  9: $b_i^{(1)} := 0$; $b_i^{(2)} := 0$; $b_i^{(3)} := 0$;
  10: if not decided then
  11: $b_i^{(1)} := 1$, with probability $p$;
  12: if $b_i^{(1)} = 1$ then
  13: dominator := true;
  14: else if message received then
  15: decided := true;
  16: fi
  17: fi
  18: if dominator then
  19: $b_i^{(2)} := 1$, with probability $q$;
  20: $b_i^{(3)} := 1$, with probability $q / \log n$;
  21: fi
  22: if $b_i^{(1)} = 1$ then send message on $\Gamma_1$ fi
  23: if $b_i^{(2)} = 1$ then send message on $\Gamma_2$ fi
  24: if $b_i^{(3)} = 1$ then send message on $\Gamma_3$ fi
  25: od
  26: if not decided then
  27: dominator := true;
  28: decided := true;
  29: fi
  30: if dominator then
  31: loop
  32: send message on $\Gamma_2$, with probability $q$;
  33: send message on $\Gamma_3$, with probability $q / \log n$;
  34: end loop
  35: fi
```
value, this sending probability $p$ is doubled (lines 6 and 7) in every round. When sending its first message, a node becomes dominator and, in addition to its sending on channel $\Gamma_1$ with probability $p$, it starts sending on channels $\Gamma_2$ and $\Gamma_3$ with probability $q$ and $q/\log n$, respectively. Once a node becomes dominators, it will remain so for the rest of the algorithm. For the algorithm to work properly, we must prevent the sum of sending probabilities on channel $\Gamma_1$ from reaching too high values. Otherwise, too many collisions will occur, leading to a large number of dominators. Hence, upon receiving its first message (without collision) on any of the channels, a node becomes decided and stops sending on $\Gamma_1$. Being decided means that the node is covered by a dominator and consequently, the node stops competing to become dominator itself by sending on $\Gamma_1$.

Thus, the basic intuition is that nodes, after some initial listening period, compete to become dominators by exponentially increasing their sending probability on $\Gamma_1$. Note that in light of asynchronous wake-up, this exponential increase is indispensable in order to achieve sublinear running time. Channels $\Gamma_2$ and $\Gamma_3$ then ensure that the number of further dominators emerging in the neighborhood of an already existing dominator remains small.

The parameters $q$, $\beta$, and $\delta$ of the algorithm are chosen as to optimize the results and guarantee that all claims hold with high probability:

$$q := \frac{1}{2^\delta \cdot \lceil \log \hat{n} \rceil}, \quad \delta := \left\lceil \frac{\log(\hat{n})}{\log(503/502)} \right\rceil, \quad \beta := 6. \quad (1)$$

The parameter $\delta$ is chosen large enough to ensure that with high probability, there is a round in which at least one competing node will send without collision. The parameter $q$ is chosen such that during the first $\delta \cdot \lceil \log \hat{n} \rceil$ “waiting time-slots”, a new node will receive a message from an existing dominator. Defining $q$ too small or too large could lead to the undesirable situation in which nodes become dominator in spite of their being covered at the time of their waking up. Finally, $\beta$ maximizes the probability of a successful execution of the algorithm.

Correctness of the algorithm and time-complexity (defined as the number of time-slots of a node between wake-up and decision) follow immediately:

**Theorem 4.1.** The algorithm computes a correct dominating set.

*Proof.* Every node which has not received a message from a dominator at the end of the algorithm will decide to become a dominator in line 28.

**Theorem 4.2.** Every node decides whether or not to become dominator in time $O(\log^2 \hat{n})$.

*Proof.* The first for-loop is executed $\delta \cdot \lceil \log \hat{n} \rceil$ times. The two nested loops of the algorithm are executed $\lceil \log \hat{n} \rceil + 1$ and $\delta$ times, respectively. At the end of these two loops, all remaining undecided nodes decide to become dominator. Let $t$ be the number of time-slots from wake-up until the final decision. Using the definition of $\delta$, we get

$$t \leq 2 \cdot (\lceil \log \hat{n} \rceil + 1) \cdot \delta \in O(\log^2 \hat{n}).$$

Note that the upper bound $\hat{n}$ does not have to be particularly tight in order to obtain a good time-complexity. If, for instance, $\hat{n} \leq n^\lambda$ for a given $\lambda \geq 1$, the time-complexity becomes only $O(\lambda^2 \log^2 n)$.

### 5 Analysis

This section contains the main theoretical contribution of this paper. It shows that the expected number of dominators in the network is within $O(1)$ of an optimal solution. As mentioned in Section 3, we can simplify the analysis by assuming all nodes operate with synchronized time-slots, such that the slot boundaries are perfectly aligned and a single packet transmission may cause a collision in exactly one time-slot. As shown in Section 3, this idealized analysis yields a result which is better only by a small constant factor as compared to the realistic unsynchronized setting.
We cover the plane with circles $C_i$ of radius $r = 1/2$ by a hexagonal lattice shown in Figure 1. Let $D_i$ be the circle centered at the center of $C_i$ having radius $R = 3/2$. It can be seen in Figure 1 that $D_i$ is (fully or partially) covering 19 smaller circles $C_j$. Note that every node in a circle $C_i$ can hear all other nodes in $C_i$. Nodes outside $D_i$ are not able to cause a collision in $C_i$.

We first give a broad outline of the proof which contains four major steps, from Lemma 5.3 to Lemma 5.7. First, we bound the sum of sending probabilities in a circle $C_i$. This leads to an upper bound on the number of collisions in a circle before at least dominator emerges. Thirdly, we give a probabilistic bound on the number of sending nodes per collision. In the last step, we then show that nodes waking up late in an already covered circle do not become dominator. More specifically, we show that all these claims hold with high probability.

We start with a lemma showing that it is sufficient to assume $\hat{n} = n$ in order to derive an upper bound for the expected number of dominators in the network.

**Lemma 5.1.** Assume that in the case of $\hat{n} = n$, the expected number of dominators of our algorithm is $d$. If the real value of $n$ is smaller than $\hat{n}$, the expected number of dominators is at most $d$.

**Proof.** The proof works by contradiction. Assume that there is a situation in which having $n' < \hat{n}$ nodes leads to more than $d$ expected dominators. Further assume that an adversary is controlling the wake-up schedule of each node. Consequently, in the case of $n = \hat{n}$ nodes, the adversary could have chosen to let $n - n'$ nodes sleep infinitely long, so that the two cases are not distinguishable anymore. For that reason, the expected number of dominators must be equal in both cases, which contradicts the assumption.

We need the following two definitions for our proofs.

**Definition 5.1.** Consider a circle $C_i$. Let $t$ be a time-slot in which a message is sent by a node $v \in C_i$ on channel $\Gamma_1$ and received (without collision) by all other nodes in $C_i$. We say that circle $C_i$ clears itself in time-slot $t$. Let $t_0$ be the first such time-slot. We say that circle $C_i$ terminates itself in time-slot $t_0$. For all time-slots $t' \geq t_0$, we call $C_i$ terminated.

**Definition 5.2.** Consider a circle $C_i$. Let $s(t) := \sum_{k \in C_i} p_k$ be the sum of sending probabilities on channel $\Gamma_1$ in $C_i$ at time $t$. We define the time slot $t_i^j$ so that for the $j^{th}$ time in $C_i$, we have

$$s(t_i^j - 1) < \frac{1}{2^j} \land s(t_i^j) \geq \frac{1}{2^j}.$$

Therefore, $t_i^0$ is the time-slot in which the sum of sending probabilities in $C_i$ exceeds $\frac{1}{2^j}$ for the first time and $t_i^j$ is the time-slot in which this threshold is surpassed for the $j^{th}$ time in $C_i$. The following lemma bounds the sum of sending probabilities in a circle.

**Lemma 5.2.** For all time-slots $t' \in [t_i^0 \ldots t_i^j + \delta - 1]$, the sum of sending probabilities in $C_i$ is bounded by

$$\sum_{k \in C_i} p_k \leq \frac{3}{2^j}.$$

**Proof.** According to the definition of $t_i^j$, the sum of sending probabilities $\sum_{k \in C_i} p_k$ at time $t_i^j - 1$ is less than $\frac{1}{2^j}$. All nodes which are active at time $t_i^j$ will double their sending probability $p_k$ exactly once in the following $\delta$ time-slots (steps 6 and 7 of the algorithm). Previously inactive nodes may wake up during that interval. There are at most $n$ of such newly active nodes and each of them will send with the initial sending probability $\frac{1}{2^j}$ in the given interval. In $[t_i^0 \ldots t_i^j + \delta - 1]$, we get

$$\sum_{k \in C_i} p_k \leq 2 \cdot \frac{1}{2^j} + \sum_{k \in C_i} \frac{1}{2^j n} \leq 2 \cdot \frac{1}{2^j} + \frac{n}{2^j n} \leq \frac{3}{2^j}.$$

Using the above lemma, we can formulate a probabilistic bound on the sum of sending probabilities in a circle $C_i$. Intuitively, we show that before the bound can be surpassed, $C_i$ does either clear itself or some nodes in $C_i$ become decided such that the sum of sending probabilities decreases.

**Lemma 5.3.** The sum of sending probabilities of nodes in a circle $C_i$ is bounded by

$$\sum_{k \in C_i} p_k \leq \frac{3}{2^j}$$

with probability at least $1 - o\left(\frac{1}{n^2}\right)$. The bound holds for all $C_i \in G$ with probability at least $1 - o\left(\frac{1}{n^2}\right)$.

**Proof.** The proof is by induction over all time-slots $t_i^j$ in ascending order. Let $t' := t_i^j$ be the very first such time-slot in the network. Lemma 5.2 states that the sum of sending probabilities in $C_i$ is bounded by $\frac{1}{2^j}$ in the interval $[t' \ldots t' + \delta - 1]$. We now show that in this interval, the circle $C_i$ either clears itself or the sum of sending probabilities falls back below $\frac{1}{2^j}$ with high probability.

If some of the active nodes in $C_i$ receive a message from a neighboring node, the sum of sending probabilities may fall
back below $\frac{1}{2^\gamma}$. In this case, the sum does obviously not exceed $\frac{1}{2^\gamma}$.

If the sum of sending probabilities does not fall back below $\frac{1}{2^\gamma}$, the following two inequalities hold for the duration of the interval $[t' \ldots t' + \delta - 1]$:

\[
\frac{1}{2^\gamma} \leq \sum_{k \in C_i} p_k \leq \frac{3}{2^\gamma} : \text{in } C_i \quad (2)
\]

\[
0 \leq \sum_{k \in C_j} p_k \leq \frac{3}{2^\gamma} : \text{in } C_j \in D_i, i \neq j. \quad (3)
\]

The second inequality holds because $t'$ is the very first time-slot in which the sum of sending probabilities exceeds $\frac{1}{2^\gamma}$. Hence, in each $C_j \in D_i$, the sum of sending probabilities is at most $\frac{1}{2^\gamma}$ in the interval $[t' \ldots t' + \delta - 1]$. (Otherwise, one of these circles would have reached $\frac{1}{2^\gamma}$ before circle $C_i$ and $t'$ is not the first time-slot considered).

We will now compute the probability that $C_i$ clears itself within the interval $[t' \ldots t' + \delta - 1]$. Circle $C_i$ clears itself when exactly one node in $C_i$ sends and no other node in $D_i \setminus C_i$ sends. The probability $P_0$ that no node in any neighboring circle $C_j \in D_i, j \neq i$ sends is

\[
P_0 = \prod_{C_j \in D_i, j \neq i} \prod_{k \in C_j} (1 - p_k)
\]

\[
\geq \prod_{C_j \in D_i, j \neq i} \left( \frac{1}{4} \right)^{k \in C_j} \geq \left[ \left( \frac{1}{4} \right)^{2^\gamma} \right]^{18} . \quad (4)
\]

Let $P_{\text{suc}}$ be the probability that exactly one node in $C_i$ sends:

\[
P_{\text{suc}} = \sum_{k \in C_i} \left( p_k \cdot \prod_{l \in C_i, l \neq k} (1 - p_l) \right)
\]

\[
\geq \sum_{k \in C_i} p_k \cdot \prod_{l \in C_i} (1 - p_l)
\]

\[
\geq \text{Fact 3.1} \sum_{k \in C_i} p_k \cdot \left( \frac{1}{4} \right)^{k \in C_i} \geq \frac{1}{2^\gamma} \left( \frac{1}{4} \right)^{2^\gamma} . \quad (5)
\]

The last inequality holds because function (5) is strictly increasing in $[\frac{1}{2^\gamma}, \frac{1}{2^\gamma}]$.

The probability $P_i$ that exactly one node in $C_i$ and no other node in $D_i$ sends is therefore given by

\[
P_i = P_0 \cdot P_{\text{suc}} \geq \left[ \left( \frac{1}{4} \right)^{2^\gamma} \right]^{18} \cdot \frac{1}{2^\gamma} \left( \frac{1}{4} \right)^{2^\gamma} \]

\[
= \frac{2^{9/32}}{256} .
\]

$P_i$ is a lower bound for the probability that $C_i$ clears itself in a time-slot $t \in [t' \ldots t' + \delta - 1]$. The reason for choosing $\beta = 6$ is that this value maximizes $P_i$.

We can now compute the probability $P_{\text{term}}$ that circle $C_i$ does not clear itself during the entire interval:

\[
P_{\text{term}} \leq \left( 1 - \frac{2^{9/32}}{256} \right)^{\delta} \leq \frac{1}{n^{\gamma}} \in o\left( \frac{1}{n^{\gamma}} \right) .
\]

We have thus shown that within the interval $[t' \ldots t' + \delta - 1]$, the sum of sending probabilities in $C_i$, either falls back below $\frac{1}{2^\gamma}$ or $C_i$ clears itself with high probability.

So far, we have only shown that the lemma holds for the very first $t'_i$ (i.e. $t'$). For the induction step, we consider an arbitrary $t'_i$. Due to the induction hypothesis, we can assume that all previous such time-slots have already been dealt with. In other words, all previously considered time-slots $t'_i$ have either lead to a clearance of circle $C_i$ or the sum of probabilities in $C_i$ has decreased below the threshold $\frac{1}{2^\gamma}$. Immediately after a clearance, the sum of sending probabilities in a circle $C_i$ is at most $\frac{1}{2^\gamma}$, which is the sending probability in the last round of the algorithm. This is true because only one node in the circle remains undecided. All others will stop sending on channel $\Gamma_1$. By Lemma 5.2, the sum of sending probabilities in all neighboring circles (both the cleared and the not cleared ones) is bounded by $\frac{1}{2^\gamma}$ in the interval $[t'_i \ldots t'_i + \delta - 1]$ (otherwise, this circle would have been considered before $t'_i$).

Therefore, we know that the bounds (2) and (3) hold with high probability. And consequently, the computation to show the induction step is the same as the one for the base case $t'$ and it also holds that $P_{\text{term}} \in o\left( \frac{1}{n^{\gamma}} \right)$.

Each step of the induction only holds with high probability. But, because there are $n$ nodes to be decided and at most $n$ circles $C_i$, the number of induction steps $t'_i$ is bounded by $n$. Hence, the probability that the lemma holds for all steps is at least $\left( 1 - o\left( \frac{1}{n^{\gamma}} \right) \right)^n \geq 1 - o\left( \frac{1}{n^{\gamma}} \right)$, which concludes the proof.

Using Lemma 5.3, we can now compute the expected number of dominators in each circle $C_i$. In the analysis, we will separately compute the number of dominators before and after the termination (i.e. the first clearance) of $C_i$. To prove our results, we will need three more lemmas.
Lemma 5.4. Let $C$ be the number of collisions (more than one node is sending in one time-slot on $\Gamma_1$) in a circle $C_i$. The expected number of collisions in $C_i$ before its termination is $E[C] < 6$. Further, $C < 7 \log n$ with probability at least $1 - o\left(\frac{1}{n^2}\right)$.

Proof. Only channel $\Gamma_1$ is considered in this proof. We assume that $C_i$ is not yet terminated and we define the following events

- $A$: Exactly one node in $D_i$ is sending
- $X$: More than one node in $C_i$ is sending
- $Y$: At least one node in $C_i$ is sending
- $Z$: Some node in $D_i \setminus C_i$ is sending

For the proof, we consider only rounds in which at least one node in $C_i$ sends. (There will be no new dominators in $C_i$ if no node sends.) We want to get a bound for the conditional probability $P[A \mid Y]$ that exactly one node in $D_i$ is sending and this one node is located in $C_i$. Using $P[Y \mid X] = 1$ and the fact that $Y$ and $Z$ are independent, we get

\[
P[A \mid Y] = P[X \mid Y] \cdot P[Z \mid Y]
\]
\[
= P[X \mid Y] \cdot P[Z] = (1 - P[X \mid Y]) \cdot (1 - P[Z])
\]
\[
= \left(1 - P[X] \cdot P[Y \mid X] \right) \cdot (1 - P[Z])
\]
\[
= \left(1 - \frac{P[X]}{P[Y]} \right) \cdot (1 - P[Z]). \tag{6}
\]

We can now compute bounds for the probabilities $P[X]$, $P[Y]$, and $P[Z]$:

\[
P[X] = 1 - \prod_{k \in C_i} (1 - p_k) - \sum_{k \in C_i} \left( p_k \prod_{l \in C_i, l \neq k} (1 - p_l) \right)
\]
\[
\leq 1 - \left(\frac{1}{4}\right) \sum_{k \in C_i} p_k - \sum_{k \in C_i} p_k \cdot \left(\frac{1}{4}\right) \sum_{k \in C_i} p_k
\]
\[
= 1 - \left(1 + \sum_{k \in C_i} p_k \right) \left(\frac{1}{4}\right) \sum_{k \in C_i} p_k. \tag{7}
\]
\[
P[Y] = 1 - \prod_{k \in C_i} (1 - p_k) \geq 1 - \left(\frac{1}{e}\right) \sum_{k \in C_i} p_k. \tag{8}
\]

The first inequality for $P[X]$ follows from Fact 3.1 and inequality (5). The inequality for $P[Y]$ also follows from Fact 3.1. In the proof for Lemma 5.3, we have already computed a bound for $P_b$, the probability that no node in $D_i \setminus C_i$ sends. Using this result, we can write $P[Z]$ as

\[
P[Z] = 1 - \prod_{C_j \in D_i \setminus C_i} \prod_{k \in C_j} (1 - p_k)
\]
\[
\leq \left(\frac{1}{4}\right)^6 \geq 1 - \left(\frac{1}{4}\right)^{18}. \tag{9}
\]

Plugging inequalities (7), (8), and (9) into equation (6) for $P[A \mid Y]$, it can be shown that the term $P[X] / P[Y]$ is maximized for $\sum_{k \in C_i} p_k = \frac{1}{2}$ and therefore

\[
P[A \mid Y] = \left(1 - \frac{P[X]}{P[Y]} \right) \cdot (1 - P[Z])
\]
\[
\geq \left(1 - \frac{1 - (1 + \frac{3}{18}) \left(\frac{1}{4}\right)^{\frac{3}{18}}}{1 - (\frac{1}{2})^{\frac{3}{18}}} \right) \left[\frac{1}{4}\right]^{\frac{3}{18}} \geq 0.18.
\]

This shows that whenever a node in $C_i$ sends, $C_i$ terminates with constant probability at least $P[A \mid Y]$. This allows us to compute the expected number of collisions in $C_i$ before the termination of $C_i$ as a geometric distribution:

\[
E[C] = \frac{1}{P[A \mid Y]} \leq 6.
\]

The high probability result can be derived as

\[
P[C \geq 7 \log n] = (1 - P[A \mid Y])^{7 \log n} \in O\left(\frac{1}{n^2}\right).
\]

So far, we have shown that the number of collisions before the clearance of $C_i$ is constant in expectation. The next lemma shows that the number of new dominators per collision is also constant. In a collision, each of the sending nodes may already be dominator. Hence, if we assume that every sending node in a collision is a new dominator, we obtain an upper bound for the true number of new dominators.

Lemma 5.5. Let $D$ be the number of nodes in $C_i$ sending in a time-slot and let $\Phi$ denote the event of a collision. Given the occurrence of a collision, the expected number of sending nodes (i.e. new dominators) is $E[D \mid \Phi] \in O(1)$. Furthermore, $P[D \leq \log n \mid \Phi] \geq 1 - o\left(\frac{1}{n^2}\right)$.

Proof. Let $m$, $m \leq n$, be the number of nodes in $C_i$ and $N = \{1 \ldots m\}$. $D$ is a random variable denoting the number of sending nodes in $C_i$ in a given time-slot. We define $A_k :=
\[ P[D = k] \] as the probability that exactly \( k \) nodes send. For example, the probability that exactly two nodes in \( C_i \) send is 

\[ A_{k-1}^i: \]

\[ A_2 = \sum_{k \in C_i} \left( p_k \cdot \prod_{l \in C_i, i \neq k} (1 - p_l) \right). \]

Let \( \binom{N}{k} \) be the set of all \( k \)-subsets of \( N \) (subsets of \( N \) having exactly \( k \) elements). We define \( A'_k \) as

\[ A'_k := \sum_{Q \in \binom{N}{k}} \prod_{i \in Q} \frac{p_i}{1 - p_i}. \]

We can then write \( A_k \) as

\[ A_k = \sum_{Q \in \binom{N}{k}} \left( \prod_{i \in Q} p_i \cdot \prod_{i \notin Q} (1 - p_i) \right) = \left( \sum_{Q \in \binom{N}{k}} \prod_{i \in Q} \frac{p_i}{1 - p_i} \right) \cdot \prod_{i = 1}^m (1 - p_i) = A'_k \cdot \prod_{i = 1}^m (1 - p_i). \]

We now continue the proof of Lemma 5.5. The conditional expected value \( E[D \mid \Phi] \) is

\[ E[D \mid \Phi] = \sum_{i = 0}^m (i \cdot P[D = i \mid \Phi]) = \sum_{i = 2}^m B_i. \quad (11) \]

where \( B_i \) is defined as \( i \cdot P[D = i \mid \Phi] \). For \( i \geq 2 \), the conditional probability reduces to

\[ P[D = i \mid \Phi] = \frac{P[D = i]}{P[\Phi]}. \quad (12) \]

In the next step, we consider the ratio between two consecutive terms of sum (11).

\[ \frac{B_{k-1}}{B_k} = \frac{(k - 1) \cdot P[D = k - 1 \mid \Phi]}{k \cdot P[D = k \mid \Phi]} = \frac{(k - 1) \cdot P[D = k - 1]}{k \cdot P[D = k]} \]

\[ = \frac{(k - 1) \cdot A_{k-1}}{k \cdot A_k} = \frac{(k - 1) \cdot A'_k}{k \cdot A'_k}. \]

It follows from Fact 5.6, that each term \( B_k \) can be upper.
bounded by

\[ B_k = \frac{kA_k'}{(k-1)A_{k-1}'} \cdot B_{k-1} \leq \frac{1}{k-1} \sum_{i=1}^{m} \frac{p_i}{1-p_i} \cdot B_{k-1} \leq 2 \cdot \sum_{i=1}^{m} p_i \cdot B_{k-1}. \]

The last inequality follows from \( \forall i : p_i < 1/2 \) and \( p_i \leq 1/2 \Rightarrow \frac{p_i}{1-p_i} \leq 2p_i. \)

From the definition of \( B_k \), it naturally follows that \( B_2 \leq 2. \) Furthermore, we can bound the sum of sending probabilities \( \sum_{i=1}^{m} p_i \) using Lemma 5.3 to be less than \( \frac{3}{2^2} \). We can thus sum up over all \( B_i \) recursively in order to obtain \( \mathbb{E} [D \mid \Phi] \):

\[ \mathbb{E} [D \mid \Phi] = \sum_{i=2}^{m} B_i \leq 2 + \sum_{i=3}^{m} \left[ \frac{2}{(i-1)!} \left( \frac{6}{2^2} \right)^{i-2} \right] = 2 + \frac{6}{2^2} + \frac{1}{3} \left( \frac{6}{2^2} \right)^2 + \ldots \leq 2 + \frac{3}{2^2} \leq 2.11. \]

In order to derive the high probability result, we solve the recursion of Fact 5.6 and obtain

\[ A_k' \leq \frac{1}{k!} \left( \sum_{i=1}^{m} \frac{p_i}{1-p_i} \right)^k. \quad (13) \]

The probability \( P_+ := \mathbb{P} [D \geq \log n \mid \Phi] \) is

\[ P_+ = \sum_{k=\lceil \log m \rceil}^{m} A_k \leq \sum_{k=\lceil \log m \rceil}^{m} A_k'. \]

\[ \leq \text{Eq } (13) \sum_{k=\lceil \log m \rceil}^{m} \left[ \frac{1}{k!} \left( \sum_{i=1}^{m} \frac{p_i}{1-p_i} \right)^k \right] \]

\[ \leq \text{Lemma } 5.3 \left( m - \lceil \log m \rceil \right) \cdot \frac{2 \cdot \sum_{i=1}^{m} p_i}{\lceil \log m \rceil} \cdot \left( \log m \right)! \]

\[ \leq m \cdot \left( \frac{6}{2^2} \right)^{\lceil \log m \rceil} \leq O \left( \frac{1}{m^2} \right). \]

\[ \square \]

The last key lemma shows that the expected number of new dominators after the termination of circle \( C_i \) is also constant.

**Lemma 5.7.** Let \( A \) be the number of new dominators after the termination of \( C_i \). Then, \( A \in O(1) \) with high probability.

**Proof.** We define \( B \) and \( B_i \) as the set of dominators in \( D_i \) and \( C_i \), respectively. Immediately after the termination of \( C_i \), only one node in \( C_i \) remains sending on channel \( \Gamma_j \), because all others will be decided. By Lemmas 5.4 and 5.5, we can bound the number of dominators in a \( C_i \) with high probability as \( |B_i| \leq \tau \log^2 n \) for a small constant \( \tau' \). Potentially, all \( C_j \in D_i \) are already terminated and therefore, \( 1 \leq |B| \leq 19 \cdot \tau' \log^2 n \) with high probability. For simplicity, we write \( \tau := 19 \cdot \tau' \).

We now distinguish the two cases \( 1 \leq |B_i| \leq \tau \log n \) and \( \tau \log n < |B_i| \leq \tau \log^2 n \). We consider channels \( \Gamma_2 \) and \( \Gamma_3 \) in the first and second case, respectively. Particularly, we will show that in either case, a new node will receive a message on one of the two channels with high probability during the waiting period at the beginning of the algorithm.

First, consider case one, i.e. \( 1 \leq |B_i| \leq \tau \log n \). The probability \( P_0 \) that one dominator is sending alone on channel \( \Gamma_2 \) is

\[ P_0 = |B| \cdot q \cdot (1-q)^{|B|-1}. \]

Note that this is a concave function in \( |B| \). For \( |B| = 1 \), we get \( P_0 = q = \frac{6}{2^2 \log m} \) and for \( |B| = \tau \log n, n \geq 2 \), we
have

\[ P_0 = \frac{\tau \log n}{2^\beta |\log n|} \left(1 - \frac{1}{2^\beta |\log n|}\right)^{\tau \log n - 1} \]
\[ \geq \frac{\tau}{2^\beta} \cdot \left(1 - \frac{\tau/2^\beta}{\log n}\right)^{\tau \log n} \]
\[ \geq \frac{\tau}{2^\beta} e^{-\frac{\tau}{2^\beta}} \left(1 - \frac{1}{\log n}\right)^{\tau \log n} \]
\[ \geq \frac{\tau}{2^\beta} e^{-\frac{\tau}{2^\beta}} \left(1 - \frac{1}{2^\beta}\right) \in O(1). \]

A newly awakened node in a terminated circle \( C_i \) will not send during the first \( \delta \cdot |\log n| \) rounds. If during this period, the node receives a message (without collision) from an existing dominator, it will become decided and hence, will not become dominator. The probability \( P_{no} \) that such an already covered node does not receive any messages from an existing dominator during the first \( \delta \cdot |\log n| \) rounds is asymptotically bounded by

\[ P_{no} \leq \left(1 - \frac{1}{2^\beta |\log n|}\right)^{\delta \cdot |\log n|} \leq e^{-\frac{\delta}{2^\beta}} \in O(1). \]

This shows that the probability of new dominators emerging in \( C_i \) after the termination of \( C_i \) is very small and with high probability, the number of new dominators is bounded by a constant in this case.

The analysis in the second case follows along the same lines. For \(|B| = \tau \log n\), we get

\[ P_0 = \frac{\tau \log n}{2^\beta |\log^2 n|} \left(1 - \frac{1}{2^\beta |\log^2 n|}\right)^{\tau \log n - 1} \]
\[ \geq \frac{\tau}{2^\beta |\log n|} \cdot \left(1 - \frac{\tau/2^\beta}{|\log n|}\right)^{\tau \log n} \]
\[ \geq \frac{\tau}{2^\beta |\log n|} e^{-\frac{\tau}{2^\beta |\log n|}} \left(1 - \frac{1}{2^\beta}\right) \in O(1). \]

For \(|B| = \tau \log^2 n\), it can be shown that \( P_0 \in O(1) \) and hence, the remainder of the analysis is analogous to the first case.

We are now ready to prove the following theorem.

**Theorem 5.8.** The expected number of dominators in circle \( C_i \) is \( E[D] \in O(1) \).

**Proof.** We consider a circle \( C_i \). By Lemma 5.4, the expected number of collisions before the termination of \( C_i \) is less than

\[ \log \frac{n}{\delta} \]
6 Single-Channel

In this section, we analyze the single-channel scenario. We prove that the constant approximation-ratio also holds in this case, by showing how each time-slot in the multi-channel model can be simulated by a number of time-slots in the single-channel model. The time complexity of the algorithm remains polylogarithmic.

Let \( s \) and \( t \) be time-slots in the single-channel and multi-channel model, respectively. We write \( \text{suc}(t) = 1 \) if a message is successfully transmitted in time-slot \( t \) and \( \text{suc}(t) = 0 \) otherwise.

**Lemma 6.1.** Time-slot \( t \) can be simulated with \( O(\log^3 n) \) time-slots \( s_i, i \in [1 \ldots 3\alpha \log^3 n] \), for a large enough constant \( \alpha \) such that \( \text{suc}(t) = 1 \Leftrightarrow \exists i : \text{suc}(s_i) = 1 \) with probability \( 1 - O(\frac{1}{n^{\alpha}}) \).

**Proof.** We first investigate the critical cases by analyzing the different sending possibilities which can occur in the multi-channel case (channels \( \Gamma_1, \Gamma_2, \text{ and } \Gamma_3 \)) and how they map to the single-channel model.

<table>
<thead>
<tr>
<th>( \Gamma_1 )</th>
<th>( \Gamma_2 )</th>
<th>( \Gamma_3 )</th>
<th>Multi</th>
<th>Single</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>( \geq 2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>( \geq 2 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>( \geq 2 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>( \geq 2 )</td>
<td>( \geq 2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>( \geq 2 )</td>
<td>( \geq 2 )</td>
<td>( \geq 2 )</td>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
</tbody>
</table>

The table shows some of the possible cases. The columns \( \Gamma_1, \Gamma_2, \text{ and } \Gamma_3 \) denote how many senders are sending on these channels in a given time-slot. The next two columns show whether or not the transmission was successful, depending on the number of channels used. For the single-channel case, we assume that all senders sending on any channel are sending on a common channel \( \Gamma \). The critical cases are those in which a node receives a message in the multi-channel case, but does not receive it in the single-channel case, due to a collision.

When simulating three channels by a single channel, we must ensure that a message can be successfully transmitted (without collision) in these critical cases.

We write \( \text{send}(t) = 1 \) if a sender sends in time-slot \( t \) and \( \text{send}(t) = 0 \), otherwise. Further, we use the abbreviations \( \lambda := \alpha \log^3 n \) and \( p := 1/\log^2 n \). Each node simulates time-slot \( t \) by \( 3\lambda \) single-channel time-slots \( s_1 \ldots s_{3\lambda} \) in the following way:

\[
\text{send}(t) = 0 \Rightarrow \forall s_i \in [s_1 \ldots s_{3\lambda}] : \\
\text{send}(s_i) := 0 \\
\text{send}(t) = 1 \Rightarrow \forall s_i \in [s_1 \ldots s_\lambda, s_{2\lambda} \ldots s_{3\lambda}] : \\
\text{send}(s_i) := 0 \\
\text{send}(t) = 1 \Rightarrow \forall s_i \in [s_\lambda \ldots s_{2\lambda}] : \\
\text{send}(s_i) := \begin{cases} 
1, \text{ with probability } p \\
0, \text{ with probability } 1 - p
\end{cases}
\]

In words, each node which sends on \( \Gamma_1, \Gamma_2, \text{ or } \Gamma_3 \) in a time-slot \( t \) sends randomly with probability \( 1/\log^3 n \) in the \( \lambda \) time-slots \( [s_1 \ldots s_{3\lambda}] \) on channel \( \Gamma \). We call \( [s_\lambda \ldots s_{2\lambda}] \) sending period, \( [s_1 \ldots s_\lambda] \) and \( [s_{2\lambda} \ldots s_{3\lambda}] \) quiet periods.

Obviously, if there is more than one sender, they may choose the same or overlapping time-slots, which will lead to collisions. Unless there is at least one time-slot in which exactly one sender is sending, the message is not transmitted successfully. Thus, there is a non-zero probability that sending a message fails in the critical cases, as defined above. We will now show, however, that this probability becomes sufficiently small to make sure the algorithm works the same way as in the multi-channel case.

Let \( T \) be the set of sending nodes in time-slot \( t \). Due to asynchronous wake-up, we can not assume that the periods \( [s_1 \ldots s_{3\lambda}] \) of sending nodes \( \nu \in T \) are aligned. It is easy to observe, however, that the probability of a successful transmission is minimized when these periods are exactly aligned. If some nodes \( \nu \in T \) are in the sending period while others are in a quiet period, the probability of a successful transmission (exactly one node sends in a time-slot \( s \)) is bigger compared to the case when all sending nodes are in the sending period at the same time. Consequently, we only have to consider the case of perfect alignment between sending periods.

By Lemma 5.5, we know that the number of sending nodes on channel \( \Gamma_1 \) in a given time-slot does not exceed \( \log n \) with high probability. By Lemmas 5.4 and 5.5 we know that the number of nodes sending on \( \Gamma_2 \) and \( \Gamma_3 \) is bounded by \( \tau \log^2 n \).

The probability \( P_1 \) that exactly one node \( \nu \in T \) sends in a time-slot \( s \) is

\[
P_1 = \frac{|T|}{\log^2 n} \left( 1 - \frac{1}{\log^2 n} \right)^{|T|-1}
\]

Since this is a concave function, we again have to consider the cases \( |T| = 2 \) and \( |T| = 2\tau \log^2 n + 1 \). In the first case, the probability \( P_{no} \) that no message is successfully transmitted in
the entire sending period is

\[ P_{no} = (1 - P_1)^\lambda \]
\[ \leq \left( 1 - \frac{2}{\log^2 n} \left( 1 - \frac{1}{\log^2 n} \right)^\alpha \log^3 n \right) \]
\[ = \left( 1 - \frac{2 \log^2 n}{\log^2 n} \right)^{\frac{\alpha \log^3 n}{\log^2 n}} \]
\[ \leq e^{-2\alpha \log n} \in O\left( \frac{1}{n^{2\alpha}} \right). \]

As for the second case, \(|T| = 2\tau \log^2 n + 1\), we have

\[ P_{no} = (1 - P_1)^\lambda \]
\[ \leq \left( 1 - 2\tau \left( 1 - \frac{1}{\log^2 n} \right) \right)^{\alpha \log^3 n} \]
\[ = \left( 1 - 2\tau e^{-2\tau} \right)^{\alpha \log^3 n} \in O\left( \frac{1}{n^{2\alpha \log^2 n}} \right). \]

Since the same computation holds for all three channels, a message from each channel is successfully transmitted with high probability.

**Theorem 6.2.** The dominator algorithm in the single-channel model has time-complexity \(O(\text{polylog}(n))\). All critical steps are executed like in the multi-channel algorithm with probability at least \(1 - O\left( \frac{\log^5 n}{n^\alpha} \right)\), for some constant \(\alpha\).

**Proof.** The time-complexity follows directly from Theorem 4.2 and Lemma 6.1. In order to show correctness, we can compute the probability \(P\) that at least one step of the entire algorithm’s execution is not handled correctly by our single-channel simulation. In the multi-channel case, the algorithm’s execution takes at most \(O(1) \cdot n \log^2 n\) steps. The probability \(P\) that none of these critical steps fails is

\[ P \geq \left( 1 - \frac{1}{n^{2\alpha}} \right)^{O(1) \cdot n \log^2 n} \in 1 - O\left( \frac{2 \log^2 n}{n^{\alpha}} \right). \]

Thus, we have shown that even with a single communication channel, we can compute an asymptotically optimal clustering in polylogarithmic time. As for a constant approximation, algorithms with sublogarithmic running time appear to be impossible, future improvements may focus on reducing the logarithm’s power.

## 7 Simulation

In this section, we show the efficiency and practicability of the algorithm in a variety of test settings. We show how the different parameters influence the quality of the dominating set, as well as the runtime of individual nodes. We assume that \(n\) nodes are randomly distributed in a \(5 \times 5\) square. Each node’s transmission range is 1. Note that non-uniform distribution improves the simulation results.

The following parameters are used. \(n\) is the number of nodes in the network. \(\lambda\) denotes the factor used to multiply \(\log n\) when obtaining \(\delta\) of the algorithm. In other words, \(\delta := \lambda \log n\). In the analysis section of the algorithm, \(\lambda := 1/\log(\frac{1003}{502})\). The third parameter, \(p\), describes the nodes’ wake-up behavior. Let \(s\) be the number of sleeping nodes at time \(t\). In time-slot \(t + 1\), each sleeping node wakes up with probability \(\frac{p}{s}\). This yields an even distribution in the first \(p^{-1}\) time-slots. In the case \(p = 1\), all nodes wake up immediately and we have a synchronous wake-up behavior. For very small \(p\), the nodes wake up widely dispersed in time. For all our simulations, we assume \(\delta = n\). For \(\delta > n\), the number of dominators subsides and the runtime increases as mentioned subsequent to Theorem 4.2 and in Lemma 5.1.

Figures 2 and 3 relate the number of dominators to the number of nodes in the graph. In both cases, the y-axis denotes the average number of dominators in the neighborhood of a node. In Figure 2, all nodes wake up synchronously in the very first time-slot, while in Figure 3, they wake-up asynchronously. In both the synchronous and asynchronous case, the number of dominators converges to a constant of about 2, if the parameter \(\lambda\) is chosen large enough. Note that this indicates a dominating set of excellent quality. For \(\lambda \geq 10\), the difference to the value \(\lambda\) as computed in the analysis section, becomes negligibly small. If \(\lambda\) is chosen too small, however, the number of dominators in each node’s neighborhood may increase, particularly in the case of asynchronous wake-up. This increase is caused by a growing number of collisions as well as the fact that the listen-only phase at the beginning of the algorithm becomes too short.

Note that the larger \(\lambda\), the slower the sending probabilities reach large values and therefore, the runtime increases. Thus, it is desirable to choose \(\lambda\) as small as possible. The question is, how small can we choose \(\lambda\) before the quality of the resulting dominating set starts worsening. We study this issue in Figures 4 and 5. The charts relate the average number of dominators in the neighborhood of a node to the parameter \(\lambda\). It can be seen that the number of dominators reaches its low at around \(\lambda = 10\) in the asynchronous case (and already at \(\lambda = 5\) in the synchronous case) and remains constant thereafter. For practical purposes, it is therefore sufficient to run
the algorithm using $\lambda := 10$ (i.e. $\delta := 10 \log \hat{n}$) which yields a good runtime, as shown in Figure 6.

The y-axis of Figure 6 denotes the average number of time-slots passing between the wake-up of a node and its decision. For large $p$, the runtime increases asymptotically as $O(\log^2 n)$. For very small values of $p$ (i.e. for very asynchronous wake-up), the average runtime is much lower and does not significantly increase in the range $n \in [10 \ldots 100000]$. This runtime gap is natural considering that each node listens during the first $\delta \cdot \lceil \log \hat{n} \rceil$ time-slots without sending. In the case of synchronous wake-up, all nodes execute these time-slots simultaneously and they all end up waiting at the same time. Obviously, this increases the average runtime. In the asynchronous case, most nodes wake up at a time existing dominators are already sending. Most of these nodes will decide not to become dominator within the first $\delta \cdot \lceil \log \hat{n} \rceil$ rounds as shown in Lemma 5.7. The runtime gap between the synchronous and asynchronous case is therefore a direct consequence of the way the algorithm works.

Concluding the simulation section, we can state that for practical purposes, the parameter $\lambda$ can be chosen to be smaller than in the analysis section. Various simulations indicate that it is sufficient to set $\delta := 10 \log \hat{n}$. This improves the runtime while maintaining an excellent quality of the dominating set. The average runtime asymptotically increases in $O(\log^2 n)$ if wake-up is more-or-less synchronous and becomes significantly smaller the more the nodes’ wake-up is dispersed in time.

8 Conclusions

How can we efficiently compute a structure completely from nothing? In this paper, we have tried to provide an answer to this question by analyzing the initialization of multi-hop
radio networks, i.e., the transition from an unstructured to a structured network. We have motivated that the problem of bringing structure into a network (i.e., organizing an efficient medium access scheme) is strongly related to the problem of clustering in absence of an established MAC layer. We formulated a model which contains many of the harsh, but realistic characteristics of unstructured networks, including the hidden terminal problem, no reliable collision detection, and asynchronous wake-up. We have then proposed a novel randomized algorithm to solve the “chicken-and-egg” problem of the initialization. The algorithm computes an asymptotically optimal dominating set in polylogarithmic time without relying on any MAC layer support. In addition to these theoretical bounds, we showed the practicability of our algorithm by simulation.

Aspiring towards the goal of modelling reality as closely as possible, it would be desirable to restrict our model assumptions even further; particularly, we would like to drop the assumption that nodes know an upper bound for the total number of nodes. Unfortunately, the theoretical lower bound proven in [17] thwarts these hopes.

Comparing our algorithm with existing dominating set algorithms shows how significantly the absence of a functioning MAC layer in an unstructured network changes the task of clustering. Collisions may occur — due to the hidden terminal problem — and a maximal independent set cannot be computed easily. We believe that in order to understand the initialization process of multi-hop radio networks thoroughly, it is necessary to analyze the problem of clustering under a model which covers the fundamental properties of unstructured networks. We further believe that — due to its being fast and simple — the algorithm presented in this paper has practical relevance, particularly in newly deployed networks with autonomous nodes, such as ad-hoc and sensor networks.

References


