An efficient communication abstraction for dense wireless networks

An Efficient Communication Abstraction for Dense Wireless Networks

Magnús M. Halldórsson[*1], Fabian Kuhn[†2], Nancy Lynch[‡3], and Calvin Newport[§4]

1 **Reykjavik University, Iceland**
mmh@ru.is
2 **University of Freiburg, Germany**
kuhn@cs.uni-freiburg.de
3 **MIT, Cambridge, USA**
lynch@csail.mit.edu
4 **Georgetown University, Washington, DC, USA**
cnewport@cs.georgetown.edu

Abstract

In this paper we study the problem of developing efficient distributed algorithms for dense wireless networks. For many problems in this setting, fast solutions must leverage the reality that radio signals fade with distance, which can be exploited to enable concurrent communication among multiple sender/receiver pairs. To simplify the development of these algorithms we describe a new communication abstraction called FadingMAC which exposes the benefits of this concurrent communication, but also hides the details of the underlying low-level radio signal behavior. This approach splits efforts between those who develop useful algorithms that run on the abstraction, and those who implement the abstraction in concrete low-level wireless models, or on real hardware.

After defining FadingMAC, we describe and analyze an efficient implementation of the abstraction in a standard low-level SINR-style network model. We then describe solutions to the following problems that run on the abstraction: max, min, sum, and mean computed over input values; process renaming; consensus and leader election; and optimal packet scheduling. Combining our abstraction implementation with these applications that run on the abstraction, we obtain near-optimal solutions to these problems in our low-level SINR model – significantly advancing the known results for distributed algorithms in this setting. Of equal importance to these concrete bounds, however, is the general idea advanced by this paper: as wireless networks become more dense, both theoreticians and practitioners must explore new communication abstractions that can help tame this density.

1998 ACM Subject Classification C.2.1 Wireless Communication, G.2.2 Graph Theory – network problems

Keywords and phrases wireless networks, abstractions, SINR, signal fading

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.25

[*] MMH is supported by Icelandic Research Fund grants 152679-05 and 174484-05.
[†] FK is supported by ERC grant no. 336495 (ACDC).
[‡] NL is supported by AFOSR contract FA9550-13-1-0042 and NSF grants 0939370-CCF, CCF-1217506, and CCF-AF-1461559.
[§] CN is supported by NSF grant CCF-1649484.

31st International Symposium on Distributed Computing (DISC 2017).
Editor: Andréa W. Richa; Article No. 25; pp. 25:1–25:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In this paper we present and study a communication abstraction designed to facilitate efficient computation in dense wireless networks. In addition to defining the abstraction, we describe and analyze an implementation on a low-level SINR-style network model, and provide multiple applications that leverage it to efficiently solve standard network problems.

Problem: Contention Management in Dense Wireless Networks. In wireless networks devices share channels. A common approach to avoiding contention among multiple transmitters on the same channel is to have devices compete to gain exclusive use of the channel. This strategy, for example, is deployed by both the 802.11 and Zigbee standards to prevent message loss due to collision. It is also implicitly enforced in the graph-based models often used by theoreticians to study distributed algorithms for wireless networks (e.g., the radio network model [6, 1] or dual graph model [16, 4]), as these models assume any contention leads to message loss at the receiver.

This *exclusive-use* strategy is effective for many existing networking scenarios. It is not well suited, however, for *dense wireless networks* in which hundreds, thousands, or even tens of thousands of wireless devices might be located in a small area. Dense networks have become increasingly relevant due to emerging trends such as the Internet-of-Things, in which networking capability is embedded into a wide variety of objects. The density of these networks render exclusive-use contention management strategies too inefficient for many problems, as the time required for each device to use the channel in isolation is prohibitive.

Solution: Abstractions that Leverage Spatial Reuse. One way to enable efficient computation in dense wireless networks is to take advantage of the reality that radio signals fade with distance. This property makes it possible to support a large amount of concurrent communication among closely packed devices – a strategy often called *spatial reuse*.[1] Leveraging spatial reuse can allow network applications to solve certain problems much faster than relying on an exclusive-use strategy (c.f., the summary of our results below).

Developing algorithms that leverage spatial reuse requires low-level wireless network models that directly model radio signal fading and determine receive behavior using a *signal to interference and noise ratio* (SINR) equation. These models have received extensive attention from specialists in the algorithms community (e.g., [7, 14, 17, 13, 12, 11, 3, 19, 20, 18]), but they are also complex, and arguably demand too high a barrier of entry for those interested in dense networks, but not necessarily willing to master the intricacies of affectance formulas and interference-bounded annuli. These difficulties impede efforts to develop useful and efficient applications that are well suited for dense networks.

We argue that a solution to these problem is to develop *communication abstractions* that present the network application developer with an intuitive interface that exposes the concurrent communication benefits of spatial reuse, while hiding the low-level details of the underlying radio signal behavior. This approach splits efforts between those developing network applications that run on the abstraction, and those who are implementing the abstraction in low-level models, and, eventually, on real hardware. By doing so it simplifies

[1] The basic mechanism behind spatial reuse is the following: If a pair of wireless devices u and v are close together compared to other nearby transmitters in the network, then the strength of a radio signal sent from u and received at v might be strong enough compared to interference from other slightly more distant transmitters to allow v to successfully receive u's message.

the study of efficient distributed algorithms in SINR models, and provides feedback to practitioners about how best to innovate network stacks to accommodate the continued growth of wireless device density.

The FadingMAC Abstraction. In this paper we propose one such communication abstraction, which we call FadingMAC. This abstraction organizes the wireless devices (called *processes* in the following) into a labelled tree structure. It then moves through the levels of the resulting tree: at each level, processes assigned to tree nodes at that level can use the abstraction to communicate with the processes assigned to their parent and/or child nodes in the tree. This abstraction allows processes to aggregate and disseminate information in time proportional to the tree height. Therefore, if the tree has a small height compared to the network size, it enables certain computations to complete faster than what is possible with exclusive-use strategies.

Our Results. We begin by describing and analyzing an implementation of the FadingMAC abstraction using a standard low-level SINR wireless network model. In the following, let n be the number of processes in the system and R be the ratio between the longest and shortest link in the low-level model (in nearly all realistic scenarios R is at most polynomial in n). Our FadingMAC implementation guarantees the following with high probability in n: its setup mode requires $O(\log R \cdot \log n \cdot \log^* n)$ rounds; the height of the resulting tree is in $O(\log R)$; and it requires only $O(1)$ rounds to handle communication at each tree level.

We then turn our attention to applications that run on the FadingMAC abstraction. We begin by adapting a classical parallel prefix scan strategy to our setting, and then use this scan as a key subroutine to implement solutions to the following problems, selected due to their relevance to dense wireless networking scenarios:

- Max, min, sum, and mean calculated over input values;
- Renaming processes with unique labels from 1 to n (or renaming any subset of k processes with unique labels from 1 to k);
- Leader election and consensus among arbitrary subsets of participating processes; and
- Optimal packet scheduling (each process has a request for a certain number of rounds it needs to send information, and the goal is to agree on a schedule that assigns each process exactly their requested number of rounds).

Our solutions require visits to at most $O(h)$ tree levels by the abstraction, where h is the tree height. Combined with our abstraction implementation, and the assumption that R is polynomial in n, we obtain new distributed algorithms for these problems in a low-level SINR model that require a setup cost of $O(\log^2 n \cdot \log^* n)$ rounds and only $O(\log n)$ rounds per instance. Given that $\Omega(\log n)$ rounds is a fundamental speed limit for solving non-trivial problems with high probability in the SINR setting (see the discussion of [8] in Section 2), these solutions are near optimal, and exponentially faster than the $\Omega(n)$ rounds required for solving these types of problems with exclusive-use strategies.

To summarize, our results provide three strong contributions to the study of dense wireless networks. First, our applications combined with our abstraction implementation provide new (and nearly optimal) solutions to multiple useful distributed algorithm problems in the SINR model. Second, our abstraction simplifies the future development of efficient distributed algorithms for the complicated SINR setting. Third, we validate the general argument that in both theory and practice, the right abstractions can help tame the increasingly important setting of dense networks.

2 Related Work

In a 2006 paper [19], Moscibroda and Wattenhofer described and analyzed a centralized algorithm that can schedule a connected structure of links in an SINR model in $O(\log^4 n)$ rounds. They note that in many standard models (such as the exclusive-use models cited above as motivation) this problem requires $\Omega(n)$ rounds – underscoring the importance of spatial reuse for efficient computation in dense networks. A subsequent series of papers [20, 18, 12] presented refined centralized scheduling algorithms, culminating in a $O(\log n)$-round solution due to Halldórsson and Mitra [12]. These same authors subsequently published a distributed version of their algorithm [11]. By comparison, our FadingMAC implementation solves this same scheduling problem in $O(\log R \cdot \log n \cdot \log^* n)$ rounds. These results, however, are not directly comparable as [20, 18, 12, 11] all assume that devices can adjust their transmission power, whereas we assume all devices use a fixed uniform power.

More recent results [14, 17] study efficient data aggregation in the SINR model with distributed algorithms and fixed uniform power. Hobbs et al. [14] describe a deterministic algorithm that aggregates data in a multihop network[2] in $O(D + \Delta \cdot \log n)$ rounds, where D is the diameter of the connectivity graph and Δ is its maximum degree (see the below footnote about connectivity graphs). Notice, in our single hop setting $D = n$. In addition, this algorithm, unlike ours, requires carrier sensing. Perhaps closer is the work of Li et al. [17], which shows how to aggregate data in a single hop setting in $O(\log R)$ rounds. Their solution, however, requires that devices are provided their position and neighbor count in advance – enabling efficient strategies built on fixed geographic regions. Our FadingMAC solution, by contrast, does not assume this information is known.

Fineman et al. [8] prove a $\Omega(\log n)$ lower bound on symmetry breaking with high probability in the same SINR model we study. A closer inspection of this bound reveals it holds for basic partition detection, and therefore applies to the applications we study in Section 7 – establishing their time complexities as optimal or near optimal (depending on assumptions about R and whether you include the one-time abstraction setup cost).

Our implementation of the FadingMAC abstraction adapts the reliable subgraph simulation strategy introduced to solve distributed one-to-all broadcast in [7], while our applications leverage a distributed prefix scan subroutine that adapts strategies from classical parallel solutions to this problem (see [2] and our discussion in Section 7). We previously explored the idea of mediating between low-level wireless models and high-level applications with our earlier work on the Abstract MAC Layer [15], including an implementation in the SINR model [10]. The Abstract MAC Layer is intended to abstract away the details of contention management (primarily) in graph-based models.

3 System Model

We assume a synchronous network with $n > 1$ computational processes. Let V be the set of processes. We assume each process is distinguished by a comparable and unique ID. Time proceeds in synchronous rounds that we label $1, 2, 3, ...$, and all processes start in round 1.

We assume the processes are connected by some underlying wireless network over which they can attempt to broadcast and receive messages. We assume this network is *single hop*, which we define in a general way here to mean that if a process $i \in V$ transmits alone in a given round, all nodes receive i's message during this round. We assume message sizes are bounded to contain at most $O(\text{polylog}(n))$ unique IDs and/or bits.

[2] In the SINR setting, *single hop* and *multihop* refer to the *connectivity graph* that results from connecting with an edge any process pair that are sufficiently close to communicate in the absence of interference.

In this paper, we present and study a *communication abstraction* called FadingMAC (defined below), that mediates between high-level network algorithms and a low-level wireless network model. Accordingly, we assume each computational process implements two distinct components:

(1) a FadingMAC abstraction implementation that interacts with other processes through the low-level wireless network model; and

(2) a high-level algorithm that interacts with other processes only through the interface provided by its local copy of the abstraction.

We sometimes say the high-level algorithm *runs on* the abstraction.

4 The FadingMAC Abstraction

The FadingMAC abstraction organizes the processes into a rooted tree structure. Each process is potentially assigned to multiple nodes in the tree. The abstraction is used in phases, with each phase dedicated to a level in the tree. A process assigned to a node u in the level corresponding to the current phase can use a children-to-parent communication primitive, provided by the abstraction, to communicate reliably with the process assigned to u's parent (assuming u is not the root). This process can also use a parent-to-children communication primitive to reliably communicate with the processes assigned to u's children (if any).

We begin below by formalizing the type of tree the abstraction uses to organize the nodes. We then detail the communication primitives provided by the abstraction, and present the parameters that describe a particular implementation's performance and correctness guarantees.

The Tournament Tree. The abstraction begins an execution in a *setup mode*, during which it organizes the processes into a *tournament tree* defined with respect to V (see below). Formally, this means that by the end of the setup mode, the abstraction outputs to each process information about its assigned nodes in the tree. When considered collectively, this output information defines a unique tree T.

In more detail, a tournament tree defined over V is a rooted tree that satisfies the following properties:

1. All leaf nodes are at the same level in the tree.
2. Each node in the tree is labeled with a process from V.[3]
3. The n leaves are labeled with the n unique processes from V.
4. Each non-leaf node in the tree is labeled with the label of one of its children.

We call a labeled tree of the type a *tournament tree* because the label of each non-leaf node can be interpreted as the winner of a competition among the processes labeling its children. Fix a tournament tree T. As is standard for rooted trees, we label the tree's levels starting with 0 for the root level and increasing the level number as we move downward toward the leaves. We define $h(T)$ to describe its height (i.e., the largest level in the tree). For a given level $\ell \in [0, h(T)]$, we define $V(T, \ell)$ to be the set of processes labeling nodes at ℓ.

[3] Technically, the node is labelled with the unique ID corresponding to process i, but for notational simplicity we use i and i's ID interchangeably.

During the setup period, the FadingMAC abstraction will identify a single tournament tree T defined with respect to V and output to each process $i \in V$ the following three pieces of information regarding the nodes it labels in T:

(1) the smallest level in T that includes i, which we denote $\ell(T, i) = \min\{\ell \in [0, h(T)] : i \in V(T, \ell)\}$;

(2) if $\ell(T, i) > 0$, i's parent in level $\ell(T, i) - 1$ (otherwise it receives some default value $\perp$);

(3) the height $h(T)$ of the tree.

Notice, the collective knowledge provided to the processes about T by the abstraction uniquely specifies T.

The Communication Primitives. After the setup period finishes producing the tree T, the abstraction informs all processes that it is shifting into *communication mode*. While in communication mode, the abstraction divides rounds into *phases*. Each phase is dedicated to a single level from the tree T organized during the setup period. The abstraction notifies all processes when each phase begins as well as the phase's corresponding level in T. Different applications using FadingMAC can use different orderings of the phases for visiting levels in the tree. (The applications we study later in this paper, for example, follow a pattern of sweeping from the leaves to the root and then back down again.)

During a phase dedicated to tree level ℓ, every process in $V(T, \ell)$ is provided the opportunity to communicate with processes assigned to nearby nodes in T using a pair of communication primitives provided by FadingMAC.

In more detail, the abstraction provides both a *parent-to-children* and *children-to-parent* broadcast primitive. During the first round of a phase dedicated to level ℓ, each process $i \in V(T, \ell)$ can provide as input to the abstraction a message m_1 for the *parent-to-children* primitive and a message m_2 for the *children-to-parent* primitive. By the definition of a tournament tree, if $i \in V(T, \ell)$, then this process is assigned to exactly one node u at this level of T. During the subsequent rounds of the phase the abstraction attempts to deliver m_1 to the process assigned to u's parent in T (if u is the root or process i is assigned to u's parent as well, then it will receive its own message back.) It also attempts to deliver m_2 to the processes assigned to u's children in T (if any).

Abstraction Parameters. The guarantees of a given implementation of the FadingMAC abstraction is characterized by the following parameters:

- t_{setup}: the number of rounds required to complete the setup period.
- t_{phase}: the number of rounds required for each phase during the communication mode.
- h_{max}: the maximum height of the tournament tree.

To accommodate probabilistic implementations of the abstraction, we include a failure probability, p_{fail}, describing the maximum probability that the abstraction fails to satisfy its specified behavior.

5 The Low-Level SINR Wireless Model

Here we define a low-level wireless model that captures the fading behavior of radio signals. We call this "the SINR" model, but note that there are many related variations on this same style of model (the variant defined below is a generalized version of the model defined in [7]).

The SINR model is motivated by the premise that transmissions are successfully received when the signal arriving at a given receiver is sufficiently strong compared to the sum of interference from other transmissions. The strength of the transmissions is modeled as

inversely proportional to a polynomial of the distance transmitted. The idea that signals fade with distance in this manner and interference sums at the receiver match experimental investigations (c.f., [21, 5]). This is the primary reason why the SINR model is viewed as one of the most realistic analytic models currently available for studying low-level wireless algorithms. We note, however, that earlier studies of the SINR model tended to place the devices in the plane and define "distance" to be Euclidean. This assumption weakens the realism of the model as real environments can distort and attenuate signals in complex ways. To accommodate this reality, we replace Euclidean distances in our below model definition with a more general metric that captures more diverse signal environments.

We now formalize the model definition. In doing so, we mostly follow notation from [7]. We assume that all processes transmit with the same power, which for simplicity we normalize as 1. The strength of a transmission from process u as received by a process v is $1/d(u,v)^\alpha$, where $d(u,v)$ is the distance from u to v and α is a constant. The transmission is *successful* if v is listening and the SINR formula holds:

$$SINR(u,v,I) := \frac{1/d(u,v)^\alpha}{N + \sum_{w \in I} 1/d(w,v)^\alpha} \geq \beta \ ,$$

where I is the set of other concurrent transmitters, β is a positive constant dependent on the hardware and coding schemes used, and N quantifies the ambient noise.

We assume that the distance metric satisfies the following doubling property: There are constants λ and d, with $\lambda > 0$ and $0 < d < \alpha$, such that if U is a set of processes of mutual distance at least d_{min}, and all processes in U are within a distance $x \cdot d_{min}$ from a given process, then $|U| \leq \lambda \cdot x^d$. For the Euclidean plane, this holds with $d = 2$. We use this relaxed definition to account for the various effects of signal propagation.

Let r_s denote the maximum distance at which successful transmission is possible, i.e., $1/r_s^\alpha = \beta N$, or $r_s = (\beta N)^{-1/\alpha}$. Let R denote the ratio between the largest and smallest process distance in the network.

We assume that processes have no collision detection or carrier sensing capability, and no advance knowledge of their distances to other processes. The processes know a polynomial upper bound on the standard deployment parameters: the network size $n = |V|$ and the distance diversity R. We also assume they know reasonable approximations of the network parameters α, β, n, and N.

6 Implementing FadingMAC in the SINR Model

We now describe and analyze a randomized distributed algorithm that implements the FadingMAC abstraction defined in Section 3 in the SINR model defined in Section 5. Our optimized version of this algorithm, presented in Section 6.3, implements the abstraction with $t_{setup} = O(\log R \cdot \log n \cdot \log^* n)$, $t_{phase} = O(1)$, $h_{max} = O(\log R)$, and $p_{fail} < n^{-c}$, for a constant $c > 1$.

We first define in Sec. 6.1 a family of graphs that captures succinctly when nodes can communicate effectively, and use it in Sec. 6.2 to construct a basic communication tree. In Sec. 6.3, we improve it so that each phase can be executed in constant number of rounds.

6.1 SINR-Induced Graphs

Our construction for each given level ℓ is based on constructing a bounded-degree graph H_ℓ. To distinguish, we shall refer to the nodes of the graph as *vertices*, reserving *nodes* for nodes of the tournament tree, and noting that computation on behalf of both types of objects is

executed by *processes*. To construct H_ℓ, we apply the concept of *SINR-induced graphs* as defined by Daum et al. in [7]. Let $U \subseteq V$ be a subset of the vertices, let p be a transmission probability, and let $\mu \in (0, p) \cap \Omega(1)$ be a reliability parameter. We define the SINR-induced graph $H_p^\mu[U]$ with vertex set U and edge set $E_p^\mu[U]$ as follows. Assume that each vertex in U transmits with probability p and each vertex in $V \setminus U$ transmits with probability 0. For $u, v \in U$, we have $\{u, v\} \in E_p^\mu[U]$ if and only if u receives a message from v with probability at least μ and also v receives a message from u with probability at least μ. Note that all distances between vertices/nodes are within the metric space, not in terms of path lengths in the graph/tree.

Distributed Computation of SINR-Induced Graphs. Given a set of vertices $U \subseteq V$ and parameters p and μ, the graph $H_p^\mu[U]$ cannot be computed exactly in a distributed way. However, it is possible to efficiently compute a close approximation of $H_p^\mu[U]$. In [7], an *ε-close approximation* of $H_p^\mu[U]$ is defined as a graph $\tilde{H}_p^\mu[U] = (U, \tilde{E}_p^\mu[U])$ for which:

$$E_p^\mu[U] \subseteq \tilde{E}_p^\mu[U] \subseteq E_p^{(1-\varepsilon)\mu}[U]$$

It is shown in [7] that an ε-close approximation $\tilde{H}_p^\mu[U]$ of $H_p^\mu[U]$ can be constructed as follows in $2T$ rounds, where $T = c\frac{\ln n}{\varepsilon^2 \mu}$ for a sufficiently large constant $c > 0$. In all $2T$ rounds, all vertices in U transmit with probability p. In the first T rounds, each vertex $u \in U$ compiles a list of all vertices of which u receives a message in at least $\left(1 - \frac{\varepsilon}{2}\right)\mu T$ of the T rounds. In the second T rounds, those lists are exchanged and an edge $\{u, v\}$ is added to $\tilde{E}_p^\mu[U]$ if and only if u is in v's list and v is in u's list. It is shown in Lemma 3 in [7] that this algorithm computes an ε-close approximation of $H_p^\mu[U]$, w.h.p. We note that because in each round, a vertex can only receive a message from a single vertex, the maximum degree of $\tilde{H}_p^\mu[U]$ is bounded by $1/\mu = O(1)$.

We can also show that $H_p^\mu[U]$ contains all relatively short edges. We first need the following lemma, which is actually somewhat surprising, even though its proof (in the appendix) deploys relatively standard techniques.

▶ **Lemma 1.** *Given a constant $\zeta \geq 1$, there exists a constant $\nu = \nu_\zeta > 0$ such that the following holds. If $U \subseteq V$ is a subset of vertices of minimum pairwise distance d_{min} and $u, v \in U$ are vertices of distance at most $d(u, v) \leq \zeta \cdot d_{\min}$, then whenever v transmits while other vertices within distance $\nu \cdot d_{\min}$ of u are silent, u is guaranteed to successfully receive the transmission of v.*

We now extend Lemma 4 in [7], to show that $H_p^\mu[U]$ contains all short edges.

▶ **Lemma 2.** *For all $p \in (0, 1/2]$ and every constant $\eta > 1$, there exists $\mu \in (0, p)$ such that $\forall U \subseteq V$ with minimum pairwise distance d_{min}, the graph $H_p^\mu[U]$ (and thus also the graph $\tilde{H}_p^\mu[U]$) contains all edges between vertices $u, v \in U$ with $d(u, v) \leq \eta \cdot d_{\min}$.*

Proof. Let ν be the constant for which Lemma 1 holds for $\zeta = \eta$. Let $u, v \in U$ be any pair of vertices with $d(u, v) \leq \eta \cdot d_{min}$. Let S be the set of vertices in V within distance $\nu \cdot d_{\min}$ of u, excluding v. By the doubling property of the metric, $|S| \leq \lambda \cdot \nu^d$. By Lemma 1, u receives the message from v if v transmits while the vertices in S are silent. The probability of the former is p while the probability of the latter is $(1-p)^{|S|}$. Since these events are independent, it follows that u successfully receives a message from v with probability $p(1-p)^{|S|} \geq p(1-p)^{\lambda \cdot \nu^d}$. By a symmetric argument, the same holds for the probability that v successfully receives a message from u. Hence, for $\mu \geq p^2(1-p)^{2\lambda \cdot \nu^d}$, $\{u, v\}$ satisfies the condition of being an edge in H_p^μ. ◀

6.2 Construction and Properties of the Basic Tournament Tree

We now first describe the construction of a tournament tree that supports basic communication primitives. In the next subsection, we extend this construction and show how to get a tree and a constant-length TDMA schedule for implementing the communication primitives on each level. For the construction of the tournament tree T, we fix a transmission probability p and a (constant) parameter $\mu > 0$ such that the conditions of Lemma 2 are satisfied for $\eta = 2$ and the graph H_p^μ contains all edges of length at most $2d_{\min}$ for every $U \subseteq V$.[4] Further, for the distributed construction of an ε-close approximation $\tilde{H}_p^\mu[U]$ of $H_p^\mu[U]$ for a vertex set U, we fix $\varepsilon := 1/10$ throughout the construction. We construct the tournament tree T bottom-up and level by level. That is, we start with level $\ell = h(T)$ connecting the leaf nodes to their parents and then move up the tree until we reach level 1 where we select a root node $r \in V(T,1)$ and connect all nodes $V(T,1)$ to the root node r. The construction for level ℓ returns the set of nodes $V(T,\ell-1)$ that proceed to the next level and it fixes the parent nodes in level $\ell-1$ for all nodes in $V(T,\ell) \setminus V(T,\ell-1)$.

In the following, we describe the construction for a given level $\ell \in \{1,\dots,h(T)\}$ (recall that $V(T,h(T)) = V$):

1. Compute the graph $H_\ell := \tilde{H}_p^\mu[V(T,\ell)]$.
2. Compute a maximal independent set (MIS) I of H_ℓ and set $V(T,\ell-1) := I$
3. Each node $u \in V(T,\ell)\setminus V(T,\ell-1)$ chooses its parent v arbitrarily among its H_ℓ-neighbors in $V(T,\ell-1)$.

The following theorem summarizes the basic properties of the resulting tournament tree T.

▶ **Theorem 3.** *With high probability, the constructed tournament tree T has the following properties:*

- *The height of the tree is $h(T) = O(\log R)$.*
- *Suppose all nodes in $V(T,\ell)$ transmit with probability p and the other nodes are silent in a given round r and level $\ell \in \{1,\dots,h(T)\}$. Then, for every node $u \in V(T,\ell)$, u receives a message from its parent in round r with constant probability, and the parent also receives a message from u in round r with constant probability.*
- *The tree T can be constructed in time $O(\log R \cdot \log n \cdot \log^* n)$ in the SINR model.*

Proof. We first prove that the height of the constructed tree T is $O(\log R)$, w.h.p. Let $d_{\min}^{(\ell)}$ be the minimum distance between vertices in $V(T,\ell)$. By Lemma 2 and the choice of the parameters p and μ, the graph H_ℓ contains an edge for every two vertices $u,v \in V(T,\ell)$ at distance at most $\min\left\{2d_{\min}^{(\ell)}, r_s\right\}$. Because $V(T,\ell-1)$ is chosen as an MIS of the graph H_ℓ, no two verticess in $V(T,\ell-1)$ can be neighbors in H_ℓ and we thus get that $d_{\min}^{(\ell-1)} > \min\left\{2d_{\min}^{(\ell)}, r_s\right\}$. Hence as long as $d_{\min}^{(\ell)} \le r_s/2$, we have $d_{\min}^{(\ell-1)} > 2d_{\min}^{(\ell)}$. As soon as $d_{\min}^{(\ell)} > r_s/2$, by Lemma 2, H_ℓ is a complete graph and thus we reached the last level ($\ell = 1$). The claim that $h(T) = O(\log R)$ now follows because from the definition of R, the minimum distance between vertices in V is at least r_s/R.

The second property follows directly from the definition of the edges of the graph H_p^μ and the fact that parents are neighbors in H_p^μ. In turn, it implies that every vertex in $V(T,\ell)$ successfully receives a message from each of its neighbors in H_ℓ within $O(\log n)$ rounds, w.h.p. Hence, a single communication round in a standard message passing model on H_ℓ can be simulated in $O(\log n)$ rounds in the SINR model. Because H_ℓ has bounded degree, an MIS of H_ℓ can be computed in $O(\log^* n)$ rounds in a standard message passing model on H_ℓ [9] and the also the third claim of the lemma follows. ◀

[4] We can for example fix $p = 1/2$ and choose $\mu > 0$ as a sufficiently small constant.

6.3 A Tournament Tree with an Efficient TDMA Schedule

We next show how to adapt and extend the construction of the previous subsection to obtain a tournament tree with the same properties as in Theorem 3, but now with the additional assumption of a constant-length TDMA schedule for implementing the communication primitives on each level.

As in Section 6.2, the tree is computed in a bottom-up fashion and we describe the construction of a single level ℓ in detail. We first give an outline of the construction. We fix the parameters p, μ, and ε as before and we again construct the graph $H_\ell = \tilde{H}^{\mu}_{p}[V(T,\ell)]$. As a next step, we compute a TDMA schedule of length $L = O(1)$ for communicating on graph H_ℓ. That is, we assign a color $\phi_\ell(v) \in \{1,\dots,L\}$ to each vertex $v \in V(T,\ell)$. We then define an L-round schedule $\mathcal{S}_\ell$ for level ℓ such that exactly the nodes $v \in V(T,\ell)$ with $\phi_\ell(v) = i$ transmit in round i. We say that an edge $\{u,v\} \in E[H_\ell]$ is successful w.r.t. schedule $\mathcal{S}_\ell$ if in round $\phi_\ell(u)$ of the schedule $\mathcal{S}_\ell$, v receives a message from u and in round $\phi_\ell(v)$ of $\mathcal{S}_\ell$, u receives a message from v. We define H'_ℓ as the subgraph of H_ℓ consisting of all edges that are successful w.r.t. $\mathcal{S}_\ell$. The level ℓ of the tree T is then constructed in the same way as before, but by using graph H'_ℓ instead of graph H_ℓ.

We will next show that the coloring ϕ_ℓ can be constructed such that the resulting subgraph H'_ℓ of H_ℓ still contains all edges of length $\min\left\{2d^{(\ell)}_{\min}, r_s\right\}$ and as a consequence, the constructed tree still satisfies the properties of Theorem 3.

In order to construct the coloring ϕ_ℓ, we consider an SINR-induced graph $H^{\mu'}_{p'}[V(T,\ell)]$ such that $H^{\mu'}_{p'}[V(T,\ell)]$ contains all edges between vertices in $V(T,\ell)$ of distance at most $\eta \cdot d^{(\ell)}_{\min}$, for a sufficiently large constant $\eta > 2$. Note that by Lemma 2, we can choose constants p' and μ' such that the graph $H^{\mu'}_{p'}[V(T,\ell)]$ satisfies this. We choose η as the value of ν obtained from applying Lemma 1 with $\zeta = 2$. The coloring ϕ_ℓ will be computed as a standard vertex coloring of an ε-close approximation of the SINR-induced graph $H^{\mu'}_{p'}[V(T,\ell)]$.

As in the basic construction, we construct the tree T in a bottom-up way, starting at level $\ell = h(T)$ where the leaf nodes are connected to their parents and ending at level 1 where a root node $r \in V(T,1)$ selected. We again describe the construction for a given level $\ell \in \{1,\dots,h(T)\}$. In the following description, $\Delta(H^+_\ell)$ denotes the maximum degree of graph H^+_ℓ.

1. Compute the graph $H^+_\ell := \tilde{H}^{\mu'}_{p'}[V(T,\ell)]$.
2. Compute a $(\Delta(H^+_\ell)+1)$-vertex coloring ϕ_ℓ of the graph H^+_ℓ
3. Compute the graph $H_\ell := \tilde{H}^{\mu}_{p}[V(T,\ell)]$.
4. Compute the graph H'_ℓ consisting of all the edges of H_ℓ that are successful w.r.t. the TDMA schedule induced by the coloring ϕ_ℓ.
5. Compute a maximal independent set (MIS) I of H'_ℓ and set $V(T,\ell-1) := I$
6. Each node $u \in V(T,\ell)\setminus V(T,\ell-1)$ chooses its parent v arbitrarily among the H'_ℓ-neighbors in $V(T,\ell-1)$.

Analogously to Theorem 3, the following theorem summarizes the properties of the constructed extended tournament tree T. The proof is deferred to the appendix.

▶ **Theorem 4.** *With high probability, the constructed tournament tree T has the following properties:*

- *The height of the tree is $h(T) = O(\log R)$.*
- *For every $\ell \in \{1,\dots,h(T)\}$, when using the TDMA schedule induced by the coloring ϕ_ℓ of $V(T,\ell)$, (bidirectional) communication between every node $u \in V(T,\ell)$ and its parent in $V(T,\ell-1)$ can be carried out in $O(1)$ rounds.*
- *The tree T can be constructed in time $O(\log R \cdot \log n \cdot \log^* n)$ in the SINR model.*

7 High-Level Algorithms

Here we describe and analyze several useful high-level algorithms that run on the FadingMAC abstraction. The key subroutine used for these applications is a *distributed prefix scan* over the processes' input values. We begin below by describing and analyzing an algorithm that computes such a scan using a small number of FadingMAC phases, before moving on to describe the algorithms that leverage these scans.

7.1 Distributed Prefix Scans Using FadingMAC

During its setup mode, the FadingMAC abstraction returns each process information about the nodes it labels in a tournament tree T. We can transform T into an ordered tree by using a comparison operator defined over process labels to order the siblings of each parent from left to right (i.e., the operator that compares the process unique ids). This ordered version of T then allows us to order the n processes in the network from 1 to n by considering the leaf labels from left to right.

In the following, in a slight abuse of notation, we use *process* i, for $i \in [1, n]$, to indicate the process ranked in position i in this ordering. Assume each process i has an input value a_i from some value domain S that includes an identity element ϵ. Let $\oplus$ be a binary associative operator defined over S. (This combination of S and $\oplus$ defines a monoid.) The goal of a *distributed prefix scan* for a given monoid and input value assignment in our model is for each process i to learn: $s_i = a_1 \oplus a_2 \oplus ... \oplus a_i$.

We emphasize that prefix scans (also called prefix sums) are a core primitive in algorithm design, as they provide a building block for efficiently solving a variety of different problems. They are therefore studied in multiple models. In an exclusive-use radio network model it is straightforward to see that $\Omega(n)$ rounds are required (as each process needs its own round to communicate its value). Below we describe and analyze an algorithm that leverages the FadingMAC abstraction to calculate a prefix scan in $2h(T)$ communication mode phases. We emphasize that for the implementation of FadingMAC provided in this paper, this translates to $O(\log n)$ rounds for most networks.[5]

Strategy and Comparison to Parallel Prefix Scans. Our distributed strategy adapts general ideas from the classical parallel prefix scan algorithms; c.f., [2]. Similar to our setup, these parallel algorithms tend to also consider an ordered tree with one input value assigned to each of the n leaves. They move up the tree level by level, starting from the leaves and heading toward the root, at each node u summing the values of u's children. The result is that each node u learns the sum of the values in the leaves of the subtree rooted at u. The algorithm then moves down the tree level by level, with each internal node u informing each child the sum of its siblings to its left in the tree ordering.

Our distributed prefix scan algorithm implements this same basic strategy. To do so, however, it must overcome two difficulties that separate our model from parallel models:

(1) in the parallel setting the tree used is fixed by the algorithm and known to all processes, while in our model the tree is returned by the abstraction and each process is only provided partial information about the tree's structure;

[5] Our implementation requires only a constant number of rounds per phase and provides $h(T) = O(\log R)$. For most realistic networks, R will be at most polynomial in n.

(2) in the parallel setting a different process can be assigned to each node in the tree, while in our model each of the n processes might be responsible for multiple tree nodes on the path from its leaf to the root, complicating the algorithm description.

Algorithm Preliminaries. The algorithm described below uses the FadingMAC abstraction parameterized to visit the tree levels in order starting from the leaves and heading up to the root (called the *up sweep* in the following), and then visiting the tree levels in order starting from the root and heading back down to the leaves (called the *down sweep* in the following). Only a single up sweep followed by a single down sweep are required for our algorithm to compute it prefix scan.

Fix some node u in the ordered tournament tree. Let u.left indicate the rank of the leftmost leaf in this interval, and r.right indicate the rank of the rightmost leaf. By definition, r.left $\leq r$.right, but it is possible that u.left $= r$.right. We define the *sum of the subtree rooted at* u, denoted $s(u)$, to be the sum (defined with respect to the $\oplus$ operator) of the input values associated with the leaves of u's subtree. Formally: $s(u) = a_{u.\text{left}} \oplus a_{u.\text{left}+1} \oplus ... \oplus a_{u.\text{right}}$.

Algorithm Description. We now describe our algorithm which computes a distributed prefix scan using the FadingMAC abstraction. We first describe its behavior during the up sweep and then during the down sweep. The following assumes we have already entered the abstraction's communication mode.

Up Sweep. During the up sweep phases, the goal is to calculate the sum of the subtree rooted at each node in the ordered tournament tree T. That is, for each node u in T, our algorithm will calculate and store $s(u)$. Recall, in a tournament tree, each node is labelled with a process. In our algorithm, it will be the responsibility of the process that labels u to calculate and store $s(u)$. This process will also end up storing a copy of $s(u')$ for each child u' of u in T, as this will be needed during the subsequent down sweep.

To accomplish this goal, each process i maintains an accumulation variable $x(i)$ initialized to the identity element ϵ at the beginning of the upsweep. It will use this variable to help calculate the sum values for which it is responsible.

In more detail, each process i has two types of responsibilities for each level ℓ visited during the upsweep. The first type of responsibilities applies if $\ell > 0$ and there exists a node u at level $\ell - 1$ that is labelled with process i. If these conditions are met, then process i will calculate $s(u)$ by the end of this phase. To do so, process i first waits to see if node u receives any values from its children at level ℓ during this phase, delivered through the children-to-parent broadcast primitive provided by the abstraction. Assume this occurs and that we label the received values $q_1, q_2, ..., q_j$. In this case, process i updates $x(i)$ as follows: $x(i) \leftarrow x(i) \oplus q_1 \oplus q_2 \oplus ... \oplus q_j$. Process i will also save these received values which, as will soon be made clear, describe the sum of the subtrees rooted at these children – knowledge process i will need during the down sweep.

Regardless of whether or not u received any values, at the end of this phase, process i locally calculates and stores $s(u)$ as follows: $s(u) \leftarrow a_i \oplus x(i)$.

The second type of responsibility applies if there exists a node v at the current level ℓ that is labelled with process i. If v is a leaf (i.e., $\ell = h(T)$), then process i can directly set $s(v) = a_i$. If v is not a leaf, then process i would have calculated $s(v)$ during the previous phase for level $\ell + 1$ as described above. Therefore, in both cases, $s(v)$ is defined. If v's parent is *not* labelled with process i, then process i will use the children-to-parent broadcast primitive to send $s(v)$ to (the process labelling) u's parent.

Down Sweep. We now describe the behavior of our algorithm during the down sweep. The goal during the down sweep is for the process labelling each node u in T to learn:

$$pred(u) = \begin{cases} a_1 \oplus a_2 \oplus ... \oplus a_{u.\text{left}-1} & \text{if } u.\text{left} > 1, \\ \epsilon & \text{else.} \end{cases}$$

That is, $pred(u)$ is the sum of the values (defined with respect to $\oplus$) associated with all the leaves (if any) to the left of the subtree rooted at u in T.

We accomplish this goal as follows. Before the down sweep begins, the root u_0 of the tree can directly set $pred(u_0) \leftarrow \epsilon$. Now assume inductively that we arrive at the phase associated with level ℓ, $0 \leq \ell < h(T)$, and $pred$ is already calculated for all nodes at level ℓ. During this phase, (the processes labeling) these nodes will send (the processes labelling) their children at level $\ell + 1$ their $pred$ values.

In more detail, fix some u at level ℓ. Assume u is labelled with process i. It is the responsibility of process i to calculate the $pred$ value for each of u's children and to then send it to the processes labelling them using the parent-to-children broadcast primitive. To do so, let $u_1, u_2, ..., u_j$ be u's $j \geq 1$ children nodes, ordered by the same comparison operator we used to order T. By the postcondition of the up sweep, process i knows $s(u_i)$ for each child u_i of u. Process i can therefore calculate $pred(u_i)$, for each child u_i, as follows:

$$pred(u_i) \leftarrow \begin{cases} pred(u) \oplus s(u_1) \oplus s(u_2) \oplus ... \oplus s(u_{i-1}) & \text{if } i > 1, \\ pred(u) & \text{else.} \end{cases}$$

Final Calculation of Scan Values. Each process i labels a single leaf node u in tree T. By the postcondition of the down sweep described above, process i knows $pred(u)$ by the end of the down sweep. Because $pred(u)$ is the sum of all values in s_i except a_i, process i can conclude the algorithm by calculating $s_i \leftarrow pred(u) \oplus a_i$.

Analysis. The correctness of the above distributed prefix scan algorithm follows from a straightforward inductive argument that establishes that during the up sweep each $s(u)$ is correctly calculated. If these subtree sum values are correct then the correctness of the $pred$ values calculated during the subsequent down sweep follows directly from the operation of the algorithm. We formalize this correctness with the following theorem, where p_{fail} describes the maximum probability that abstraction fails to satisfy its t_{phase} and h_{max} guarantees.

▶ **Theorem 5.** *This algorithm computes a distributed prefix scan using the FadingMAC abstraction in $O(t_{phase} \cdot h_{max})$ rounds after the abstraction has entered the communication mode, with probability at least $1 - p_{fail}$.*

7.2 Applications

We now describe multiple applications that leverage the distributed prefix scan algorithm from the previous section as a subroutine to efficiently solve useful numerical and distributed coordination problems. Every application below requires at most a constant number of distributed prefix scans, which each require at most $O(t_{phase} \cdot h_{max})$ rounds. When combined with our FadingMAC implementation in the SINR model, we obtain solutions to the below problems that run with high probability in the SINR model in at most $O(\log R)$ rounds, in addition to the one-time time cost of $O(\log R \cdot \log n \cdot \log^* n)$ rounds for the abstraction setup mode. As argued above, R is at most polynomial in n for most realistic networks, meaning these SINR solutions require only $O(\log n)$ rounds per instance, and a one-time setup cost of $O(\log^2 n \cdot \log^* n)$ rounds.

Renaming and Network Sizing. Our system model assumes that the n processes are only provided comparable unique IDs. In many applications it might useful if the processes were assigned unique IDs from the set $\{1, 2, ..., n\}$. This can be easily accomplished with a single distributed prefix scan. In more detail, set the domain S to be the natural numbers and $\oplus$ to describe addition. Each process sets its initial value to 1 then computes a prefix scan. It follows directly from the definition of the scan that each process will end up with a sum that describes its rank order in the tournament tree leaves – providing the needed renaming from 1 to n. To rename only a subset of k participating processes, it is sufficient for these participants to set their initial value to 1 and the non-participants to set their initial value to 0. The sum values at the participants provide a renaming of the participants from 1 to k. In both cases, the process labeling the root of the tournament tree will learn the sum of all values during the prefix scan. It can then announce this information to the rest of the network by transmitting alone in the first round after the scan or disseminating the information during the scan's down sweep. The result is that all processes learn the number of participants in the renaming, which, in the case of all processes participating, is also the network size n.

Basic Aggregation Functions: Max, Min, Sum, Mean. It is straightforward to compute basic aggregation function on the process inputs using distributed prefix scans. Here we consider max, min, sum and mean functions (where the mean function assumes the values are numerical). In all four cases, the first step is to run the renaming and network sizing strategy described above. At the end of step, the processes are renamed from 1 to n and all processes know n. This requires one prefix scan. To calculate max or min, the processes can perform another prefix scan with $\oplus$ as the max or min operator, respectively. In both cases, both the root and the process renamed n will end up with the correct value. To calculate the sum of the initial values, it is sufficient to perform a scan with $\oplus$ describing addition. As before, the root and the process renamed n will end up with the sum. Either of these processes can then calculate the mean by simply dividing this sum by the network size n.

Leader Election and Consensus. Leader election and consensus are key primitives in distributed system design. If we assume all processes are participating in the primitive, we can solve both problems directly after the abstraction setup mode completes: the process that labels the root can declare itself leader or announce its initial value as the decision for consensus. On the other hand, if we assume only an arbitrary subset of the processes have an initial value or are interested in becoming leader, we can leverage the distributed prefix scan to break symmetry among this unknown set of participants. In more detail, the processes execute a max computation using the prefix scan as described above. In the case of leader election, the participating processes use their unique id as their input value, and in the case of consensus they use their input value. The non-participating processes set their input value to some default minimum value from the domain that is guaranteed to be at least as small as any id or initial value. At the end of the scan, the initial value sum identifies a single leader or correct decision value, which can then be spread to the full network as above.

Packet Scheduling. The packet scheduling problem assumes that each process i has some request $r_i \geq 0$ describing the number of rounds it needs to broadcast alone on the channel to deliver some important information to the network or perhaps a nearby base station. Let $T = \sum_{i=1}^{n} r_i$ be the total number of rounds needed by all the processes. The packet scheduling problem requires the processes to agree on a transmission schedule that provides

each process i at least r_i consecutive rounds to send its information. A distributed prefix scan provides an eloquent and efficient solution to this problem. In more detail, the processes use the natural numbers as the value domain and use addition for $\oplus$. Each process i then sets its initial value for the scan as $a_i \leftarrow r_i$. At the end of the scan, the sum s_i learned by each process i describes the sum of all request values of processes at its rank or below, with $s_n = T$. The processes can use these values to define a T-round packet schedule. In particular, each process i claims the rounds $s_i - r_i + 1$ to s_i in the schedule. The result is an optimal length schedule in which each process gets its requested number of rounds in a contiguous interval of the schedule.

References

1. R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Computer and System Sciences*, 45(1):104–126, 1992.
2. Guy E Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11):1526–1538, 1989.
3. Marijke H.L. Bodlaender, Magnús M. Halldórsson, and Pradipta Mitra. Connectivity and aggregation in multihop wireless networks. In *PODC*, pages 355–364. ACM, 2013.
4. K. Censor-Hillel, S. Gilbert, F. Kuhn, N. Lynch, and C. Newport. Structuring unreliable radio networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2011.
5. Yin Chen and Andreas Terzis. On the mechanisms and effects of calibrating RSSI measurements for 802.15.4 radios. In *EWSN*, pages 256–271. Springer, 2010.
6. I. Chlamtac and S. Kutten. On broadcasting in radio networks–problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.
7. Sebastian Daum, Seth Gilbert, Fabian Kuhn, and Calvin Newport. Broadcast in the ad hoc SINR model. In *Proc. 27th Symposium on Distributed Computing (DISC)*, pages 358–372, 2013.
8. Jeremy T Fineman, Seth Gilbert, Fabian Kuhn, and Calvin Newport. Contention resolution on a fading channel. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 155–164. ACM, 2016.
9. A. Goldberg, S. Plotkin, and G. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM J. Discrete Math.*, 1(4):434–446, 1988.
10. Magnús M. Halldórsson, Stephan Holzer, and Nancy A. Lynch. A local broadcast layer for the SINR network model. In *PODC*, pages 129–138, 2015.
11. Magnús M. Halldórsson and Pradipta Mitra. Distributed connectivity of wireless networks. In *PODC*, pages 205–214, 2012.
12. Magnús M. Halldórsson and Pradipta Mitra. Wireless connectivity and capacity. In *SODA*, pages 516–526, 2012.
13. Magnús M Halldórsson, Yuexuan Wang, and Dongxiao Yu. Leveraging multiple channels in ad hoc networks. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 431–440. ACM, 2015.
14. Nathaniel Hobbs, Yuexuan Wang, Qiang-Sheng Hua, Dongxiao Yu, and Francis CM Lau. Deterministic distributed data aggregation under the SINR model. In *International Conference on Theory and Applications of Models of Computation*, pages 385–399. Springer, 2012.
15. Fabian Kuhn, Nancy Lynch, and Calvin Newport. The abstract MAC layer. *Distributed Computing*, 24(3):187–206, 2011.

16 Fabian Kuhn, Nancy Lynch, Calvin Newport, Rotem Oshman, and Andrea Richa. Broadcasting in unreliable radio networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 336–345. ACM, 2010.

17 Hongxing Li, Chuan Wu, Qiang-Sheng Hua, and Francis CM Lau. Latency-minimizing data aggregation in wireless sensor networks under physical interference model. *Ad Hoc Networks*, 12:52–68, 2014.

18 Thomas Moscibroda. The worst-case capacity of wireless sensor networks. In *IPSN*, pages 1–10, 2007.

19 Thomas Moscibroda and Roger Wattenhofer. The complexity of connectivity in wireless networks. In *INFOCOM*, pages 1–13, 2006.

20 Thomas Moscibroda, Roger Wattenhofer, and Aaron Zollinger. Topology control meets SINR: the scheduling complexity of arbitrary topologies. In *MobiCom*, pages 310–321, 2006.

21 Dongjin Son, Bhaskar Krishnamachari, and John Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *SenSys*, pages 237–250. ACM, 2006.