Exponential Separations in the Energy Complexity of Leader Election*

Yi-Jun Chang
University of Michigan
cyijun@umich.edu

Tsvi Kopelowitz
University of Michigan
kopelot@gmail.com

Seth Pettie
University of Michigan
pettie@umich.edu

Ruosong Wang
Tsinghua University
wrs13@mails.tsinghua.edu.cn

Wei Zhan
Tsinghua University
jollwish@gmail.com

Abstract

Energy is often the most constrained resource for battery-powered wireless devices and the lion's share of energy is often spent on transceiver usage (sending/receiving packets), not on computation. In this paper we study the *energy complexity* of Leader Election and Approximate Counting in several models of wireless radio networks. It turns out that energy complexity is very sensitive to whether the devices can generate random bits and their ability to *detect collisions*. We consider four collision-detection models: Strong-CD (in which transmitters and listeners detect collisions), Sender-CD and Receiver-CD (in which only transmitters or only listeners detect collisions), and No-CD (in which no one detects collisions.)

The take-away message of our results is quite surprising. For randomized Leader Election algorithms, there is an exponential gap between the energy complexity of Sender-CD and Receiver-CD:

$$\text{Randomized: No-CD} = \text{Sender-CD} \gg \text{Receiver-CD} = \text{Strong-CD}$$

and for deterministic Leader Election algorithms, there is another exponential gap in energy complexity, *but in the reverse direction*:

$$\text{Deterministic: No-CD} = \text{Receiver-CD} \gg \text{Sender-CD} = \text{Strong-CD}$$

In particular, the randomized energy complexity of Leader Election is $\Theta(\log^* n)$ in Sender-CD but $\Theta(\log(\log^* n))$ in Receiver-CD, where n is the (unknown) number of devices. Its deterministic complexity is $\Theta(\log N)$ in Receiver-CD but $\Theta(\log\log N)$ in Sender-CD, where N is the (known) size of the devices' ID space.

There is a tradeoff between time and energy. We give a new upper bound on the time-energy tradeoff curve for randomized Leader Election and Approximate Counting. A critical component of this algorithm is a new deterministic Leader Election algorithm for *dense* instances, when $n = \Theta(N)$, with inverse-Ackermann-type ($O(\alpha(N))$) energy complexity.

*Supported by NSF grants CNS-1318294, CCF-1514383, and CCF-1637546.

1 Introduction

In many networks of wireless devices the scarcest resource is *energy*, and the lion's share of energy is often spent on *radio transceiver* usage [41, 4, 37, 43]—sending and receiving packets—not on computation per se. In this paper we investigate the *energy complexity* of fundamental problems in wireless networks: Leader Election, Approximate Counting, and taking a Census.

In all the models we consider *time* to be partitioned into discrete slots; all devices have access to a single shared channel and can choose, in each time slot, to either transmit a message m from some space $\mathcal{M}$, listen to the channel, or remain idle. Transmitting and listening each cost one unit of energy; we measure the energy usage of an algorithm on n devices by the worst case energy usage of any device. For the sake of simplicity we assume computation is free and the message size is unbounded. If exactly one device transmits, all listeners hear the message m, and if zero devices transmit, all listeners hear a special message λ_S indicating silence. We consider four models that differ in how *collisions* can be detected.

Strong-CD. Each sender and listener receives one of three signals: λ_S (silence, if zero devices transmit), λ_N (noise, if ≥ 2 devices transmit), or a message $m \in \mathcal{M}$ (if one device transmits).

Sender-CD. (Often called "No-CD" [25]) Each sender and listener receives one of two signals: λ_S (zero or ≥ 2 devices transmit), or a message $m \in \mathcal{M}$ (if one device transmits). Observe that the Sender-CD model has no explicit collision detection, but still allows for sneaky collision detection: if a sender hears λ_S, it can infer that there was at least one other sender.

Receiver-CD. Senders receive no signal. Each listener receives one of three signals: λ_S (silence, if zero devices transmit), λ_N (noise, if ≥ 2 devices transmit), or a message $m \in \mathcal{M}$ (if one device transmits).

No-CD. Senders receive no signal. Listeners receive one of two signals: λ_S (zero or ≥ 2 devices transmit) or a message $m \in \mathcal{M}$. To avoid trivial impossibilities, it is promised that there are at least 2 devices.

In addition we distinguish randomized and deterministic algorithms. In the randomized model all n devices begin in exactly the same state (n is unknown and unbounded), and can break symmetry by generating private random bits. In the deterministic model we assume that all n devices have unique IDs in the range $[N] \stackrel{\text{def}}{=} \{0, \ldots, N-1\}$, where N is common knowledge but $n \leq N$ is unknown.

We consider the following fundamental problems.

Leader Election. Exactly one device designates itself the leader and all others designate themselves follower. The computation ends when the leader sends a message while every follower listens to the channel.

Approximate Counting. At the end of the computation all devices agree on an $\tilde{n}$ such that $\tilde{n} = \Theta(n)$.

Census. We only study this problem in the deterministic setting. At the end of the computation some device announces a list (or *census*) of the IDs of all devices.

Remark 1. It could be argued that real-world devices rarely endow transmitters with more collision-detection power than receivers, so the Sender-CD model does not merit study. We feel this thinking gets the order backwards. There is a certain cost for equipping tiny devices with extra capabilities (such as generating random bits or detect collisions) so how are we to tell whether adding these capabilities is *worth the expense*? To answer that question we need to first determine the complexity of the algorithms that will ultimately run on the device. The goal of this work is to understand the power of various abstract models, not to cleave closely to existing real-world technologies, simply because they exist.

New Results. In the randomized model, we show that the energy complexity of Leader Election and Approximate Counting are $\Theta(\log^* n)$ in Sender-CD and No-CD but $\Theta(\log(\log^* n))$ in Strong-CD and Receiver-CD. The lower bounds also apply to the *contention resolution* problem, and this establishes that the recent $O(\log(\log^* n))$ contention resolution protocol of Bender et al. [7] is optimal. Our algorithm naturally offers a time-energy tradeoff. See Table 1 for the energy costs of our algorithm under different runtime constraints.

Time Complexity	Energy Complexity	
	Strong-CD or Receiver-CD	Sender-CD or No-CD with $n > 1$
$O(n^{o(1)})$	$O(\log(\log^* n))$	$O(\log^* n)$
$O(\log^{2+\epsilon} n)$, $0 < \epsilon \le O(1)$	$O(\log(\epsilon^{-1} \log\log\log n))$	$O(\epsilon^{-1} \log\log\log n)$
$O(\log^2 n)$	$O(\log\log\log n)$	$O(\log\log n)$

Table 1: Time-energy tradeoff of randomized Approximate Counting and Leader Election.

For Leader Election we establish matching bounds in all the deterministic models. In Strong-CD and Sender-CD, Leader Election requires $\Omega(\log\log N)$ energy (even when $n = 2$) and can be solved with $O(\log\log N)$ energy and $O(N)$ time, for any $n \le N$. However, in No-CD and Receiver-CD, the complexity of these problems jumps to $\Theta(\log N)$ [27]. Meanwhile, Census can be solved with $O(\log^2 \log N)$ energy and $O(N)$ time in the Strong-CD and Sender-CD models, and with $\Theta(\log N)$ energy in the Receiver-CD and No-CD models.

Finally, we prove that when the input is *dense* in the ID space, meaning $n = \Theta(N)$, Census can actually be computed with only $O(\alpha(N))$ energy and $O(N)$ time, even in No-CD. To our knowledge, this is the first time inverse-Ackermann-type recursion has appeared in distributed computing.

1.1 Prior Work

Jurdzinski et al. [25] studied the deterministic energy complexity of Leader Election in the Sender-CD model. They proved that dense instances ($n = \Theta(N)$) can be solved with $O(\log^* N)$ energy, and claimed that the complexity of the sparse problem is between $\Omega(\log\log N / \log\log\log N)$ and $O(\log^\epsilon N)$. While the lower bound is correct, the algorithm presented in [25] is not.[1] The most efficient published algorithm uses $O(\sqrt{\log N})$ energy, also due to Jurdzinski et al. [28]. The same authors [26] gave a reduction from randomized Sender-CD Approximate Counting to deterministic Leader Election over ID space $N = O(\log n)$, which, using [28], leads to an $O(\sqrt{\log\log n})$ energy algorithm for Approximate Counting. In [27] the authors gave a method for simulating Sender-CD

[1] T. Jurdzinski. (Personal communication, 2016.)

protocols in the No-CD model, and proved that deterministic No-CD Leader Election takes $\Omega(\log N)$ energy. Nakano and Olariu [38] showed that n devices in the Sender-CD model can assign themselves distinct IDs in $\{1, \ldots, n\}$ with $O(\log \log n)$ energy.

Very recently Bender et al. [7] gave a method for circuit-simulation in the Strong-CD model, which led to randomized Approximate Counting and Leader Election protocols using $O(\log(\log^* n))$ energy and $n^{o(1)}$ time. An earlier algorithm of Kardas et al. [30] solves Leader Election in the Strong-CD model in $O(\log^\epsilon n)$ time using $O(\log \log \log n)$ energy, in expectation.

Most of the previous work in the radio network model has been concerned with *time*, not energy. Willard [44] proved that $O(\log \log n)$ time is necessary and sufficient for one device to successfully transmit in the Strong-CD model with constant probability; see [39] for tradeoffs between time and success probability. In the Sender-CD model this problem requires $\Theta(\log^2 n)$ time to solve, with probability $1 - 1/\text{poly}(n)$ [16, 29, 40]. Greenberg and Winograd [24] proved that if *all* devices need to send a message, $\Theta(n \log_n(N))$ time is necessary and sufficient in the deterministic Strong-CD model.

The related problems of broadcasting, leader election, and gossiping have been studied extensively in multi-hop radio networks, where the bounds typically depend on both the diameter and size of the network, whether it is directed, and whether randomization and collision detection are available. See, e.g., [2, 3, 9, 10, 13, 35, 1, 33, 34, 19, 12]. Schneider and Watterhofer [42] investigated the use of collision detection in multihop radio networks when solving archetypal problems such as MIS, $(\Delta + 1)$-coloring, and broadcast. Their results showed that the value of collision detection depends on the problem being solved.

Cornejo and Kuhn [11] introduced the *beeping* model, where no messages are sent; the only signals are $\{\lambda_N, \lambda_S\}$: noise and silence. The complexity of Approximate Counting was studied in [8] and the state-complexity of Leader Election was studied in [23].

In adversarial settings a *jammer* can interfere with communication. See [36, 14] for leader election protocols resilient to jamming. In a *resource-competitive* protocol [6] the energy cost of the devices is some function of the energy cost of the jammer. See [5] for resource-competitive contention resolution, and [22, 32] for resource-competitive point-to-point communication and broadcast protocols.

1.2 Organization and Technical Overview

To establish the two sharp exponential separations we need 8 distinct upper and lower bounds. The $O(\log N)$ upper bound on deterministic No-CD Leader Election is trivial and the matching lower bound in Receiver-CD is from [27]. The $O(\log(\log^* n))$ upper bound from [7] on randomized Leader Election/Approximate Counting works only in Strong-CD. This paper contains matching upper and lower bounds of $\Theta(\log \log N)$ in the deterministic Sender-CD/Strong-CD models, and matching $\Theta(\log^* n)$ bounds in the randomized No-CD/Sender-CD models. We adapt the randomized $O(\log(\log^* n))$ upper bound [7] from Strong-CD to Receiver-CD, and provide a matching $\Omega(\log(\log^* n))$ lower bound in Strong-CD. In addition, we give a simpler proof of the $\Omega(\log N)$ lower bound in deterministic Receiver-CD.

In Section 2 we begin with a surprisingly simple proof that protocols solving any non-trivial problem in the deterministic Strong-CD model require $\Omega(\log \log N)$ energy if the devices are adaptive and $\Omega(\log N)$ if they are non-adaptive. It turns out that Receiver-CD algorithms are essentially forced to be non-adaptive, so this yields $\Omega(\log N)$ lower bounds for deterministic Leader Election in

Receiver-CD. The $\Omega(\log\log N)$ lower bound combines a decision-tree representation of the algorithm with the encoding argument that Katona and Szemerédi [31] used to solve the biclique covering problem of Erdős, Rényi, and Sós [15].

In Section 3 we prove the $\Omega(\log^* n)$ and $\Omega(\log(\log^* n))$ lower bounds on randomized Approximate Counting/Leader Election. These lower bounds begin by embedding any algorithm into an infinite *universal* decision-DAG (basically a decision tree with some reconvergent paths). In this model two algorithms only differ in their transition probabilities and halting criteria. The proof is information-theoretic. We consider the only two methods for devices in Strong-CD/Receiver-CD to learn new information (i) via direct communication (in which one device transmits and some subset of devices listen) and (ii) via inference (in which transmitting/listening devices detect a collision or silence, which informs their future decisions). The information-theoretic capacity of method (i) is essentially unbounded whereas method (ii) is bounded by 1-bit per unit energy in Strong-CD and usually less in Receiver-CD. We show that any algorithm with a reasonable time bound can be forced to learn an approximation of n via the information-theoretically well-behaved method (ii).

In Sections 4 and 5 we present all deterministic upper bounds: an $O(\log\log N)$-energy protocol for Leader Election, an $O(\log^2\log N)$-energy protocol for Census, and an $O(\alpha(N))$ energy protocol for *dense* Leader Election/Census, when $n = \Theta(N)$. The first two protocols combine *van Emde Boas*-like recursion with a technique that lets a group of devices function *as one device* and thereby share energy costs.

In Section 6 we present upper bounds on randomized Leader Election/Approximate Counting. When time is not too constrained (suppose the budget is $n^{o(1)}$), the Sender-CD and Receiver-CD protocols have energy complexity $O(\log^* n)$ and $O(\log(\log^* n))$. Our protocols naturally adapt to any time bound that is $\Omega(\log^2 n)$, where the energy complexity gradually increases as we approach this lower limit. See Table 1. These protocols are randomized (and do not assume distinct IDs); nonetheless, they use the deterministic $\alpha(N)$ dense Leader Election/Census algorithm of Section 5.

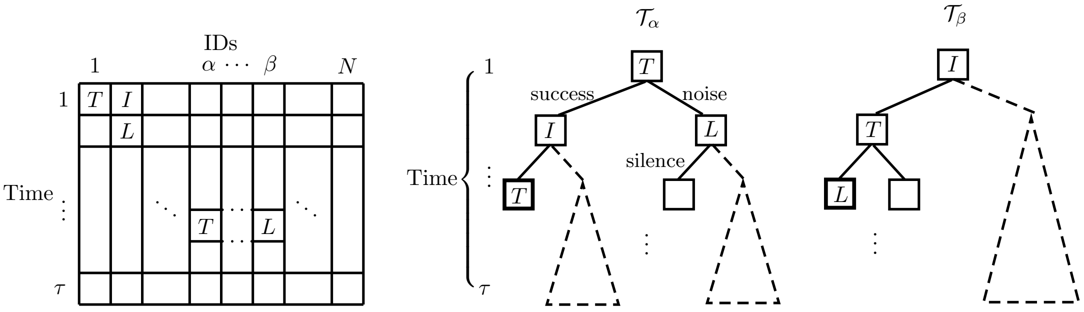
2 Deterministic Lower Bounds

In this section we prove deterministic lower bounds on the Successful Communication problem, which is no harder than Leader Election. The goal is to have *some* time slot where exactly one device sends a message while at least one device listens. Once successful communication occurs the algorithm is terminated on all devices. We focus on the special case when $n = 2$, both devices know $n = 2$, but not the ID of the other device. Since Strong-CD and Sender-CD models are the same when $n = 2$, our lower bound applies to both.

We first address the case of non-adaptive algorithms, where the actions of a device are solely a function of its ID, not the signals it receives by transmitting/listening to the channel. We then extend the proof technique to the more general adaptive case.

Theorem 2.1. *The energy cost of a non-adaptive Leader Election algorithm is $\Omega(\log N)$, even when $n = 2$.*

Proof. Let $\tau = \tau(N)$ be the running time of a non-adaptive algorithm for Successful Communication. This algorithm can be encoded by a table in the set $\{T, L, I\}^{\tau\times N}$; see Figure 1a. The (j, i)-th entry of the table is the action (transmit, listen, or idle) made by the device with ID i at time j. The energy cost of device i, denoted E_i, is the number of T or L entries in the ith column. The two active devices must successfully communicate in some time slot. If their IDs are α and β, there



(a) (b)

Figure 1: (a) The non-adaptive action table. Any two columns must have a corresponded pair of T and L. (b) Two binary decision trees $\mathcal{T}_\alpha$ and $\mathcal{T}_\beta$ in the adaptive case. There is a corresponded pair of T and L in the two trees with the same left-right path from the roots.

must be a row in the table where the entries of the αth and the βth columns contain one T and one L.

We now prove that $\max_i E_i \geq \log N$. The proof is inspired by Katona and Szemerédi's [31] lower bound of the biclique covering problem. Encode the ith column as a binary string of length τ by replacing T with 0, L with 1, and I with either 0 or 1. There are clearly $2^{\tau - E_i}$ possible encodings for column i. Since every pair of columns must have a row where they contain T and L, no binary string is an eligible encoding of two distinct columns. Since there are 2^τ possible encodings, we have:

$$\sum_{i=1}^{N} 2^{\tau - E_i} \leq 2^\tau, \quad \text{which implies that} \quad \sum_{i=1}^{N} \frac{1}{2^{E_i}} \leq 1.$$

The convexity of $f(x) = 1/2^x$ implies $\sum_i E_i \geq N \log N$. So the energy cost of the algorithm is $\Omega(\log N)$, even on average. □

Theorem 2.2. *The energy cost of a deterministic* Leader Election *is* $\Omega(\log\log N)$ *in the* Strong-CD *and* Sender-CD *models, even when* $n = 2$.

Proof. Suppose we have an algorithm for Successful Communication running in τ time. Fixing $n = 2$, we represent the behavior of the algorithm on the device with ID i as a binary decision tree $\mathcal{T}_i$. Each node in $\mathcal{T}_i$ is labeled with an action in $\{T, L, I\}$. An I (idle) node has one left child and no right child; a T (transmit) node has two children, a left one indicating collision-free transmisssion and a right one indicating a collision. An L node has two children, a left one indicating silence and a right one indicating the transmission of some message (and the termination of the algorithm).

The left-right ordering of children is meaningless but *essential* to making the following arguments work. Suppose that we run the algorithm on devices with IDs α and β, and halt the algorithm at the first time t in which successful communication occurs. We claim the paths taken through $\mathcal{T}_\alpha$ and $\mathcal{T}_\beta$ are encoded by the same sequence of $t-1$ left/right turns. At any time slot before t the possible actions performed by $\{\alpha, \beta\}$ are $\{I, I\}, \{I, T\}, \{I, L\}, \{L, L\}, \{T, T\}$. In all cases, both α and β branch to the left, except for $\{T, T\}$, in which case they both branch right. At time t the actions are $\{T, L\}$ and it is only here that they branch in opposite directions. See Figure 1b.

To encode $\mathcal{T}_i$ as a bit-string we extend it to a full binary tree of depth τ: add a dummy right child to every I-node, and repeatedly add two dummy children to any leaf at depth less than τ. The right child of an L-node is considered a dummy. The number of nodes in $\mathcal{T}_i$ is now $2^\tau - 1$. We encode $\mathcal{T}_i$ as a bit-string of length $2^\tau - 1$ by listing the nodes in any fixed order (say preorder), mapping each T-node to 0, L-node to 1, and each I-node or dummy node to either 0 or 1. Since any α, β must engage in successful communication, there must be a position in the full binary tree that is a T-node in $\mathcal{T}_\alpha$ and an L-node in $\mathcal{T}_\beta$ or vice versa. Hence no bit-string is an eligible encoding of two distinct $\mathcal{T}_\alpha, \mathcal{T}_\beta$. If device i spends energy E_i, then the number of T or L nodes in $\mathcal{T}_i$ is at most $2^{E_i} - 1$ and therefore $\mathcal{T}_i$ has $2^{(2^\tau-1)-(2^{E_i}-1)}$ possible encodings. Thus,

$$\sum_{i=1}^{N} 2^{(2^\tau-1)-(2^{E_i}-1)} \leq 2^{2^\tau-1} \quad \text{which implies that} \quad \sum_{i=1}^{N} \frac{1}{2^{2^{E_i}-1}} \leq 1.$$

The convexity of $f(x) = 1/2^{2^x-1}$ implies $\sum_i E_i \geq N \log(\log N + 1)$, so the energy cost of the algorithm is $\Omega(\log\log N)$, even on average. □

Notice that in the decision trees above, only transmitting nodes are able to branch before the algorithm halts with a successful communication. In the No-CD model, however, transmitters receives no feedback. In this case, no branching is possible before the first successful communication and the problem reduces to the non-adaptive case. Therefore we also obtain an $\Omega(\log N)$ lower bound for Leader Election in the No-CD model, which matches the trivial upper bound, and an earlier lower bound of [27]. The lower bound also applies to the stronger Receiver-CD model since when $n = 2$, the Receiver-CD model is the same as No-CD.

Corollary 2.3. *The energy cost of Leader Election is $\Theta(\log N)$ in the No-CD and Receiver-CD models, even when $n = 2$.*

3 Randomized Lower Bounds

In this section we prove energy lower bounds of randomized algorithms for Approximate Counting. Our lower bounds hold even in the scenario where the devices in a failed execution may consume unbounded energy and never halt.

Theorem 3.1. *The energy cost of a polynomial time Approximate Counting algorithm with failure probability $1/n$ is $\Omega(\log^* n)$ in the Sender-CD model.*

Theorem 3.2. *The energy cost of a polynomial time Approximate Counting algorithm with failure probability $1/n$ is $\Omega(\log(\log^* n))$ in the Strong-CD model.*

Since No-CD is strictly weaker than Sender-CD, the $\Omega(\log^* n)$ lower bound also applies to No-CD. Similarly, the $\Omega(\log(\log^* n))$ lower bound applies to Receiver-CD.

3.1 Randomized Decision Trees

The process of a device s interacting with the network at time slot t has two phases. During the first phase (action performing phase), s decides on its action, and if this action is to transmit, then s chooses a message $m \in \mathcal{M}$ and transmits m. During the second phase (message receiving phase), if s chose to listen or transmit during the first phase, then s hears the signal (depending on the model) which depends on the transmissions occurring at this time slot. The phases partition the timeline into *layers*. We write layer t to denote the time right before the first phase of time slot t, and layer $t + 0.5$ to denote the time right before the second phase of time slot t. The choice of the message space $\mathcal{M}$ is irrelevant to our lower bound proof. The cardinality of $\mathcal{M}$ may be finite or infinite.

For a device s, the *state* of s at layer t includes the ordered list of actions taken by s and signals received from the channel until layer t. Layer 1 consists of only the *initial state*, which is the common state of all devices before the execution of an algorithm.

Our lower bounds are proved using a *single* decision tree $\mathcal{T}$ of unbounded branching factor (if $|\mathcal{M}|$ is unbounded). A special directed acyclic graph (DAG) $\mathcal{G}$ is defined to capture the behaviour of *any* randomized algorithm, and then the decision tree $\mathcal{T}$ is constructed by "short-cutting" some paths in $\mathcal{G}$.

The DAG $\mathcal{G}$. The set of all nodes in $\mathcal{G}$ represent all possible states of a device during the execution of any algorithm. Similarly, the set of all arcs represent all legal transitions between states during the execution of any algorithm. Therefore each arc connects only nodes in adjacent layers, and the root of $\mathcal{G}$ is the initial state.

Let $t \in \mathbb{Z}^+$. A transition from a state u in layer t to a state v in layer $t + 0.5$ corresponds to one of the possible $|\mathcal{M}| + 2$ actions that can be performed in the first phase of time slot t. The transitions from a state u in layer $t + 0.5$ to a state v in layer $t + 1$ are more involved. Based on the action performed in the first phase of time slot t that leads to the state u, there are three cases:

- If the action is idle, then u has one outgoing arc corresponding to doing nothing.
- If the action is listen, then u has $|\mathcal{M}| + 2$ outgoing arcs in the Strong-CD model (or $|\mathcal{M}| + 1$ in the Sender-CD model), corresponding to all possible messages that can be heard.
- If the action is transmit, then u has two outgoing arcs. The first (second) outgoing arc corresponds to the message transmission succeeding (failing). If a failure took place, then no other device knows which message was sent by the device, and so the content of this message is irrelevant. Thus, all states u in layer $t + 0.5$ that correspond to the action transmit and share the same parent have the same child node in layer $t + 1$ corresponding to a failure in transmitting the message. Notice that the arcs corresponding to failed transmissions are what make $\mathcal{G}$ a DAG and not a tree.

Embedding an algorithm. Any algorithm can be embedded into $\mathcal{G}$. An algorithm $\mathcal{A}$ is embedded into $\mathcal{G}$ as follows. First of all, appropriate states, depending on $\mathcal{A}$, are designated as terminal states. Without loss of generality, we require that any terminal state must be in layer t for some $t \in \mathbb{Z}^+$. Each terminal state is associated with a specific output for the problem at hand. A device entering a terminal state u terminates with the output associated with the state u. Any randomized algorithm is completely described by designating the terminal states together with their outputs, and specifying the transition probabilities from states in layer t to states in layer $t + 0.5$ for all $t \in \mathbb{Z}^+$. We require that the transition probabilities of all transitions from a given state in layer t sum up to 1.

The randomized decision tree $\mathcal{T}$. The tree $\mathcal{T}$ is derived from $\mathcal{G}$ as follows. The set of nodes of $\mathcal{T}$ is the set of nodes in $\mathcal{G}$ that are in layer t such that $t \in \mathbb{Z}^+$. For any two states u in layer $t \in \mathbb{Z}^+$ and v in layer $t+1$ that are linked by a directed path, there is a transition from u to v in $\mathcal{T}$. It is straightforward to see that $\mathcal{T}$ is a rooted tree. See Figure 2 for an illustration of both $\mathcal{G}$ and $\mathcal{T}$ in the Strong-CD model with $\mathcal{M} = \{m_1, \ldots, m_k\}$.[2] For a state u in layer $t \in \mathbb{Z}^+$, and for an action $x \in \{\mathsf{idle}, \mathsf{listen}, \mathsf{transmit}\}$, we write $p_{u \rightsquigarrow x}$ to denote the probability that a device belonging to u chooses to do the action x in the first phase of time slot t.

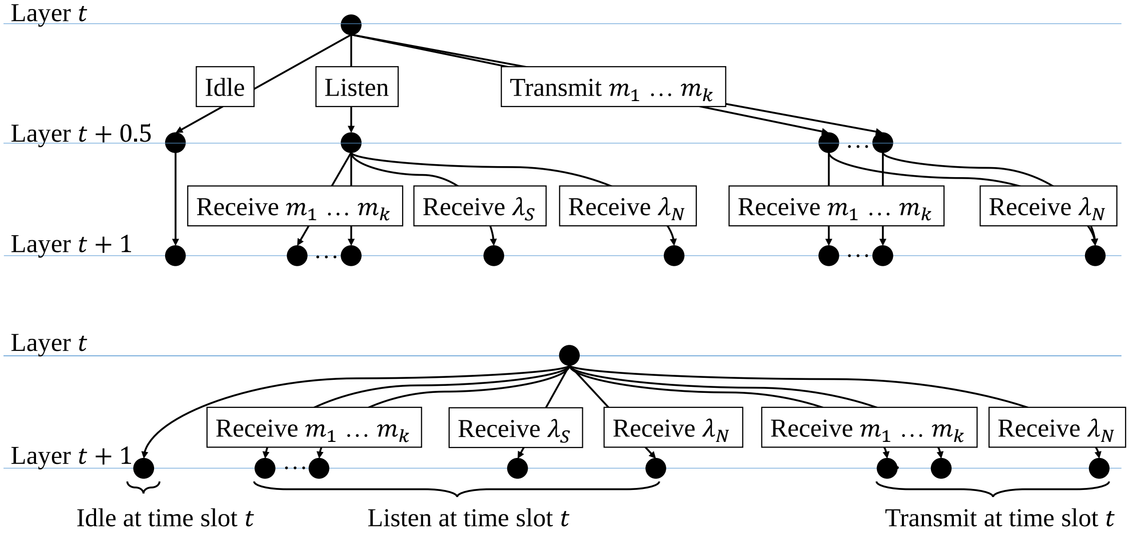


Figure 2: Upper: a portion of $\mathcal{G}$. Lower: the corresponding portion in $\mathcal{T}$.

Time and energy complexity. An execution of an algorithm for a device is completely described by a directed path $(u_1, u_2, \ldots, u_k)$ in $\mathcal{T}$ such that u_t is in time slot t for each $1 \leq t \leq k$, and u_k is the only terminal state. The runtime of the device is k. The amount of energy the device spends is the number of transitions corresponding to listen or transmit in $(u_1, u_2, \ldots, u_k)$. The time (energy) of an execution of an algorithm is the maximum time (energy) spent by any device.

3.2 Lower Bound in the Sender-CD Model

Let $\mathcal{A}$ be any $T(n)$ time algorithm for Approximate Counting in the Sender-CD model with failure probability at most $1/n$. We assume that $T(n) \geq n$ throughout the section.[3] An energy lower bound of $\mathcal{A}$ is shown by carefully selecting a sequence of network sizes $\{n_i\}$ with *checkpoints* $\{d_i\}$ such that $d_i < n_i \leq T(n_i) < d_{i+1}$.

There are two main components in the lower bound proof. The first component is to demonstrate that, with probability $1 - 1/\text{poly}(n_i)$, no message is successfully transmitted before time d_i when

[2] Notice that in the Strong-CD model, a device transmitting a message m_i to the channel at a time slot must not hear λ_S in the same time slot. If the transmission is successful, it hears the message m_i; otherwise it hears λ_N.

[3] The following simple fix increases the runtime of any Approximate Counting algorithm to $T(n) \geq n$ without affecting the energy cost. Suppose that the estimate $\tilde{n}$ all devices agree on is within a multiplicative factor $c \geq 1$ of n. After a device decides its estimate $\tilde{n}$, let it remains idle for $c \cdot \tilde{n}$ additional time slots before it terminates.

running $\mathcal{A}$ on n_i devices (that is, every transmission ends in a collision). This limits the amount of information that could be learned from a device, as the only thing a device can learn at a given time slot is whether there are zero or at least two devices trying to transmit some message to the channel. The second component is to prove that, for $j > i$, in order for a device to have enough information to distinguish between n_i and n_j within $T(n_i) < d_{i+1}$ time slots, that device must use at least one unit of energy within time interval $[d_i, d_{i+1} - 1]$. Intuitively, this is because a device only gains information from either listen or transmit.

Truncated decision tree. The *no-communication tree* $\mathcal{T}_{\text{no-comm}}$ is defined as the subtree of $\mathcal{T}$ induced by the set of all states u such that no transition in the path from the root to u corresponds to receiving a message in $\mathcal{M}$. In other words, $\mathcal{T}_{\text{no-comm}}$ contains exactly the states whose execution history contains no successful communication. Notice that in Sender-CD each state in $\mathcal{T}_{\text{no-comm}}$ has exactly three children, and the three children correspond to the following three pairs of action performed and message received: $(\mathsf{transmit}, \lambda_S)$, $(\mathsf{listen}, \lambda_S)$, and $(\mathsf{idle}, \text{N/A})$. For each state u at layer t of the tree $\mathcal{T}_{\text{no-comm}}$, we define the *probability estimate* p_u inductively as follows. If u is the root, $p_u = 1$; otherwise $p_u = p_v \cdot p_{v \rightsquigarrow x}$, where v is the parent of u, and x is the action performed at time slot $t-1$ that leads to the state u. Intuitively, if no message is successfully sent in an execution of $\mathcal{A}$, the proportion of devices entering u is concentrated around p_u, given that p_u is high enough. See Figure 3 for an illustration of no-communication tree $\mathcal{T}_{\text{no-comm}}$ and probability estimates in the Sender-CD model.

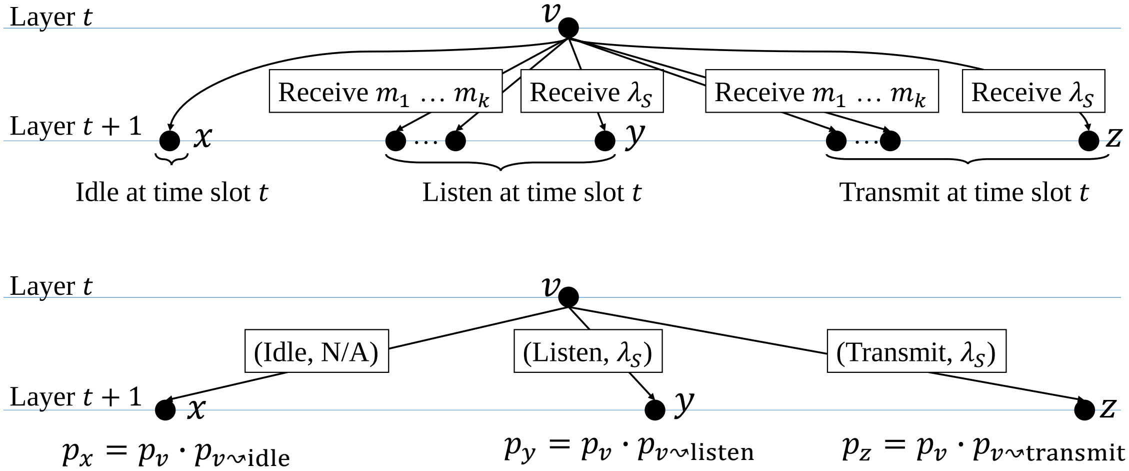


Figure 3: Upper: a portion of the tree $\mathcal{T}$. Lower: the corresponding portion in the no-communication tree $\mathcal{T}_{\text{no-comm}}$.

Checkpoints d_i and network sizes n_i. Given the runtime constraint $T(n)$, we define an infinite sequence of *checkpoints* as follows: d_1 is a large enough constant to be determined; for all $i > 1$, $d_i = T\left(2^{2^{2^{2^{d_{i-1}}}}}\right)$.

Lemma 3.3. *For each index i, there exists a network size n_i with $2^{2^{d_i}} < n_i < 2^{2^{2^{2^{d_i}}}}$ such that for each state $u \in \mathcal{T}_{\textit{no-comm}}$ at layer at most d_i, either $p_u \le {n_i}^{-10}$ or $p_u \ge {n_i}^{-1/10}$.*

Proof. Define $m_1 = 2^{2^{d_i}} + 1$; for $1 < k \le 3^{d_i}$, define $m_k = {m_{k-1}}^{100} + 1$. It is straightforward to see that $2^{2^{d_i}} < m_1 < m_2 < \ldots < m_{3^{d_i}} < 2^{2^{2^{2^{d_i}}}}$, as long as d_i is at least a large enough constant. For each state $u \in \mathcal{T}_{\text{no-comm}}$ at layer at most d_i, there exists at most one m_k with ${m_k}^{-10} < p_u < {m_k}^{-1/10}$. Recall that $\mathcal{T}_{\text{no-comm}}$ has branching factor 3, and hence the number of states up to layer d_i is less than 3^{d_i}. By the pigeonhole principle, among the 3^{d_i} distinct integers $m_1, m_2, \ldots$, there exists one integer n_i such that, for each state $u \in \mathcal{T}_{\text{no-comm}}$ at layer at most d_i, either $p_u \le {n_i}^{-10}$ or $p_u \ge {n_i}^{-1/10}$. □

For each index i, n_i is chosen to meet the statement of Lemma 3.3, and the first checkpoint d_1 is chosen to be a large enough number such that n_i and n_{i+1} are not within a constant approximation. We define $\mathcal{T}_i$ as the subtree of $\mathcal{T}_{\text{no-comm}}$ that consists of all states u up to layer d_i such that $p_u \ge {n_i}^{-1/10}$. Notice that $\mathcal{T}_i \subseteq \mathcal{T}_{i+1}$ for all i. For an execution of $\mathcal{A}$ on n_i devices, we write $\mathcal{P}_i$ to denote the event where for each state u in layer $1 \le t \le d_i$ of the decision tree $\mathcal{T}$, the number of devices entering u is within $n_i \cdot p_u \pm t \cdot {n_i}^{0.6}$ if u is in layer t of $\mathcal{T}_i$, and is 0 if $u \notin \mathcal{T}_i$.

Lemma 3.4. *Let $t \le d_i$, and let v be a state in layer $t-1$ of $\mathcal{T}_i$ such that the number of devices entering v is within $n_i \cdot p_v \pm (t-1) \cdot {n_i}^{0.6}$, and let x be an action in $\{\textsf{transmit}, \textsf{listen}, \textsf{idle}\}$. With probability $1 - O({n_i}^{-9})$, the number of devices that are in state v at time $t-1$ and perform action x at time $t-1$ is (i) within $n_i \cdot p_v \cdot p_{v \rightsquigarrow x} \pm t \cdot {n_i}^{0.6}$ when $p_v \cdot p_{v \rightsquigarrow x} \ge {n_i}^{-1/10}$, or is (ii) zero when $p_v \cdot p_{v \rightsquigarrow x} \le {n_i}^{-10}$.*

Proof. Define m as the number of devices that are in state v at time $t-1$ and do action x at time t. According to our choice of n_i, for each state $u \in \mathcal{T}_{\text{no-comm}}$ at layer at most d_i, either $p_u \le {n_i}^{-10}$ or $p_u \ge {n_i}^{-1/10}$. Since $p_v \cdot p_{v \rightsquigarrow x} = p_u$ for some $u \in \mathcal{T}_{\text{no-comm}}$ at layer at most d_i, we have the following two cases:

- *Case 1:* ${n_i}^{-10} \ge p_v \cdot p_{v \rightsquigarrow x}$. We have $p_{v \rightsquigarrow x} \le {n_i}^{-10}/p_v \le {n_i}^{-9.9}$, and we have $\mathrm{E}[m] \le \left(n_i \cdot p_v + (t-1) \cdot {n_i}^{0.6}\right) \cdot p_{v \rightsquigarrow x} \le {n_i}^{-9} + (t-1) \cdot {n_i}^{0.6} \cdot p_{v \rightsquigarrow x} \le {n_i}^{-9} + (t-1) \cdot {n_i}^{-9.3} < 2{n_i}^{-9}$ (recall that $t \le d_i < \log\log n_i$). By Markov's inequality, $m = 0$ with probability $\ge 1 - 2{n_i}^{-9}$.
- *Case 2:* $p_v \cdot p_{v \rightsquigarrow x} \ge {n_i}^{-1/10}$. The value of $\mathrm{E}[m]$ is within $p_{v \rightsquigarrow x} \cdot (n_i \cdot p_v \pm (t-1) \cdot {n_i}^{0.6})$, which is within the range $n_i \cdot p_v \cdot p_{v \rightsquigarrow x} \pm (t-1) \cdot {n_i}^{0.6}$. Let $\delta = {n_i}^{-0.4}/2$. Notice that we have $\delta \cdot \mathrm{E}[m] < {n_i}^{0.6}$ and $\mathrm{E}[m] > {n_i}^{0.9}/2$. Meanwhile, each device in the state v decides which action to perform next independently. By Chernoff bound, the probability that m is within $1 \pm \delta$ fraction of $\mathrm{E}[m]$ is at least $1 - 2\exp(-\delta^2 \cdot \mathrm{E}[m]/3) \ge 1 - 2\exp(-({n_i}^{-0.4}/2)^2 \cdot ({n_i}^{0.9}/2)/3) > 1 - O({n_i}^{-9})$. Therefore, with such probability, m is in the range $\mathrm{E}[m](1 \pm \delta)$, which is within $n_i \cdot p_v \cdot p_{v \rightsquigarrow x} \pm t \cdot {n_i}^{0.6}$.

□

Lemma 3.5. *For an execution of $\mathcal{A}$ on n_i devices, $\mathcal{P}_i$ holds with probability at least $1 - {n_i}^{-7}$.*

Proof. For the base case of $t = 1$, it is straightforward to see that $\mathcal{P}_i$ holds for the initial state. For each $1 < t \le d_i$, assuming $\mathcal{P}_i$ holds for all states at layer $t-1$, then we show that $\mathcal{P}_i$ holds for all states at layer t with probability at least $1 - O({n_i}^{-8})$.

That $\mathcal{P}_i$ holds for all states at layer $t-1$ guarantees that for each state v in layer $t-1$ of $\mathcal{T}_i$, the number of devices entering v is within $n_i \cdot p_v \pm (t-1) \cdot {n_i}^{0.6}$. By the union bound on at most 3^{t-1} choices of v and 3 choices of x, the statement of Lemma 3.4 holds for all choices of v and x with probability at least $1 - \left(3^t\right) \cdot O({n_i}^{-9}) \geq 1 - O({n_i}^{-8})$ (recall that $t < \log\log n_i$). This implies that the number of devices transmitting at time $t-1$ is either 0 or at least ${n_i}^{0.9} - t \cdot {n_i}^{0.6} > 1$, and hence no message is successfully sent by the layer t (i.e. right before the time slot t). Therefore, at layer t, all devices are confined in states within $\mathcal{T}_{\text{no-comm}}$. Let u be the child of v in $\mathcal{T}_{\text{no-comm}}$ such that the action leading to u is x. By the choice of n_i, $u \in \mathcal{T}_i$ implies that $p_u = p_v \cdot p_{v \rightsquigarrow x} \geq {n_i}^{-1/10}$; and $u \notin \mathcal{T}_i$ implies that $p_u = p_v \cdot p_{v \rightsquigarrow x} \leq {n_i}^{-10}$. Therefore, assuming $\mathcal{P}_i$ holds for all states at layer $t-1$, Lemma 3.4 guarantees that, with probability at least $1 - O({n_i}^{-8})$, $\mathcal{P}_t$ holds for all states at layer t.

By the union bound on all $t \in \{1, \ldots, d_i\}$, $\mathcal{P}_i$ holds with probability at least $1 - {n_i}^{-7}$. Notice that $d_i < \log\log n_i$. □

We claim that $\mathcal{T}_{\text{no-comm}}$ has no terminal state u with $p_u \neq 0$. Suppose that $u \in \mathcal{T}_{\text{no-comm}}$ is a terminal state with $p_u \neq 0$. Then there exists an index i such that for *all* $j \geq i$, $u \in \mathcal{T}_j$. Among all $\{n_j\}_{j \geq i}$, the decision of u is a correct estimate of at most one n_j (recall that any two network sizes in $\{n_j\}_{j \geq 1}$ are not within a constant factor of each other). Therefore, the adversary can choose one network size $n_{j'}$ from $\{n_j\}_{j \geq i}$ such that when $\mathcal{A}$ is executed on $n_{j'}$ devices, any device entering u gives a wrong estimate of $n_{j'}$. By Lemma 3.5, with probability $1 - {n_{j'}}^{-7} > 1/n_{j'}$, there is a device entering u, and hence the algorithm fails with probability higher than $1/n_{j'}$, a contradiction.

Forcing energy expenditure. Let p_{idle} be the probability for a device entering a state u in layer d_i of $\mathcal{T}_i$ to be idle throughout the time interval $[d_i, d_{i+1} - 1]$. For a device s to correctly give an estimate of network size n_i within the time constraint $T(n_i) < d_{i+1}$, the device s must successfully hear some message $m \in \mathcal{M}$. Recall that $\mathcal{T}_{\text{no-comm}}$ has no terminal state u with $p_u \neq 0$, and hence a device must *leave* the tree $\mathcal{T}_{\text{no-comm}}$ before it terminates. If $\mathcal{P}_i$ holds, then one unit of energy expenditure in the time interval $[d_i, d_{i+1} - 1]$ is required for any device s in a state in layer d_i of $\mathcal{T}_i$. Therefore, for all devices to terminate by time $T(n_i)$ with probability at least $1 - 1/n_i$ when $\mathcal{A}$ is executed on n_i devices, we need $1/n_i \geq \Pr[\mathcal{P}_i] \cdot p_{\mathsf{idle}}$, and this implies $p_{\mathsf{idle}} < 2/n_i \leq 2 \cdot 2^{-2^{d_i}} < 2^{-d_i}$. In other words, for any device in a state in layer d_i of $\mathcal{T}_i$, the device spends at least one unit of energy in the time interval $[d_i, d_{i+1} - 1]$ with probability at least $1 - p_{\mathsf{idle}} > 1 - 2^{-d_i}$.

The lower bound. Consider an execution of $\mathcal{A}$ on n_i devices, and let s be any one of the n_i devices. Let $j \in \{1, \ldots, i\}$. We claim that, given that $\mathcal{P}_i$ holds, with probability $1 - 2 \cdot 2^{-d_j}$ the device s spends at least one unit of energy in the interval $[d_j, d_{j+1} - 1]$.

- First, we show that the probability that s enters a state in layer d_j of $\mathcal{T}_j$ is at least $1 - 2^{-d_j}$. Recall that $\mathcal{T}_j$ is a subtree of $\mathcal{T}_i$. By definition, if u is a state in layer d_j but not belongs to $\mathcal{T}_j$, then $p_u < {n_j}^{-1/10}$. Therefore, the proportion of the devices that do not enter a state in layer d_j of $\mathcal{T}_j$ is at most $\frac{1}{n_i}\left(n_i \cdot {n_j}^{-1/10} + d_j \cdot {n_i}^{0.6}\right) \cdot 3^{d_j} = \left({n_j}^{-1/10} + d_j \cdot {n_i}^{-0.4}\right) \cdot 3^{d_j} < 2^{-d_j}$.
- Second, as argued in the above paragraph, a device that enters a state in layer d_j of $\mathcal{T}_j$ spends at least one unit of energy in the time interval $[d_j, d_{j+1} - 1]$ with probability $1 - 2^{-d_j}$.

Therefore, the claim is concluded.

By the union bound on $j = 1, \ldots, i$, the probability that the device s spends at least one unit of energy in each of the intervals $[d_j, d_{j+1} - 1]$, $1 \le j \le i$, is at least $\Pr[\mathcal{P}_i] - 2\sum_{j=1}^{i} 2^{-d_j} > 1/2$ (we choose d_1 to be a large enough number to make the inequality hold). We conclude the theorem.

Theorem 3.6. *For any $i \ge 1$, there exists a network size n with $d_i + 1 \le n \le d_{i+1}$ such that if $\mathcal{A}$ is executed on n devices, for any device s, with probability at least $1/2$ the device s spends at least one unit of energy in each of the time intervals $[d_j, \ldots, d_{j+1} - 1]$, $1 \le j \le i$.*

As long as $T(n) = O\big(\overbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}}^{\alpha}\big)$ for a constant α, the energy cost of $\mathsf{Approximate\ Counting}$ in the $\mathsf{Sender\text{-}CD}$ model is $\Omega(\log^* n)$. Our lower bound proof naturally offers a time-energy tradeoff. For example, the lower bound generalizes to *higher* time constraints as follows. For *any* function $g(n) \le \log n$, under time constraint $T(n) = g^{-1}(n)$, the energy cost is $\Omega(g^*(n))$.

3.3 Lower Bound in the Strong-CD Model

Let $\mathcal{A}$ be any $T(n)$ time algorithm for $\mathsf{Approximate\ Counting}$ in the $\mathsf{Strong\text{-}CD}$ model with failure probability at most $1/n$. Similar to the Section 3.2, we construct a sequence of network sizes $\{n_i\}$ with checkpoints $\{d_i\}$ such $d_i < n_i \le T(n_i) < d_{i+1}$. Each pair (n_i, d_i) is associated with a truncated decision tree $\mathcal{T}_i$ such that with probability $1 - 1/\mathrm{poly}(n_i)$ the execution history of all devices are confined in $\mathcal{T}_i$ when running $\mathcal{A}$ on n_i devices by time d_i. Due to the capability of differentiating between $\mathsf{silence}$ and noise in the $\mathsf{Strong\text{-}CD}$ model, a device is able to perform a binary search on a set of candidate network sizes $\{n_i\}_{1 \le i \le k}$, which costs only $O(\log k)$ unit of energy. In comparison, in the $\mathsf{Sender\text{-}CD}$ model $O(k)$ energy consumption is needed in the worst case. In the lower bound proof, we construct a path P in $\mathcal{T}_{\text{no-comm}}$ which captures the worst case scenario of the binary search, and we will see that the energy consumption of a device whose execution history follows the path P is $\Omega(\log(\log^* n))$.

Checkpoints d_i, network sizes n_i, and subtrees $\mathcal{T}_i$. The definitions of no-communication tree $\mathcal{T}_{\text{no-comm}}$ and probability estimate p_u are the same as in the previous section. However, notice that in the $\mathsf{Strong\text{-}CD}$ model each state in $\mathcal{T}_{\text{no-comm}}$ has exactly four children, corresponding to all valid combinations of $\{\lambda_S, \lambda_N\}$ and $\{\mathsf{transmit}, \mathsf{listen}, \mathsf{idle}\}$: $(\mathsf{transmit}, \lambda_N)$, $(\mathsf{listen}, \lambda_S)$, $(\mathsf{listen}, \lambda_N)$, and $(\mathsf{idle}, \mathrm{N/A})$. Notice that a device transmitting a message never hear $\mathsf{silence}$ in the $\mathsf{Strong\text{-}CD}$ model. The definition of the checkpoints d_i and how we select the network sizes n_i are also the same as in the previous section.

For an index i, the subtree $\mathcal{T}_i$, along with the sequence $\{m_{i,t}\}_{1 \le t \le d_i - 1}$ indicating $\mathsf{noise}/\mathsf{silence}$ of the channel at time slot t, is defined inductively as follows. Initially $\mathcal{T}_i$ consists of the initial state (the base case of $t = 1$). For each $1 < t \le d_i$, suppose that $\mathcal{T}_i$ has been defined up to layer $t - 1$. If there exists a state v in layer $t - 1$ of $\mathcal{T}_i$ with $p_v \cdot p_{v \rightsquigarrow \mathsf{transmit}} \ge {n_i}^{-1/10}$, define $m_{i,t-1} = \lambda_N$; otherwise define $m_{i,t-1} = \lambda_S$. A state u in layer t that is a child of a state in $\mathcal{T}_i$ is added to the subtree $\mathcal{T}_i$ if the following two conditions are met:

- The probability estimate of u is high: $p_u \ge {n_i}^{-1/10}$.
- Either the action performed in the first phase of time slot $t - 1$ that leads to the state u is idle, or the message received in the second phase of time slot $t - 1$ that leads to the state u is $m_{i,t-1}$.

Observe that all states in $\mathcal{T}_i$ are confined to layer at most d_i, and each state in $\mathcal{T}_i$ that is not a leaf has 3 children, corresponding to the three actions.

Similar to the previous section, for an execution of $\mathcal{A}$ on n_i devices, we write $\mathcal{P}_i$ to denote the event that for each state u in layer $1 \leq t \leq d_i$ of the decision tree $\mathcal{T}$, the number of devices entering u is within $n_i \cdot p_u \pm t \cdot {n_i}^{0.6}$ if u is in layer t of $\mathcal{T}_i$, and is 0 if $u \notin \mathcal{T}_i$.

Lemma 3.7. *For an execution of $\mathcal{A}$ on n_i devices, $\mathcal{P}_i$ holds with probability at least $1 - {n_i}^{-7}$.*

Proof. This is essentially the same as the proof of Lemma 3.5. □

High energy path and active indices. Observe that, unlike the Sender-CD model, we do not have $\mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \ldots$. Therefore, to obtain a lower bound, a new technique is needed. Let $k \geq 3$ be an integer. Among $\{n_i\}_{1 \leq i \leq k}$, our goal is to find an index $\hat{\imath}$ such that for an execution of $\mathcal{A}$ on $n_{\hat{\imath}}$ devices, with probability $1 - 1/\text{poly}(n_{\hat{\imath}})$ there exists a device that uses $\Omega(\log k)$ unit of energy. This is achieved by constructing a *high energy path* $P = (u_1, u_2, \ldots, u_{\hat{t}})$, along with a sequence of sets of *active indices* $\{K_t\}_{1 \leq t \leq \hat{t}}$ such that $i \in K_t$ implies $u_t \in \mathcal{T}_i$. The path P is a directed path in the tree $\mathcal{T}_{\text{no-comm}}$, and u_t belongs to layer t, for each t. The number $\hat{t}$ will de chosen later. Intuitively, any device entering the state u_t is unable to distinguish between $\{n_i\}_{i \in K_t}$, and we will later see that $\{1, \ldots, k\} = K_1 \supseteq K_2 \supseteq \ldots$.

The path P is chosen to contain at least $\Omega(\log k)$ transitions that corresponds to listen or transmit. Therefore, by setting $\hat{\imath}$ as any index in $K_{\hat{t}}$, we have $u_{\hat{t}} \in \mathcal{T}_{\hat{\imath}}$, and hence $\hat{t} \leq d_{\hat{\imath}}$. By Lemma 3.7, in an execution of $\mathcal{A}$ on $n_{\hat{\imath}}$ devices, with probability $1 - {n_{\hat{\imath}}}^{-7}$ there is at least $n_{\hat{\imath}} \cdot p_{u_{\hat{t}}} - \hat{t} \cdot {n_{\hat{\imath}}}^{0.6} = \Omega({n_{\hat{\imath}}}^{0.9}) > 1$ device entering the state $u_{\hat{t}}$, and such a device uses at least $\Omega(\log k)$ unit of energy.

One may attempt to construct the path P by a greedy algorithm which iteratively extends the path by choosing the child state with the highest probability estimate. But this is insufficient to warrant *any* energy expenditure in P. To force an energy expenditure in P, we make use of the property that a state entering u_t is unable to distinguish between the network sizes in $\{n_i\}_{i \in K_t}$ (recall that $i \in K_t$ implies $u_t \in \mathcal{T}_i$). If $i \in K_{d_i}$, the probability that a device s entering the state u_{d_i} remains idle in all the time slots $\{d_i, \ldots, d_{i+1} - 1\}$ must be *small*. Notice that for an execution of $\mathcal{A}$ to be successful, the device s needs to know whether the underlying network size is n_i by the time constraint $T(n_i) < d_{j+1}$. In light of the above, in our construction of P, a special update rule, which ensures one energy expenditure in the time interval $[d_i, d_{i+1} - 1]$, is invoked when a checkpoint d_i is met with $i \in K_{d_i}$.

Formally, the high energy path P and the sequence of sets of active indices $\{K_t\}$ are defined by the following procedure: Initially $P = (u_1)$ contains only the initial state, and $K_1 = \{1, 2, \ldots, k\}$. The following update rules are applied repeatedly until either a terminal state is reached or $K_t = \emptyset$.

Let the current P be $(u_1, u_2, \ldots, u_t)$.

- *Case "regular update": $t \neq d_i$ for all $i \in K_t$.* Let $x \in \{\text{transmit}, \text{listen}, \text{idle}\}$ be chosen to maximize $p_{u_t \rightsquigarrow x}$. If $x = \text{idle}$, append the child of u_t that corresponds to being idle at time slot t to the end of P, and set $K_{t+1} = K_t$. Otherwise, let $m \in \{\lambda_S, \lambda_N\}$ be chosen to maximize the number of indices $j \in K_t$ with $m_{j,t} = m$, and append the child of u_t that corresponds to performing action x and receiving message m at time slot t to the end of P, and set K_{t+1} to the set of indices $j \in K_t$ with $m_{j,t} = m$.

- *Case "special update": $t = d_i$ for some $i \in K_t$.* Let $t' \in \{d_i + 1, \ldots, d_{i+1}\}$ and $x \in \{\mathsf{transmit}, \mathsf{listen}\}$ be chosen to maximize the probability for a device entering u_t to remain idle during all the time slots $\{t, \ldots, t' - 2\}$ and to perform x in time slot $t' - 1$. Let $m \in \{\lambda_S, \lambda_N\}$ be chosen to maximize the number of indices $j \in K_t \setminus \{i\}$ with $m_{j,t'-1} = m$. We let $u_{t'}$ be the unique descendant of u_t resulting from applying $t' - t$ idle actions during time slots $t, \ldots, t' - 2$, and then performing action x and receiving message m at time slot $t' - 1$. The path P is extended to have $u_{t'}$ as the new end point. For each $t'' \in \{t + 1, \ldots, t'\}$, we let $K_{t''}$ to be the set of indices $j \in K_t \setminus \{i\}$ with $m_{j,t'-1} = m$.

We select $\hat{t}$ to be the largest number such that $K_{\hat{t}} \neq \emptyset$. See Figure 4 for an illustration of the update rules. Notice that, in the special update, the reason i must be removed from the set of the active indices is that $\mathcal{T}_i$ only contains states up to layer d_i.

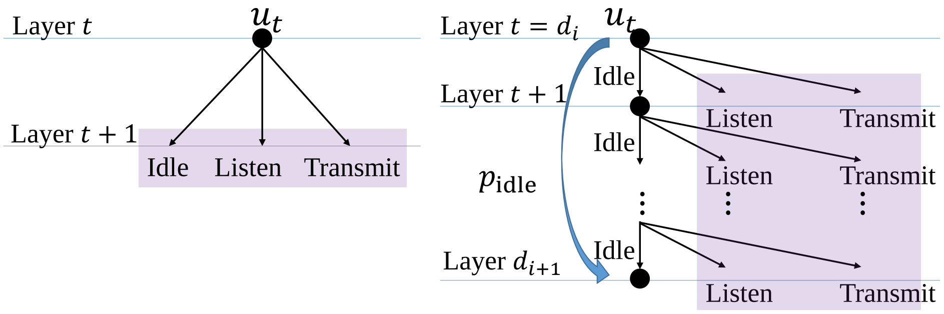


Figure 4: Left: regular update. Right: special update. The shaded region indicates the set of candidate endpoints to extend the current path P.

The lower bound. To obtain the desired lower bound, we need to show that (i) $i \in K_t$ implies $u_t \in \mathcal{T}_i$, and that (ii) the energy cost of the path P is $\Omega(\log k)$. To prove (i), we need to lower bound the probability estimate of the states in P. We observe the following:

- In a regular update at time t, we have $p_{u_{t+1}} \geq \frac{1}{3} p_{u_t}$.
- In a special update at time $t = d_j$, we have $p_{u_{t'}} \geq \frac{p_{u_t}}{2 \cdot (d_{j+1} - d_j)} (1 - p_{\mathsf{idle}}) > \frac{p_{u_t}}{2 d_{j+1}} (1 - p_{\mathsf{idle}})$, where p_{idle} is defined as the probability that a device entering the state u_{d_j} is idle in all the time slots $\{d_j, \ldots, d_{j+1} - 1\}$.

Lemma 3.8. *For each $i \in \{1, \ldots, k\}$ and each $t \in \{1, \ldots, \hat{t}\}$ such that $i \in K_t$, we have $u_t \in \mathcal{T}_i$.*

Proof. Suppose that by induction hypothesis the statement of the lemma holds for all $i' \leq i$ and $t' \leq t$ such that $(i, t) \neq (i', t')$. Proving $p_{u_t} \geq \left(2^{2^{d_i}}\right)^{-1/10} > {n_i}^{-1/10}$ suffices to guarantee that $u_t \in \mathcal{T}_i$. Notice that the procedure constructing P and $\{K_t\}$ guarantees that $i \in K_t$ implies that either (i) u_{t-1} is idle, or (ii) the message received at time slot $t - 1$ that leads to the state u_t is $m_{i,t-1}$. Moreover, the special update implies that $t \leq d_i$.

We claim that $p_{\mathsf{idle}} < \frac{2}{n_j} < 1/2$. This implies that $p_{u_{t'}} \geq \frac{p_{u_t}}{4d_{j+1}}$ in the special update at time $t = d_j$. Therefore, by the above two observations, we have:

$$p_{u_t} \geq 3^{-t} \prod_{j=1}^{\arg\max_{j'}\left\{d_{j'}<t\right\}} \frac{1}{4 \cdot d_{j+1}} = 3^{-d_i} \cdot \left(\frac{1}{4 \cdot d_i}\right)^{i-1} > \left(2^{2^{d_i}}\right)^{-1/10},$$

and hence the lemma holds (notice that $\arg\max_{j'} \left\{d_{j'} < t\right\} = O(\log^* t)$, and also $t \leq d_i$).

What remains to do is to prove that $p_{\mathsf{idle}} < 2/n_j$ when a special update is applied at time d_j. Let v be the state at time d_{j+1} that results from being idle in time slots $\{d_j, \ldots, d_{j+1} - 1\}$ after leaving the state u_{d_j}. Since a special update is applied at time d_j, we have $j \in K_{d_j}$. Also, the fact that $i \in K_t$ guarantees $i \in K_{d_j}$. Therefore, by induction hypothesis, u_{d_j} belongs to both $\mathcal{T}_i$ and $\mathcal{T}_j$. Since $d_j < t \leq d_i$, i and j are distinct indices.

- If v does not give a correct estimate of n_j, then we have $1/n_j \geq (1 - {n_j}^{-7}) \cdot p_{\mathsf{idle}}$, since otherwise Lemma 3.7 implies that with probability higher than $1/n_j$ there is a device in the state v when we execute $\mathcal{A}$ on n_j devices, and such a device does not give a correct estimate of n_j by time $T(n_j) < d_{j+1}$.
- Otherwise, the devices entering the state v have made a decision, and such a decision is not a correct estimate of n_i. Similarly we need to have $1/n_i \geq (1 - {n_i}^{-7}) \cdot p_{\mathsf{idle}}$.

As a result, $p_{\mathsf{idle}} \leq \frac{1}{n_j \cdot (1 - {n_j}^{-7})} < \frac{2}{n_j}$. □

To lower bound the energy cost of P, we observe the following:

- In a regular update at time t, either (i) $|K_{t+1}| = |K_t|$, or (ii) $|K_{t+1}| \geq |K_t|/2$, and there is one unit of energy expenditure at time slot t.
- In a special update at time $t = d_j$, for each $t'' \in \{t+1, \ldots, t'\}$, $|K_{t''}| \geq (|K_t| - 1)/2$, and there is one unit of energy expenditure at time slot $t' - 1$.

Therefore, the total energy expenditure in the path $P = (u_1, \ldots, u_{\hat{t}})$ is $\Omega(\log |K_1| - \log |K_{\hat{t}}|)$. As long as $|K_{\hat{t}}| \leq 2$, the energy cost is at least $\Omega(\log k)$ (recall that $|K_1| = k$). The only possibility to cause $|K_{\hat{t}}| > 2$ is that $u_{\hat{t}}$ is a terminal state, and so no more update rule can be applied to extend the path P any further. However, by Lemma 3.8, $u_{\hat{t}} \in \mathcal{T}_i$ for each $i \in K_{\hat{t}}$, and hence Lemma 3.7 guarantees that, with probability $1 - {n_i}^{-7}$, there is a device s that enters $u_{\hat{t}}$ when $\mathcal{A}$ is executed on n_i devices. If $u_{\hat{t}}$ is a terminal state, then there must exist some $i \in K_{\hat{t}}$ such that s gives an incorrect estimate of n_i. Therefore, $|K_{\hat{t}}| \leq 2$, and we conclude the theorem.

Theorem 3.9. *Let $k \geq 3$ be any integer. Then there exists a network size n with $d_1 \leq n \leq d_{k+1}$ such that if $\mathcal{A}$ is executed on n devices, with probability at least $1 - n^{-7}$ there exists a device s that uses $\Omega(\log k)$ units of energy.*

As long as $T(n) = O\big(\overbrace{2^{2^{\cdots^{2^n}}}}^{\alpha}\big)$ for a constant α and $T(n) \geq n$, the energy cost of $\mathsf{Approximate\ Counting}$ in the $\mathsf{Sender\text{-}CD}$ model is $\Omega(\log(\log^* n))$. Similarly, the lower bound generalizes to higher time constraints as follows: For *any* function $g(n) \leq \log n$, under time constraint $T(n) = g^{-1}(n)$, the energy cost is $\Omega(\log(g^*(n)))$.

3.4 Lower Bound for Other Problems

In this section we demonstrate how our lower bounds proofs can be adapted to Leader Election and contention resolution. First of all, observe that for both Leader Election and contention resolution, before a device terminates, it must successfully transmit or listen once. Let $\mathcal{A}$ be an algorithm for Leader Election or contention resolution. Suppose that the underlying model is Sender-CD, by Lemma 3.5, with probability at least $1 - {n_i}^{-7}$, the runtime of all devices in an execution of $\mathcal{A}$ is at least d_i and is at most $T(n_i) < d_{i+1}$. Therefore, the runtime of a device can be seen as a very loose estimate of the network size.

Now we relax the criteria of Approximate Counting by considering any estimate $d_i \leq \tilde{n} < d_{i+1}$ as a correct estimate of n_i, and we also allow different devices to have different estimates. Then $\mathcal{A}$ actually solves such a relaxed version of Approximate Counting with high probability when $n = n_i$ for some i. Since one estimate $\tilde{n}$ only works for at most one n_i, the lower bound proof for Theorem 3.1 still applies in such a relaxed setting[4], and hence the same lower bound applies to both Leader Election and contention resolution. Similarly, Theorem 3.2 applies to these two problems.

4 Deterministic Algorithms for **Leader Election** and **Census**

In this section we prove a tight upper bound on the deterministic energy complexity of Leader Election in Sender-CD, and also prove a nearly tight upper bound on Census. Our algorithms are inspired by the energy-sharing technique of Jurdzinski et al. [25].

Theorem 4.1. *There exists a deterministic Sender-CD algorithm that solves Leader Election in $O(N)$ time with energy $O(\log \log N)$.*

Theorem 4.2. *There exists a deterministic Sender-CD algorithm that solves Census in $O(N)$ time with energy $O(\log^2 \log N)$.*

4.1 Groups in Deterministic Algorithms

We first introduce the concept of groups, which will be used by our deterministic algorithms. A *group* is an ordered list $(s_0, s_1, \ldots, s_{k-1})$ of active device IDs and each device is in at most one group. The group containing a device s_i is denoted $G = G(s_i)$ and its *rank* is $r(s_i) = i$. The size of the group is $|G| = k$. The representative device s_0 is called the *master* of G, denoted $m(G)$. Each group has a unique group ID, and each device s_i knows the group ID of $G(s_i)$, $m(G(s_i))$, $|G(s_i)|$ and $r(s_i)$.

The notion of groups already breaks symmetry: if an algorithm outputs a single group, the current master of this group can be considered the leader. To solve the more difficult Census problem, we formalize the *information* known by a device s as a set $I(s)$ maintained by s.

Definition 4.1. A group G is *centralized* if $I(m(G)) = \bigcup_{s \in G} I(s)$.

Notice that, by initializing $I(s) = \{s\}$ for each active device ID s, the Census problem is solved when there is one device collecting $\{I(s)\}_s$ over all active IDs s.

[4] Also observe that our lower bounds work for estimating $\log^* n$ within a constant additive error.

4.2 A Simple Census/Leader Election Algorithm

In this section we show how to leverage the notion of groups in order to distribute energy costs, by presenting a simple deterministic Census algorithm SimpleCensus($\hat{N}$). Prior to executing the algorithm, it is guaranteed that the devices are partitioned into centralized groups with group ID space $[\hat{N}]$, and each group has size larger than $\log \hat{N}$. After the execution of SimpleCensus($\hat{N}$), one device s of rank $h = \lceil \log \hat{N} \rceil$ identifies itself as the leader which collects all input information.

The algorithm SimpleCensus($\hat{N}$) is executed recursively. If $\hat{N} = 1$, the leader device is simply the master of the only group. If $\hat{N} > 1$, the algorithm recursively calls SimpleCensus($\lceil \hat{N}/2 \rceil$) two times on the two halves of the ID space, respectively. Suppose the two leader devices of the two recursive calls are s_0 and s_1, which both have rank $h - 1$ in their own groups.

In the final two time slots, s_0 announces its group ID and $I(s_0)$, and s_1 does likewise. Each device with rank h in its group listens to these two slots. If s_0 exists, the leader is the device in $G(s_0)$ of rank h; if s_0 does not exist and s_1 exists, the leader is the device in $G(s_1)$ with rank h. The leader device s then sets $I(s) \leftarrow I(s_0) \cup I(s_1)$. Observe that any device listens to at most two time slots and transmits in at most one.

Theorem 4.3. *The algorithm SimpleCensus($\hat{N}$) successfully assigns a leader device s in $O(\hat{N})$ time, where $I(s)$ contains $\bigcup_G I(m(G))$ for every group G. Moreover, in each group G, devices with ranks $0, 1, \ldots, \lceil \log \hat{N} \rceil$ spend constant energy and all others spend zero energy.*

The algorithm SimpleCensus(N) solves Census and Leader Election efficiently, and the strategy behind our proofs of Theorem 4.1 and Theorem 4.2 is to merge the devices into groups to $\lceil \log N \rceil$, so that an application of SimpleCensus(N) solves the problem.

4.3 An $O(\log \log N)$ Leader Election Algorithm

In this section we prove Theorem 4.1. Without loss of generality, we assume $\log N$ is an integer. Our algorithm consists of $\log \log N$ *phases*. All devices participate initially and may drop out in the middle of the algorithm. We maintain the following invariants:

1. At the beginning of the i-th phase, all participating devices are organized into active groups of size exactly 2^i.
2. The number of active groups is at least one.

At the beginning of Phase 0, each device forms a singleton group. During each phase, some pairs of groups are merged into groups of doubled size; the devices in the unmerged groups are terminated. There are two possible outcomes of our algorithm.

- After Phase i there is only one active group G remaining, for some $i \leq \log \log N - 1$. Then the algorithm is terminated at Phase i with the master of G being the leader.
- More than one active group remains after Phase $\log \log N - 1$. As the groups that survive until the end have size $\log N$, a leader can be elected by applying SimpleCensus(N).

Intuitively, in Phase i of our algorithm, each active group attempts to find another active group to merge into a group with size 2^{i+1}. All groups that are not merged during Phase i are terminated. We will later see that the energy cost per active group is $O(\log \log N)$. For each phase, there is a *representative* in each group which is responsible for carrying out the procedure in this phase. At the end of a phase the representative of a group announces their new group membership to other group members. The energy cost in a phase for a device s is $O(\log\log N)$ if s serves as a representative in this phase, and is $O(1)$ otherwise. We will later see that each device is responsible for no more than 2 phases throughout the algorithm. Therefore, the total energy cost of the algorithm is

$$2 \cdot O(\log\log N) + \sum_{i=0}^{\log\log N-1} O(1) = O(\log\log N).$$

In the subsequent discussion we focus on describing and analyzing one phase of the algorithm, which is based on the procedure $\mathsf{DetLE}(\hat{N})$. Prior to executing the procedure $\mathsf{DetLE}(\hat{N})$, it is guaranteed that the devices are partitioned into centralized groups with group ID space $[\hat{N}]$, each group has the same size, and the number of groups is at least 2. By the end of the execution, each group has either dropped out or merged with another group. The procedure $\mathsf{DetLE}(\hat{N})$ is defined recursively as follows.

Base Case. There are exactly two groups G_0 and G_1, and $\hat{N} = 2$. Using two time slots, the representatives of the two groups exchange the information $I(G_0)$ and $I(G_1)$. Then the two groups are merged.

Inductive Step. Uniformly divide the group ID space $[\hat{N}]$ into $\hat{N}' = \lceil\sqrt{\hat{N}}\rceil$ intervals. For each $j \in [\hat{N}']$, if there are at least two groups whose ID belongs to the j-th interval of $\hat{N}'$ IDs (this can be checked in one time slot), recursively call $\mathsf{DetLE}(\hat{N}')$ on the j-th interval of $\hat{N}'$ IDs. Each group that does not participate in any recursive call change its ID from d to $\lfloor d/\hat{N}'\rfloor$. If the number of remaining groups is at least two (again, this can be checked in one time slot), then call $\mathsf{DetLE}(\hat{N}')$ on these groups.

By initially assigning each active group G the group ID $\min_{s\in G}\mathrm{ID}(s)$, an execution of $\mathsf{DetLE}(N)$ fulfills the task of one phase of our algorithm. The inductive step guarantees that (i) each recursive call invoked has at least two groups participating, and (ii) as long as the number of active groups is at least 2, at least one recursive call is invoked. Therefore, given that the number of active groups is at least 2 in the beginning of a phase, we must reach the base case and have two groups merged.

Let $E(N)$ ($T(N)$) denote the energy complexity (time complexity) of $\mathsf{DetLE}(N)$. Since each representative is involved in at most one recursive call of $\mathsf{DetLE}(\lceil\sqrt{\hat{N}}\rceil)$, we have $E(\hat{N}) = E(\lceil\sqrt{\hat{N}}\rceil) + O(1)$. Hence the energy cost is $O(\log\log N)$. Similarly, since $\mathsf{DetLE}(N)$ invokes at most $\lceil\sqrt{\hat{N}}\rceil + 1$ recursive calls of $\mathsf{DetLE}(\lceil\sqrt{\hat{N}}\rceil)$, we have $T(\hat{N}) = (\lceil\sqrt{\hat{N}}\rceil + 1) \cdot T(\lceil\sqrt{\hat{N}}\rceil) + O(\lceil\sqrt{\hat{N}}\rceil)$. Hence the runtime is $O(N)$.

Energy Sharing. As mentioned earlier, in order to save energy, only the representatives of the groups participate in the procedure of each phase. For Phase i ($i > 0$), the device with rank $2^{i-1}-1$ in group G will be selected as the representative of G. It is straightforward to verify that, after Phase 0, each device s serves as a representative for at most one time.

We allocate N extra time slots after each phase to let the representatives announce their new group membership to other group members. Let G and G' be two groups that are merged in phase i, and let s (s') be the representative of the group G (G') in phase i. If the ID of the group G

(G') is g (g'), then s (s') announces the ID of the merged group $G \cup G'$ to all other members in G (G') at the g-th (g'-th) time slot. Each device in the merged group then recomputes their new ranks and updates the group size locally. The message size complexity is $O(\log N)$. Also, if G is not successfully merged during phase i, the representative of G can also transmit to terminate all devices in G.

Time complexity. Recall that there are $O(\log\log N)$ phases during the execution of the algorithm, each uses $O(N)$ time slots, thus the total time complexity of the algorithm is $O(N \log\log N)$.

The running time can be further reduced to $O(N)$ by a preprocessing which uniformly divides the ID space into $\frac{N}{\log\log N}$ intervals and calls $\mathsf{SimpleCensus}(\log\log N)$ on every interval. Here the initial group size is 1, and each device simulates $\log(\log\log N)$ devices in its group. The preprocessing uses $O(N)$ time and $O(\log\log\log N) = o(\log\log N)$ energy, and after that the leader devices from the $\frac{N}{\log\log N}$ intervals execute the algorithm in this section on the ID space $\left[\frac{N}{\log\log N}\right]$.

4.4 An $O(\log^2 \log N)$ Census Algorithm

In this section we prove Theorem 4.2. We describe the deterministic algorithm $\mathsf{DetCensus}(\hat{N}, l)$. Prior to executing the algorithm $\mathsf{DetCensus}(\hat{N}, l)$, it is guaranteed that the devices are partitioned into centralized groups with group ID space $[\hat{N}]$, and each group has the same size 2^l. It is ensured that the group size is always a power of 2 ranging from 2^0 to $2^{\lceil \log\log N \rceil}$. After the execution, there is one centralized group $G^\star$, which is the union of some input groups. Devices in $G^\star$ all identify $G^\star$ as the leader group, and $I(m(G^\star)) = \bigcup_s I(s)$ for every device s in the input groups, even if $s \notin G^\star$.

The algorithm is executed recursively. There are three base cases of the recursion.

Base Case 1. If $l = \lceil \log\log N \rceil$, all the devices execute $\mathsf{SimpleCensus}(\hat{N})$ over the ID space $[\hat{N}]$. The leader device s then transmits the group ID of $G^\star = G(s)$ and all devices listen. Any device not in $G^\star$ halts.

Base Case 2. If $l < \lceil \log\log N \rceil$, and there is only one group in the ID space, then the only group is the leader group $G^\star$. In $\mathsf{Sender\text{-}CD}$ the devices can check whether they are in this case using unit energy. In the first time slot the masters of each group transmit and all devices listen.

Base Case 3. If $l < \lceil \log\log N \rceil, \hat{N} = 2$, and the two groups G_0 and G_1 both exist. In two time slots, $m(G_0)$ and $m(G_1)$ transmit $I(m(G_0))$ and $I(m(G_1))$ and all devices in G_0 and G_1 listen. Then the two groups are merged into a leader group $G^\star = G_0 \cup G_1$ of size 2^{l+1}, such that $m(G^\star)$ is the former $m(G_0)$. Every device s formerly in G_1 sets $r(s) \leftarrow r(s) + 2^l$.

When the conditions for the above three base cases do not fit, the algorithm uses three phases, where the first two are recursive:

Phase 1. Uniformly divide the group ID space $[\hat{N}]$ into $\hat{N}' = \lceil \sqrt{\hat{N}} \rceil$ intervals. For each $j \in [\hat{N}']$, recursively call $\mathsf{DetCensus}(\hat{N}', l)$ on the j-th interval of $\hat{N}'$ IDs, and denote the leader group by G_j which uses j as the new group ID. Notice that because of mergers, $|G_j|$ varies from 2^l to $2^{\lceil \log\log N \rceil}$.

Phase 2. For each $k = l, l+1, \ldots, \lceil \log\log N \rceil$, recursively call $\mathsf{DetCensus}(\hat{N}', k)$ on the ID space $[\hat{N}']$, restricting the input groups to those among $\{G_j\}$ with $|G_j| = 2^k$. Denote the leader

group of this recursive call by $G^{(k)}$. Again, $|G^{(k)}|$ varies between 2^k and $2^{\lceil \log\log N \rceil}$. Let k_0 be the maximum index such that $G^{(k_0)}$ exists. The goal of the next phase is to aggregate all the information in $G^{(k_0)} = G^\star$.

Phase 3. This phase has only $2(\lceil \log\log N \rceil - l)$ time slots, partitioned into 2 slots for each rank k from $\lceil \log\log N \rceil - 1$ to l, in *decreasing* order. In the first slot, if $k_0 > k$, the device with rank k in $G^{(k_0)}$ transmits $\bigcup_{k'>k} I(m(G^{(k')}))$; all devices in $G^{(k)}$ listen and the device with rank $k-1$ in $G^{(k_0)}$ listens. If $k_0 > k$, the leader $m(G^{(k)})$ transmits $I(m(G^{(k)}))$ in the second slot, the device with rank $k-1$ in $G^{(k_0)}$ listens, and all devices in $G^{(k)}$ halt. If the first slot is silent, devices in $G^{(k)}$ deduce that $k_0 = k$. All members of $G^\star = G^{(k_0)}$ listen to the two time slots for rank l to learn $\bigcup_k I(m(G^{(k)}))$.

To solve the Census problem, each device in the ID space $[N]$ regards itself as a group of size 1, whose group ID is the same as the device ID, and then executes DetCensus$(N, 0)$. Let $T(\hat{N})$ be the maximum running time of DetCensus$(\hat{N}, l)$ over all $l < \lceil \log\log N \rceil$. The function $T(\hat{N})$ satisfies the following recursive formula:

$$T(\hat{N}) \le (\sqrt{\hat{N}} + \log\log N)T(\sqrt{\hat{N}}) + O(\sqrt{\hat{N}}) + O(\log\log N),\ T(0) = O(1).$$

The additive $O(\sqrt{\hat{N}})$ reflects the time for DetCensus$(\hat{N}', \lceil \log\log N \rceil)$ in Phase 2 and the $O(\log\log N)$ is the time for Phase 3. Let $f(N) = (\log\log N)^{\log\log\log\log N + 2}$. In Appendix A we prove that $T(N) = O(N \cdot f(N))$. The running time can be further reduced to $O(N)$ by a preprocessing which uniformly divides the ID space into $N/f(N)$ intervals and calls SimpleCensus$(f(N))$ on every interval, as what we have done in Section 4.3.

To analyze the energy cost of DetCensus, we make use of the following lemma, which is proved by induction on $\hat{N}$.

Lemma 4.4. *If $l < \lceil \log\log N \rceil$, and the leader group of DetCensus$(\hat{N}, l)$ has size 2^l, it must have halted in Base Case 2.*

Let $E(\hat{N}, l, k)$ be the maximum energy cost of a device s during the execution of DetCensus$(\hat{N}, l)$ which satisfies $|G(s)| = 2^k$ after the execution. Suppose $s \in G_j$ after Phase 1 and $|G_j| = 2^{k'}$ for some $k' \ge l$. The energy spent by s is $E(\hat{N}', l, k') + E(\hat{N}', k', k) + c$, for some $c = O(1)$. Thus, E satisfies the following inductive definition.

$$\begin{aligned}
E(2, l, l+1) &= c && \text{Base Case 3}\\
E(\hat{N}, l, l) &= c && \text{Base Cases 1 and 2, } \hat{N} > 2, l \le \lceil \log\log N \rceil\\
E(\hat{N}, l, k) &\le \max_{k' \in [l,k]} \left(E(\lceil \sqrt{\hat{N}} \rceil, l, k') + E(\lceil \sqrt{\hat{N}} \rceil, k', k) + c \right) && \text{If } \hat{N} > 2, k > l, l < \lceil \log\log N \rceil
\end{aligned}$$

The bound on $E(\hat{N}, l, l)$ follows from Lemma 4.4 when $l < \lceil \log\log N \rceil$ and Theorem 4.3 when $l = \lceil \log\log N \rceil$. By induction, $E(\hat{N}, l, k) \le 2c((\log\log \hat{N})(k - l) + 1/2)$. Thus, the energy cost of DetCensus$(N, 0)$ is $O(\log^2 \log N)$.

5 Deterministic Dense Leader Election and Census

In this section we present a deterministic algorithm that solves Leader Election and Census with energy cost $O(\alpha(N))$ when the input is *dense* in the ID space, i.e., the number of active devices

n is at least $c \cdot N$ for a fixed constant $c > 0$. Here $\alpha(N)$ denotes the inverse-Ackermann function. Formally, we prove the following theorem:

Theorem 5.1. *There exists a deterministic algorithm in No-CD model that solves Leader Election and Census with energy cost $O(\alpha(N))$ and time $O(N)$ for the case $N = \Theta(n)$.*

At the beginning of our algorithm, each device is a group[5] with only one member. A key subroutine of our algorithm, $\mathsf{DenseAlgo}_i(\hat{N}, j)$, recursively merges groups into fewer and larger ones. When only one group G remains, a leader is successfully elected by making the master s of G identify itself as the leader and all other devices identify themselves as follower.

A group G is said to be *j-rich* if the size of G is at least j. We write *j-density* to denote the proportion of *j-rich* groups in all groups. From the above definition, it is straightforward to see that the 1-density is at least c at the beginning of the algorithm.

The input/output specification and the efficiency of the subroutine $\mathsf{DenseAlgo}_i(\hat{N}, j)$ are described as follows:

- Input: Before the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$, all groups have group IDs in $[\hat{N}]$, and the j-density is at least $\frac{1}{\log j}$. All groups have size either zero or at least j.

- Output: After the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$, all groups have group IDs in $[\lceil \hat{N}/b_i(j) \rceil]$, and the $a_i(j)$-density is at least $\frac{1}{\log a_i(j)}$, for some functions $a_i(j)$ and $b_i(j)$ to be determined. All groups have size either zero or at least $a_i(j)$ and are centralized[6]. In addition, there are $[\lceil \hat{N}/b_i(j) \rceil]$ *output time slots*. In the k-th output time slot, the leader of the new group G with group ID k announces the list of all members of G.

- Energy cost: Each device spends $O(i)$ energy during the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$.

- Time slots: $\mathsf{DenseAlgo}_i(\hat{N}, j)$ uses $O(\hat{N})$ time slots (with taking the output time slots into consideration).

In addition, we always assume $j > 32$.

In Section 5.1, we give the formal description of the subroutine $\mathsf{DenseAlgo}_i(\hat{N}, j)$. In Section 5.2 and Section 5.3 we illustrate how to use $\mathsf{DenseAlgo}_i(\hat{N}, j)$ to solve Leader Election and Census, and thus prove Theorem 5.1.

5.1 The Subroutine $\mathsf{DenseAlgo}_i(\hat{N}, j)$

In this section we present the subroutine $\mathsf{DenseAlgo}_i(\hat{N}, j)$ that meets the required specification. The first step of $\mathsf{DenseAlgo}_i(\hat{N}, j)$ is the *initialization step*, whose purpose is to create groups with enough number of devices to run the subsequent steps.

Initialization step. The group ID space is partitioned into $\lceil \hat{N}/2^j \rceil$ consecutive parts, and each part contains 2^j group IDs, possibly except for the last part. For each part p, we run SimpleCensus to merge all j-rich groups in part p into a single centralized group. Notice that it is guaranteed in the input specification of $\mathsf{DenseAlgo}_i(\hat{N}, j)$ that all groups have size either zero or at least j. After merging each part into a single group, all devices in groups with size smaller than j^5 are terminated. The following lemma summarizes the performance of the initialization step.

[5] See Section 4.1 for the definition of group.

[6] See Section 4.1 for the definition of centralized.

Lemma 5.2. *Suppose $\hat{N} > 2^j$ and $j > 32$, all groups have IDs in $[\hat{N}]$ and the j-density is at least $\frac{1}{\log j}$. After the initialization step, all groups have IDs in $[\lceil \hat{N}/2^j \rceil]$ and the j^5-density is at least $\frac{1}{4\log j}$. The initialization step uses $O(\hat{N})$ time slots and each device spends constant energy during the initialization step.*

Proof. The worst case occurs when all j-rich groups have precisely size j and all other groups have size zero. Before the initialization step, the total number of devices is at least $\frac{1}{\log j} \cdot j \cdot \hat{N}$. After the initialization step, each group contains at most $j \cdot 2^j$ devices and the number of devices which are in groups with size smaller than j^5 is at most

$$j^5 \cdot \lceil \hat{N}/2^j \rceil < \hat{N}.$$

After the initialization step, the number of j^5-rich groups is minimized when they have the maximum size, which is

$$\left\lfloor \frac{1}{\log j} \cdot j \cdot \hat{N}/(j \cdot 2^j) \right\rfloor > \frac{1}{4\log j} \lceil \hat{N}/2^j \rceil.$$

Meanwhile, the number of time slots and energy cost of the initialization step directly follows Theorem 4.3. So the lemma holds. □

Once $\hat{N} \le 2^j$, we can then use SimpleCensus to merge all j-rich groups into a single group and skip all further time slots.

Algorithm. The subroutine $\mathsf{DenseAlgo}_i(\hat{N}, j)$ is defined inductively as follows. For the base case of $i = 0$, $\mathsf{DenseAlgo}_0(\hat{N}, j)$ consists of only the initialization step. For $i > 0$, after the initialization step, all members of groups with size less than j^5 have been terminated. For each j^5-rich group G, each member of G constructs j disjoint subgroups $G^1, G^2, \ldots, G^j$, with each subgroup containing $\lfloor |G|/j \rfloor \ge j^4$ devices. For $1 \le r < j$, as G^r and G^{r+1} have the same size, we set up a bijection $\phi_r : G^r \to G^{r+1}$. There are j recursive calls to $\mathsf{DenseAlgo}_{i-1}$ during the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$ and only devices in the r-th subgroup participate in the r-th recursive call. For each device s in the r-th subgroup G^r, after s finishes the r-th recursive call, if s is not terminated yet, $\phi_r(s)$ continues to play the role of s in the $(r+1)$-th recursive call. $\phi_r(s)$ learns all the information known by s, by listening to output time slot of the r-th recursive call.

The parameters of the j recursive calls are chosen to make sure the input specification of $\mathsf{DenseAlgo}_{i-1}$ is always met. By Lemma 5.2, after the initialization step, the j^5-density is at least $\frac{1}{4\log j}$, and thus the j^4-density is at least $\frac{1}{4\log j} = \frac{1}{\log j^4}$ after partitioning each j^5-rich group into j subgroups, each with size at least j^4. $\mathsf{DenseAlgo}_i(\hat{N}, j)$ can therefore invoke $\mathsf{DenseAlgo}_{i-1}\left(\lceil \hat{N}/2^j \rceil, j^4\right)$. After the first recursive call, all groups have IDs in $\lceil \frac{\hat{N}}{2^j \cdot b_{i-1}(j^4)} \rceil$ and the $a_{i-1}(j^4)$-density is at least $\frac{1}{\log a_{i-1}(j^4)}$, and thus $\mathsf{DenseAlgo}_i(\hat{N}, j)$ invokes $\mathsf{DenseAlgo}_{i-1}\left(\lceil \frac{\hat{N}}{2^j \cdot b_{i-1}(j^4)} \rceil, a_{i-1}(j^4)\right)$ as the second recursive call. In general, $\mathsf{DenseAlgo}_i(\hat{N}, j)$ invokes

$$\mathsf{DenseAlgo}_{i-1}\left(\left\lceil \frac{\hat{N}}{2^j \cdot \prod_{t=1}^{r-1} b_{i-1}\left(a_{i-1}^{(t-1)}(j^4)\right)} \right\rceil, a_{i-1}^{(r-1)}(j^4)\right)$$

as the r-th recursive call.

After the last recursive call ends, all groups have IDs in

$$\left[\!\left[\left\lceil \frac{\hat{N}}{2^j \cdot \prod_{t=1}^{j} b_{i-1}\left(a_{i-1}^{(t-1)}(j^4)\right)} \right\rceil\right]\!\right],$$

and the $a_{i-1}^{(j)}(j^4)$-density is at least $\frac{1}{\log\left(a_{i-1}^{(j)}(j^4)\right)}$. Thus, we define

$$b_i(j) = 2^j \cdot \prod_{r=1}^{j} b_{i-1}\left(a_{i-1}^{(r-1)}(j^4)\right),$$

and $a_i(j) = a_{i-1}^{(j)}(j^4)$ for $i > 0$. For the base case of $i = 0$, by Lemma 5.2 we define $a_0(j) = j^5$ and $b_0(j) = 2^j$, as $\frac{1}{4\log j} > \frac{1}{5\log j} = \frac{1}{\log j^5} = \frac{1}{\log a_0(j)}$. Therefore, the choices of $a_i(j)$ and $b_i(j)$ are valid, and they are lower bounded by the standard Ackermann function.

Recall that after the last recursive call ends, all members of each group are from the j-th subgroup. Notice that it is guaranteed that all groups are centralized after the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$, thus after the last recursive call ends, the leader has the information of all its groups members and can thus merge devices in the first $j-1$ subgroups back into corresponding groups and centralize the group.

Analysis. During $\mathsf{DenseAlgo}_i(\hat{N}, j)$, each device uses constant energy in the initialization step. For each device s in the r-th subgroup, s uses constant energy to listen for the information $\phi_{r-1}^{-1}(s)$. Each device only participates in one recursive call, which takes $O(i-1)$ energy by induction. Thus each device spends $O(i)$ energy during the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$.

Denote $T_i(\hat{N}, j)$ to be the total number of time slots used by $\mathsf{DenseAlgo}_i(\hat{N}, j)$. The initialization step uses $O(\hat{N})$ time slots. Also, the number of output time slots is $\lceil \hat{N}/b_i(j) \rceil \le \hat{N}$. Thus, we have

$$T_i(\hat{N}) \le O(\hat{N}) + T_{i-1}(\lceil \hat{N}/2^j \rceil) + T_{i-1}\left(\left\lceil \frac{\hat{N}}{2^j \cdot b_{i-1}(j^4)} \right\rceil\right) + \ldots .$$

for $i > 0$ and

$$T_0(\hat{N}) = O(\hat{N}).$$

Noticing that $j > 32$ and $b_{i-1}(j^4) > 2$ when $i > 0$, thus we have $T_i(\hat{N}, j) = O(\hat{N})$.

Notice that during the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$, all the recursive calls have the second parameter higher than j, thus the assumption $j > 32$ is valid.

5.2 Algorithm for Leader Election

The algorithm of Theorem 5.1 for Leader Election is a *preprocessing step* followed by an execution of DenseAlgo. The purpose of the preprocessing step is to guarantee that the input specification of $\mathsf{DenseAlgo}_i(\hat{N}, j)$ is met. In the preprocessing step, we first partition the ID space into $\left\lceil \hat{N}/\lceil 2^{\frac{8}{c}} \rceil \right\rceil$ parts, where each part contains at most $\lceil 2^{\frac{8}{c}} \rceil$ IDs, and then use the algorithm of Theorem 4.3 to merge all devices in each part into a single group by letting each device emulate $\lceil 8/c \rceil$ devices. Members of groups with size smaller than $\lceil 2^{\frac{4}{c}} \rceil$ are terminated. Each device spends $O(1/c)$ energy during the preprocessing step and the time complexity of the preprocessing step is $O(N)$. After the preprocessing step, we calculate the minimum i such that $b_i\left(\lceil 2^{\frac{4}{c}}\rceil\right) \geq \left\lceil \hat{N}/\lceil 2^{\frac{8}{c}}\rceil \right\rceil$ and then execute $\mathsf{DenseAlgo}_i\left(\left\lceil N/\lceil 2^{\frac{8}{c}}\rceil \right\rceil, \lceil 2^{\frac{4}{c}}\rceil\right)$. By using an averaging argument we can show that after the preprocessing step, the input specification of $\mathsf{DenseAlgo}_i\left(\left\lceil N/\lceil 2^{\frac{8}{c}}\rceil \right\rceil, \lceil 2^{\frac{4}{c}}\rceil\right)$ is satisfied. Readers can refer to the proof of Lemma 5.2 for a formal proof.

According to the output specification, after the execution of $\mathsf{DenseAlgo}_i\left(\left\lceil N/\lceil 2^{\frac{8}{c}}\rceil \right\rceil, \lceil 2^{\frac{4}{c}}\rceil\right)$, only one group remains, and thus a leader is successfully elected. Later we have shown in Section 5.1, $a_i(j)$ and $b_i(j)$ are two functions lower bounded by the standard Ackermann function, and thus $i = O(\alpha(N))$. Since the energy cost of $\mathsf{DenseAlgo}_i(\hat{N}, j)$ is $O(i)$ and $\mathsf{DenseAlgo}_i(\hat{N}, j)$ use $O(\hat{N})$ time slots, thus there exists an algorithm that solves Leader Election with energy cost $O(\alpha(N))$ and time $O(N)$ when $N = \Theta(n)$.

5.3 Algorithm for Census

In this section, we further modify the algorithm in Section 5.2 to solve Census. We first show that after the execution of the algorithm in Section 5.2, the only remaining group G has size $\Omega(n)$. We first partition the ID space into parts with size $O(1/c)$. For each part p, we assign one device in G to collect the list of active stations in p, by using constant energy. Finally, we merge the information of all group members by using $O(N)$ time slots and constant energy, and thus Census is solved.

Number of terminated devices. During the execution of the whole algorithm, a device is terminated only in the following two cases.

- During the preprocessing step, devices in groups with size smaller than $\lceil 2^{\frac{4}{c}}\rceil$ are terminated.
- During the initialization step, devices in groups with size smaller than j^5 are terminated.

During the preprocessing step, the fraction of devices that are terminated is at most

$$\frac{\left\lceil N/\lceil 2^{\frac{8}{c}}\rceil \right\rceil \cdot \lceil 2^{\frac{4}{c}}\rceil}{c \cdot N} = \frac{1}{2^{\Omega(1/c)}}.$$

During the initialization step, the fraction of devices that are terminated is at most

$$\frac{\lceil N/2^j\rceil \cdot j^5}{\frac{1}{\log j} \cdot j \cdot N} < 1/j$$

by noticing that $j > 32$.

Denote $\textsc{Terminated}_i(j)$ to be the fraction of devices the are terminated during the execution of $\mathsf{DenseAlgo}_i(\hat{N}, j)$, now we prove that $\textsc{Terminated}_i(j) \leq \frac{2}{j}$.

$$\begin{aligned}
&\textsc{Terminated}_i(j)\\
\leq& 1 - \left(1 - \frac{1}{j}\right) \cdot \left(1 - \textsc{Terminated}_{i-1}(j^4)\right) \cdot \left(1 - \textsc{Terminated}_{i-1}(a_{i-1}(j^4))\right) \cdots \\
\leq& 1 - \left(1 - \frac{1}{j}\right) \cdot \left(1 - \frac{2}{j^4}\right) \cdot \left(1 - \frac{2}{a_{i-1}(j^4)}\right) \cdots && \text{(By induction hypothesis)}\\
\leq& \frac{2}{j}.
\end{aligned}$$

Thus, the overall fraction of devices that are terminated during the execution of the whole algorithm is at most

$$1-\left(1-\frac{1}{2^{\Omega(1/c)}}\right)\cdot\left(1-\frac{2}{\lceil 2^{8/c}\rceil}\right)=\frac{1}{2^{\Omega(1/c)}},$$

which implies the only remaining group after the execution of DenseAlgo has size $\Omega(n)$.

Algorithm. After the execution of DenseAlgo, we partition the ID space into consecutive parts with size $O(1/c)$. Denote $G=(s_0,s_1,s_2,\ldots,s_{|G|-1})$ to be the only remaining group after the execution of DenseAlgo. For each part p, we assign one device s in G to collect all active devices in p, by letting s listen for constant times and each device in p transmits once. We denote $I(s_i)$ to be the list of active devices collected by s_i.

After that, s_0 trasnmits $I(s_0)$ and s_1 listens, and then s_1 transmits $I(s_0)\cup I(s_1)$ and s_2 listens, $\ldots$. Finally, $s_{|G|-1}$ transmits $I(s_0)\cup I(s_1)\cup\ldots\cup I(s_{|G|-1})$ and all devices listen, and thus Census is solved.

Clearly, this step uses $O(N)$ time slots and each device spends constant energy.

6 Randomized Upper Bounds

In this section we design algorithms for Approximate Counting matching the energy complexity lower bound proved in Section 3. In [7], an algorithm for Approximate Counting in Strong-CD model using $O(\log(\log^* n))$ energy is devised. They showed that any circuit of constant fan-in, with input bits encoded as noise/silence time slots, can be simulated with $O(1)$ energy cost. An estimate of the network size can be computed by such a circuit. In this section we demonstrate a different approach to Approximate Counting that is based on our dense Census algorithm and works for all four collision detection models.

Theorem 6.1. *There is an algorithm that, with probability $1-1/\mathrm{poly}(n)$, solves Approximate Counting in $n^{o(1)}$ time with energy cost:*

- *$O(\log^* n)$, if the model is Sender-CD, or is No-CD model with $n>1$.*[7]
- *$O(\log(\log^* n))$, if the model is Strong-CD or Receiver-CD.*

When the algorithm fails, the devices are allowed to behave *arbitrarily.* In particular, a device in a failed execution may consume unbounded amount of energy and never halt. This is a standard assumption in some prior works (e.g. [26, 27, 8]). The exponential difference in energy complexity reflects the fact that receiver-side collision detection enables the devices to perform a binary search on a set of candidate estimates of network sizes.

6.1 Testing Network Size

In this section we show that testing whether a given estimate $\tilde{n}$ of the network size n is accurate can be done by dense Census.

[7] In this section, we always assume $n>1$ in the No-CD model. In the No-CD model a sender cannot simultaneously listen to the channel, and so a device never hears any message if it is the only device in the network. However, when n is large enough, with high probability a device also does not hear any message in the first few time slots. It seems hopeless to have an algorithm that detects loneliness of a device, i.e. distinguishes between $n=1$ and $n>1$. See [21, 20] for other work dealing with the $n\overset{?}{=}1$ issue.

Goal. There are n participating devices agreeing on a number $\tilde{n}$. The goal is to decide whether $\tilde{n}$ is a good estimate of n. We require that a leader elected if $n/1.5 \leq \tilde{n} \leq 1.5 \cdot n$, and no leader is elected if $\tilde{n} \geq 1.9 \cdot n$ or $\tilde{n} \leq n/1.9$.

Assigning IDs. Running a deterministic algorithm requires IDs. Suppose that the underlying model is Strong-CD or Sender-CD. We allocate $N = \Theta(\log \tilde{n})$ time slots. Each participating device transmits a message in each time slot with probability $1/\tilde{n}$. If a device s hears its message at i^{th} time slot, then s assigns itself the ID i.

Recall that in the dense Census with parameter c and ID space $[N]$, if $n \geq c \cdot N$, a leader who collects all IDs of the participating devices is elected. Therefore, the algorithm allows us to distinguish between the case where the number of participating devices is at least $c \cdot N$ from the case the number of participating devices is less than $c \cdot N$.

Fact 1. *For any two positive integers $n, \tilde{n}$ such that $\tilde{n} \geq 100$, we have:*

- $\Pr[\text{binom}(n, p = 1/\tilde{n}) = 1] < 0.32$ *when $\tilde{n} \geq 1.9n$ or $\tilde{n} \leq n/1.9$.*[8]
- $\Pr[\text{binom}(n, p = 1/\tilde{n}) = 1] > 0.33$ *when $n/1.5 \leq \tilde{n} \leq 1.5n$.*

By Fact 1, a standard application of Chernoff bounds leads to the following lemma.

Lemma 6.2. *For any $\tilde{n} \geq 100$, with probability $1 - \min\{n^{-\Omega(1)}, \tilde{n}^{-\Omega(1)}\}$, we have:*

- *The number of the devices assigned an ID is less than $0.325 \cdot N$ when $\tilde{n} \geq 1.9n$ or $\tilde{n} \leq n/1.9$.*
- *The number of the devices assigned an ID is more than $0.325 \cdot N$ when $n/1.5 \leq \tilde{n} \leq 1.5n$.*

Algorithm. The algorithm Test-Network-Size($\tilde{n}$) is described as following. First we do the ID assignment, and then the dense Census with ID space $[N]$ and parameter $c = 0.325$ is executed on the devices that are assigned IDs. Notice that it is possible that a device is assigned to multiple IDs, and in such case the device simulates multiple devices of different IDs in the dense Census algorithm. Only the devices that are assigned less than β tasks participate in the algorithm, where β is a constant to be determined. If the leader elected collects less than $c \cdot N$, it resets itself as follower.

Analysis. Observe that the probability that there exists a device assigned to at least β tasks is $n^{-\Omega(\beta)}$ for the case $n/1.5 \leq \tilde{n} \leq 1.5 \cdot n$. Therefore, by selecting β as a large enough constant, with probability $n^{-\Omega(1)}$ all devices are assigned to less than $\beta = O(1)$ tasks. By Lemma 6.2, with probability $1 - \min\{n^{-\Omega(1)}, \tilde{n}^{-\Omega(1)}\}$, a leader is elected if $n/1.5 \leq \tilde{n} \leq 1.5 \cdot n$, and no leader is elected if $\tilde{n} \geq 1.9 \cdot n$ or $\tilde{n} \leq n/1.9$.

The time spent on both assigning IDs and the dense Census are $O(N) = O(\log \tilde{n})$, and hence the algorithm takes $O(\log \tilde{n})$ time. The energy complexity is $(\beta - 1) \cdot O(\alpha(N)) = O(\alpha(\tilde{n}))$.

[8] $\text{binom}(n, p)$ is defined as the random variable that follows binomial distribution with parameters n and p.

Handling networks with no sender-side collision detection. One issue arises when there is no sender-side collision detection (i.e. Receiver-CD and No-CD). During the ID assignment, a device transmitting in a time slot does not know whether it is the only device transmitting. One way to get around this issue is as follows. The number of time slots allocated for ID assignment is increased to $2 \cdot N$. If the number of devices transmitting at time slot $2 \cdot i$ and the number of devices transmitting at time slot $2 \cdot i + 1$ are both one, the device that transmits at time slot $2 \cdot i$ takes the ID i. A device can decide whether to take the ID i using two additional time slots t_1 and t_2.

1. In time slot t_1, all devices who transmit at time slot $2 \cdot i$ speak while all devices who transmit at time slot $2 \cdot i + 1$ listen.

2. In time slot t_2, all devices who transmit at time slot $2 \cdot i + 1$ and receive a message at time slot t_1 speak while all devices who transmit at time slot $2 \cdot i$ listen.

It is straightforward to see that a device receives a message at time slot t_2 if and only if the number of devices transmitting at time slot $2 \cdot i$ and the number of devices transmitting at time slot $2 \cdot i + 1$ are both one.

Observe that the probability that an ID i is assigned to a device is $\Pr[\mathrm{binom}(n, p = 1/\tilde{n}) = 1]^2$. Therefore, similar to Lemma 6.2, by setting $c' = c^2$, with probability $1 - \min\{n^{-\Omega(1)}, \tilde{n}^{-\Omega(1)}\}$, the number of devices assigned IDs is at least $c' \cdot N$ when $n/1.5 \leq \tilde{n} \leq 1.5n$, and the number of devices assigned IDs is less than $c' \cdot N$ when $\tilde{n} \geq 1.9n$ or $\tilde{n} \leq n/1.9$. Hence dense Census with parameter c' fulfills our goal.

We conclude the following theorem.

Theorem 6.3. *In all four collision detection models, with probability* $1 - \min\{n^{-\Omega(1)}, \tilde{n}^{-\Omega(1)}\}$*,* Test-Network-Size$(\tilde{n})$ *accomplishes the following with energy* $O(\alpha(\tilde{n}))$ *and in time* $O(\log \tilde{n})$*.*

- *If* $n/1.5 \leq \tilde{n} \leq 1.5 \cdot n$*, a leader is elected.*

- *If* $\tilde{n} \geq 1.9 \cdot n$ *or* $\tilde{n} \leq n/1.9$*, no leader is elected.*

The asymptotic time complexity of the algorithm Test-Network-Size$(\tilde{n})$ is the same as the algorithm in [7] which works in Strong-CD and is based on circuit simulation. However, the circuit simulation takes only $O(1)$ energy while Test-Network-Size$(\tilde{n})$ needs $O(\alpha(\tilde{n}))$ energy.

6.2 Exponential Search

As mentioned earlier, the capability of distinguishing noise and silence in the Strong-CD and the Receiver-CD models allows a device to do binary search on a set of candidate estimates of network sizes. Such an observation is formalized in this subsection. Suppose that we have an infinite set $D = \{d_1, d_2, \ldots\}$ of positive integers such that $d_{i+1} \geq \gamma \cdot d_i$, for a large enough constant $\gamma > 1$. We let the index $\hat{i}$ be the one such that $d_{\hat{i}-1} < \log n \leq d_{\hat{i}}$. The goal is to estimate $\hat{i}$ within ± 1 additive error. We prove that in Strong-CD model this can be done with high probability in $O(\log \hat{i})$ time, and hence using at most $O(\log \hat{i})$ energy.

Algorithm. We define the 1-round subroutine Test(i) as follows. Each device transmits a message with probability 2^{-d_i}, and all other devices listen to the channel. All devices that listen to the channel decide $i \geq \hat{i}$ if the channel is silent, and decide $i < \hat{i}$ otherwise. Any device that transmits a message decides $i < \hat{i}$. Based on the subroutine Test(i), the procedure Exponential-Search(D) is defined as follows:

1. Repeatedly run Test(i) for $i = 1, 2, 4, 8, \ldots$ to find the smallest number $i' \in \{1, 2, 4, 8, \ldots\}$ such that Test(i') returns $i' \geq \hat{i}$.

2. Use Test(i) to perform a binary search on $\{1, 2, 3, \ldots, i'\}$ to find an index $\tilde{i}$ that estimates $\hat{i}$.

Theorem 6.4. *In the Strong-CD and the Receiver-CD models, the algorithm* Exponential-Search(D) *finds an index $\tilde{i}$ with $\tilde{i} \in \{\hat{i}-1, \hat{i}, \hat{i}+1\}$ in $O(\log \hat{i})$ time with probability $1 - n^{-\Omega(1)}$.*

Proof. Observe that if the following statements hold, then Exponential-Search(D) finds an index $\tilde{i}$ with $\tilde{i} \in \{\hat{i}-1, \hat{i}, \hat{i}+1\}$ in $O(\log \hat{i})$ time.

- For $i \in \{1, 2, \ldots, \hat{i} - 2\}$, Test($i$) returns $i < \hat{i}$ for all devices.

- For $i \in \{\hat{i}+1, \hat{i}+2, \ldots, 2\hat{i}\}$, Test($i$) returns $i \geq \hat{i}$ for all devices.

We show that the above statements hold with probability $1 - n^{-\Omega(1)}$. For any $i \leq \hat{i} - 2$, the probability that Test(i) returns $i \geq \hat{i}$ is $\Pr[\text{binom}(n, 2^{-d_i}) = 0] = n(1 - 2^{-d_i})^n \leq n(1 - 2^{\frac{\log n}{\gamma}})^n = n \cdot (1 - n^{-1/\gamma})^n \leq n \cdot \exp(-n^{1-1/\gamma})$. For any $i \geq \hat{i} + 1$, the probability that Test(i) returns $i < \hat{i}$ is $\Pr[\text{binom}(n, 2^{-d_i}) > 0] \leq n \cdot 2^{-d_i} \leq n \cdot 2^{-\gamma \log n} = n^{-\gamma+1}$.

Since $\hat{i} - 1 \leq d_{\hat{i}-1} < \log n$, we have $\hat{i} \leq \log n$. By the union bound, the probability that the above statement is false is at most $(\hat{i} - 2) \cdot n \cdot \exp(-n^{1-1/\gamma}) + (\hat{i} - 1) \cdot n^{-\gamma+1} = n^{-\Omega(1)}$. □

6.3 The Algorithm

In this section we present an algorithm for Approximate Counting that matches the lower bounds proved in Section 3.

First of all, for any constant d, it is straightforward to devise an algorithm Trivial-Algorithm(d) such that in constant time, with probability $1 - 1/\text{poly}(n)$, the algorithm either (i) decides $n > d$ or (ii) finds an estimate $\tilde{n}$ of n with $n/2 \leq \tilde{n} \leq 2n$.

Suppose that we are given an infinite set $D = \{d_1, d_2, \ldots\}$ of positive integers such that $d_{i+1} \geq \gamma d_i$ (to make Exponential-Search(D) work), $\sum_{k=d_1}^{\infty} 1/\sqrt{2^k} \leq 1$, and $\sqrt{2^{d_1}} \geq 100$. We will later see that the purpose of the constraint $\sqrt{2^{d_1}} \geq 100$ is to meet the condition of Lemma 6.2 when Test-Network-Size is invoked. As in the previous section, the index $\hat{i}$ is defined as the one such that $d_{\hat{i}-1} < \log n \leq d_{\hat{i}}$. Notice that the elements of the set D play the roles of *checkpoints* in our algorithm. We will see that the set D indicates the iterations to examine whether a leader has been elected. Notice that different choices of D lead to different time-energy tradeoffs.

Estimate-Network-Size(D)

Initial setup:

1. For any $k \geq d_1$, a device is labeled k with probability $1/\sqrt{2^k}$ such that each device is labeled by at most one number. Notice that this implicitly requires $\sum_{k=d_1}^{\infty} 1/\sqrt{2^k} \leq 1$.

2. Run Trivial-Algorithm(2^{d_1}). If an estimate $\tilde{n}$ of n is found, the algorithm is terminated.

3. If the model is Strong-CD, let $\tilde{i}$ be the result of Exponential-Search(D), and set $k_0 = d_{\tilde{i}-2}$. Otherwise, set $k_0 = d_1$.

For $k = k_0, k_0+1, k_0+2, \ldots,$ do the following:

1. The devices with label k collaboratively run Test-Network-Size($\sqrt{2^k}$).
2. If $k = d_i$ for some i, do the following:
   - (a) All leaders with an *odd* label announce their labels, while all other devices listen. If exactly one message $\tilde{k}$ is sent, the algorithm is terminated with all devices agreeing on the same estimate $\tilde{n} = 2^{\tilde{k}}$.
   - (b) All leaders with an *even* label announce their labels, while all other devices listen. If exactly one message $\tilde{k}$ is sent, the algorithm is terminated with all devices agreeing on the same estimate $\tilde{n} = 2^{\tilde{k}}$.

Lemma 6.5. *Let $\hat{k} = \lceil \log n \rceil$. With probability $1 - O(\exp(\Omega(\sqrt{n})))$ the following is true:*

- *For each $k < \hat{k} - 1$, the number of devices labeled k is at least $0.95\sqrt{2n} \ge 1.9\sqrt{2^k}$.*
- *For each $k > \hat{k}$, the number of devices labeled k is at most $\sqrt{2^k}/1.9$.*
- *For at least one of $k \in \{\hat{k}-1, \hat{k}\}$, the number of devices labeled k is within $\sqrt{2^k}/1.5$ and $1.5\sqrt{2^k}$.*

Proof. First of all, with probability $1 - n \cdot \sum_{k=n+1}^{\infty} 1/\sqrt{2^k} = 1 - O(\exp(-\Omega(n)))$, no device has label greater than n. Therefore, in what follows we only consider the labels in the range $\{1, 2, \ldots, n\}$.

- For any $k < \hat{k} - 1$, the expected number of devices labeled k is at least $n \cdot \sqrt{2^{-k}} \ge n \cdot \sqrt{2^{-(\log n - 1)}} = \sqrt{2n}$. Also, $1.9\sqrt{2^k} \le 1.9\sqrt{n/2} = 0.95\sqrt{2n} = (1-0.05)\sqrt{2n}$. Therefore, by Chernoff bound, the probability that the number of devices labeled k is less than $0.95\sqrt{2n}$ is bounded by $\exp(0.05^2\sqrt{2n}/2)$.

- For any $n \ge k > \hat{k}$, the probability that the number of devices labeled k is more than $\sqrt{2^k}/1.9$ is $\Pr[\mathrm{binom}(n, \sqrt{2^{-k}}) > \sqrt{2^k}/1.9] \le \Pr[\mathrm{binom}(n, \sqrt{2^{-(\log n+1)}}) > \sqrt{2^{\log n+1}}/1.9] = \Pr[\mathrm{binom}(n, 1/\sqrt{2n}) > \sqrt{2n}/1.9]$. The expected value of $\mathrm{binom}(n, 1/\sqrt{2n})$ is $\sqrt{n/2}$. By setting $\delta = 0.1/1.9$, we have $\sqrt{2n}/1.9 = (1+\delta)\,\mathrm{E}\left[\mathrm{binom}(n, 1/\sqrt{2n})\right]$. Therefore, by Chernoff bound, the probability that the number of devices labeled k is more than $\sqrt{2^k}/1.9$ is bounded by $\exp(-\delta^2\sqrt{n/2}/3)$.

- We let k' be any one of $\hat{k}-1, \hat{k}$ such that $n/\sqrt{2} \le 2^{k'} \le \sqrt{2}n$. The expected number of devices labeled k' is within $\sqrt{2^{k'}}/\sqrt{2}$ and $\sqrt{2} \cdot \sqrt{2^{k'}}$. Since $\sqrt{2} < 1.5$ and $\sqrt{2^{k'}} = \Theta(\sqrt{n})$, using Chernoff bound we can deduce that with probability $1 - O(\exp(\Omega(\sqrt{n})))$ the number of devices labeled k' is within $\sqrt{2^{k'}}/1.5$ and $1.5\sqrt{2^{k'}}$.

By the union bound, the probability that the statement of the lemma is not met is bounded by $(\log n)\exp(0.05^2\sqrt{2n}/2) + n\exp(-\delta^2\sqrt{n/2}/3) + O(\exp(\Omega(\sqrt{n}))) = O(\exp(\Omega(\sqrt{n})))$. □

Theorem 6.6. *In an execution of* Estimate-Network-Size(D)*, with probability $1 - n^{-\Omega(1)}$, all devices agree on an estimate $\tilde{n}$ of the network size n such that $n/2 \le \tilde{n} \le 2n$ with the following time T and energy E cost:*

- *If the model is Strong-CD or Receiver-CD, then* $T = O(d_{\hat{i}}^2)$, $E = O(\log \hat{i})$.
- *If the model is Sender-CD or No-CD with* $n > 1$, *then* $T = O(d_{\hat{i}}^2)$, $E = O(\hat{i})$.

Proof. Suppose that all subroutines Trivial-Algorithm(2^{d_1}), Exponential-Search(D), and Test-Network-Size$(\sqrt{2^k})$, for all k, do not fail. Then the statement of Lemma 6.5, which is true with probability $1 - O(\exp(\Omega(\sqrt{n})))$, implies that only devices with label $k \in \{\lceil \log n \rceil - 1, \lceil \log n \rceil\}$ can be elected as leader through Test-Network-Size$(\sqrt{2^k})$, and hence the algorithm ends by the iteration $k = d_{\hat{i}}$ with a correct estimate of n.

Since each Test-Network-Size$(\sqrt{2^k})$ takes $O(k)$ time, the total time complexity is $d_{\hat{i}} \cdot O(d_{\hat{i}}) = O(d_{\hat{i}}^2)$.

The energy cost per device of Trivial-Algorithm(2^{d_1}) and Test-Network-Size$(\sqrt{2^k})$ are bounded by a constant. In Sender-CD and No-CD model, the asymptotic energy cost is the number of times we encounter $k = d_i$ for some i, which is $O(\hat{i})$. In Strong-CD and Receiver-CD, the number of times we encounter $k = d_i$ for some i is a constant since we start with $k_0 = d_{\tilde{i}-2}$, where the index $\tilde{i}$ is returned by Exponential-Search(D), and we have $\tilde{i} \in \{\hat{i}-1, \hat{i}, \hat{i}+1\}$. Therefore, the asymptotic energy cost equals the energy cost of Exponential-Search(D), which is $O(\log \hat{i})$.

To complete the proof, we show that under the assumption that the statement of Lemma 6.5 holds, with probability $1 - n^{-\Omega(1)}$, none of the Test-Network-Size$(\sqrt{2^k})$ fails. Similar to the proof of Lemma 6.5, we observe that with probability $1 - n \cdot \sum_{k=n+1}^{\infty} 1/\sqrt{2^k} = 1 - O(\exp(-\Omega(n)))$, no device has label greater than n. Therefore, in what follows we only consider $k \in \{1, 2, \ldots, n\}$.

- *Case 1: the number of devices labeled* k *is at least* $0.95\sqrt{2n} \geq 1.9\sqrt{2^k}$. By Theorem 6.3, the probably that no leader is elected in Test-Network-Size$(\sqrt{2^k})$ is at least $1 - \left(0.95\sqrt{2n}\right)^{-\Omega(1)} = 1 - n^{-\Omega(1)}$.
- *Case 2: the number of devices labeled* k *is at most* $\sqrt{2^k}/1.9$. In this case, Lemma 6.5 guarantees that $k \geq \log n - 1$. By Theorem 6.3, the probably that no leader is elected in Test-Network-Size$(\sqrt{2^k})$ is at least $1 - (\sqrt{2^k}/1.9)^{-\Omega(1)} = 1 - n^{-\Omega(1)}$.
- *Case 3: the number of devices labeled* k *is within* $\sqrt{2^k}/1.5$ *and* $1.5\sqrt{2^k}$. In this case, Lemma 6.5 guarantees that $\sqrt{2^k} = \Theta(\sqrt{n})$. By Theorem 6.3, the probably that a leader is elected in Test-Network-Size$(\sqrt{2^k})$ is at least $1 - \sqrt{2^k}^{-\Omega(1)} = 1 - n^{-\Omega(1)}$.

By the union bound on all labels $1, \ldots, n$, the probability that at least one of Test-Network-Size$(\sqrt{2^k})$ fails is bounded by $O(\exp(-\Omega(n))) + n \cdot n^{-\Omega(1)} = n^{-\Omega(1)}$. □

In addition to solving Approximate Counting, the algorithm Estimate-Network-Size(D) also solves Leader Election. Recall that by the end of the algorithm, a unique device s announces its label while all other devices listen to the channel.

Setting the checkpoints. Similar to the lower bound proofs in the section 3, Theorem 6.6 naturally offers a time-energy tradeoff. We demonstrate how different choices of the checkpoints D give rise to different time and energy cost. The first checkpoint d_1 is always chosen as a large enough constant so as to meet the three conditions: $d_{i+1} \geq \gamma d_i$, $\sum_{k=d_1}^{\infty} 1/\sqrt{2^k} \leq 1$, and $\sqrt{2^{d_1}} \geq 100$. In the subsequent discussion we only focus on defining d_i inductively based on d_{i-1}.

To obtain $O(\log^2 n)$ runtime, we set $d_i = \gamma d_{i-1}$ for some constant γ. With such checkpoints, the energy cost in Sender-CD and No-CD is $\hat{i} = O(\log\log n)$; the energy cost in Strong-CD and Receiver-CD is $\log \hat{i} = O(\log\log\log n)$. Let $0 < \epsilon \leq O(1)$. To obtain $O(\log^{2+\epsilon} n)$ runtime, we set $d_i = d_{i-1}^{1+\epsilon/2}$. With such checkpoints, the energy cost in Sender-CD and No-CD is $\hat{i} = O(\log_{1+\epsilon/2} \log\log n) = O(\epsilon^{-1} \log\log\log n)$; the energy cost in Strong-CD and Receiver-CD is $\log \hat{i} = O(\log(\epsilon^{-1} \log\log\log n))$.

Proof of Theorem 6.1. Observe that setting $d_i = b^{d_{i-1}}$ for any constant $b > 1$ gives a polynomial time algorithm achieving the desired energy complexity. To obtain *sub-polynomial* runtime while maintaining the same asymptotic energy complexity, one may set $d_i = 2^{2^{(\log d_{i-1})^\epsilon}}$, $0 < \epsilon < 1$. This leads to time complexity of the form $O(2^{2^{(\log\log n)^{\epsilon'}}})$, for some $0 < \epsilon' < 1$. □

Final Remarks. As mentioned earlier in this section, a device in a failed execution may run forever and consume unbounded amount of energy. In particular, both the expected runtime and the expected energy are unbounded. In the following we suggest several remedies to ease the problem. The root of the unbounded complexity problem is that, if the algorithm fails to terminate by the checkpoint $d_{\hat{i}}$, very likely the algorithm runs forever. To get around this issue, the algorithm can be modified to run all Test-Network-Size($\sqrt{2^k}$) from $k = k_0$ to $k = d_i$ (instead of from $k = d_{i-1} + 1$ to $k = d_i$) for each checkpoint d_i. Another cause for the algorithm to not terminate by the checkpoint $d_{\hat{i}}$ is that Test-Network-Size($\sqrt{2^k}$) can be accidentally passed for some $k \notin \{\hat{k}-1, \hat{k}\}$. Recall that Test-Network-Size is implemented using dense Census algorithm, so a leader elected in Test-Network-Size($\sqrt{2^k}$) collects $\Theta(\log(\sqrt{2^k})) = \Theta(k)$ number of IDs. The leader can schedule the $\Theta(k)$ devices with these IDs to *sabotage* the next $\Theta(k)$ runs of Test-Network-Size. We leave it as an open problem to formally derive good expected energy and time bounds for Approximate Counting.

7 Conclusion and Open Problems

In this paper we exposed two exponential separations in the energy complexity of Leader Election on various wireless radio network models. The upshot is that randomized algorithms in {Strong-CD, Receiver-CD} are exponentially more efficient than those in {Sender-CD, No-CD}, but deterministic algorithms in {Strong-CD,Sender-CD} are exponentially more efficient than those in {Receiver-CD,No-CD}. This exponential separation also occurs in the closely related problem of Approximate Counting.

There are a few intriguing problems left open by this work. Is $\Theta(\alpha(N))$ the correct complexity of dense Leader Election/Census, and is there an $O(\log\log N)$-energy deterministic Census algorithm? The randomized complexity of Approximate Counting should exhibit a 3-way tradeoff between energy, time, and error probability (that the estimate $\tilde{n}$ is not $\Theta(n)$). Understanding every feasible tradeoff between these three measures is a difficult problem. Consider one particular tradeoff: it there a Strong-CD Approximate Counting algorithm taking $O(\log n)$ time, $O(\log\log n)$ energy, with error probability $1/\mathrm{poly}(n)$?

Acknowledgement. We would like to thank Tomasz Jurdziński for discussing the details of [25, 26, 27, 28] and to Calvin Newport for assisting us with the radio network literature.

References

[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. Efficient emulation of single-hop radio network with collision detection on multi-hop radio network with no collision detection. *Distributed Computing*, 5(2):67–71, 1991.

[3] 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. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

[4] M. Barnes, C. Conway, J. Mathews, and D. K. Arvind. ENS: An energy harvesting wireless sensor network platform. In *Proceedings of the 5th International Conference on Systems and Networks Communications*, pages 83–87, 2010.

[5] M. A. Bender, J. T. Fineman, S. Gilbert, and M. Young. How to scale exponential backoff: Constant throughput, polylog access attempts, and robustness. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 636–654, 2016.

[6] M. A. Bender, J. T. Fineman, M. Movahedi, J. Saia, V. Dani, S. Gilbert, S. Pettie, and M. Young. Resource-competitive algorithms. *SIGACT News*, 46(3):57–71, 2015.

[7] M. A. Bender, T. Kopelowitz, S. Pettie, and M. Young. Contention resolution with log-logstar channel accesses. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 499–508, 2016.

[8] P. Brandes, M. Kardas, M. Klonowski, D. Pajak, and R. Wattenhofer. Approximating the size of a radio network in beeping model. In *Proceedings of the 23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2016.

[9] B. S. Chlebus, D. R. Kowalski, and A. Pelc. Electing a leader in multi-hop radio networks. In *The 16th International Conference On Principles Of Distributed Systems (OPODIS)*, pages 106–120. Springer, 2012.

[10] A. E. F. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology. *Theoretical Computer Science*, 302(1):337–364, 2003.

[11] A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *Proceedings of The 24th International Symposium on Distributed Computing (DISC)*, pages 148–162. Springer, 2010.

[12] A. Czumaj and P. Davies. Brief announcement: Optimal leader election in multi-hop radio networks. In *Proceedings 35th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 47–49, 2016.

[13] A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 492–501, 2003.

[14] S. Daum, S. Gilbert, F. Kuhn, and C. Newport. Leader election in shared spectrum radio networks. In *Proceedings of the 31st ACM symposium on Principles of distributed computing (PODC)*, pages 215–224, 2012.

[15] P. Erdős, A. Rényi, and V. T. Sós. On a problem of graph theory. *Studia Sci. Math. Hung.*, 1:215–235, 1966.

[16] M. Farach-Colton, R. J. Fernandes, and M. A. Mosteiro. Lower bounds for clear transmissions in radio networks. In *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 447–454, 2006.

[17] J. T. Fineman, S. Gilbert, F. Kuhn, and C. C. Newport. Contention resolution on a fading channel. In *Proceedings 35th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 155–164, 2016.

[18] J. T. Fineman, C. Newport, and T. Wang. Contention resolution on multiple channels with collision detection. In *Proceedings 35th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 175–184, 2016.

[19] M. Ghaffari and B. Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (PODC)*, pages 748–766, 2013.

[20] M. Ghaffari, N. A. Lynch, and S. Sastry. Leader election using loneliness detection. *Distributed Computing*, 25(6):427–450, 2012.

[21] M. Ghaffari and C. Newport. Leader Election in Unreliable Radio Networks. In *Proceedings 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 138:1–138:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[22] S. Gilbert, V. King, S. Pettie, E. Porat, J. Saia, and M. Young. (near) optimal resource-competitive broadcast with jamming. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 257–266, 2014.

[23] S. Gilbert and C. Newport. The computational power of beeps. In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, pages 31–46. Springer, 2015.

[24] A. G Greenberg and S. Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *J. ACM*, 32(3):589–596, 1985.

[25] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Efficient algorithms for leader election in radio networks. In *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 51–57, 2002.

[26] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Energy-efficient size approximation of radio networks with no collision detection. In *Proceedings of the 8th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 279–289, 2002.

[27] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Weak communication in radio networks. In *Proceedings of the 8th International European Conference on Parallel Computing (Euro-Par)*, pages 965–972, 2002.

[28] T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Weak communication in single-hop radio networks: adjusting algorithms to industrial standards. *Concurrency and Computation: Practice and Experience*, 15(11–12):1117–1131, 2003.

[29] T. Jurdzinski and G. Stachowiak. Probabilistic algorithms for the wakeup problem in single-hop radio networks. In *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC)*, pages 535–549, 2002.

[30] M. Kardas, M. Klonowski, and D. Pajak. Energy-efficient leader election protocols for single-hop radio networks. In *Proceedings of the 42nd International Conference on Parallel Processing*, pages 399–408, 2013.

[31] G. Katona and E. Szemerédi. On a problem of graph theory. *Studia Scientiarum Mathematicarum Hungarica*, 2:23–28, 1967.

[32] V. King, J. Saia, and M. Young. Conflict on a communication channel. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 277–286, 2011.

[33] D. R. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. *Distributed Computing*, 18(1):43–57, 2005.

[34] D. R. Kowalski and A. Pelc. Leader election in ad hoc radio networks: A keen ear helps. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 521–533, 2009.

[35] E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

[36] M. Kutyłowski and W. Rutkowski. Adversary immune leader election in ad hoc radio networks. In *Proceedings of the 11th European Symposium on Algorithms (ESA)*, pages 397–408. Springer, 2003.

[37] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester, and D. Blaauw. A modular 1 mm^3 die-stacked sensing platform with low power I^2C inter-die communication and multi-modal energy harvesting. *IEEE Journal of Solid-State Circuits*, 48(1):229–243, 2013.

[38] K. Nakano and S. Olariu. Energy-efficient initialization protocols for single-hop radio networks with no collision detection. *IEEE Trans. Parallel Distrib. Syst.*, 11(8):851–863, 2000.

[39] K. Nakano and S. Olariu. Randomized initialization protocols for ad hoc networks. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):749–759, 2000.

[40] C. C. Newport. Radio network lower bounds made easy. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*, pages 258–272, 2014.

[41] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 364–369, 2005.

[42] J. Schneider and R. Wattenhofer. What is the use of collision detection (in wireless networks)? In *Proceedings 24th International Symposium on Distributed Computing (DISC)*, pages 133–147, 2010.

[43] K. M. Sivalingam, M. B. Srivastava, and P. Agrawal. Low power link and access protocols for wireless multimedia networks. In *Proceedings of the 47th IEEE Conference on Vehicular Technology*, volume 3, pages 1331–1335, 1997.

[44] D. E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal on Computing*, 15(2):468–477, 1986.

A Upper Bound on the Recursive Function $T(\hat{N})$

In this section we show the omitted proof for $T(N) = O(N \cdot f(N))$ in Section 4.4, where $f(N) = (\log\log N)^{\log\log\log\log N+2}$.

Recall the recursive formula of $T(\hat{N})$:

$$T(\hat{N}) \leq (\sqrt{\hat{N}} + \log\log N)T(\sqrt{\hat{N}}) + c(\sqrt{\hat{N}} + \log\log N)),\ T(0) = O(1).$$

Here $c \geq 1$ is a constant. When $\hat{N} \leq \log^2\log N$, we have $T(\hat{N}) \leq 2\log\log N \cdot T(\sqrt{\hat{N}}) + 2c\log\log N$, which implies that $T(\hat{N}) \leq O((2\log\log N)^{\log\log\hat{N}})$.

Furthermore, when $\log^2\log N < \hat{N} \leq \log^4\log N$, we have

$$T(\hat{N}) \leq 2\sqrt{\hat{N}}T(\log^2\log N) + 2\sqrt{\hat{N}} \leq c(\hat{N} - 2\sqrt{\hat{N}})T(\log^2\log N).$$

Then we move on to $\hat{N} \geq \log^4\log N$, where it holds $T(\hat{N}) \leq (\sqrt{\hat{N}} + \sqrt[4]{\hat{N}})T(\sqrt{\hat{N}}) + 2c\sqrt{\hat{N}}$. We prove by induction that in this case, $T(\hat{N}) \leq c(\hat{N} - 2\sqrt{\hat{N}})T(\log^2\log N)$ still holds, since if the inequality is true for $T(\hat{N})$, it implies that

$$\begin{aligned} T(\hat{N}^2) &\leq (\hat{N} + \sqrt{\hat{N}})T(\hat{N}) + 2c\hat{N} \\ &\leq c(\hat{N} + \sqrt{\hat{N}})(\hat{N} - 2\sqrt{\hat{N}})T(\log^2\log N) + 2c\hat{N} \\ &\leq c(\hat{N}^2 - 2\hat{N})T(\log^2\log N) - c\hat{N}(\sqrt{\hat{N}} - 2). \end{aligned}$$

Combining the two parts together, it turns out that when $\hat{N} \geq \log^4\log N$,

$$\begin{aligned} T(\hat{N}) &\leq c(\hat{N} - 2\sqrt{\hat{N}})T(\log^2\log N) \\ &\leq c(\hat{N} - 2\sqrt{\hat{N}})O((2\log\log N)^{\log\log(\log^2\log N)}). \end{aligned}$$

Thus we conclude that $T(N) = O(N \cdot f(N))$.