Distributed MIS in $O(\log\log n)$ Awake Complexity

Fabien Dufoulon
fabien.dufoulon.cs@gmail.com
University of Houston
Houston, USA

William K. Moses Jr.*
wkmjr3@gmail.com
Durham University
Durham, UK

Gopal Pandurangan
gopalpandurangan@gmail.com
University of Houston
Houston, USA

ABSTRACT

Maximal Independent Set (MIS) is one of the fundamental and most well-studied problems in distributed graph algorithms. Even after four decades of intensive research, the best known (randomized) MIS algorithms have $O(\log n)$ round complexity on general graphs [Luby, STOC 1986] (where n is the number of nodes), while the best known lower bound is $\Omega(\sqrt{\log n/\log\log n})$ [Kuhn, Moscibroda, Wattenhofer, JACM 2016]. Breaking past the $O(\log n)$ round complexity upper bound or showing stronger lower bounds have been longstanding open problems.

Energy is a premium resource in various settings such as battery-powered wireless networks and sensor networks. The bulk of the energy is used by nodes when they are *awake*, i.e., when they are sending, receiving, and even just listening for messages. On the other hand, when a node is *sleeping*, it does not perform any communication and thus spends very little energy. Several recent works have addressed the problem of designing *energy-efficient* distributed algorithms for various fundamental problems. These algorithms operate by minimizing the number of rounds in which *any* node is *awake*, also called the (worst-case) *awake complexity*. An intriguing open question is whether one can design a distributed MIS algorithm that has significantly smaller awake complexity compared to existing algorithms. In particular, the question of obtaining a distributed MIS algorithm with $o(\log n)$ awake complexity was left open in [Chatterjee, Gmyr, Pandurangan, PODC 2020].

Our main contribution is to show that MIS can be computed in awake complexity that is *exponentially* better compared to the best known round complexity of $O(\log n)$ and also bypassing its fundamental $\Omega(\sqrt{\log n/\log\log n})$ round complexity lower bound exponentially. Specifically, we show that MIS can be computed by a randomized distributed (Monte Carlo) algorithm in $O(\log\log n)$ awake complexity with high probability.[1] However, this algorithm has a round complexity that is $O(\text{poly}(n))$. We then show how to drastically improve the round complexity at the cost of a slight increase in awake complexity by presenting a randomized distributed (Monte Carlo) algorithm for MIS that, with high probability computes an MIS in $O((\log\log n)\log^* n)$ awake complexity and $O((\log^3 n)(\log\log n)\log^* n)$ round complexity. Our algorithms work in the CONGEST model where messages of size $O(\log n)$ bits can be sent per edge per round.

*Part of this work was done while the author was a Post Doctoral Fellow at the University of Houston.
[1]Throughout, we use "with high probability (w.h.p.)" to mean with probability at least $1 - n^{-1}$.

PODC '23, June 19–23, 2023, Orlando, FL, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0121-4/23/06…$15.00
https://doi.org/10.1145/3583668.3594574

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; • **Mathematics of computing → Probabilistic algorithms**; *Discrete mathematics.*

KEYWORDS

Maximal Independent Set, Sleeping model, energy-efficient, awake complexity, round complexity, trade-offs

ACM Reference Format:
Fabien Dufoulon, William K. Moses Jr., and Gopal Pandurangan. 2023. Distributed MIS in $O(\log\log n)$ Awake Complexity. In *ACM Symposium on Principles of Distributed Computing (PODC '23), June 19–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583668.3594574

1 INTRODUCTION

1.1 Maximal Independent Set Problem

Computing the *maximal independent set* (*MIS*) is one of the fundamental and most well-studied problems in distributed graph algorithms. Given a graph with n nodes, each node must (irrevocably) commit to being in a subset $M \subseteq V$ (called the MIS) or not such that (i) every node is either in M or has a neighbor in M and (ii) no two nodes in M are adjacent to each other.

Because of the importance of MIS, distributed algorithms for MIS have been studied extensively for the last four decades mainly with a focus on improving the time complexity (i.e., the number of rounds). In 1986, Alon, Babai, and Itai [1] and Luby [33] presented a randomized distributed MIS algorithm that takes $O(\log n)$ rounds (n is the number of nodes in the graph) with high probability. Since these seminal results, there has been a lot of significant progress in recent years in designing progressively faster distributed MIS algorithms. For n-node graphs with maximum degree Δ, Ghaffari [20] presented a randomized MIS algorithm running in $O(\log\Delta) + 2^{O(\sqrt{\log\log n})}$ rounds, improving over the algorithm of Barenboim, Elkin, Pettie and Schneider [6] that runs in $O(\log^2\Delta) + 2^{O(\sqrt{\log\log n})}$ rounds. We note that these two results assume the LOCAL model. The run time was further improved by Rozhon and Ghaffari [39, Corollary 3.2] to $O(\log\Delta + \text{poly}(\log\log n))$ rounds, which is currently the best known bound for randomized algorithms in the LOCAL model. The currently best known randomized algorithm in the CONGEST model takes $O(\log\Delta\log\log n + \text{poly}(\log\log n))$ rounds [21]. Thus, the currently known best algorithms of MIS ([20, 21, 39]) are dependent on Δ (the maximum degree), and hence still take $O(\log n)$ rounds (even in the LOCAL model) for graphs with $O(\text{poly}(n))$ degree. As far as deterministic algorithms are concerned, the best known algorithms take $O(\text{poly}(\log n))$ rounds in the LOCAL as well as CONGEST models [21, 39].

There are faster distributed algorithms known for special classes of graphs such as trees [20, 32] and Erdos-Renyi random graphs [20, 30], but they still take $\Omega(\sqrt{\log n/\log\log n})$ rounds. There are also MIS algorithms that run faster on graphs with low arboricity, but they nevertheless take $O(\log n)$ rounds on high arboricity graphs [5, 20].

While the above results make significant progress in the round complexity of the MIS problem for some specific graphs, however, in general graphs, the best known running time is still $O(\log n)$ (even for randomized algorithms and even in the LOCAL model). Furthermore, there is a fundamental lower bound of $\Omega\left(\min\left\{\frac{\log\Delta}{\log\log\Delta}, \sqrt{\frac{\log n}{\log\log n}}\right\}\right)$ rounds due to Kuhn, Moscibroda, and Wattenhofer [31] that also applies to randomized algorithms and holds even in the LOCAL model. Thus, for example, say, when $\Delta = 2^{\Omega(\sqrt{\log n \log\log n})}$, it follows that one cannot hope for algorithms faster than $\sqrt{\log n/\log\log n}$ rounds. Balliu, Brandt, Hirvonen, Olivetti, Rabie, and Suomela [3] showed that one cannot hope for algorithms that run within $o(\Delta) + O(\log^* n)$ rounds for the regimes where $\Delta \ll \log\log n$ (for randomized algorithms) [3, Corollary 5] and $\Delta \ll \log n$ (for deterministic algorithms) [3, Corollary 6]. (See also results on an improved lower bound [4].)

1.2 Awake Complexity

Energy is a premium resource in various settings such as battery-powered wireless networks and sensor networks. The bulk of the energy is used by the nodes (devices) when they are *"awake"*, i.e., when they are sending, receiving, and even just listening for messages. It is well-known that the energy used by a node when it is idle and just listening (waiting to hear from a neighbor) is only slightly smaller than the energy used in a transmitting or receiving state [18, 42]. On the other hand, the energy used in the "sleeping" state, i.e., when a node has switched off its communication devices and is not sending, receiving or listening, is *significantly less* than in the transmitting/receiving/idle (listening) state [18, 28, 40–42]. A node may choose to enter and exit this sleeping mode in a judicious way to save energy during the course of an algorithm.[2]

There has been a lot of recent theoretical interest in designing energy-efficient distributed algorithms for various fundamental problems such as maximal independent set, maximal matching, coloring, broadcasting, spanning tree construction, breadth-first tree construction, etc. (see e.g., [2, 7, 10–13, 15, 22]). These algorithms operate by minimizing the number of rounds in which any node is *awake*, also called the *awake complexity*. An intriguing question is whether one can design distributed algorithms for various problems that have significantly smaller awake complexity compared to existing algorithms. However, this is challenging, since a node can only communicate with a neighboring node that is awake (note that a sleeping node does not send or receive messages and also messages sent to it are lost). As a result, coordinating (or scheduling) such communication in an efficient manner (without keeping any node awake for a long time) becomes non-trivial.

[2]This has been exploited by protocols to save power in ad hoc wireless networks by switching between two states — *sleeping* and *awake* — as needed (the MAC layer provides support for switching between these states [36, 41, 42]).

1.3 Model and Complexity Measures

Distributed Computing Model. We are given an anonymous distributed network of n nodes, modeled as an undirected graph $G = (V, E)$. Each node hosts a processor with limited initial knowledge. We assume that each node has ports (each port having a unique port number); each incident edge is connected to one distinct port. We assume that each node knows a common value N, a polynomial upper bound on n.

Nodes are allowed to communicate through the edges of the graph G and it is assumed that communication is *synchronous* and occurs in rounds. In particular, we assume that each node knows the current round number (starting from round 0). In each round, each node can perform some local computation (which finishes in the same round) including accessing a private source of randomness, and can exchange messages of size $O(\log n)$ bits with each of its neighboring nodes.

This standard model of distributed computation is called the CONGEST model [38]. We note that our algorithms also, obviously, apply to the LOCAL model, another standard model [38] where there is no restriction on the size of the messages sent per edge per round. Though the CONGEST and LOCAL models do not put any constraint on the computational power of the nodes, our algorithms perform only light-weight computations (each node processes only poly$(\log n)$ bits per round and takes computation time essentially linear in the number of bits processed).

Sleeping Model. We assume the sleeping model [13], where a node can be in either of the two states — sleeping or awake. (At the beginning, we assume that all nodes are awake.) This is a simple generalization of the standard distributed computing model, where nodes are always assumed to be awake. In the sleeping model, each node decides to be either *awake* or *asleep* in each round (till it terminates), corresponding to whether the node can receive/send messages and perform computations in that round or not, respectively. That is, any node v can decide to *sleep* starting at any (specified) round of its choice. We assume that all nodes know the correct round number whenever they are awake. A node can *wake up* again later at any *specified* round and enter the *awake* state. We note that the model allows a node to cycle through the process of sleeping in some round and waking up at a later round as many times as it wants. To summarize, distributed computation in the sleeping model proceeds in *synchronous* rounds and each round consists of the following steps: (1) Each awake node can perform local computation. (2) Each awake node can send a message to its adjacent nodes. (3) Each awake node can receive messages sent to it in this round (in the previous step) by other awake nodes.

In the sleeping model, let A_v denote the number of awake rounds for a node v before it terminates (i.e., finishes the execution of the algorithm, locally). We define the *(worst-case) awake complexity* as $\max_{v\in V} A_v$. For a randomized algorithm, A_v will be a random variable and our goal is to obtain high probability bounds on the awake complexity. While the main goal is to reduce awake complexity, we also strive to minimize the *round complexity*, where both, sleeping and awake rounds, are counted.

Several recent works (see Section 1.5) have designed distributed algorithms in the sleeping model for fundamental problems such as MIS, approximate matching and vertex cover, spanning tree, minimum spanning tree, coloring, and other problems [2, 7, 13, 22].

1.4 Our Contributions

In light of the difficulty in breaking the $o(\log n)$ round barrier of MIS and the lower bound of $\Omega(\sqrt{\log n/\log\log n})$ rounds in the standard model (that applies even for LOCAL algorithms), as well as motivated by resource considerations discussed above, a fundamental question that we address in this paper is:

Can we design a distributed MIS algorithm with $o(\log n)$ awake complexity?

We answer the above question in the affirmative and actually show a much stronger bound. Our main contribution is that we show that MIS can be computed in (worst-case) awake complexity of $O(\log\log n)$ rounds, bypassing the $\Omega(\sqrt{\log n/\log\log n})$ lower bound on the round complexity in an exponentially better fashion. Specifically, we present the following results.

(1) We first present a randomized distributed (Monte Carlo) algorithm for MIS that with high probability computes an MIS and has $O(\log\log n)$ awake complexity.
This algorithm has *round complexity* that is polynomial in n. Our bounds hold even in the CONGEST model where messages of $O(\log n)$ bits can be sent per edge per round.
(2) We then show that we can drastically reduce the round complexity at the cost of a slight increase in awake complexity by presenting a randomized MIS algorithm with $O((\log\log n)\log^* n)$ awake complexity and $O((\log^3 n)(\log\log n)\log^* n)$ round complexity in the CONGEST model.

Our work answers a key question left open in [13], namely whether one can design MIS algorithms with (even) $o(\log n)$ (worst-case) awake complexity. We note that prior results [13, 22] presented algorithms in the sleeping model with $O(\log n)$ awake complexity (see Section 1.5). Our results show that we can compute an MIS in an awake complexity that is *exponentially* better compared to the best known round complexity of $O(\log n)$. Since a node spends a significant amount of energy only in its awake rounds, our algorithms are highly energy-efficient compared to the existing algorithms.

1.5 Related Work and Comparison

Chatterjee, Gmyr, and Pandurangan [13] posit the sleeping model and showed that MIS in general graphs can be solved in $O(1)$ *node-averaged* awake complexity. Node-averaged awake complexity is measured by the *average* number of rounds a node is awake. They also defined *worst-case* awake complexity (used in this paper) which is the worst-case number of rounds a node is awake until it finishes the algorithm. The worst-case awake complexity of their MIS algorithm is $O(\log n)$, while the worst-case complexity (that includes all rounds, sleeping and awake) is $O(\log^{3.41} n)$ rounds. An important question left open in [13] is whether one can design an MIS algorithm with $o(\log n)$ worst-case awake complexity (even in the LOCAL model).[3]

Subsequently Ghaffari and Portmann [22] developed a randomized MIS algorithm that has worst-case awake complexity of $O(\log n)$, round complexity of $O(\log n)$, while having $O(1)$ node-averaged awake complexity (all bounds hold with high probability). They also present algorithms for $(1+\varepsilon)$ approximation of maximum matching and $(2+\varepsilon)$ approximation of minimum vertex cover with the same bounds on round complexity and node-averaged awake complexity. Hourani, Pandurangan, and Robinson [25] presented randomized MIS algorithms that have $O(\text{poly}(\log\log n))$ awake complexity for certain special classes of *random graphs*, including random geometric graphs (of arbitrary dimension). These algorithms work only in the LOCAL model as linear (in n) sized messages need to be sent per edge per round. This result is subsumed by the results of the current paper.

Barenboim and Maimon [7] showed that many problems, including broadcast, construction of a spanning tree, and leader election can be solved deterministically in $O(\log n)$ awake complexity in the sleeping model. They construct a spanning tree called Distributed Layered Tree (DLT) in $O(\log n)$ awake complexity deterministically. In this tree, broadcast and convergecast can be accomplished in $O(1)$ awake rounds. They also showed that fundamental symmetry breaking problems such as MIS and $(\Delta+1)$-coloring can be solved deterministically in $O(\log\Delta+\log^* n)$ awake rounds in the LOCAL model, where Δ is the maximum degree. (Note that, in general, this can take $O(\log n)$ awake rounds when $\Delta=\Omega(\text{poly}\, n)$.) Their algorithm only works in the LOCAL model (as opposed to the CONGEST model), as it needs large-sized (polynomial number of bits) messages to be sent over an edge. They also define the class of *O-LOCAL* problems (that includes MIS and coloring), where such a problem is one that can be solved sequentially according to some acyclic orientation of the edges of the input graph where the decision of a node depends on the decisions of the nodes in the subtree rooted at it. They showed that problems in this class admit a deterministic algorithm that runs in $O(\log\Delta)$ awake time and $O(\Delta^2)$ round complexity. Maimon [34] presents trade-offs between awake and round complexity for O-LOCAL problems.

Augustine, Moses Jr., and Pandurangan [2] give an $O(\log n)$ awake complexity algorithm for the minimum spanning tree (MST) problem (in the CONGEST model). They use a spanning tree construction called the Labelled Distance Tree (LDT) which we also use in our algorithm.

There is a lot of other work on energy-efficient algorithms over the years which is too vast to survey here, and we restrict ourselves to those that are most relevant. A relevant recent line of work is that of Chang, Kopelowitz, Pettie, Wang, and Zhan [12] on radio networks (see also the references therein and its follow up papers [10, 11, 15, 16] and also much earlier work on energy efficient algorithms in radio networks e.g., [26, 27, 37]). This work defines the measure of *energy complexity* which is the same as (worst-case)

[3]In this paper, we do not focus on the node-averaged awake complexity measure, and only focus on the (worst-case) awake complexity. However, it is fairly straightforward to augment the algorithms of this paper so that they also give an $O(1)$-node averaged complexity in addition to their (worst-case) awake complexity guarantees by using the approach of [13].

awake complexity (i.e., both measures count only the rounds that a node is awake). While the awake complexity used here and several other papers [2, 7, 13, 22] assumes the usual CONGEST (or LOCAL) communication model (and hence the model can be called *SLEEPING-CONGEST* (or *SLEEPING-LOCAL*)), the energy complexity measure used in [12] (and also papers mentioned above) has some additional communication restrictions that pertain to radio networks (and can be called *SLEEPING-RADIO* model). One important restriction in the radio model is that nodes can only broadcast messages (hence the same message is sent to all neighbors). Also, collisions can occur at a listening node if two neighboring nodes transmit simultaneously in the same round. There are a few variants depending on how collisions are handled. Energy-efficient algorithms for several problems such as broadcast, leader election, breadth-first search, and maximal matching have been studied in the radio network model [10–12, 15, 16]. An interesting open problem is whether our sub-logarithmic bounds on MIS awake complexity can be obtained in the SLEEPING-RADIO model.

King, Phillips, Saia, and Young [28] also study a similar model where nodes can be in two states: sleeping or awake (listening and/or sending). They present an energy-efficient algorithm in this model to solve a reliable broadcast problem. We also refer to the literature on resource competitive algorithms where there is limited energy available both to the algorithm and the (jamming) adversary (e.g., [8, 23, 24]).

2 HIGH-LEVEL OVERVIEW AND TECHNIQUES

The best known distributed MIS algorithms ([20, 21, 39]) in the traditional setting suffer from a $\log \Delta$ dependency in the round complexity, where Δ is the maximum degree (see Section 1). Prior to this work, that was also the case in the sleeping model as well.[4] Rather than improve these algorithms to remove this dependency (which appears very difficult), we improve a different algorithm: the well-known *randomized greedy MIS* algorithm [9, 14, 19], a variant of Luby's algorithm [33].

The (sequential) randomized greedy MIS algorithm considers nodes (of some graph $G = (V, E)$) in random order and adds them to the output set unless one of their neighbors is already in it. If $v_1, \ldots, v_n$ is the random node ordering considered by the algorithm, then it is well-known that the output is the lexicographically first MIS (LFMIS) [14] with respect to that (random) ordering.[5] Fischer and Noever [19] showed that the randomized greedy MIS can be implemented in the (traditional) distributed computing model in $O(\log n)$ rounds with high probability and also that this bound is tight. On the other hand, we show that randomized greedy MIS can be implemented in $O(\log \log n)$ awake complexity. We build to this result in three steps.

Algorithm VT-MIS. First, we give a simple awake-efficient variant (Algorithm VT-MIS in Subsection 4.3) of the naive distributed implementation of the above (sequential) algorithm. Let I be an upper bound on the randomly chosen IDs. Then, the naive distributed implementation works as follows. In each round $i \in [1, I]$, all nodes transmit whether they have joined the MIS or not to their neighbors. After which, the node with ID i (if it exists) enters the MIS if none of its neighbors already have. Clearly, the naive implementation has an excessive $O(I)$ awake complexity. However, we can reduce the awake complexity exponentially, that is, to $O(\log I)$.

To reduce the awake complexity, we use a *virtual binary tree structure* (see Subsection 4.1), similar to that of [7], to carefully coordinate the communication between the nodes. More precisely, a (virtual) tree with t leaves (where the same tree is locally determined by each node using the parameter $t = 2^{\lceil \log I \rceil}$) tells each node in which round to be awake and communicate to its neighbors whether it is in the MIS or not. This virtual tree ensures that for any two neighboring nodes v, v' with $id_v < id_{v'}$, v and v' are both awake in some round i that satisfies $id_v < i \leqslant id_{v'}$. As a result, a node needs to be awake in only $O(\log I)$ well-chosen rounds (where $\log t = O(\log I)$ is the depth of the virtual tree) to correctly implement randomized greedy MIS. This virtual tree coordination framework is reused in our third algorithm, Awake-MIS, and we believe it can be useful in general for designing awake-efficient algorithms.

Algorithm LDT-MIS. Second, we give a more awake-efficient variant (Algorithm LDT-MIS in Subsection 4.3) with an improved dependency on the ID upper bound I. This improved dependency comes into play when I is super-polynomial (or even worse) in the number of nodes n. Having good awake complexity in this particular scenario, which happens in Awake-MIS, is crucial for our $O(\log \log n)$ awake complexity main result.

To obtain an improved dependency on I, we use a spanning tree structure, called a *labeled distance tree* (LDT), introduced in [2] (an improvement on a similar structure, introduced previously in [7]). Crucially, these trees can be used to broadcast and rank nodes (i.e, assign IDs in $[1, n]$) in $O(1)$ awake complexity, while the LDT itself can be constructed in $O(\log n)$ awake complexity deterministically [2]. Hence, one can first build a LDT and rank the nodes in $O(\log n)$ awake complexity. Since nodes are ranked arbitrarily, we can then have the root broadcast a uniformly random permutation of $[1, n]$ in $O((n \log n)/\log I)$ consecutive broadcasts (recall that messages can be of size $O(\log I) = O(\log n)$ bits, and thus can be sent in CONGEST). Consequently, nodes obtain new IDs in $[1, n]$ that correspond to a uniformly random ordering. It remains only to run Algorithm VT-MIS, which now takes $O(\log n)$ awake complexity (due to the smaller IDs).

Algorithm Awake-MIS. For our main result, we use the previously described techniques as well as the following two key properties of the *randomized greedy MIS algorithm*. The first is the *composability property*. One can run the randomized greedy MIS algorithm on the first $k > 0$ nodes, then run that algorithm again but on the remaining nodes, that is, those which are not neighbors of the first computed MIS. The union of the two computed MIS's is the output of the randomized greedy MIS algorithm on G. The second is the *residual sparsity property* — stated formally in Lemma 2 in Subsection 3.3 — which shows that after processing the first k nodes in the random ordering, the degree of the residual graph (i.e., the subgraph induced by the rest of the nodes minus the neighbors

[4] In particular, the algorithms of [13, 22] which had optimal $O(1)$ rounds *node-averaged* awake complexity, however, had $O(\log n)$ (worst-case) awake complexity.

[5] Given two (MIS) subsets $X \neq Y$ of V, such that $X \not\subseteq Y$ and $Y \not\subseteq X$, X is lexicographically smaller (with respect to that ordering) than Y if and only if the smallest differing element between X and Y is in X.

of MIS nodes among the first k nodes) is reduced (essentially) to $O(n/k)$ with high probability.

In Algorithm Awake-MIS (described in Section 5), nodes are partitioned into $O(\log^2 n)$ batches. More precisely, each node picks a pair $(i, j) \in [1, \ell] \times [1, 2\Delta']$ with some well-chosen probabilities, where $\ell = O(\log n)$ and $\Delta' = O(\log n)$ are two parameters. To compute the MIS, we consider batches sequentially in phases (according to the lexicographical ordering). During the first "communication" round of each phase, nodes inform their neighbors whether they are already in the MIS or not. Moreover, just as in VT-MIS, nodes use a virtual binary tree structure to coordinate in which of these rounds to be awake or to sleep. (Hence, any given node is awake for at most $O(\log\log n)$ communication rounds.) For the remaining rounds of some phase (i, j), nodes of batch (i, j) with no neighbors already in the MIS run Algorithm LDT-MIS to compute an MIS over the batch's nodes. Crucially, we show that the subgraph induced by the nodes running Algorithm LDT-MIS is *shattered*: that is, it consists only of $O(\log n)$-sized components with high probability. Hence, nodes run LDT-MIS using $O(\log\log n)$ awake complexity only. (Here, it is important that the second term of LDT-MIS, caused by the CONGEST bandwidth, adds up to $O(\log\log n)$.) From this, it is easy to see that Algorithm Awake-MIS has $O(\log\log n)$ awake complexity.

However, how do we ensure that the subgraph induced by the nodes running Algorithm LDT-MIS is shattered? First, we adjust the probabilities that nodes choose a given (batch) pair to ensure that the first $2\Delta'$ batches contain with high probability half as many nodes as the next $2\Delta'$, and so on. Hence, by the residual sparsity property, the subgraph induced by the nodes (with no MIS neighbors) within any of these collections of $2\Delta'$ batches has small $O(\log n)$ degree. (In fact, we must first use the composability property to show that the MIS computed throughout all previous phases is the output of randomized greedy MIS on the nodes in all previous batches.) Furthermore, nodes within any such collection independently and uniformly chose any of the $2\Delta'$ batches. Hence, for each node, the expected number of neighbors (themselves with no MIS neighbors) is less than 1/2. In this case, a simple branching process argument — stated formally in Lemma 3 in Subsection 3.4 — shows that the subgraph induced by a given batch's nodes (with no MIS neighbors) is shattered.

3 PRELIMINARIES: NOTATION AND RANDOMIZED TECHNIQUES

3.1 Notation

For any subset $V' \subseteq V$, let $G[V']$ denote the subgraph of G induced by V'. For any node v, let $N(v)$ denote the union of v and the set of its neighbors. Similarly, for any set of vertices $V' \subseteq V$, let $N(V')$ denote the union of V' and the set of neighbors of any node in V'. For any two integers i and j, $[i, j]$ is used to denote the set $\{i, \ldots, j\}$.

3.2 Chernoff Bounds

The following Chernoff bounds [35] are used in the later sections, where the second bound is obtained by applying the inequality $\ln(1+\delta) \geqslant (2\delta)/(2+\delta)$ to Theorem 4.4 (Inequality 1) in [35].

Lemma 1. *Let $X_1, \ldots, X_k$ be independent Bernoulli random variables with parameter p. Then,*

- *For any $0 \leqslant \delta \leqslant 1$, $\Pr[\sum_{i=1}^k X_i \leqslant (1-\delta)pk] \leqslant e^{-\frac{\delta^2 kp}{2}}$,*
- *For any $\delta \geqslant 0$, $\Pr[\sum_{i=1}^k X_i \geqslant (1+\delta)pk] \leqslant e^{-\frac{\delta^2 kp}{2+\delta}}$.*

3.3 Sequential Randomized Greedy MIS

The *sequential randomized greedy MIS* algorithm processes each node in a sequential but random order. Each node is added to the output set if it is not a neighbor of a node already in that set. It is well-known that this algorithm outputs the lexicographically first MIS (LFMIS), with respect to the random node ordering.

Given a random node ordering $v_1, v_2, \ldots, v_n$, Lemma 2 — a slight generalization of Lemma 1 in [29] — shows that after the first t nodes (according to the node ordering) are processed by the sequential randomized greedy order MIS, the maximum degree of the subgraph induced by the remaining nodes among the first $t' > t$ nodes (those which have not been added, nor are neighbors of an already added node) has decreased (almost) linearly in t. In fact, the lemma is more general. For example, it applies to distributed algorithms that compute the LFMIS over the subgraph induced by the first t nodes, according to the random node ordering mentioned above.

Lemma 2. *Let t, t' be two integers such that $1 \leqslant t < t' \leqslant n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any constant $\varepsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t}\ln(n/\varepsilon)$ with probability at least $1-\varepsilon$.*

Proof. Note that $(V_{t'} \setminus N(M_t)) \subseteq \{v_{t+1}, \ldots, v_{t'}\}$. We show that, with probability at least $1-\varepsilon/n$, for any $j \in [t+1, t']$, either v_j has degree at most $\frac{t'}{t}\ln(n/\varepsilon)$ in $G[V_{t'} \setminus N(M_t)]$ or $v_j \in N(M_t)$. The lemma statement holds by a union bound (over j).

Let $j \in [t+1, t']$. We apply the principle of deferred decisions [35]. More precisely, we first fix (the random choice of) which node is in position j — that is, v_j. After which, we fix (the random choices of) which nodes are in position 1 to t sequentially — that is, v_1 to v_t. For any integer $i \in [1, t]$, let V_i denote the first i (fixed) nodes and M_i be the LFMIS over $G[V_i]$. Additionally, let $U_i = (N(v_j) \cap V_{t'}) \setminus N(M_{i-1})$ and $d_i = |U_i|$. Then, $\Pr[v_i \in U_i \mid v_j, v_1, \ldots, v_{i-1}] \geqslant \frac{d_i}{t'-1-(i-1)} \geqslant \frac{d_i}{t'}$.

The sequence $(d_i)_{i\in[1,t]}$ is decreasing. If $d_t \leqslant \frac{t'}{t}\ln(n/\varepsilon)$, then v_j has degree at most $\frac{t'}{t}\ln(n/\varepsilon)$ in $G[V_{t'} \setminus N(M_t)]$. Otherwise, $d_t > \frac{t'}{t}\ln(n/\varepsilon)$. Then, $\Pr[\forall i \leqslant t, v_i \notin U_i \mid v_j] \leqslant \prod_{i=1}^t \Pr[v_i \notin U_i \mid v_j, v_1, \ldots, v_{i-1}] \leqslant \prod_{i=1}^t (1-\frac{d_i}{t'}) \leqslant (1-\frac{d_t}{t'})^t \leqslant e^{-\frac{d_t}{t'}t} \leqslant e^{-\ln(n/\varepsilon)} \leqslant \varepsilon/n$. In other words, there exists $i \in [1, t]$ such that $v_i \in U_i$ with probability at least $1-\varepsilon/n$. In which case, since M_i is the LFMIS over $G[V_i]$, $v_i \in M_i$. Thus, $v_i \in M_t$ and $v_j \in N(M_t)$ (with probability at least $1-\varepsilon/n$). □

3.4 Simple Graph Shattering

Consider some n-node graph H with maximum degree Δ and partition the nodes into 2Δ sets uniformly at random. More precisely, each node is in set U_j, for any $j \in [1, 2\Delta]$, with probability $1/(2\Delta)$. Then, a simple branching process argument implies that the corresponding induced subgraphs $H[U_j]$ are "shattered": that is, they are composed of small $O(\log n)$-sized connected components with high probability.

Lemma 3. *For any $j \in [1, 2\Delta]$, the connected components of $H[U_j]$ are of size at most $6 \ln(n/\varepsilon)$ with probability at least $1 - \varepsilon$.*

Proof. For any $j \in [1, 2\Delta]$, consider some node $v \in U_j$. (If $|U_j| = 0$, the lemma statement obviously holds.) We assume, by the principle of deferred decisions, that v is the only initially revealed node of U_j (and for all other nodes, we do not know whether they are in U_j or not) and we find out which nodes are in U_j through a BFS search (over $H[U_j]$ only) starting at v. In more detail, the BFS queue initially consists only of v. In the first step, v is dequeued and for each unrevealed neighbor $w \in N(v)$, we reveal whether w is in U_j. For each such $w \in N(v)$, if $w \in U_j$ then w is added to the queue. Once all neighbors have been revealed, the first step is done. Subsequent steps are executed similarly, but with that step's dequeued node, until the queue is empty. Importantly, the number of steps executed before the queue is empty is the size $C(v)$ of the connected component of $H[U_j]$ containing v. Moreover, it is a random variable that depends on each revealed node — that is, whether that revealed node is in U_j or not. Finally, each unrevealed node w, once revealed, is in U_j with probability at most $\frac{1}{2\Delta}$.

The above (BFS search) randomized process is hard to analyze since a node dequeued in some step k might have neighbors that were revealed in previous steps. Instead, we consider an easier-to-analyze but related randomized process: BFS on a branching process. Let the number of nodes in the queue at the start of step $k \geqslant 0$ be denoted by A_t. Initially, there is a single node in the queue — that is, $A_0 = 1$. Subsequently, for any step $k \geqslant 1$, $A_k = A_{k-1} - 1 + Y_k$, where the random variables $(Y_k)_{k \geqslant 1}$ are independent and binomially distributed with parameters Δ, the total number of events, and $\frac{1}{2\Delta}$, the probability of each event being successful. Then, the *size* of this randomized process is $C' = \min\{k \geqslant 0 \mid A_t = 0\}$. Importantly, C' dominates the random variable $C(v)$ — that is, $\Pr[C(v) \geqslant k] \leqslant \Pr[C' \geqslant k]$ for any positive integer k. By the definition of C', $\Pr[C' > k] = \Pr[A_1 > 0, \ldots, A_k > 0] \leqslant \Pr[A_k > 0] \leqslant \Pr[A_0 + \sum_{i=1}^{k} Y_i > k] = \Pr[\sum_{i=1}^{k} Y_i \geqslant k]$. Note that since the Y_i are independent binomially distributed random variables (with parameters Δ and $\frac{1}{2\Delta}$), $\sum_{i=1}^{k} Y_i$ is the sum of $k\Delta$ Bernoulli random variables with parameter $\frac{1}{2\Delta}$ and thus $\mathbb{E}[\sum_{i=1}^{k} Y_i] = k/2$. Hence, by the Chernoff bound (see Lemma 1), $\Pr[\sum_{i=1}^{k} Y_i \geqslant k)] \leqslant e^{-k/6}$. Consequently, $\Pr[C' \geqslant 6 \ln(n/\varepsilon)] \leqslant \varepsilon/n$ for any constant $\varepsilon > 0$. Since C' dominates $C(v)$, $\Pr[C(v) \geqslant 6 \ln(n/\varepsilon)] \leqslant \varepsilon/n$. By union bound over all connected components of $H[U_j]$ (of which there are at most n), the lemma statement holds. □

4 AUXILIARY PROCEDURES

4.1 The Virtual Binary Tree Technique

We provide a virtual binary tree construction similar to that in [7]. Let i be an integer, provided as a parameter. The virtual (full) binary tree $\mathcal{B}([1, i])$ has depth $d = \lceil \log i \rceil$ and thus $y = 2^{d+1} - 1$ nodes. These nodes are labeled with integers in $[1, y]$ according to an in-order tree traversal. See Figure 1 (left). Given $\mathcal{B}([1, i])$, we can define the more convenient node-labeled (full) binary tree $\mathcal{B}^*([1, i])$ as follows. The tree structure is the same, but the node labels of $\mathcal{B}^*([1, i])$ are obtained by applying $g(x) = \lfloor x/2 \rfloor + 1$ to the node labels of $\mathcal{B}([1, i])$. Using $\mathcal{B}^*([1, i])$, we define, for any integer $k \in [1, i]$, a *communication set* $S_k([1, i])$ of d integers in $[1, i]$, as follows: $S_k([1, i])$ consists of the labels of all ancestors of the leaf node labeled k in $\mathcal{B}^*([1, i])$. See Figure 1 (right).

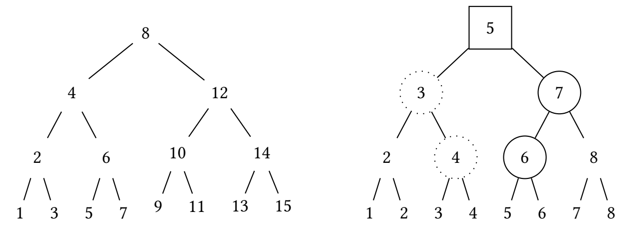


Figure 1: Binary tree $\mathcal{B}([1, 6])$ on the left and binary tree $\mathcal{B}^*([1, 6])$ on the right. $S_3([1, 6])$ consists of the dotted circle and rectangle nodes' labels and $S_5([1, 6])$ of the (non-dotted) circle and rectangle nodes' labels. Note that $5 \in S_3([1, 6]) \cap S_5([1, 6])$ and $3 < 5 \leqslant 6$.

These sets have the following property (see Observation 2 below): for any integers $k, k' \in [1, i]$ such that $k < k'$, there exists an integer r in both $S_k([1, i])$ and $S_{k'}([1, i])$ such that $k < r \leqslant k'$. Informally, we later use the sets $S_k([1, i])$ to decide when nodes with IDs in $[1, i]$ are awake or asleep. The above property allows us to decide, for any two nodes, on a common round in which they are guaranteed to be awake simultaneously (and thus communicate with each other). Remember that in rounds in which nodes are not awake simultaneously, any message sent between two neighboring nodes is *lost if any one of them is asleep.*

Observation 1. *For any positive integers k, i such that $1 \leqslant k \leqslant i$, $|S_k([1, i])| \leqslant \lceil \log i \rceil$.*

Observation 2. *For any positive integers k, k', i such that $1 \leqslant k < k' \leqslant i$, there exists an integer $r \in S_k([1, i]) \cap S_{k'}([1, i])$ such that $k < r \leqslant k'$.*

Proof. Let the lowest common ancestor node in $\mathcal{B}^*([1, i])$ of the leaves labeled k and k' be labeled with r'. Then, according to the definition of a communication set, r' is in both $S_k([1, i])$ and $S_{k'}([1, i])$. Moreover, note that the corresponding nodes in $\mathcal{B}([1, i])$ are the internal (lowest common ancestor) node labeled $2(r' - 1)$ and the leaf nodes labeled $2k - 1$ and $2k' - 1$. By the property of the in-order tree traversal, $2k - 1 < 2(r' - 1) < 2k' - 1$. Hence, $k < r' - 1/2 < k'$. Since r' and k' are integers, $k < r' \leqslant k'$. □

4.2 Labeled Distance Trees

For any $V' \subseteq V$ where $G[V']$ is connected, a *labeled distance tree* (introduced in [2]) is an oriented node labeled spanning tree over $G[V']$ rooted at some node $r \in V'$. Each node's label is its distance to r in the spanning tree. (Note that the distance to r in the spanning tree may be much larger than that in G.) In the distributed implementation, the LDT satisfies the following properties: (i) all nodes in the tree know the ID of r, called the ID of the LDT, (ii) each node knows its depth in the tree (i.e., the hop-distance from itself to the root of the tree via tree edges), and (iii) each node knows the

IDs of its parent and children, if any, in the LDT. If a given graph is disconnected, one can compute a disjoint set of such LDTs, one per connected component, which we refer to as a *forest of labeled distance trees (FLDT).*

Multiple distributed LDT construction algorithms (both randomized and deterministic) are presented in [2]. We are interested in their two deterministic construction algorithms, which give different guarantees. The first lemma below captures the worst-case awake and round complexities of their first construction algorithm: Algorithm LDT-Construct-Awake. On the other hand, the second lemma captures the properties of a distributed LDT construction algorithm (Algorithm LDT-Construct-Round) with faster round complexity but larger awake complexity; Corollary 1 of [2] states such an algorithm exists, and we give a full construction procedure in the full version of this paper [17].

Lemma 4 (Theorem 2, [2]). *For any connected $V' \subseteq V$ of at most n' nodes with unique IDs in $[1, I]$, where n' and I are known to all nodes, LDT-Construct-Awake deterministically constructs an LDT over $G[V']$ with $O(\log n')$ awake complexity, $O(n' I \log n')$ round complexity and $O(\log I)$ bit messages.*

Lemma 5. *For any connected $V' \subseteq V$ of at most n' nodes, where n' is known to all nodes, with unique IDs in $[1, I]$, LDT-Construct-Round deterministically constructs an LDT over $G[V']$ with $O((\log n') \log^* I)$ awake complexity, $O(n' (\log n') \log^* I)$ round complexity and $O(\log I)$ bit messages.*

Next, we define two operations over a labeled distance tree.

Definition 1. *For any connected $V' \subseteq V$ and an LDT spanning $V' \subseteq V$:*

- *The* broadcast *operation consists of sending the root's (input) message, denoted by m_r, to all of the LDT's nodes.*
- *The* ranking *operation consists of having the nodes of V' compute a total ordering — more precisely, each node knows its rank in the ordering — as well as the size $|V'|$.*

[2] provides a distributed algorithm for broadcasting over an LDT (see Observation 2 in [2]). A distributed algorithm for ranking over an LDT is presented in the full version of this paper [17]. The properties of these two algorithms are captured by the following lemma.

Lemma 6. *For any LDT over at most n' nodes, where n' is known to all nodes, with unique IDs in $[1, I]$, broadcast and ranking can be executed deterministically with $O(1)$ awake complexity, $O(n')$ round complexity and the size of the messages are $O(|m_r|)$ bits and $O(\log I)$ bits respectively.*

4.3 Simple Awake-Efficient MIS Algorithms

We provide two simple deterministic distributed lexicographically first MIS (LFMIS) algorithms with good awake complexity. Note that it is important for our main result (see Section 5) that these algorithms compute the LFMIS rather than any arbitrary MIS.

The first algorithm (VT-MIS) runs in $O(\log I)$ awake time, where I is an upper bound on the nodes' IDs, and illustrates how to use the virtual binary tree technique (see Section 4.1). At a high-level, VT-MIS is an awake-efficient version of the naive distributed implementation of sequential greedy MIS. Recall that the naive algorithm runs in I rounds, and in the ith round, the node with ID i (if it exists) communicates with all neighbors to check if any neighbor with smaller ID is already in the MIS. If not, it joins the MIS. Although VT-MIS also has poor $O(I)$ round complexity, its $O(\log I)$ awake complexity is exponentially smaller.

The second algorithm (LDT-MIS) runs in $O(\log n')$ awake time, where n' is the number of nodes. Of particular interest to us is when $\log n'$ may be much smaller than the length of the node's IDs. This naturally happens if LDT-MIS is run on a subgraph obtained after shattering a much larger graph (e.g., with exponentially more nodes n) whose nodes had unique IDs. At a high-level, we compute labeled distance trees, then the root computes n' and assigns uniformly random and small enough $O(\log n')$-length IDs to each node, all in $O(\log n')$ awake time. All nodes then compute the LFMIS using VT-MIS and these new IDs in $O(\log n')$ awake time.

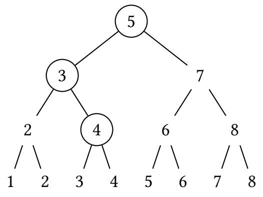
VT-MIS: MIS in $O(\log I)$ awake time. We assume nodes are given unique IDs in $[1, I]$, for some known value I. Initially, each node starts in the undecided state and computes the virtual binary tree $\mathcal{B}^*([1, I])$ locally (for the description, see Subsection 4.1). Then, nodes compute the LFMIS in I rounds. In round r, the node that has ID r as well as all nodes u for which $r \in S_{id_u}([1, I])$ wake up — where $S_{id_u}([1, I])$ is the *communication set* defined using $\mathcal{B}^*([1, I])$, see Figure 2 for an example. Intuitively, the additional nodes wake up to communicate information on their state (i.e., undecided, in MIS or not in MIS), and the use of communication sets results in low awake time. All awake nodes send their state to all neighbors. Any awake undecided node that learns one of its neighbors is in the MIS sets its state to "not in MIS". If the node with ID r remains undecided despite the received messages, then it joins the MIS and sets its state to "in MIS".

Lemma 7. *VT-MIS computes the LFMIS with $O(\log I)$ awake complexity, $O(I)$ round complexity and $O(\log I)$ bit messages.*

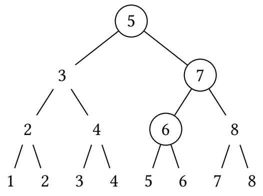
Proof. We first prove the correctness. The main difference with the naive distributed implementation of sequential greedy MIS is that here, in any round $r \in [1, I]$, not all nodes may be awake and thus the node v with ID r may be unaware that one of its neighbors is already in the MIS. However, by Observation 2, there exists for any node u with ID $r' < r$, a round r^* such that $r' < r^* \leqslant r$ and both u and v are awake in r^*. Since u can only decide to join the MIS in round r' by the algorithm's definition, u successfully communicates whether it joins the MIS or not to v in round r^*. It follows that VT-MIS implements sequential greedy MIS, and thus correctly computes an LFMIS.

As for the awake complexity, note that by Observation 1, each communication set of $\mathcal{B}^*([1, I])$ is of size $O(\log I)$. Since each node u is only awake in rounds $r \in S_{id_u}([1, I])$, each node is awake for $O(\log I)$ rounds. □

LDT-MIS: MIS in $O(\log n')$ awake time. We assume that each node knows the value n', an upper bound on the number of nodes of the connected subgraph to which the node belongs. We also assume that the (at most) n' nodes are given unique IDs in $[1, I]$, for some known value I (where I may be exponentially larger than n'), and that messages can contain up to $O(\log I)$ bits. First, all nodes participate in the construction of an LDT. (The correctness is guaranteed by the fact that all nodes know the same bound n'.)



(a) $S_3([1,6])$ **consists of the circle nodes' labels.**



(b) $S_5([1,6])$ **consists of the circle nodes' labels.**

Figure 2: For the example, consider $I = 6$ and two nodes u, v with IDs 3 and 5 respectively. Node u is awake in rounds 3, 4 and 5 and v is awake in rounds 5 and 6 (but not in round 7, since there are only I rounds). It can be seen that u communicates whether it has joined the MIS to v in round 5.

Second, all nodes participate in an operation to (i) compute the exact number of nodes n'' in the LDT, and (ii) allow each node to know its rank in a given total ordering of the nodes. Third, the root of the LDT locally computes a uniformly random permutation of $[1, n'']$ and broadcasts it in $O((n' \log n')/\log I)$ consecutive broadcasts over the LDT. (In each broadcast, we can transmit $O(\log I)$ bits of the total $O(n'' \log n'')$ bits. Once all of these bits have been transmitted, "null" messages are sent for the remaining broadcasts, if any.) Each node uses its previously computed rank to retrieve its ID in the permutation. Finally, all nodes run VT-MIS using these smaller IDs.

Lemma 8. *LDT-MIS computes an LFMIS with respect to some uniformly random node ordering, and not to the ID-based ordering, with $O(\log n' + (n' \log n')/\log I)$ awake complexity, $O(n' I \log n' + ((n')^2 \log n')/\log I)$ round complexity and $O(\log I)$ bit messages.*

Proof. First, note that LDT-MIS implements (sequential) randomized greedy MIS. Indeed, constructing the LDT, computing n' and sending down smaller IDs is simply equivalent to computing a uniformly random node ordering on the n' nodes. After which, VT-MIS implements sequential greedy MIS with respect to that order by Lemma 7.

Now we prove the rest of the lemma statement. Constructing the LDT takes $O(\log n')$ awake complexity, $O(n' I \log n')$ round complexity and $O(\log I)$ bit messages (by Lemma 4). The ranking operation takes $O(1)$ awake complexity, $O(n')$ round complexity, and $O(\log I)$ bit messages (by Lemma 6). The $O((n' \log n')/\log I)$ consecutive broadcasts used to transmit the permutation chosen by the root (at most $O(n' \log n')$ bits), via messages of $O(\log I)$ bits, take altogether $O((n' \log n')/\log I)$ awake complexity and $O(((n')^2 \log n')/\log I)$ round complexity (by Lemma 6). Finally, computing the LFMIS (via VT-MIS) with respect to the new node ordering (from the new IDs in $[1, n']$) takes $O(\log n')$ awake complexity, $O(n')$ round complexity, and $O(\log n') = O(\log I)$ bit messages. □

By replacing the awake efficient LDT construction (Algorithm LDT-Construct-Awake) with the round efficient one (Algorithm LDT-Construct-Round, whose properties are described in Lemma 5), we obtain a round efficient version of LDT-MIS, which we call LDT-MIS-ROUND. Its properties — a faster round complexity but larger awake complexity than LDT-MIS — are formally stated in the following corollary.

Corollary 1. *LDT-MIS-ROUND computes an LFMIS with respect to some uniformly random node ordering with $O((\log n') \log^* I + (n' \log n')/\log I)$ awake complexity, $O((n' \log n') \log^* I + ((n')^2 \log n')/\log I)$ round complexity and $O(\log I)$ bit messages.*

5 RANDOMIZED GREEDY MIS IN $O(\log \log n)$ AWAKE ROUNDS

We present our main result: an $O(\log \log n)$ awake complexity randomized MIS algorithm. We first give a high-level description. Algorithm Awake-MIS computes the lexicographically first MIS, with respect to some uniformly random ordering, in "batches". More precisely, the LFMIS is computed over the first t nodes, then over the next t' nodes, and so on. To (energy efficiently) ensure all batches know which nodes (of prior batches) are in the MIS, we coordinate communication using the virtual binary tree technique with only $O(\log \log n)$ awake complexity. Moreover, by batching nodes we can leverage the following "graph shattering" property: the subgraph induced by the yet undecided nodes within each batch can be decomposed into small $O(\log n)$-sized connected components. Finally, for each such component, it suffices to run LDT-MIS from Section 4 to compute the LFMIS with respect to a uniformly random ordering (of the component's nodes) in $O(\log \log n)$ awake time.

Description of Awake-MIS. Let $\ell = \lceil \log n - \log \log n \rceil$ and $\Delta' = O(\log n)$ be some parameters decided in the analysis. The output variable *state* takes value in $\{undecided, inMIS, notinMIS\}$. The MIS problem is said to be solved if all nodes have chosen a state in $\{inMIS, notinMIS\}$ and the set of all nodes with the *inMIS* output forms an MIS.

Initially, each node starts in the "undecided" state. Moreover, each node $v \in V$ picks a pair $p(v) = (i, j) \in [1, \ell] \times [1, 2\Delta']$ at random (but not uniformly) which decides what batch the node falls in—that batch is denoted by $B_{i,j}$. Batches are ordered via lexicographical order (of the pairs). Next, we describe precisely how nodes choose a batch. Each node chooses $i \in [1, \ell - 1]$ with probability $(10 \cdot 2^i \log n)/n$ and $i = \ell$ with the remaining probability. Moreover, each node chooses $j \in [1, 2\Delta']$ uniformly at random, that is, with probability $1/(2\Delta')$.

After the batches have been decided, there are $2\ell\Delta' = O(\log^2 n)$ phases. (The number of rounds per phase is determined in the analysis, allowing some nodes to sleep through the phase.) In each phase $(i, j) \in [1, \ell] \times [1, 2\Delta']$, the first *communication round* is used to update which of the batch's nodes have a neighbor in the MIS. (Let $g : [1, \ell] \times [1, 2\Delta'] \mapsto [1, 2\ell\Delta']$ be the natural bijection that preserves lexicographic order.) In more detail, node $v \in V$ is awake if $g(i, j) \in S_{g(p(v))}([1, 2\ell\Delta'])$, and is asleep otherwise—see

Subsection 4.1 for the formal definition of the communication set $S_{g(p(v))}([1, 2\ell\Delta'])$ and Subsection 4.3 for an example. Awake nodes send their state to their neighbors. At the end of the round, awake nodes that received a *inMIS* message from a neighboring node set their state to *notInMIS* (thus becoming decided). After which, the remaining rounds are used by all undecided batch nodes (i.e., with $p(v) = (i, j)$ and in the "undecided" state) to execute LDT-MIS described in Subsection 4.3. (In the analysis, we show that w.h.p., the remaining rounds are sufficient for this algorithm to terminate.)

Algorithm 1 AWAKE-MIS for node v

```
1:  Input: $n$
2:  $v$ chooses $i \in [1,\ell]$ with probability $(10 \cdot 2^i \log n)/n$ and $j \in [1, 2\Delta']$ with
    probability $1/(2\Delta')$
3:  $p(v) := (i, j)$, $state_v := undecided$
4:  for phase $p = 1$ to $2\ell\Delta'$ do
    // For the first (communication) round:
5:      if $p \in S_{g(p(v))}([1, 2\ell\Delta'])$ then        // Nodes with $g(p(v)) \neq p$ can also
    participate
6:          if $state_v = undecided$ then
7:              Listen for one round
8:              if $v$ receives an "in MIS" message then    $state_v := notinMIS$
9:          else    Send $state_v$ to all neighbors
10:     else    Sleep for one round
    // For the next $O(\text{poly}(n)(\log n)\log\log n)$ rounds:
11:     if $p \neq g(p(v))$ or $state_v \neq undecided$ then    Sleep for the remainder of
    the phase
12:     else    $state_v$ = LDT-MIS()
```

Analysis of AWAKE-MIS. Next, we show correctness and complexity bounds of AWAKE-MIS. Note that we assume nodes know n exactly (or at least some constant factor approximation) in the above description and below analysis for clarity. This assumption can be removed through straightforward changes, so that nodes can use instead a polynomial upper bound N on n.

Theorem 1. *MIS can be solved (w.h.p.) in $O(\log\log n)$ awake complexity and $O(\text{poly}(n))$ round complexity in CONGEST.*

PROOF. Let us start with some notations. For any $(i, j) \in [1,\ell] \times [1, 2\Delta']$, denote by $V_{i,j}$ the union of all batches up to, and including, batch (i, j), and for any $i \in [1,\ell]$, let $V_i = V_{i,2\Delta'}$. Then, we bound the size of V_i for any $i \in [1,\ell]$. Recall that each node is in V_i independently and with probability $\sum_{k=1}^{i}(10 \cdot 2^k \log n)/n = (10(2^{i+1} - 1)\log n)/n$. Hence, by linearity of expectation, $\mathbb{E}[|V_i|] = 10(2^{i+1} - 1)\log n$. After which, we apply the Chernoff bound (for the two tails, see Lemma 1) with $\delta = 1/2$: $\Pr[|V_i| \geqslant 15(2^{i+1} - 1)\log n] \leqslant \exp(-(10(2^{i+1} - 1)\log n)/10)$ and $\Pr[|V_i| \leqslant 5(2^{i+1} - 1)\log n] \leqslant \exp(-(10(2^{i+1} - 1)\log n)/8)$. Thus, by union bound, $5(2^{i+1} - 1)\log n \leqslant |V_i| \leqslant 15(2^{i+1} - 1)\log n$ holds with probability at least $1 - 1/n^3$.

Second, we show by induction that by the end of phase $(i, j) \in [1,\ell] \times [1, 2\Delta']$, the algorithm has computed the LFMIS over $G[V_{i,j}]$ with respect to a uniformly random ordering of $V_{i,j}$ and with probability at least $1 - 1/n$. Consider the base case. During the first phase, different connected components $C_1, \ldots, C_k$ of $G[V_{1,1}]$ run independent LDT-MIS executions. Given the upper bound on V_1 shown above, these components are of size $O(\log n)$ with probability at least $1 - 1/n^3$. By setting the number of rounds within the first phase accordingly — to $O(\text{poly}(n)(\log n)\log\log n)$ rounds — all LDT-MIS executions terminate. Thus, by Lemma 8, the output of LDT-MIS for each component C_h is an LFMIS with respect to a uniformly random node ordering (of C_h), which we denote M_h. Note that $M = \bigcup_{h=1}^{k} M_h$ is exactly the set of nodes with state *inMIS* when the first phase ends. Clearly, M is an MIS since two nodes from different connected components are non-adjacent. It remains to show that M is also a LFMIS with respect to a uniformly random ordering of $V_{1,1}$. Consider the following node ordering. First, pick some node w uniformly at random among all $|V_{1,1}|$ nodes and then choose the first node of the ordering uniformly at random among all nodes of the component w belongs to. Next, pick some node w' uniformly at random among the remaining $|V_{1,1}| - 1$ nodes and then choose the second node of the ordering uniformly at random among all nodes in the component w' belongs to (excluding the first node in the ordering). Repeat this process until all nodes in $V_{1,1}$ have been ordered. It is straightforward to show that this node ordering is a uniformly random ordering of $V_{1,1}$. Moreover, two nodes u, v from different components are not connected in $G[V_{1,1}]$ and thus whether u is ordered before v (or inversely) does not influence the resulting LFMIS over $G[V_{1,1}]$. Hence, M is the LFMIS over $G[V_{1,1}]$ with respect to that uniformly random ordering of $V_{1,1}$, concluding the base case.

Now, consider some phase $(i, j) \neq (1, 1)$ and assume that the induction hypothesis holds for the (lexicographically) previous phase (i', j'). By the induction hypothesis, the algorithm has computed the LFMIS M' over $G[V_{i',j'}]$ with respect to some uniformly random order of $V_{i',j'}$ by the end of phase (i', j'). Since $V_{i',j'} = V_{i,j} \setminus B_{i,j}$, the induction step follows if we show that the algorithm computes a LFMIS over $G[B^*_{i,j}]$ with respect to a uniformly random ordering of $B^*_{i,j}$, where $B^*_{i,j}$ are the nodes of $B_{i,j}$ with no neighbors in M'. To do so, we first note that nodes know whether they are in $B^*_{i,j}$ or not by the end of the communication round of phase (i, j). This can be shown using properties of the communication sets (as in the proof of Lemma 7). In the remainder of the phase, different connected components $C_1, \ldots, C_{k'}$ of $G[B^*_{i,j}]$ run independent LDT-MIS executions. If $i = 1$, then just as in the base case, these connected components have size $O(\log n)$ with probability at least $1 - 1/n^3$, and thus $O(\text{poly}(n)(\log n)\log\log n)$ rounds are sufficient for the LDT-MIS executions to terminate with probability at least $1 - 1/n^3$. Otherwise, if $i > 1$, let M_{i-1} be the MIS computed by the end of phase $(i - 1, 2\Delta')$. Due to the induction hypothesis, we can apply Lemma 2 and thus $G[V_i \setminus N(M_{i-1})]$ has maximum degree upper bounded by $\frac{|V_i|}{|V_{i-1}|}\ln(n^4)$ with probability at least $1 - 1/n^3$. Using the above bounds on V_i and V_{i-1}, this is at most $9\ln(n^4)$. By choosing $\Delta' = 9\ln(n^4)$ (and using the principle of deferred decisions), Lemma 3 implies that $G[B_{i_j} \setminus N(M_{i-1})]$ consists of small $O(\log n)$-sized connected components with probability at least $1 - 1/n^3$. Given that $B^*_{i,j} \subseteq B_{i_j} \setminus N(M_{i-1})$, $G[B^*_{i_j}]$ also consists of small $O(\log n)$-sized connected components with probability at least $1 - 1/n^3$. And thus, $O(\text{poly}(n)(\log n)\log\log n)$ rounds are sufficient for the LDT-MIS executions to terminate with probability at least $1 - 1/n^3$. Next, we point out that one can show, using an argument similar to that of the base case, that the union M'' of the computed MIS—which is exactly the set of nodes whose state becomes *inMIS* during phase (i, j)—is a LFMIS with respect to a uniformly random order of $B^*_{i,j}$. Moreover, it is straightforward to extend this ordering of $B^*_{i,j}$ to a uniformly random order of $V_{i,j}$, as long as all nodes in $B_{i,j}$ are ordered after those in $V_{i',j'}$. (In which case, no matter how some node $z \in B_{i,j} \setminus B^*_{i,j}$ is ordered, z is not in the LFMIS.) Consequently, when phase (i, j) ends, the set $M' \cup M''$ is a LFMIS over $G[V_{i,j}]$ with respect to some uniformly random order of $V_{i,j}$ with probability at least $1 - 1/n^2$ and the induction step follows. (Note that the error probability of all induction steps added together is at most $1/n$.)

To conclude, we note that all communication is done through $O(\log n)$ bit messages, and in particular the communication within the LDT-MIS calls. Next, we upper bound the awake complexity. Each node $v \in V$ is awake for at most $O(\log \log n)$ communication rounds over all $O(\log^2 n)$ phases. Moreover, within v's chosen phase $p(v)$, recall that the subgraph induced by the awake nodes is composed of $O(\log n)$-sized connected components with high probability. Then, by Lemma 8, v is awake for at most $O(\log \log n + ((\log n) \log \log n)/\log n) = O(\log \log n)$ rounds (w.h.p.) during the LDT-MIS execution. Finally, a simple modification of LDT-MIS limiting the number of rounds a node can be awake to $O(\log \log n)$ (where the precise value can be obtained from the analysis and the desired error probability) implies any failure affects correctness rather than the awake complexity. Therefore, the awake complexity of Awake-MIS is (deterministically) upper bounded by $O(\log \log n)$. As for the round complexity, the above analysis implies that each phase takes $O(\text{poly}(n)(\log n) \log \log n)$ rounds. Hence, the algorithm's round complexity is $O(\text{poly}(n))$. □

By using the more round efficient LDT-MIS-ROUND (see Corollary 1) instead of LDT-MIS, an MIS can be computed with a significantly better round complexity, but at the cost of a small $O(\log^* n)$ overhead to the awake complexity.

Corollary 2. *MIS can be solved (w.h.p.) in $O((\log \log n) \log^* n)$ awake complexity and $O((\log^3 n)(\log \log n) \log^* n)$ round complexity in CONGEST.*

Proof. When using LDT-MIS-ROUND instead of LDT-MIS, phases of $O((\log n)(\log \log n) \log^* n)$ rounds suffice (see Corollary 1). □

6 CONCLUSION

In this paper, we show that the fundamental MIS problem on general graphs can be solved in $O(\log \log n)$ awake complexity, i.e., the worst-case number of awake (non-sleeping) rounds taken by all nodes is $O(\log \log n)$. This is the first such result that we are aware of where we can obtain even a $o(\log n)$ bound on the awake complexity for MIS. A long-standing open question is whether a similar bound (i.e., $o(\log n)$) can be shown for the round complexity.

Several open problems arise from our work. An important one is determining whether one can improve the awake complexity bound of $O(\log \log n)$, or showing that is optimal by showing a lower bound. Another one is whether one can obtain an $O(\log \log n)$ awake complexity MIS algorithm that has $O(\log n)$ round complexity. More generally, can one obtain good trade-offs between awake and round complexity of MIS?

Finally, it would be useful to design algorithms for other symmetry breaking problems such as maximal matching, coloring, etc., that have better awake complexity compared to the traditional round complexity.

ACKNOWLEDGMENTS

F. Dufoulon was supported in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419. W. K. Moses Jr. was supported in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419. G. Pandurangan was supported in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419.

REFERENCES

[1] Noga Alon, László Babai, and Alon Itai. 1986. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms* 7, 4 (1986), 567–583.
[2] John Augustine, William K. Moses Jr., and Gopal Pandurangan. 2022. Brief Announcement: Distributed MST Computation in the Sleeping Model: Awake-Optimal Algorithms and Lower Bounds. *Proceedings of the 41st ACM Symposium on Principles of Distributed Computing (PODC)* (2022), 51–53. Full version available on arXiv: https://arxiv.org/abs/2204.08385.
[3] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. 2019. Lower Bounds for Maximal Matchings and Maximal Independent Sets. In *IEEE FOCS*. 481–497.
[4] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. 2021. Improved Distributed Lower Bounds for MIS and Bounded (Out-)Degree Dominating Sets in Trees. In *PODC '21: ACM Symposium on Principles of Distributed Computing*. ACM, 283–293.
[5] Leonid Barenboim and Michael Elkin. 2010. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Comput.* 22, 5-6 (2010), 363–379.
[6] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. 2016. The Locality of Distributed Symmetry Breaking. *Journal of the ACM* 63, 3, Article 20 (June 2016), 45 pages. Conference version: *IEEE FOCS* 2012.
[7] Leonid Barenboim and Tzalik Maimon. 2021. Deterministic Logarithmic Completeness in the Distributed Sleeping Model. In *35th International Symposium on Distributed Computing, DISC*, Vol. 209. 10:1–10:19.
[8] Michael A. Bender, Jeremy T. Fineman, Mahnush Movahedi, Jared Saia, Varsha Dani, Seth Gilbert, Seth Pettie, and Maxwell Young. 2015. Resource-Competitive Algorithms. *SIGACT News* 46, 3 (2015), 57–71.
[9] Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. 2012. Greedy Sequential Maximal Independent Set and Matching Are Parallel on Average. In *ACM Symposiumon Parallelism in Algorithms and Architectures (SPAA)*. 308–317.
[10] Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, Qizheng He, Wenzheng Li, and Seth Pettie. 2018. The Energy Complexity of Broadcast. In *ACM PODC*. 95–104.
[11] Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, and Seth Pettie. 2020. The Energy Complexity of BFS in Radio Networks. In *ACM PODC*. 273–282.
[12] Yi-Jun Chang, Tsvi Kopelowitz, Seth Pettie, Ruosong Wang, and Wei Zhan. 2019. Exponential Separations in the Energy Complexity of Leader Election. *ACM Trans. Algorithms* 15, 4 (2019), 49:1–49:31. Conference version: *ACM STOC* 2017..
[13] Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan. 2020. Sleeping is Efficient: MIS in $O(1)$-rounds Node-averaged Awake Complexity. In *ACM Symposium on Principles of Distributed Computing, PODC*. 99–108.
[14] Don Coppersmith, Prabhakar Raghavan, and Martin Tompa. 1989. Parallel graph algorithms that are efficient on average. *Information and Computation* 81, 3 (1989), 318–333.
[15] Varsha Dani, Aayush Gupta, Thomas P. Hayes, and Seth Pettie. 2021. Wake up and Join Me! an Energy-Efficient Algorithm for Maximal Matching in Radio Networks. In *35th International Symposium on Distributed Computing (DISC)*. 19:1–19:14.
[16] Varsha Dani and Thomas P. Hayes. 2022. How to Wake up Your Neighbors: Safe and Nearly Optimal Generic Energy Conservation in Radio Networks. In *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA (LIPIcs, Vol. 246)*, Christian Scheideler (Ed.). 16:1–16:22.
[17] Fabien Dufoulon, William K Moses Jr., and Gopal Pandurangan. 2022. Sleeping is Superefficient: MIS in Exponentially Better Awake Complexity. *arXiv preprint arXiv:2204.08359* (2022).
[18] Laura Marie Feeney and Martin Nilsson. 2001. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, Vol. 3. 1548–1557.
[19] Manuela Fischer and Andreas Noever. 2018. Tight Analysis of Parallel Randomized Greedy MIS. In *SODA*. 2152–2160.
[20] Mohsen Ghaffari. 2016. An Improved Distributed Algorithm for Maximal Independent Set. In *SODA*. 270–277.
[21] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhon. 2021. Improved Deterministic Network Decomposition. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*. 2904–2923.

[22] Mohsen Ghaffari and Julian Portmann. 2022. Average Awake Complexity of MIS and Matching. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 45–55.

[23] Seth Gilbert, Valerie King, Seth Pettie, Ely Porat, Jared Saia, and Maxwell Young. 2014. (Near) optimal resource-competitive broadcast with jamming. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014*, Guy E. Blelloch and Peter Sanders (Eds.). ACM, 257–266.

[24] Seth Gilbert and Maxwell Young. 2012. Making evildoers pay: resource-competitive broadcast in sensor networks. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, Darek Kowalski and Alessandro Panconesi (Eds.). ACM, 145–154.

[25] Khalid Hourani, Gopal Pandurangan, and Peter Robinson. 2022. Awake-Efficient Distributed Algorithms for Maximal Independent Set. In *IEEE Conference on Distributed Computing Systems (ICDCS)*. 1338–1339.

[26] Tomasz Jurdzinski, Miroslaw Kutylowski, and Jan Zatopianski. 2002. Efficient algorithms for leader election in radio networks. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002*, Aleta Ricciardi (Ed.). ACM, 51–57.

[27] Marcin Kardas, Marek Klonowski, and Dominik Pajak. 2013. Energy-Efficient Leader Election Protocols for Single-Hop Radio Networks. In *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013*. IEEE Computer Society, 399–408.

[28] Valerie King, Cynthia A. Phillips, Jared Saia, and Maxwell Young. 2011. Sleeping on the Job: Energy-Efficient and Robust Broadcast for Radio Networks. *Algorithmica* 61, 3 (2011), 518–554.

[29] Christian Konrad. 2018. MIS in the Congested Clique Model in $O(\log\log\Delta)$ Rounds. *arXiv preprint arXiv:1802.07647* (2018).

[30] K. Krzywdziński and K. Rybarczyk. 2015. Distributed algorithms for random graphs. *Theoretical Computer Science* 605 (2015), 95–105.

[31] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. 2016. Local Computation: Lower and Upper Bounds. *Journal of the ACM* 63, 2, Article 17 (2016), 44 pages.

[32] Christoph Lenzen and Roger Wattenhofer. 2011. MIS on trees. In *ACM PODC*. 41–48.

[33] Michael Luby. 1986. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.* 15, 4 (1986), 1036–1053. Conference version: *ACM STOC 1985*.

[34] Tzalik Maimon. 2021. Sleeping Model: Local and Dynamic Algorithms. arXiv:2112.05344

[35] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and computing: randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press.

[36] Chebiyyam Siva Ram Murthy and Balakrishnan Manoj. 2004. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, USA.

[37] Koji Nakano and Stephan Olariu. 2000. Randomized Leader Election Protocols in Radio Networks with No Collision Detection. In *Algorithms and Computation, 11th International Conference, ISAAC 2000, Taipei, Taiwan, December 18-20, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1969)*, D. T. Lee and Shang-Hua Teng (Eds.). Springer, 362–373.

[38] David Peleg. 2000. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics.

[39] Václav Rozhon and Mohsen Ghaffari. 2020. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 350–363.

[40] Qin Wang, Mark Hempstead, and Woodward Yang. 2006. A Realistic Power Consumption Model for Wireless Sensor Network Devices. In *The Third Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON '06, Vol. 1)*. 286–295.

[41] Ou Yang and Wendi Heinzelman. 2013. An Adaptive Sensor Sleeping Solution Based on Sleeping Multipath Routing and Duty-Cycled MAC Protocols. *ACM Transactions on Sensor Networks* 10, 1 (2013).

[42] Rong Zheng and Robin Kravets. 2005. On-demand power management for ad hoc networks. *Ad Hoc Networks* 3, 1 (2005), 51–68.