Distributed Computing (2026) 39:9
https://doi.org/10.1007/s00446-025-00498-4

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

Fabien Dufoulon[1] · **William K. Moses Jr.**[2] · **Gopal Pandurangan**[3]

Received: 1 November 2023 / Accepted: 19 November 2025

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 (i.e., with probability at least $1-n^{-1}$). This algorithm has a round complexity of $O((\log^7 n)\log\log n)$. We also show that we can 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 $\mathcal{CONGEST}$ model where messages of size $O(\log n)$ bits can be sent per edge per round.

Keywords Maximal independent set · Sleeping model · Energy-efficient · Awake complexity · Round complexity · Trade-offs

✉ William K. Moses Jr.
wkmjr3@gmail.com

Fabien Dufoulon
fabien.dufoulon.cs@gmail.com

Gopal Pandurangan
gopalpandurangan@gmail.com

1 School of Computing and Communications, Lancaster University, Lancaster, UK

2 Department of Computer Science, Durham University, Durham, UK

3 Department of Computer Science, University of Houston, Houston, TX, USA

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.

Published online: 19 March 2026

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 [39] 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 [25] 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 $\mathcal{LOCAL}$ model. The run time was further improved by Rozhon and Ghaffari [45, 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 $\mathcal{LOCAL}$ model. The currently best known randomized algorithm in the $\mathcal{CONGEST}$ model takes $O(\log \Delta \log \log n + \text{poly}(\log \log n))$ rounds [26]. Thus, the currently known best algorithms of MIS ([25, 26, 45]) are dependent on Δ (the maximum degree), and hence still take $O(\log n)$ rounds (even in the $\mathcal{LOCAL}$ 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 $\mathcal{LOCAL}$ as well as $\mathcal{CONGEST}$ models [26, 45].

There are faster distributed algorithms known for special classes of graphs such as trees [25, 38] and Erdos-Renyi random graphs [25, 36], but they still take $\Omega(\sqrt{\log n / \log \log n})$ rounds. There is also a fast deterministic distributed algorithm running in $\mathcal{O}(\log^* n)$ for growth-bounded graphs (which includes unit-disk graphs) [46]. 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, 25].

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 $\mathcal{LOCAL}$ 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 [37] that also applies to randomized algorithms and holds even in the $\mathcal{LOCAL}$ 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 [22, 49]. 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 [22, 34, 47–49]. A node may choose to enter and exit this sleeping mode in a judicious way to save energy during the course of an algorithm.[1]

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, 11, 12, 15, 16, 18, 27]). 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.

1.3 Model and complexity measures

1.3.1 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

[1] 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 [42, 48, 49]).

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 $\mathcal{CONGEST}$ model [44]. We note that our algorithms also, obviously, apply to the $\mathcal{LOCAL}$ model, another standard model [44] where there is no restriction on the size of the messages sent per edge per round. Though the $\mathcal{CONGEST}$ and $\mathcal{LOCAL}$ models do not put any constraint on the computational power of the nodes, our algorithms perform only light-weight computations (each node processes only $\mathrm{poly}(\log n)$ bits per round and takes computation time essentially linear in the number of bits processed).

1.3.2 Sleeping model

We assume the sleeping model [16], 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.

This paper assumes the sleeping model in the $\mathcal{CONGEST}$ setting, which we call the $\mathcal{SLEEPING}-\mathcal{CONGEST}$ model.

Several recent works (see Sect. 2) have designed distributed algorithms in the $\mathcal{SLEEPING}-\mathcal{CONGEST}$ model for fundamental problems such as MIS, approximate matching and vertex cover, spanning tree, minimum spanning tree, coloring, and other problems [2, 7, 16, 27, 28].

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 $\mathcal{LOCAL}$ 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 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 is the first distributed MIS algorithm with $O(\log\log n)$ awake complexity. This algorithm has *round complexity* $O(\log^7 n\log\log n)$. Our bounds hold even in the $\mathcal{CONGEST}$ model where messages of $O(\log n)$ bits can be sent per edge per round.
2. We show that we can 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 $\mathcal{CONGEST}$ model.

Our work answers a key question left open in [16], namely whether one can design MIS algorithms with (even) $o(\log n)$ (worst-case) awake complexity. We note that prior results [16, 27] presented algorithms in the sleeping model with $O(\log n)$ awake complexity (see Sect. 2). 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.

2 Related work and comparison

2.1 Prior work in the sleeping model for MIS

Chatterjee, Gmyr, and Pandurangan [16] 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 [16] is whether one can design an MIS algorithm with $o(\log n)$ worst-case awake complexity (even in the $\mathcal{LOCAL}$ model).[2]

Ghaffari and Portmann [27] subsequently improved the round complexity bound of Chatterjee, Gmyr, and Pandurangan [16]. They present 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 [31] 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 $\mathcal{LOCAL}$ 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.

Subsequent to the publication of the arxiv version of our paper [21], Ghaffari and Portmann [28] presented a distributed MIS algorithm with the same worst-case awake complexity of $O(\log\log n)$, but a better round complexity of round complexity $O(\log^2 n)$. They also present a second algorithm that has worse awake complexity of $O((\log\log n)^2)$ but with a better round complexity of $O(\log n \log\log n \log^* n)$.

[2] 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 [16].

2.2 Other works in the sleeping model

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 $\mathcal{LOCAL}$ 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 $\mathcal{LOCAL}$ model (as opposed to the $\mathcal{CONGEST}$ 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 $O(\log\Delta + \log^* n)$ awake time and $O(\Delta^2 + \log^* n)$ round complexity. Maimon [40] 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 $\mathcal{CONGEST}$ model). They use a spanning tree construction called the Labelled Distance Tree (LDT) which we also use in our algorithm.

2.3 Other prior works on distributed energy-efficient algorithms

There has been significant research on energy-efficient distributed algorithms over the years—more than we can survey here—and we restrict ourselves to those that are most relevant.

A first, closely linked line of work is that of Chang, Kopelowitz, Pettie, Wang, and Zhan [15] on radio networks (inspired by earlier work on energy efficient algorithms in radio networks e.g., [32, 33, 43]). Unlike our paper (and the other papers mentioned in the previous paragraph) that use the $\mathcal{SLEEPING-CONGEST}$ model, this line of work assumes a model with additional communication restrictions that reflect the behavior of radio networks; we refer to it as the $\mathcal{SLEEPING-RADIO}$ model. More concretely, in this model, nodes can only broadcast messages; hence the same message is sent to all neighbors. Additionally, 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.) In this model, they study the *energy complexity* measure, which is defined identically to (worst-case) awake complexity (i.e., both measures count only the rounds that a node is awake.

Energy-efficient algorithms for several problems such as broadcast, leader election, breadth-first search, maximal matching, diameter and minimum-cut computation have been studied in the $\mathcal{SLEEPING-RADIO}$ model [10–15, 18, 19]. An interesting open problem is whether our sub-logarithmic bounds on MIS awake complexity in the $\mathcal{SLEEPING-CONGEST}$ can be obtained in the $\mathcal{SLEEPING-RADIO}$ model.

King, Phillips, Saia, and Young [34] 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, 29, 30]).

Finally, Fraigniaud, Montealegre, Rapaport and Todinca [24] consider energy-efficiency from a slightly different angle. They do not assume nodes have a sleeping capability, but instead seek to minimize the maximum, taken over all nodes (or all edges), of the number of rounds in which these nodes (or edges) are active in sending messages (or transmitting messages). These two complexity measures are called respectively node-activation and edge-activation complexity, and the authors show that every Turing-computable problem can be solved with $O(1)$ node- and edge-activation complexity in the $\mathcal{CONGEST}$ setting, at the cost of possibly exponential in n round complexity. In the $\mathcal{LOCAL}$ setting, the round complexity can be reduced to polynomial in n.

3 High-level overview and techniques

The best known distributed MIS algorithms ([25, 26, 45]) in the traditional setting suffer from a $\log \Delta$ dependency in the round complexity, where Δ is the maximum degree (see Sect. 1). Prior to this work, that was also the case in the sleeping model as well.[3] 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, 17, 23], a variant of Luby's algorithm [39].

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) [17] with respect to that (random) ordering.[4] Fischer and Noever [23] 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 5.3) 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 5.1), 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 5.3) with an improved dependency on the ID upper bound I. This improved dependency comes into play when I is

[3] In particular, the algorithms of [16, 27] which had optimal $O(1)$ rounds *node-averaged* awake complexity, however, had $O(\log n)$ (worst-case) awake complexity.

[4] 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.

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 $\mathcal{CONGEST}$). 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 4.3—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 of MIS nodes among the first k nodes) is reduced (essentially) to $O(n/k)$ with high probability.

In Algorithm Awake-MIS (described in Sect. 6), 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 $\mathcal{CONGEST}$ 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 4.4—shows that the subgraph induced by a given batch's nodes (with no MIS neighbors) is shattered.

4 Preliminaries: notation and randomized techniques

4.1 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 $\{i, \ldots, j\}$.

4.2 Chernoff bounds

The following Chernoff bounds [41] 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 [41].

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

4.3 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 [35]—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) is at most $\frac{t'}{t}\ln(n)$. 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 $\varepsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t}\ln(n/\varepsilon)$ with probability at least $1-\varepsilon$.*

Proof Note that $(V_{t'} \backslash N(M_t)) \subseteq \{v_{t+1}, \ldots, v_{t'}\}$. 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 $G[V_{t'} \backslash N(M_t)]$ 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 [41]. 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 $U_i = (N(v_j) \cap V_{t'}) \backslash N(M_{i-1})$ 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,

$$
\begin{aligned}
&Pr[\forall it, v_i \notin U_i \mid v_j] \\
&\leq \prod_{i=1}^{t} \Pr[v_i \notin U_i \mid v_j, v_1, \ldots, v_{i-1}] \\
&\leq \prod_{i=1}^{t} (1 - \frac{d_i}{t'}) \leq (1 - \frac{d_t}{t'})^t \leq e^{-\frac{d_t}{t'}t} \\
&\leq e^{-\ln(n/\varepsilon)} \leq \varepsilon/n
\end{aligned}
$$

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 $1-\varepsilon/n$). □

4.4 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', $\Pr[C' > k] = \Pr[A_1 > 0, \ldots, A_k > 0]$

$$\leq \Pr[A_k > 0] \leq \Pr[A_0 + \sum_{i=1}^{k} Y_i > k] = \Pr[\sum_{i=1}^{k} Y_i \geq k].$$

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. □

5 Auxiliary procedures

5.1 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

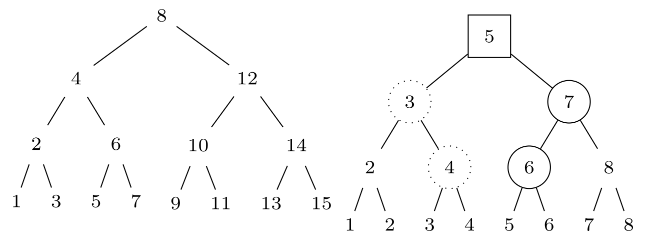


Fig. 1 Binary tree $\mathcal{B}([1,6])$ on the left and binary tree $\mathcal{B}^*([1,6])$ on the right. $S_3([1,6])$ consists of the dotted circle and rectangle nodes' labels and $S_5([1,6])$ of the (non-dotted) circle and rectangle nodes' labels. Note that $5 \in S_3([1,6]) \cap S_5([1,6])$ and $3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

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 Fig. 1 (right).

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, $k < r' \leqslant k'$. □

5.2 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. (Here, we use the improved version of Theorem 4 in [2] which can be found in the arXiv version.) 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 Appendix A.

Lemma 4 *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, $O(n'(\log n')\log^4 I)$ 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 Appendix A. 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.*

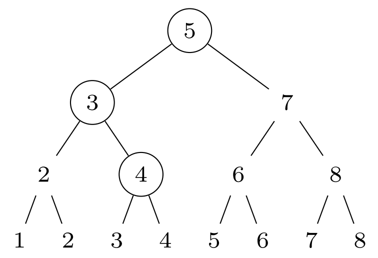
5.3 Simple awake-efficient MIS algorithms

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 Sect. 6) that these algorithms compute the LFMIS rather than any arbitrary MIS.

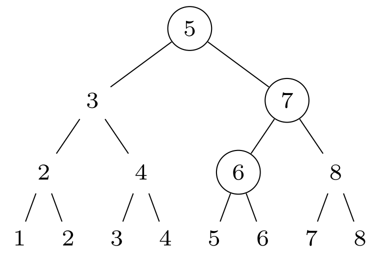
The first algorithm (VT-MIS) runs in $O(\log I)$ awake time, where I is an upper bound on the nodes' IDs, and illustrates how to use the virtual binary tree technique (see Sect. 5.1). At a high-level, VT-MIS is an awake-efficient version of the naive distributed implementation of sequential greedy MIS. 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 5.1). 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 Fig. 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".



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



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

Fig. 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

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'.) 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, $O(n'(\log n') \log^4 I + ((n')^2 \log n')/\log I)$ 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, $O(n'(\log n') \log^4 I)$ 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.*

6 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 Sect. 5 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 5.1 for the formal definition of the communication set $S_{g(p(v))}([1, 2\ell\Delta'])$ and Subsection 5.3 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 5.3. (In the analysis, we show that w.h.p., the remaining rounds are sufficient for this algorithm to terminate.)

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

Algorithm 1 Awake-MIS for node *v*

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*.[5]

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, the next lemma bounds the size of V_i for any $i \in [1, \ell]$.

Lemma 9 *For any $i \in [1, \ell]$, with probability at least $1 - 1/n^3$ it holds that:*

$$5(2^{i+1} - 1)\log n \leqslant |V_i| \leqslant 15(2^{i+1} - 1)\log n$$

[5] Indeed, in that case one can use $\ell = \lceil \log N - \log \log n \rceil$, the probability of joining batch $i \in [1, \ell]$ becomes $(10 \cdot 2^i \log N)/N$, and the remaining parameters change to $\Delta' = O(\log N)$, and the number of phases to $O(\log^5 N \log \log N)$. In turn, Lemma 9 is affected: for $i \leqslant \ell - \log n$, we get an $O(\log n)$ upper bound on $|V_i|$ but no lower bound. However, this does not significantly impact the remainder of the analysis.

Proof 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 $\delta = 1/2$: $\Pr[|V_i| \geq 15(2^{i+1}-1)\log n] \leq \exp(-(10(2^{i+1}-1)\log n)/10)$ 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 $1-1/n^3$. □

Next, we give a lemma that leads to the correctness of Awake-MIS.

Lemma 10 *With probability $1-1/n^2$, for any phase $(i,j) \in [1,\ell] \times [1, 2\Delta']$, it holds that:*

i) *The different connected components of $G[B^*_{i,j}]$ have size $O(\log n)$, where $B^*_{i,j}$ denotes the nodes in $B_{i,j}$ that are undecided when the phase (i, j) starts,*
ii) *Awake-MIS has computed the LFMIS over $G[V_{i,j}]$ with respect to a uniformly random ordering of $V_{i,j}$.*

Proof We show the lemma statement by induction on phase $(i,j) \in [1,\ell] \times [1,2\Delta']$. More precisely, we show the statement holds for any phase $(i,j) \in [1,\ell] \times [1,2\Delta']$ with probability at least $1 - \frac{4(i \cdot 2\Delta' + j)}{n^3}$.

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. By Lemma 9, these components are of size $O(\log n)$ with probability at least $1-1/n^3$, which satisfies the first half of the base case. For the remainder of the base case, assume that the components have size $O(\log n)$. Then, setting the number of rounds within the first phase accordingly—to $O(\log^5 n \log\log n)$ rounds—ensures that all LDT-MIS executions terminate. Thus, by Lemma 8, the output of LDT-MIS for each component C_h is an LFMIS M_h with respect to a uniformly random node ordering (of C_h). Their union, $M = \bigcup_{h=1}^{k} M_h$, is exactly the set of nodes with state *inMIS* when the first phase ends. Moreover, *M* is itself an LFMIS with respect to a uniformly random ordering of $V_{1,1}$. Indeed, one can straightforwardly construct such an ordering of $V_{1,1}$ (for the sake of the analysis only) out of the multiple components' orderings, since two nodes *u*, *v* from different components are not connected in $G[V_{1,1}]$, and thus ordering *u* before *v* (or inversely) does not influence the resulting LFMIS over $G[V_{1,1}]$.

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'), with probability $1 - \frac{4(i \cdot 2\Delta' + j - 1)}{n^3}$.

Let us first bound the size of the connected components of $G[B^*_{i,j}]$—we remind that $B^*_{i,j}$ are the nodes of $B_{i,j}$ with no neighbors in M'. 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). As such, for 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 $1-1/n^3$. 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. This implies that $G[V_i \setminus N(M_{i-1})]$ has maximum degree upper bounded by $\frac{|V_i|}{|V_{i-1}|}\ln(n^4)$ with probability at least $1-1/n^3$. By Lemma 9 and a union bound on the failure probabilities, $\frac{|V_i|}{|V_{i-1}|} \leqslant \frac{15(2^{i+1}-1)\log n}{5(2^i-1)\log n} \leqslant 9$ with probability at least $1-2/n^3$. Hence, by a union bound on the failure probabilities, we get that the maximum degree of $G[V_i \setminus N(M_{i-1})]$ is at most $9\ln(n^4)$ with probability at least $1-3/n^3$. By choosing $\Delta' = 9\ln(n^4)$ (and using the principle of deferred decisions), as well as another union bound, 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 $1-1/n^3-3/n^3 = 1-4/n^3$. 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 $1-4/n^3$.

For the remainder of this case, assume that the connected components of $G[B^*_{i_j}]$ have size $O(\log n)$. In which case, $O(\log^5 n \log\log n)$ rounds are sufficient for the LDT-MIS executions on these connected components to terminate. Then, similarly to the base case, we can show that the union M' of the different MIS computed over the connected components of $G[B^*_{i_j}]$—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.) As such, 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 $V_{i,j}$. Finally, the induction step follows from a union bound over the different failure probabilities for that last statement. □

Theorem 1 *MIS can be solved (w.h.p.) in $O(\log\log n)$ awake complexity and $O(\log^7 n \log\log n)$ round complexity in $\mathcal{CONGEST}$.*

Proof The correctness w.h.p. follows from item (ii) of Lemma 10 applied to phase $(\ell, 2\Delta')$.

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, note that in the proof of Lemma 10, we set the number of rounds per phase to $O(\log^5 n \log\log n)$ rounds, and that this is a deterministic upper bound. Since Awake-MIS has $O(\log^2 n)$ phases, the round complexity is $O(\log^7 n \log\log n)$.

Finally, note that all communication is done through $O(\log n)$ bit messages, and in particular that within the LDT-MIS calls. □

By using the more round-efficient LDT-MIS-ROUND (see Corollary 1) instead of LDT-MIS, an MIS can be computed with a 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 $\mathcal{CONGEST}$.*

Proof In Awake-MIS, if LDT-MIS-ROUND is used instead of LDT-MIS, phases of $O((\log n)(\log\log n)\log^* n)$ rounds suffice (see Corollary 1). □

7 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.

Labeled distance tree (LDT) and a faster round complexity

In this section, we describe a useful data structure called a labeled distance tree (LDT), introduced in [2]. We first describe the structure and relevant procedures needed to construct it in Sect. A.1. Subsequently, we give an informal description on how one might construct an LDT in Sect. A.2, i.e., we describe algorithm $LDT - Construct - Round$. Finally, in Sect. A.3 we give a few useful procedures that can be run once an LDT is constructed.

LDT description and relevant procedures

For a given graph, a *labeled distance tree (LDT)* is a spanning tree of that graph such that (i) all nodes in the tree know the ID of the root of the tree, 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 tree. If a given graph can be partitioned into a disjoint set of such LDTs, we refer to that graph as a *forest of labeled distance trees (FLDT)*.

In order to construct an LDT over a given graph, we start from a situation where all nodes are considered LDTs of their own (i.e., the overall graph is an FLDT) and we successively merge LDTs together in phases until we arrive at a situation where all nodes in the graph belong to the same LDT. In the course of this process, we utilize several simple procedures (e.g., upcast, broadcast) that are modified to be efficient in awake complexity. We first describe those procedures before explaining how to construct an LDT in the next section.

For the procedures described below, it is assumed that the initial graph has already been divided into an FLDT where each node u knows the ID of the root, *root*, of the LDT it belongs to, u's distance to *root* within the LDT, as well as u's parent and children, if any, in the LDT it belongs to. First of all, we define a transmission schedule that is used in each of the procedures and will be directly utilized in the algorithms. Then we describe the procedures themselves.

Transmission schedule of nodes in an LDT Consider an LDT rooted at the node *root* and a node *u* in that tree *at distance i from the root*. Let *n* be an upper bound on the number of nodes in the LDT. We describe a transmission schedule and an upper bound on the number of nodes in the LDT *n*. The transmission schedule $Transmission - Schedule(root, u, n)$ assigns a set of rounds to *u* to be awake in from a block of $2n + 1$ possible rounds. For ease of explanation, we assign names to each of these rounds as well. For all non-root nodes *u*, the set of rounds that $Transmission - Schedule(root, u)$ assigns to *u* includes rounds $i, i + 1, n + 1, 2n - i + 1$, and $2n - i + 2$ with corresponding names Down - Receive, Down - Send, Side - Send - Receive, Up - Receive,andUp-Send,respectively. $Transmission - Schedule(root, root, n)$ assigns to *root* the set containing only the rounds 1, $n + 1$, and $2n + 1$ with names Down-Send, Side-Send-Receive, and Up-Receive, respectively.[6]

This transmission schedule can be used to modify typical procedures on a tree (e.g., upcast, broadcast) to procedures that have small awake complexity. During one instance of $Transmission - Schedule(root, u, n)$, by having each node wake up in a carefully selected non-empty subset of its at most 5 named rounds, we guarantee that all nodes in the LDT have woken up at least once and that information is propagated in the correct "direction" in the LDT as needed. We name useful procedures to construct an LDT and give guarantees on these procedures below.

Broadcasting a message in an LDT Procedure $Fragment - Broadcast(n)$, run by all nodes in a given LDT, allows the root node of that LDT to transmit a given message to all nodes in the LDT in $O(1)$ awake complexity and $O(n)$ round complexity.

Upcasting the minimum value in an LDT Procedure $Upcast - Min(n)$, run by all nodes in a given LDT, allows the smallest value among all values held by the nodes of that LDT, to be propagated to the root of the tree in $O(1)$ awake complexity and $O(n)$ round complexity.

Transmitting a message between nodes of adjacent LDTs Procedure $Transmit - Adjacent(n)$, run by all nodes in an LDT, allows each node in the LDT to transfer a message, if any, to neighboring nodes belonging to other LDTs in $O(1)$ awake complexity and $O(n)$ round complexity.

[6] We assumed that $Transmission - Schedule(\cdot, \cdot, n)$ was started in round 1 when assigning rounds. If $Transmission - Schedule(\cdot, \cdot, n)$ is started in round *r*, then the correct round numbers can be obtained by adding $r - 1$ to the values mentioned in the description.

Construction of an LDT

In this section, we describe how to deterministically construct an LDT in algorithm $LDT - Construct - Round$. We note that the original process [2] was designed to construct a minimum spanning tree while simultaneously constructing an LDT over the original graph. We describe a simplified version of the process that focuses on just constructing an LDT over the original graph.

Algorithm. At a high level, we first spend a single round so that each node is aware of the IDs of each of its neighbors and each edge can be assigned a unique ID known to both endpoints (the ID can be the composition of the edge's endpoints' IDs in increasing order). Subsequently, we run a version of the classical GHS algorithm, modified to ensure that the awake complexity of nodes is small and to also ensure that at the beginning of each phase of the algorithm, an FLDT is maintained. Initially, each node is considered to be its own LDT (and so the overall graph is an FLDT) and we describe how to merge LDTs together in $O(\log n)$ phases until all nodes in the graph belong to the same LDT.

In each phase, we perform the following sequence of steps in three stages. Recall that at the beginning of the phase, we have a forest of LDTs. We explain each phase of the algorithm from a bird's eye perspective with the details of what each LDT does as bullet points.

1. **Stage 1.** Each LDT finds its "minimum" outgoing edge and both nodes of an outgoing edge are made aware of it. (Since the edges are unweighted, this is just the outgoing edge whose ID is smallest among all outgoing edges.) Also, all nodes of an a given LDT should know both the name of the outgoing edge and the ID of the LDT that edge leads to.
   (a) (The nodes in) each LDT run procedure $Upcast - Min(n)$ to collect the ID of the "minimum" outgoing edge at the root of the LDT.
   (b) Each LDT runs procedure $Fragment - Broadcast(n)$ to let all nodes in the LDT know the ID of the chosen outgoing edge.
   (c) Each LDT L runs $Transmit - Adjacent(n)$ to let nodes from other LDTs know if they are part of the chosen outgoing edge from L.

2. **Stage 2.** Consider the supergraph H where each LDT is a node and the outgoing edges identified in the previous step are the edges. Denote the connected components in H as C_1, C_2, , C_p. Our final goal is to decompose the connected components in H in to a forest F of small-depth trees SDT_1, SDT_2, ,

SDT_k.[7] This is done as follows. For each connected component C_i in H, the LDTs that make up C_i work together to construct a spanning tree T_i on top of C_i. Then the LDTs in each T_i simulate a Cole-Vishkin style coloring of T_i and use this coloring to then perform a maximal matching. Each edge of this maximal matching is added to F. Finally, after the maximal matching, if there exist isolated LDTs (i.e., unmatched LDTs), those LDTs add their outgoing edge to their parent LDT to F. If the isolated LDT is a root of some T_i, then it will not have a parent. In this case, this isolated LDT chooses an outgoing edge to one of its children in T_i. (a) Notice that in each connected component C_i of the supergraph H, there will exist *exactly* two LDTs with outgoing edges into each other. The LDT with the smaller ID becomes the root of the tree T_i for that connected component and the other LDT becomes its child. By having all LDTs use procedures $Upcast-Min(n)$ and $Fragment-Broadcast(n)$ a constant number of times, the nodes in both of these LDTs identify that they belong to such LDTs in the connected component and can identify if they are in the LDT that forms the root of the tree in H.

(b) Subsequently, each remaining LDT in C_i identifies its outgoing edge as leading to its parent in the spanning tree T_i. This does not require communication between nodes as this is just local computation performed by the roots of the LDTs.

(c) Now, a Cole-Vishkin style coloring algorithm to 6-color the LDTs of each T_i (i.e., each LDT in T_i is colored a single color in [1, 6], known to all nodes within that LDT) can be easily simulated. Each round of the $O(\log^* n)$ algorithm consists of some local computation and then having each node communicate with each of its children. One round of local computation can be simulated by the roots of each of the LDTs in one round. One round of communication can be simulated in $O(1)$ awake complexity and $O(n)$ round complexity via a constant number of uses of the procedures $Upcast-Min(n)$, $Fragment-Broadcast(n)$, and $Transmit-Adjacent(n)$.

(d) Previously, a spanning tree was constructed over each connected component of the supergraph H and each of the LDTs, constituting nodes in this spanning tree, were colored in via a 6-coloring. Now, in order to obtain a maximal matching on these components, the following process is run for 6 phases. We maintain the invariant that at the beginning of each phase, each LDT (all nodes belonging to that LDT) knows if it is matched or not and for every inter-LDT edge, both nodes of that edge know the matching status of both LDTs. In phase i, each unmatched LDT of color i chooses one of its unmatched children, if any, arbitrarily (say by performing procedure $Upcast-Min(n)$ to return the smallest inter-LDT edge among edges leading to unmatched children). Subsequently the LDT informs the child of being matched to it and informs all adjacent LDTs that it is no longer unmatched. Each phase can be simulated through a constant number of uses of $Upcast-Min(n)$, $Fragment-Broadcast(n)$, and $Transmit-Adjacent(n)$.

(e) Each LDT that remains unmatched in T_i (except the root LDT of T_i) now informs its parent LDT in T_i that the inter-LDT edge between them also belongs to F. This can be done through a constant number of uses of $Upcast-Min(n)$, $Fragment-Broadcast(n)$, and $Transmit-Adjacent(n)$.

(f) Finally, if the parent LDT in T_i is unmatched, it chooses one of its children in T_i and informs that child (really the node within that LDT at the other end of the edge) that the inter-LDT edge between them is also in F. This can be done through a constant number of uses of $Upcast-Min(n)$, $Fragment-Broadcast(n)$, and $Transmit-Adjacent(n)$.

3. **Stage 3.** At the end of the previous stage, a forest F of small-depth trees SDT_i was formed. (These trees consist of the LDTs and any edges added to F. While all nodes within an LDT may not be aware of these edges in F, the endpoints (nodes within LDTs) of every such edge are aware of it.) The final part of each phase consists of merging together the LDTs in each small-depth tree SDT_i into one large LDT. Care must be taken to ensure that each node of a resulting merged LDT has the correct ID for that LDT (i.e., the ID of the root of the LDT) and furthermore, each node in the merged LDT maintains the correct distance-from-root value. Additionally, each node may need to be re-oriented, i.e., each node may need to update information about who its parent and children in the LDT are. (a) Notice that every tree SDT_i in F is of diameter at most 4 (i.e., any two LDTs in the same tree SDT_i in F are at

[7] Note that each tree SDT_i is itself a supergraph consisting of LDTs as nodes and edges between LDTs forming the edges in the supergraph. That is, both F and H are supergraphs with the same set of nodes but possibly different edges. Also note that the number of small-depth trees in F, k, may be different than the number of connected components in H, p.

most distance 4 apart).[8] First, for each SDT_i, we choose its constituent LDT with the smallest ID to be the core of the merged LDT, around which we re-orient the nodes of the other LDTs. It is easy to see that, for any tree SDT_i in F, the smallest LDT ID in SDT_i can be progagated to every LDT in SDT_i through a constant number of instances of $Upcast - Min(n)$, $Fragment - Broadcast(n)$, and $Transmit - Adjacent(n)$.

(b) Now, all nodes within a given tree SDT_i in F know the ID that will become the ID of the final merged LDT. Let LDT L have this ID. Consider an LDT L' that is adjacent to the LDT L in SDT_i. The nodes in L' must re-align themselves and update their distance-to-root values in the merged LDT consisting of L and L'. Suppose the edge (u, v) is the inter-LDT edge between L and L' such that $u \in L$ and $v \in L'$. Now, the nodes in L' update their values by utilizing two instances of a transmission schedule parameterized by n. Recall that v is aware of u's distance-to-root and so knows its own distance-to-root. In the first instance of the transmission schedule the nodes in the branch from v to the root of L' update their distance-to-root values and re-orient themselves using the Up-Send and Up-Receive rounds. In the second instance, the remaining nodes in L' update their values in the Down-Send and Down-Receive rounds and can re-orient themselves. This process allows all LDTs at distance 1 from LDT L in SDT_i to update their values. By running this process 4 times, we guarantee that all LDTs up to and including distance 4 from LDT L in SDT_i can update their values and re-orient themselves.***Analysis.*** We briefly analyze the algorithm above to give the intuition of correctness and complexities of the construction. We note that the full analysis was already given in [2]. The correctness comes from the fact that we maintain a forest of LDTs at the beginning of each phase. The construction of the forest F in the final part of each phase and the merging of LDTs in F guarantees that a constant number of the LDTs are merged together in each phase. To see this, notice that, in every phase, every tree SDT_i in F consists of at least two LDTs, resulting in at least a constant fraction of the LDTs "disappearing" in the phase (but really just being merged into one another).[9] As a result, after $O(\log n)$ phases, all LDTs are merged together into one single LDT.

Regarding the awake complexity and round complexity, we see that the first and third stage each use a constant number of transmission schedules parameterized by n, as well as a constant number of calls to $Upcast - Min(n)$, $Fragment - Broadcast(n)$, and $Transmit - Adjacent(n)$. Totally, all of these contribute $O(1)$ awake complexity and $O(n)$ round complexity. However, in the second stage, the process of simulating the Cole-Vishkin style coloring takes a total of $O(\log^* n)$ awake complexity and $O(n \log^* n)$ round complexity. Since, there are a total of $O(\log n)$ phases, we see that the total awake complexity is $O((\log n) \log^* n)$ and the total round complexity is $O(n(\log n) \log^* n)$.

The following lemma captures the above guarantees.

Lemma 11 *For a given graph of at most n nodes, where n is known to all the nodes, $LDT - Construct - Round$ deterministically constructs an LDT over the given graph in the $\mathcal{CONGEST}$ setting in $O((\log n) \log^* n)$ awake complexity and $O(n(\log n) \log^* n)$ round complexity.*

Useful procedures

Once an LDT is constructed, we can then leverage it to run fast awake complexity procedures. In particular, we describe the operations of broadcast and ranking from Definition 1.

Broadcasting in an LDT We describe Procedure $LDT - Broadcast$, run by each node v of an LDT, which takes the message msg_r of the root of the LDT as input and results in all nodes of the LDT receiving msg_r. Consider an LDT of size at most n', where the root of the LDT wants to broadcast a message of size at most m_r bits to all nodes in the LDT. Assume that messages of size $O(m_r)$ can be sent over each edge. Broadcast can be performed by having all nodes in the LDT participate in one instance of a transmission schedule parameterized by n', where each node receives this message in their Down-Receive round and propagates the message to its children in its Down-Send round. This takes $O(1)$ awake complexity and $O(n')$ round complexity.

Ranking in an LDT We describe Procedure $LDT - Ranking$, run by each node v of an LDT,

[8] To see why this is true, notice how the edges were added to F to construct each SDT_i. First, edges were added from a maximal matching. This created small-depth trees of diameter 1. Now, for each tree T_i , isolated (non-root) LDTs added edges to their parents. This could result in small-depth trees of diameter 3. Finally, for each tree T_i, if the root LDT is isolated, it could add an edge to one of its children. This could increase the diameter of the resulting small-depth tree by 1.

[9] Notice that each tree SDT_i in F is formed from the LDTs in H as well as edges added to F in stage two. In stage two, every LDT in each tree T_i is either unmatched or matched with another LDT. Each LDT that is matched ensures that the small-depth tree its belongs to comprises of at least two LDTs. For each LDT that remained unmatched, it is still guaranteed to merge with another LDT because it will either add an edge to its parent in T_i to F (if it is not the root of T_i) or else add an edge to one of its children in T_i to F (if it is the root of T_i).

which calculates *v*'s rank in some overall total order of the nodes of the LDT, and additionally allows *v* to learn the size of the LDT. (The given total ordering may be intuitively understood as the in-order ranking of a binary tree, when extended to an *n*-ary tree. For an *n*-ary tree, one possible generalization, that we use here, would be to recursively do the following: visit the leftmost subtree, then the root, then the remaining subtrees in some arbitrary order.) Consider an LDT of size at most n'. The procedure consists of two instances of a transmission schedule parameterized by n'. In the first instance of a transmission schedule, each node listens in its Up-Receive round for the number of nodes in each of its children's subtrees and remembers these values, and then in its Up-Send round it sends the sum of these values plus one to its parent. At the end of this instance of a transmission schedule, the root will know the total number of nodes in the tree n'. In the second instance of the transmission schedule, each node *v* receives in its Down-Receive round the value of n' and a value from its parent, which *v* uses to calculate its rank in a manner described below, and subsequently sends down the value of n' and unique values to each of *v*'s children in its Down-Send round. For a given node *v* that receives a value *x* and has *k* children $c_1, c_2, \ldots, c_k$ with corresponding number of nodes in their subtrees $n_1, n_2, \ldots, n_k$, do the following. We assume that the root of the LDT "receives" $x = 0$.[10] Set *v*'s rank to $x + n_1$. During *v*'s Down-Send round, it sends down *x* to c_1, and for $i \in [2, k]$, *v* sends the value $1 + \sum_{j=1}^{i-1} n_j$ to c_i. This procedure takes *O*(1) awake complexity and $O(n')$ round complexity.

The guarantees on constructing the LDT as well as the procedures described above are reflected in the following lemma.

Lemma 12 *For any connected $V' \subseteq V$ of at most n' nodes, where n' is known to all nodes, with unique IDs in [1, I]:*

- An LDT over $G[V']$ can be constructed deterministically with $O(\log I)$ bit messages, with $O((\log n') \log^* I)$ awake complexity and $O(n'(\log n') \log^* I)$ round complexity.
- Broadcast over an LDT—in which the LDT root v_r has a message of size at most m_r—can be executed deterministically with $O(m_r)$ bit messages, with *O*(1) awake complexity and $O(n')$ round complexity.
- Ranking over an LDT—in which each node $v \in V'$ learns its rank in some total ordering of the tree and also learns $|V'|$—can be executed deterministically with $O(\log I)$ bit messages, with *O*(1) awake complexity and $O(n')$ round complexity.

[10] Each node maintains an arbitrary internal ordering of its children in the LDT, if any.

Acknowledgements We thank the program committee of PODC 2023 for inviting our paper to the special issue of the Distributed Computing journal devoted to the best papers of PODC 2023. Part of the work was done while Fabien Dufoulon and William K. Moses Jr. were postdoctoral fellows at the University of Houston in Houston, USA. F. Dufoulon was supported in part by National Science Foundation (NSF) grants CCF-1540512, IIS-1633720, and CCF-1717075 and U.S.-Israel Binational Science Foundation (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. **Note:** A preliminary version of this paper appeared in PODC 2023 [20]. It was invited to the special issue of this journal.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Alon, N., Babai, L., Itai, A.: A fast and simple randomized parallel algorithm for the maximal independent set problem. Journal of Algorithms **7**(4), 567–583 (1986)
2. Augustine, J., Moses Jr. W.K., Pandurangan, G.: 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), pages 51–53, (2022). Full version available on arXiv:2204.08385
3. Balliu, A., Brandt, S., Hirvonen, J., Olivetti, D., Rabie, M., Suomela, J.: Lower bounds for maximal matchings and maximal independent sets. In IEEE FOCS, pages 481–497, (2019)
4. Balliu, A., Brandt, S., Kuhn, F., Olivetti, D.: Improved distributed lower bounds for MIS and bounded (out-)degree dominating sets in trees. In PODC '21: ACM Symposium on Principles of Distributed Computing, pages 283–293. ACM, (2021)
5. Barenboim, L., Elkin, M.: Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. Distributed Comput. **22**(5–6), 363–379 (2010)
6. Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. Journal of the ACM **63**(3), 1 (2016)
7. Barenboim, L., Maimon, T.: Deterministic logarithmic completeness in the distributed sleeping model. In 35th International Symposium on Distributed Computing, DISC, volume 209, pages 10:1–10:19, (2021)

8. Bender, M.A., Fineman, J.T., Movahedi, M., Saia, J., Dani, V., Gilbert, S., Pettie, S., Young, M.: Resource-competitive algorithms. SIGACT News **46**(3), 57–71 (2015)
9. Blelloch, G.E., Fineman, J.T., Shun, J.: Greedy sequential maximal independent set and matching are parallel on average. In ACM symposiumon parallelism in algorithms and architectures (SPAA), pages 308–317, (2012)
10. Chang, Y.-J.: The energy complexity of diameter and minimum cut computation in bounded-genus networks. In Sergio Rajsbaum, Alkida Balliu, Joshua J. Daymude, and Dennis Olivetti, editors, Structural information and communication complexity (SIROCCO), pages 262–296, Cham, (2023). Springer Nature Switzerland
11. Chang, Y.-J., Dani, V., Hayes, T.P., He, Q., Li, W., Pettie, S.: The energy complexity of broadcast. In ACM PODC, pages 95–104, (2018)
12. Chang, Y.-J., Dani, V., Hayes, T.P., Pettie, S.: The energy complexity of BFS in radio networks. In ACM PODC, pages 273–282, (2020)
13. Chang, Y.-J., Duan, R., Jiang, S.: Near-optimal time–energy tradeoffs for deterministic leader election. ACM Trans. Algorithms (2023). https://doi.org/10.1145/3614429
14. Chang, Y.-J., Jiang, S.: The energy complexity of las vegas leader election. In Proceedings of the 34th ACM symposium on parallelism in algorithms and architectures, SPAA '22, page 75–86, New York, NY, USA, (2022). Association for Computing Machinery. https://doi.org/10.1145/3490148.3538586
15. Chang, Y.-J., Kopelowitz, T., Pettie, S., Wang, R., Zhan, W.: Exponential separations in the energy complexity of leader election. ACM Trans. Algorithms **15**(4), 49:1–49:31 (2019)
16. Chatterjee, S., Gmyr, R., Pandurangan, G.: Sleeping is efficient: MIS in O(1)-rounds node-averaged awake complexity, pp. 99–108. In ACM Symposium on Principles of Distributed Computing, PODC (2020)
17. Coppersmith, D., Raghavan, P., Tompa, M.: Parallel graph algorithms that are efficient on average. Inf. Comput. **81**(3), 318–333 (1989)
18. Dani, V., Gupta, A., Hayes, T.P., Pettie, S.: Wake up and join me! an energy-efficient algorithm for maximal matching in radio networks. In 35th international symposium on distributed computing (DISC), pages 19:1–19:14, (2021)
19. Dani, V., Hayes, T.P.: How to wake up your neighbors: Safe and nearly optimal generic energy conservation in radio networks. In Christian Scheideler, editor, 36th international symposium on distributed computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA, volume 246 of LIPIcs, pages 16:1–16:22, (2022)
20. Dufoulon, F., Moses Jr, W.K., Pandurangan, G.: Distributed MIS in O(log log n) awake complexity. In Rotem Oshman, Alexandre Nolin, Magnús M. Halldórsson, and Alkida Balliu, editors, Proceedings of the 2023 ACM symposium on principles of distributed computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023, pages 135–145. ACM, (2023). https://doi.org/10.1145/3583668.3594574
21. Dufoulon, F., Moses Jr, W.K., Pandurangan, G.: Sleeping is superefficient: MIS in exponentially better awake complexity. arXiv preprint arXiv:2204.08359, (2022)
22. Feeney, L.M., Nilsson, M.: Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In IEEE INFOCOM **3**, 1548–1557 (2001)
23. Fischer, M., Noever, A.: Tight analysis of parallel randomized greedy MIS. In SODA, pages 2152–2160, (2018)
24. Fraigniaud, P., Montealegre, P., Rapaport, I., Todinca, I.: Energy-efficient distributed algorithms for synchronous networks. In Sergio Rajsbaum, Alkida Balliu, Joshua J. Daymude, and Dennis Olivetti, editors, structural information and communication complexity (SIROCCO), pages 482–501, Cham, (2023). Springer Nature Switzerland
25. Ghaffari, M.: An improved distributed algorithm for maximal independent set. In SODA, pages 270–277, (2016)
26. Ghaffari, M., Grunau, C., Rozhon, V.: Improved deterministic network decomposition. In Proceedings of the 2021 ACM-SIAM symposium on discrete algorithms, SODA, pages 2904–2923, (2021)
27. Ghaffari, M., Portmann, J.: Average awake complexity of MIS and matching. In ACM Symposium on parallelism in algorithms and architectures (SPAA), pages 45–55, (2022)
28. Ghaffari, M, Portmann, J.: Distributed MIS with low energy and time complexities. In Rotem Oshman, Alexandre Nolin, Magnús M. Halldórsson, and Alkida Balliu, editors, Proceedings of the 2023 ACM symposium on principles of distributed computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023, pages 146–156. ACM, (2023). https://doi.org/10.1145/3583668.3594587
29. Gilbert, S., King, V., Pettie, S., Porat, E., Saia, J., Young, M.: (near) optimal resource-competitive broadcast with jamming. In Guy E. Blelloch and Peter Sanders, editors, 26th ACM Symposium on parallelism in algorithms and architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014, pages 257–266. ACM, (2014)
30. Gilbert, S., Young, M.: Making evildoers pay: resource-competitive broadcast in sensor networks. In Darek Kowalski and Alessandro Panconesi, editors, ACM symposium on principles of distributed computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012, pages 145–154. ACM, (2012)
31. Hourani, K., Pandurangan, G., Robinson, P.: Awake-efficient distributed algorithms for maximal independent set. In IEEE conference on distributed computing systems (ICDCS), pages 1338–1339, (2022)
32. Jurdzinski, T., Kutylowski, M., Zatopianski, J.: Efficient algorithms for leader election in radio networks. In Aleta Ricciardi, editor, Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002, pages 51–57. ACM, (2002)
33. Kardas, M., Klonowski, M., Pajak, D.: 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, pages 399–408. IEEE Computer Society, (2013)
34. King, V., Phillips, C.A., Saia, J., Young, M.: Sleeping on the job: Energy-efficient and robust broadcast for radio networks. Algorithmica **61**(3), 518–554 (2011)
35. Konrad, C.: MIS in the congested clique model in $O(\log\log\Delta)$ rounds. arXiv preprint arXiv:1802.07647, (2018)
36. Krzywdziński, K., Rybarczyk, K.: Distributed algorithms for random graphs. Theoret. Comput. Sci. **605**, 95–105 (2015)
37. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: Lower and upper bounds. Journal of the ACM (2016). https://doi.org/10.1145/2742012
38. Lenzen, C., Wattenhofer, R.: MIS on trees. In ACM PODC, pages 41–48, (2011)
39. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing **15**(4), 1036–1053 (1986)
40. Maimon, T.: Sleeping model: Local and dynamic algorithms, (2021). arXiv:2112.05344
41. Mitzenmacher, M., Upfal, E.: Probability and computing: randomization and probabilistic techniques in algorithms and data analysis. Cambridge university press, (2017)
42. Murthy, C.S.R., Manoj, B.: Ad Hoc wireless networks: architectures and protocols. Prentice Hall PTR, USA (2004)

43. Nakano, K., Olariu, S.: Randomized leader election protocols in radio networks with no collision detection. In D. T. Lee and Shang-Hua Teng, editors, Algorithms and Computation, 11th International Conference, ISAAC 2000, Taipei, Taiwan, December 18-20, 2000, Proceedings, volume 1969 of Lecture Notes in Computer Science, pages 362–373. Springer, (2000)
44. Peleg, D.: Distributed computing: a locality-sensitive approach. Society for industrial and applied mathematics, (2000)
45. Rozhon, V., Ghaffari, M.: Polylogarithmic-time deterministic network decomposition and distributed derandomization. In 52nd Annual ACM SIGACT symposium on theory of computing, STOC, pages 350–363, (2020)
46. Schneider, J., Wattenhofer, R.: A log-star distributed maximal independent set algorithm for growth-bounded graphs. pages 35–44. ACM, (2008)
47. Wang, Q., Hempstead, M., Yang, W.: 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, volume 1 of SECON '06, pages 286–295, (September 2006)
48. Yang, O., Heinzelman, W.: An adaptive sensor sleeping solution based on sleeping multipath routing and duty-cycled mac protocols. ACM Transactions on Sensor Networks (2013). https://doi.org/10.1145/2529977
49. Zheng, R., Kravets, R.: On-demand power management for ad hoc networks. Ad Hoc Networks **3**(1), 51–68 (2005)