Sleeping is Superefficient: MIS in Exponentially Better Awake Complexity

Fabien Dufoulon∗ William K. Moses Jr.† Gopal Pandurangan ‡

Abstract

Maximal Independent Set (MIS) is one of the central and most well-studied problems in distributed computing. Even after four decades of intensive research, the best-known (randomized) MIS algorithms take $O(\log n)$ worst-case rounds on general graphs (where $n$ is the number of nodes), while the best-known lower bound is $\Omega\left(\sqrt{\frac{\log n}{\log \log n}}\right)$ rounds. Breaking past the $O(\log n)$ worst-case bound or showing stronger lower bounds have been longstanding open problems.

Motivated by resource-considerations in energy-constrained distributed networks such as ad hoc wireless and sensor networks, we study MIS algorithms in the sleeping model [Chatterjee, Gmyr, and Pandurangan, PODC 2020], a model for design and analysis of resource-efficient distributed algorithms. The sleeping model is a generalization of the traditional model, that allows nodes to enter either “sleep” or “waking” states in any round. While the waking state corresponds to the default state in the traditional model, in the sleeping state a node does not send or receive messages (and messages sent to it are also lost) and does not incur any time, communication, or local computation cost. Thus, in the sleeping state, a node spends little or no energy. Hence, in this model, only rounds in which a node is awake are counted and we are interested in minimizing the worst-case number of rounds a node spends in the awake state, i.e., the awake complexity. While the primary goal is to minimize awake complexity, a secondary goal is to reduce the (traditional) worst-case round complexity (that counts both the awake and sleeping rounds).

Our main contribution is that we show that MIS can be computed in (worst-case) awake complexity of $O(\log \log n)$ rounds that is (essentially) exponentially better compared to the (traditional) round complexity lower bound of $\Omega\left(\sqrt{\frac{\log n}{\log \log n}}\right)$. 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 $O(\log \log n)$-rounds awake complexity.¹ This algorithm has (traditional) round complexity that is $O(\text{poly}(n))$. Our bounds hold in the $\text{CONGEST}(O(\text{poly}(\log n)))$ model where only $O(\text{poly}(\log n))$ (specifically $O(\log^3 n)$) bits are allowed to be sent per edge per round.

2. We also show that we can drastically reduce the round complexity at the cost of a slight increase in awake complexity by presenting a randomized MIS algorithm with

---

¹Throughout, we use “with high probability (w.h.p.)” to mean with probability at least $1 - n^{-1}$. 

*Department of Computer Science, University of Houston, Houston, TX, USA. fabien.dufoulon.cs@gmail.com. https://orcid.org/0000-0003-2977-4109. Supported, in part, by NSF grants IIS-1633720, CCF-1717075, and BSF award 2016419.

†Department of Computer Science, University of Houston, Houston, TX, USA. wkmjr3@gmail.com. https://orcid.org/0000-0002-4533-7593. Supported, in part, by NSF grants IIS-1633720, CCF-1717075, and BSF award 2016419.

‡Department of Computer Science, University of Houston, Houston, TX, USA. gopalpandurangan@gmail.com. https://orcid.org/0000-0001-5833-6592. Supported, in part, by NSF grants IIS-1633720, CCF-1717075, and BSF award 2016419.
$O(\log \log n \log^* n)$ awake complexity and $O(\log^3 n \log \log n \log^* n)$ round complexity in the \textit{CONGEST}(O(poly(\log n))) model.

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 $O(\log n)$ in the traditional model and bypassing its $\Omega\left(\sqrt{\frac{\log n}{\log \log n}}\right)$ lower bound. Since a node spends resources only in its awake rounds, our algorithms are very resource-efficient compared to the traditional algorithms.

1 Introduction

1.1 Maximal Independent Set Problem

The \textit{maximal independent set} or \textit{MIS} problem is one of the central and most well-studied problems in distributed computing. The MIS problem is: Given a graph with $n$ nodes, each node must (irrevocably) commit to being in the set $M$ 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.

The MIS problem arises in many applications (e.g., resource allocation, dominating set construction) as well as a basic primitive in distributed algorithms (e.g., [31, 27]). A canonical application of MIS is to break local symmetry. For example, in a wireless network, neighboring nodes may not be able to communicate simultaneously due to signal interference and hence have to compete for channel allocation. One way to solve this problem is to find an MIS; only nodes that belong to the MIS (and thus are not neighbors) are allowed to communicate simultaneously which reduces interference. Another property of MIS is that it is also a \textit{dominating set}; in fact it is a \textit{minimal} dominating set, which has many applications, e.g., it can be used as a routing backbone for efficient communication (see e.g., [35]).\footnote{Given a graph $G = (V, E)$, a dominating set $D \subseteq V$ is a subset of nodes such that every vertex in $V$ either belongs to $D$ or has a neighbor in $D$.} These applications have been studied extensively in the context of ad hoc wireless and sensor networks [33].

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., number of rounds). In 1986, Alon, Babai, and Itai [1] and Luby [26] presented a randomized distributed algorithm for MIS, that takes $O(\log n)$ ($n$ is the number of nodes in the graph) rounds with high probability. Since these results, there has been a lot of progress in recent years in designing progressively faster distributed MIS algorithms. For $n$-node graphs with maximum degree $\Delta$, Ghaffari [17] presented a randomized MIS algorithm running in $O(\log \Delta) + 2O(\sqrt{\log \log n})$ rounds, improving over the algorithm of Barenboim, Elkin, Pettie and Schneider [6] that runs in $O(\log^2 \Delta) + 2O(\sqrt{\log \log n})$ rounds. It was further improved by Rozhon and Ghaffari [32, 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 \textit{LOCAL} model.\footnote{\textit{LOCAL} and \textit{CONGEST} are standard models in distributed computing, see Section 1.3.} The currently best-known randomized algorithm in the \textit{CONGEST} model takes $O(\log \Delta \log \log n + \text{poly}(\log \log n))$ rounds [18]. Thus, the current known best algorithms of MIS ([32, 18, 17]) are dependent on $\Delta$ (the maximum degree), and hence still take $O(\log n)$ rounds 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 \textit{LOCAL} as well as \textit{CONGEST} models [32, 18].

There are faster distributed algorithms known for special classes of graphs such as trees [17, 25] and Erdos-Renyi random graphs [23, 17], but they still take $O\left(\sqrt{\frac{\log n}{\log \log n}}\right)$ rounds. There are also...
MIS algorithms that run faster on graphs with low arboricity, but they nevertheless take $O(\log n)$ time on high arboricity graphs [17, 5].

While the above results make significant progress in the round complexity of the MIS problem, however, in general graphs, the best-known running time is still $O(\log n)$ (even for randomized algorithms). 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 Wattenhoffer [24] that also applies to randomized algorithms and holds even in the $\text{LOCAL}$ model. Thus, for example, say, when $\Delta = \omega(2^{\sqrt{\log n}})$, it follows that one cannot hope for algorithms faster than $\sqrt{\frac{\log n}{\log \log n}}$ 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 $\Delta << \log \log n$ (for randomized algorithms) [3, Corollary 5] and $\Delta << \log n$ (for deterministic algorithms) [3, Corollary 6]. (See also results on an improved lower bound that applies to trees [4].)

1.2 Energy Considerations and the Sleeping Model

The sleeping model (introduced in [12], see also the related energy complexity model [11] — see Section 1.5) is motivated by resource-constrained networks such as ad hoc wireless and sensor networks. In such networks, a node’s energy consumption depends on the amount of time it is actively communicating with nodes. It is well-known that the energy consumption by node when it is idle and just listening (waiting to hear from a neighbor) is only slightly smaller than that in a transmitting or receiving state [37, 15]. Thus, even though there might be no messages exchanged between a node and its sender, a node might be spending quite a bit of energy if it is just waiting to receive a message.

On the other hand, the energy consumption 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 [37, 15, 21, 34, 36]. A node may choose to enter and exit sleeping mode in a judicious way to save energy during the course of an algorithm.\(^4\)

Motivated by the above considerations, in the sleeping model (see Section 1.3 for details), a node can be in either of the two states — sleeping or awake. While in the traditional model nodes are only in the awake state, in the sleeping model nodes have the option of entering sleeping state at any round as well as exiting the sleeping state and entering the awake state at a later round. In the sleeping state, a node does not send or receive messages and messages sent to it by other nodes are lost; it also does not do any local computation. If a node enters a sleeping state, then it is assumed that it does not incur any cost, including messages, time, or resources such as energy. However, it becomes challenging to design distributed algorithms that are “awake-efficient” in the sleeping model; since a node can only communicate with a neighboring node that is awake, coordinating such communication efficiently is non-trivial.

1.3 Model and Complexity Measures

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

\(^4\)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 [37, 36, 30]).
Nodes are allowed to communicate through the edges of the graph \( G \) and it is assumed that communication is \textit{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 small size \( (O(\text{poly}(\log n))) \) bits with each of its neighboring nodes.

This standard model of distributed computation is called the \textit{CONGEST}(\( O(\text{poly}(\log n)) \)) model [31]. We note that our algorithms also, obviously, apply to the \textit{LOCAL} model, another standard model [31] where there is no restriction on the size of the messages sent per edge per round. Though the \textit{CONGEST} and \textit{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 \( \text{poly}(\log n) \) bits per round and takes computation time essentially linear in the number of bits processed).

\textbf{Sleeping Model.} We consider the sleeping model [12], 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 generalization of the above traditional model, where nodes are always assumed to be awake. In the sleeping model, each node decides to be either \textit{awake} or \textit{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 \textit{sleep} starting at any (specified) round of its choice; we assume all nodes know the correct round number whenever they are awake. It can \textit{wake up} again later at any specified round and enter the \textit{awake} state.\footnote{For details on how this can be implemented in wireless networks we refer to [12].} 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.

\textbf{Complexity Measures.} 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 \textit{(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. Apart from minimizing the awake complexity, we also strive to minimize the overall (traditional) \textit{round complexity} (also called \textit{time complexity}), where both, sleeping and awake rounds, are counted.

\textbf{Additional 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 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 \( \{i, \ldots, j\} \).

\textbf{1.4 Our Contributions}

In light of the difficulty in breaking the \( o(\log n) \) round barrier of MIS and the lower bound of \( \Omega\left( \min\left\{ \frac{\log \Delta}{\log \log \Delta}, \frac{\sqrt{\log n}}{\log \log n} \right\} \right) \) rounds in the traditional model (even for \textit{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 \( o(\log n) \) round 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 (worst-case) awake complexity of \( O(\log \log n) \) rounds, bypassing the \( \Omega\left(\sqrt{\frac{\log n}{\log \log n}}\right) \) traditional lower bound on the round complexity in an almost 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 \( O(\log \log n) \)-rounds awake complexity.\(^6\) This algorithm has (traditional) round complexity that is polynomial in \( n \). Our bounds hold in the \( \text{CONGEST}(O(\text{poly}(\log n))) \) model where up to \( O(\log^3 n) \) bits can be sent per edge per round.

2. We also show that we can drastically reduce the round complexity at the cost of a slight increase in awake complexity by presenting a randomized MIS algorithm with \( O(\log \log n \log^* n) \) awake complexity and \( O(\log^3 \log n \log^* n) \) round complexity in the \( \text{CONGEST}(O(\text{poly}(\log n))) \) model.

Our work answers a key question left open in [12], namely whether one can design MIS algorithms with (even) \( o(\log n) \) (worst-case) awake complexity. 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 \( O(\log n) \) in the traditional model. Since a node spends energy only in its awake rounds, our algorithms are very energy-efficient compared to the traditional algorithms.

### 1.5 Related Work

Chatterjee, Gmyr, and Pandurangan [12] showed that MIS in general graphs can be solved in \( O(1) \) rounds 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 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 [12] is whether one can design an MIS algorithm with even \( o(\log n) \) worst-case awake complexity (even in the \( \text{LOCAL} \) model).\(^7\)

Subsequently Ghaffari and Portmann [19] have developed a randomized MIS algorithm that has worst-case awake complexity of \( O(\log n) \), round complexity of \( O(\log n) \), while having \( O(1) \) node-averaged awake complexity (all bounds hold with high probability). Hourani, Pandurangan, and Robinson [20] presented randomized MIS algorithms that have \( O(\text{poly}(\log \log n)) \) rounds awake complexity for certain special classes of random graphs, including random geometric graphs (of arbitrary dimension). This algorithm works only in the \( \text{LOCAL} \) model as linear (in \( n \)) sized messages need to be sent per edge per round. This result is subsumed by the results of the current paper.

Barenboim and Maimon [7] showed that many problems, including broadcast, construction of a spanning tree, and leader election can be solved deterministically in \( O(\log n) \) awake complexity

\(^6\)Throughout, we use “with high probability (w.h.p.)” to mean probability at least \( 1 - n^{-1} \).

\(^7\)In this paper, we do not focus on the node-averaged awake complexity measure, and only focus on the (worst-case) awake complexity — see Section 7.
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. We use the DLT crucially in our algorithm. 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 \(\text{LOCAL}\) model, where \(\Delta\) is the maximum degree. (Note that, in general, this can take \(O(\log n)\) awake rounds when \(\Delta = \Theta(n)\).) Their algorithm only works in the \(\text{LOCAL}\) model (as opposed to the \(\text{CONGEST}\) model), as it needs large-sized (polynomial number of bits) messages to be sent over an edge. They also define the class of \(O\text{-LOCAL}\) problems (that includes MIS and coloring) and showed that problems in this class admit a deterministic algorithm that runs in \(O(\log \Delta)\) awake time and \(O(\Delta^2)\) round complexity. Maimon [28] presents trade-offs between awake and round complexity for \(O\text{-LOCAL}\) problems.

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

There have been other models proposed for energy-efficient distributed computation, most notably the energy complexity model of Chang, Kopelewitz, Pettie, Wang, and Zhan [11]. The energy complexity model is similar to the sleeping model in the sense that only awake rounds are counted; the number of awake rounds in an algorithm is called its energy complexity. However, the energy complexity model has some additional restrictions that pertain to radio networks. One important restriction is that nodes can only broadcast messages (hence the same message is sent to all neighbors). Also, collisions can occur at a listening node if two neighboring nodes transmit simultaneously in the same round. The energy complexity model has a few variants depending on how collisions are handled. There is a variant called the local model where collisions are ignored and nodes can transmit messages at the same time; this is quite similar to the sleeping model (apart from the notion of broadcast only in the energy model). (On the other hand, the sleeping model is a simple extension of the traditional \(\text{CONGEST}\) or \(\text{LOCAL}\) model.) Several problems such as broadcast, leader election, breadth-first search, and maximal matching have been studied in the energy model [11, 9, 10, 14].

King, Phillips, Saia, and Young [21] 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.

2 Techniques and High-level Overview

The best known distributed MIS algorithms ([32, 18, 17]) in the traditional setting suffer from a \(\log \Delta\) dependency in the round complexity, where \(\Delta\) is the maximum degree (see Section 1). Prior to this work, that was also the case in the sleeping model as well.\(^8\) Rather than improve these algorithms to remove this dependency (which appears very difficult), we instead judiciously use an algorithm with such a dependency as a building block.

To break past the barrier of \(O(\log \Delta)\), and get (essentially) an exponentially better awake complexity, our algorithms use two simple but main ideas. On a high-level, the first key idea is to implement the well-known randomized greedy MIS algorithm [13, 8, 16], a variant of Luby’s algorithm [26], in the sleeping model in an awake-efficient manner. The (sequential) randomized greedy MIS algorithm considers nodes in random order and adds them to the output set unless

\(^8\)In particular, the algorithms of [12, 19] which had optimal \(O(1)\) rounds node-averaged awake complexity, however, had \(O(\log n)\) (worst-case) awake complexity.
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) [13] with respect to that (random) ordering.\(^9\) Fischer and Noever [16] 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 was tight.

Our first main idea, which can be of independent interest, is to implement the randomized greedy MIS in the sleeping model with an awake complexity that depends on \( \Delta \) (and not on \( n \) as in the traditional model). We note that traditional distributed algorithms with \( \log \Delta \) dependency ([32, 18, 17]) compute an arbitrary MIS, and no known algorithms (even in the sleeping model) computed LFMIS faster than \( O(\log n) \) rounds. In particular, we present an algorithm (Section 4) that implements randomized greedy MIS in \( O(\log \Delta + \log \log n) \) awake complexity.

The reason we use LFMIS is because of two key properties of the randomized greedy MIS algorithm that are crucially exploited by our algorithm. The first is the residual sparsity property — stated formally in Lemma 1 in Section 3 — which shows that after processing the first \( t \) 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 \( t \) nodes) is reduced (essentially) to \( O(n/t) \). The second is the composability property. One can run the greedy MIS algorithm on the first \( t > 0 \) nodes, and then run the greedy MIS algorithm on the remaining nodes which are not neighbors of the first computed MIS. The union of the two computed MIS’s is the output of the greedy MIS on the original graph.

While there are several technical details, the main high level idea of our awake-efficient algorithms is captured in the following simple schema. The schema works in phases. These phases take advantage of the sleeping model. Indeed, each node is awake for a single (communication) round in all phases, except one in which it may be awake for more rounds. In more detail, we initially partition (e.g., by sampling) the node set \( V \) into \( t \) subsets (\( t \) is a parameter) \( V_1, \ldots, V_t \). Then, in phase \( i \in [1, t] \), all nodes use the first (communication) round to communicate and know which nodes are neighbors of \( M \), the MIS on \( G[\bigcup_{k=1}^{t-1} V_k] \). In the remaining rounds, nodes in \( V_i' = V_i \setminus N(M) \) compute the MIS \( M_i \) on \( G[V_i'] \) — such that each node in \( V_i' \) is awake for a small number of rounds only. After which, nodes in \( M_i \) are added to \( M \) and the phase ends. In fact, not all nodes need to be awake in each phase’s communication round. We use a virtual binary tree structure (see Section 3), similar to that of [7], to carefully coordinate the communication between the nodes. The virtual tree (where the same tree is locally determined by each node using the parameter \( t \)) tells each node what rounds it should be awake to coordinate computation with its neighboring nodes. Using this virtual tree, we show that a node needs to be awake in only \( O(\log t) \) well-chosen communication rounds (where \( \log t \) is the depth of the virtual tree). We believe this virtual tree coordination framework can be useful in general for designing awake-efficient algorithms in the sleeping model.

In the above schema if we set \( t \) to \( O(\Delta) \) (more precisely \( 2\Delta \)), we can obtain an awake complexity that is proportional (essentially) to \( O(\log t) = O(\log \Delta) \) if we can compute the MIS in \( V_i' \) (defined above) using a small number of awake rounds — specifically, in \( O(\log \log n) \) awake rounds as follows. We partition \( V \) into \( 2\Delta \) (i.e., set \( t = 2\Delta \) in the above schema) subsets \( V_1, \ldots, V_{2\Delta} \) (by sampling). Since each node in some \( V_i \) has, in expectation, less than \( 1/2 \) neighbors in \( V_i \), we can show through a branching process type argument — see Lemma 5 in Section 4 — that for all integers \( i \geq 1 \), \( G[V_i] \) consists of only components of \( O(\log n) \) size (with high probability). Thus, this shatters the graph into small \( O(\log n) \)-sized connected components which can hence be computed in parallel. But can we compute the LFMIS of each such component in a small number of awake rounds?

\(^9\)Given two (MIS) subsets \( X \neq Y \) of \( V \), \( X \) is lexicographically smaller (with respect to that ordering) than \( Y \) if and only if the minimum element of \( X \setminus Y \) is smaller than that of \( Y \setminus X \) or \( X \subseteq Y \).
Note that it is important for our schema (since we need the residual sparsity property as explained below) that the overall MIS is the LFMIS (with respect to an initial random ordering). To achieve this, we use the second main idea, namely a spanning tree structure called a Distributed Layered Tree (DLT) and introduced in [7], which can be used to upcast and broadcast (downcast) in $O(1)$ awake complexity. A DLT itself can be constructed in $O(\log n)$ awake complexity deterministically [7]. Hence, the LFMIS of $G[V_i]$ can actually be computed deterministically by building a DLT, upcasting the topology of an $O(\log n)$-sized graph to the root which computes the LFMIS, and then subsequently broadcasts it over the DLT. Note that since each connected component is of size $O(\log n)$, we can compute a DLT in $O(\log n)$ awake rounds deterministically and then compute the LFMIS by upcasting and broadcast in an additional $O(1)$ awake rounds.\footnote{Since each component is of size $O(\log n)$, w.h.p, the upcast and broadcast can be implemented in $O(1)$ rounds in a $CONGEST(poly(\log n))$ model, since at most $O(\log^* n)$ edges (each edge is represented by $O(\log n)$ bits) need to be sent per edge per round.} Thus, the above schema, by setting $t = 2\Delta$, implements randomized greedy MIS in $O(\log \Delta + \log \log n)$ awake rounds (see Algorithm SLEEPY-SIMPLE-GREEDY-MIS in Section 4).

To obtain an $O(\log \log n)$ awake complexity algorithm (see Algorithm SLEEPY-GREEDY-MIS in Section 5), we use the above algorithm as a subroutine and crucially exploit the residual sparsity property (which is why we insisted on computing the LFMIS). Using the residual sparsity property, we can partition $V$ into only $t = O(\log n)$ subsets while keeping the residual degree small. Indeed, notice first that in phase $i$, we compute the MIS on $G[V_i'] = G[V_i \setminus N(M)]$ rather than $G[V_i]$. Hence, we only need $G[V_i']$ to have small maximum degree, not $G[V_i]$. Moreover, the residual sparsity property states that the maximum degree of $G[V_i']$ is upper bounded by $O\left(\frac{|V_i|}{|\bigcup_{k=1}^{i-1} V_k|} \log n\right)$ with high probability. Therefore, by partitioning $V$ (by sampling) such that $|V_1| \leq 2$ and $|V_i| \leq 2|\bigcup_{k=1}^{i-1} V_k|$ for all integers $i > 1$, we upper bound the maximum degree $\Delta'$ of $G[V_i']$ by $O(\log n)$. We now use Algorithm SLEEPY-SIMPLE-GREEDY-MIS as a subroutine to compute the LFMIS of $G[V_i']$ in $O(\log \Delta' + \log \log n) = O(\log \log n)$ awake rounds. Hence, the above schema using the parameter $t = O(\log n)$ implements randomized greedy MIS in $O(\log \log n)$ awake rounds — in particular, this includes the $O(\log t) = O(\log \log n)$ rounds used to communicate according to the virtual tree framework.

One drawback of the above algorithm is that its traditional round complexity is large mainly because the construction of a DLT has a large $\tilde{O}(I^2)$ round complexity, where $I$ is the largest ID. To obtain an algorithm that also has a small round complexity, the main change we do is to replace the DLT in the above algorithm by a different spanning tree called a Labeled Distance Tree (LDT) [2]. As we show (see Appendix A), an LDT can be constructed deterministically in $O(\log n \log^* n)$ awake complexity (slightly worse than a DLT) and $O(n \log n \log^* n)$ round complexity. Crucially, the round complexity of the LDT construction depends on the number of nodes in the graph rather than the ID space. Hence, the LDT construction in the $O(\log n)$-sized components has a small $O(\log(\log n))$ round complexity at the cost of a $O(\log^* n)$ factor increase in the awake complexity. This allows us to obtain an MIS algorithm that has $O(\log n \log^* n)$ awake complexity and $O(\log(\log n))$ round complexity (see Section 6).

3 Preliminaries

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 \(v_1, \ldots, v_n\), Lemma 1 — a slight generalization of Lemma 1 in [22] — shows that after the first \(t\) nodes (according to the node ordering) are processed by the sequential randomized greedy order MIS, the maximum degree over the subgraph induced by the remaining nodes among the first \(t' > t\) nodes (those which have not been added, nor are neighbors of an already added node) has decreased (almost) linearly in \(t\). In fact, the lemma is more general. For example, it applies to distributed algorithms that compute the LFMIS over the subgraph induced by the first \(t\) nodes, according to the random node ordering mentioned above.

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

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

Let \(j \in [t+1, t']\). We apply the principle of deferred decisions [29]. 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 \(M_t\) be the LFMIS over \(G[V_i]\). Additionally, let \(U_i = (N(v_j) \cap V_t) \setminus N(M_t)\) and \(d_i = |U_i|\). Then, \(\Pr[v_i \in U_i \mid v_j, v_1, \ldots, v_{i-1}] \geq \frac{d_i}{e^{-\frac{d_i}{t-1}} - 1} \geq \frac{\varepsilon}{n}\).

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

**Virtual Binary Tree.** 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 \(B([1, i])\) has depth \(d = \lceil \log i \rceil\) and thus \(y = 2^{\lceil \log i \rceil + 1} - 1\) nodes. These nodes are labeled with integers in \([1, y]\) as follows. The root is labeled with \(\frac{1 + y}{2}\), the midpoint of \([1, y]\).[11] The rest of the tree is defined inductively, as follows. Each internal node is labeled with \(m\), the midpoint of some interval \([a, b]\). Its left and right children are respectively labeled with the midpoints of \([a, m - 1]\) and \([m, b]\). Note that the leaves are labeled with the odd integers in \([1, y]\). See Figure 1 (left).

Given \(B([1, i])\), we can define the more convenient node-labeled (full) binary tree \(B^\ast([1, i])\) as follows. The tree structure is the same, but the node labels of \(B^\ast([1, i])\) are obtained by applying \(g(x) = \lceil x/2 \rceil + 1\) to the node labels of \(B([1, i])\). Using \(B^\ast([1, i])\), 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 node labeled \(k\) in \(B^\ast([1, i])\). See Figure 1 (right).

These sets have the following property: 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 \(k < r \leq k'\). 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.

---

[11] The midpoint of an interval \([a, b]\) is defined as \(\frac{a + b}{2}\).
Figure 1: Binary tree $\mathcal{B}([1,6])$ on the left and binary tree $\mathcal{B}^*([1,6])$ on the right. $S_3([1,6])$ consists of the dotted circle and rectangle nodes’ labels and $S_5([1,6])$ of the (non-dotted) circle and rectangle nodes’ labels. Note that $5 \in S_3([1,6]) \cap S_5([1,6])$ and $3 < 5 \leq 6$.

Figure 2: Binary tree $\mathcal{B}([12,17])$ on the left and binary tree $\mathcal{B}^*([12,17])$ on the right. $S_{12}([12,17])$ consists of the dotted circle and rectangle nodes’ labels and $S_{15}([12,17])$ of the (non-dotted) circle and rectangle nodes’ labels. Note that $14 \in S_{12}([12,17]) \cap S_{15}([12,17])$ and $12 < 14 \leq 15$. 
The virtual binary tree constructions can easily be generalized to arbitrary intervals \([i, j]\), for any two integers \(i, j\). More precisely, \(B([i, j])\) and \(B^*([i, j])\) are both obtained by applying \(f(x) = x + i - 1\) to the node labels of \(B([1, i])\) and \(B^*([1, i])\). (Hence, \(B([i, j])\) and \(B^*([i, j])\) have depth \(d' = \lceil \log(j - i + 1) \rceil\) and \(y' = 2^{\log(j - i + 1)} + 1 - 1\) nodes.) In particular, the sets \(S_k([i, j])\) are obtained by applying \(f(x) = x + i - 1\) to all elements of \(S_k([1, j - i])\). See Figure 2. Clearly, the above property holds also for the sets \(S_k([i, j])\) — see Lemma 2 below.

**Observation 1.** For any positive integers \(i, j\) such that \(i \leq k \leq j\), \(|S_k([i, j])| \leq \lceil \log(j - i + 1) \rceil\).

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

**Proof.** First, let us show the lemma statement for \(i' = 1\), \(j' = j - i + 1\), \(k_1 = k - i + 1\) and \(k_2 = k' - i + 1\). Let the lowest common ancestor node in \(B^*([1, j'])\) of the leaves labeled \(k_1\) and \(k_2\) be labeled \(r'\). Then, according to the definition of a communication set, \(r'\) is in both \(S_{k_1}([1, j'])\) and \(S_{k_2}([1, j'])\). Moreover, note that the corresponding nodes in \(B([1, j'])\) are the internal (lowest common ancestor) node labeled \(2(r' - 1)\) and the leaf nodes labeled \(2k_1 - 1\) and \(2k_2 - 1\). By the inductive definition of \(B([1, j'])\), \(2k_1 - 1 < 2(r' - 1) < 2k_2 - 1\). Hence, \(k_1 < r' - 1/2 < k_2\). Since \(r'\) and \(k_2\) are integers, \(k_1 < r' \leq k_2\).

Finally, note that applying \(f(x) = x + i - 1\) to \(i', j', k_1, k_2\) results in \(i, j, k, k'\). Additionally, \(S_k([i, j])\) and \(S_{k'}([i, j])\) are respectively obtained by applying \(f(x) = x + i - 1\) to all elements of \(S_{k_1}([1, j'])\) and \(S_{k_2}([1, j'])\). Hence, \(r = r' + i - 1\) satisfies \(r \in S_k([i, j]) \cap S_{k'}([i, j])\) and \(k < r \leq k'\). Thus, the lemma statement holds.

**Distributed Layered Trees (DLT).** For any connected \(V' \subseteq V\), a distributed layered tree (introduced in [7]) is a node-labeled spanning tree over \(G[V']\). The node-labeling \(L\) satisfies \(L(v) \leq L(u)\) when \(v\) is the child of \(u\). In the distributed implementation, each node in the DLT knows its label, its parent and the parent’s label.

[7] gives a distributed DLT construction algorithm, as well as distributed upcast and broadcast algorithms (over DLTs).\(^{12}\) Their worst-case awake and round complexities are captured by the following lemma (see Lemma 2.1, Theorem 2.5 and Theorem 6.2 in [7]).

**Lemma 3 ([7]).** For any connected \(V' \subseteq V\) of at most \(n'\) nodes with unique IDs in \([1, I]\):

- A DLT over \(G[V']\) can be constructed deterministically with \(O(\log I)\) bit messages, with \(O(\log n')\) awake complexity and \(O(I^2 \log n')\) round complexity.
- **Upcast over a DLT** — in which each node \(v \in V'\) has a message of size at most \(m_v\) — can be executed deterministically with \(O(\sum_{v \in V'} m_v)\) bit messages, \(O(1)\) awake complexity and \(O(I)\) round complexity.
- **Broadcast over a DLT** — in which the DLT root \(v_r\) has a message of size at most \(m_r\) — can be executed with \(O(m_r)\) bit messages, with \(O(1)\) awake complexity and \(O(I)\) round complexity.

\(^{12}\)Upcast and broadcast over a DLT respectively consist of gathering all of the messages held by the DLT’s nodes at the root and sending the root’s message to all of the DLT’s nodes.
**Chernoff Bound.** The following Chernoff bounds [29] are used in the later sections, where the second bound is obtained by applying the inequality \( \ln(1 + \delta) \geq (2\delta)/(2 + \delta) \) to Theorem 4.4 (Inequality 1) in [29].

**Lemma 4.** Let \( X_1, \ldots, X_k \) be independent Bernoulli random variables with parameter \( p \). Then,

- For any \( 0 \leq \delta \leq 1 \), \( \Pr[\sum_{i=1}^{k} X_i \leq (1 - \delta)pk] \leq e^{-k^2p/2} \),
- For any \( \delta \geq 0 \), \( \Pr[\sum_{i=1}^{k} X_i \geq (1 + \delta)pk] \leq e^{-k^2p/2+\delta} \).

### 4 Randomized Greedy MIS in \( O(\log \Delta + \log \log n) \) Awake Rounds

Our main result — a distributed MIS algorithm with \( O(\log \log n) \) awake complexity — is presented in Section 5. Building up to that result, we first present, in this section, a distributed (Monte Carlo) randomized greedy MIS algorithm — Algorithm \textsc{Sleepy-Simple-Greedy-MIS} — with an awake complexity comparable to the best known distributed MIS algorithms, is crucial for our main result.

**Assumptions.** We assume that nodes have (uniformly) random unique IDs in \([1, N^3]\) and that the maximum degree \( \Delta \) of \( G \) is known. The first assumption can be implemented by having each node initially choose an ID in \([1, N^3]\) uniformly at random. Clearly, these IDs are unique with high probability. (To obtain a stronger guarantee, one only needs to increase the range accordingly.) Note that these IDs imply a random node ordering — that in which nodes are ordered according to increasing IDs.

**Algorithm Description.** Algorithm \textsc{Sleepy-Simple-Greedy-MIS} works in \( 2\Delta \) phases, each consisting of \( T_{\Delta} + 1 \) rounds, for some \( T_{\Delta} = O(N^6 \log \log n) \). Let \( V_1, \ldots, V_{2\Delta} \) be a partition of the node set \( V \) as follows: \( V_j \) consists of all nodes with IDs in \([\lfloor 2^{-j}N^3 \rfloor + 1, \frac{j}{2\Delta}N^3]\) for any integer \( j \in [1, 2\Delta] \). Initially, all nodes start in the “undecided” state. A node is said to be *decided* if it sets its state to either “in MIS” or “not in MIS”. We denote by \( M \) the set of nodes with state “in MIS”. For any integer \( j \in [1, 2\Delta] \), we denote by \( V_j' \) the set of undecided nodes in \( V_j \).

Informally (and from a more centralized perspective), in each phase \( j \in [1, 2\Delta] \), the first *communication round* is used to update \( V_j' \) in accordance with \( M \): that is, \( V_j' \) is updated to \( V_j' \setminus N(M) \). Moreover, to have good worst-case awake time complexity, some of the other subsets \( V_z' \) (for \( z \neq j \) and \( z \in [1, 2\Delta] \)) are also updated to \( V_z' \setminus N(M) \). The remaining \( T_{\Delta} \) rounds are used to compute the LFMIS \( M_j \) on \( G[V_j'] \). To have good worst-case awake complexity, only nodes in \( V_j' \) are awake during these \( T_{\Delta} \) rounds, and moreover, each such node can be awake at most \( T_{\Delta}' = O(\log \log n) \) of these rounds. Finally, nodes in \( M_j \) are added to \( M \) and the phase is done.

Let us now give a more precise (and distributed) description of phase \( j \in [1, 2\Delta] \). First, note that for any node \( v \in V \), there exists \( z(v) \in [1, 2\Delta] \) such that \( v \in V_{z(v)} \). Then, in the *communication round* node \( v \in V \) is awake if \( j \in S_{z(v)}([1, 2\Delta]) \), and asleep otherwise – see Figure 3 for an example, and Section 3 for the formal definition of the communication set \( S_{z(v)}([1, 2\Delta]) \).\(^{13}\)

\(^{13}\)Note that using communication sets to decide whether to stay awake or asleep in each communication round results in exponentially smaller awake-time complexity compared to simply staying awake in each communication round.

---

---
Figure 3: For the example, consider $\Delta = 3$. Then, the node set $V$ is partitioned into $V_1, \ldots, V_6$. On the one hand, nodes in $V_3$ are awake in the communication rounds of phase 3, 4 and 5. On the other hand, nodes in $V_5$ are awake in the communication rounds of phase 5 and 6 (but not in phase 7, since there are only 6 phases). It can be seen that nodes in $V_3$ communicate their state — and in particular, if their state is “in MIS” — to nodes in $V_5$ in the communication round of phase 5.

“in MIS” message from a neighboring node set their state to “not in MIS”.

For the remaining $T_\Delta$ rounds of phase $j$, nodes in $V'_j$ stay awake and all other nodes sleep. Importantly, $G[V'_j]$ is separated into small $O(\log n)$ size maximal connected components with high probability — see Lemma 5. Each node in $V'_j$ first participates in the construction of a distributed layered tree (DLT) over the maximal connected component it belongs to — see Section 3 for the definition of a DLT and how to construct it and communicate over it. Then, each node upcasts its ID and the list of its neighbors’ IDs using the DLT to the root. When the upcast terminates, the root computes which nodes are part of the LFMIS (over that maximal connected component) and broadcasts that list using the DLT. Finally, nodes in that list set their state to “in MIS”, and other nodes to “not in MIS”. (Note that $M_j$ is simply the set of nodes that set their state to “in MIS” in phase $j$.)

Additionally, a node in $V'_j$ can only be awake for (at most) $T'_\Delta + 1 = O(\log \log n)$ rounds within phase $j$. After which, that node sleeps for the rest of the phase — such a node, which sleeps because it has been awake for $T'_\Delta + 1$ rounds within that phase, is said to be bad. Although a bad node may cause an incorrect output, we show that with high probability, no node becomes bad. Indeed, since the size of the maximal connected components of $G[V'_j]$ is $O(\log n)$ at the start of phase $j$, a well-chosen $T'_\Delta$ along with Lemma 3 together imply that, with high probability, no node is bad in phase $j$.

The above description does not take message size into account. We bound the number of bits in a message by some $s = O(\log^3 n)$ as follows. Whenever the above description forces a node to send a message of more than $s$ bits, that node is also said to be bad and instead sends no message. Although this may cause an incorrect output, we show in the proof of Theorem 1 that, with high probability, this also does not happen.

**Analysis.** Lemma 5 uses a branching process type argument to show that in each phase, the awake nodes are separated into maximal connected components of size at most $O(\log n)$ with high probability. By computing the LFMIS for each such component in a “brute-force” and deterministic yet efficient manner, we obtain that throughout the algorithm, the components’ LFMIS are computed correctly and efficiently with high probability. (Note that simply executing randomized algorithms within these $O(\log n)$ size components do not imply with high probability in $n$, but just in $O(\log n)$.)
Lemma 5. For any phase \( j \in [1, 2\Delta] \), \( G[V_j] \) is composed of maximal connected components of size at most \( O(\log n) \) with high probability. More precisely, for any constant \( \varepsilon > 0 \), the maximal connected components are of size at most \( 6 \log(n/\varepsilon) \) with probability at least \( 1 - \varepsilon \).

Proof. Consider an arbitrary phase \( j \) and some node \( v \in V_j \). (If \( |V_j| = 0 \), the lemma statement obviously holds.) We consider a BFS search, over \( G[V_j] \), starting at \( v \). By the principle of deferred decisions, this BFS search is in fact a randomized process (since the order in which nodes are processed by the BFS search is random). Indeed, we can assume \( v \) is the only initially revealed node of \( V_j \) (and for all other nodes, we do not know whether they are in \( V_j \) or not). 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 \( V_j \). For each such \( w \in N(v) \), if \( w \in V_j \) 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 \( C \) similarly, but with that step’s dequeued node, until the queue is empty.

It uses \( O(\log^3 n) \) bit messages, has \( O(\log \Delta + \log \log n) \) awake complexity and \( O(\Delta N^6 \log \log n) \) round complexity.

Proof. Let \( \varepsilon > 0 \) be (an upper bound on) the failure probability. To obtain correctness with high probability, take \( \varepsilon = \frac{\Delta}{n} \). Let us first show, by induction on (phase) \( j \in [0, 2\Delta] \), the following statement. At the end of phase \( j \) (where the end of phase 0 simply denotes the start of the algorithm), \( M \) is the LFMIS (with respect to the randomly chosen IDs) over \( G[\cup_{i=1}^j V_i] \) with probability at least \( 1 - j(\varepsilon/2n) \). This statement with \( j = 2\Delta \) implies that Algorithm SLEEPY-SIMPLE- GREEDY-MIS computes the LFMIS with probability at least \( 1 - \varepsilon \).

The base case is trivially correct. Next, consider some \( j \geq 1 \), such that the induction hypothesis holds for \( j - 1 \). For any integer \( i < j \), there exists an integer \( r(i) \in S_j([1, 2\Delta]) \cap S_i([1, 2\Delta]) \) such...
that $i < r(i) \leq j$, by Lemma 2 (see Section 3). Since, for any $V_i$ such that $i < j$, nodes in $V_i$ have decided whether they are in the MIS or not prior to phase $r(i)$ (by the algorithm description), then they communicate their (decided) state to nodes in $V_j$ in the communication round of phase $r(i)$. Hence, after the communication round of phase $j$, $V'_j = V_j \setminus N(M)$. By Lemma 5, $G[V'_j]$ (and thus $G[V'_j]$) consists of maximal connected components of size at most $C_\varepsilon = 6\ln(2n^2/\varepsilon)$ with probability at least $1 - \varepsilon/2n$. We adjust $T_\Delta$ and $T'_\Delta$ to be greater than, respectively, the (worst-case) round complexity and (worst-case) awake complexity of one execution of the DLT construction, upcast and broadcast algorithms on a graph of size (at most) $C_\varepsilon$. By Lemma 3 in Section 3, we can choose $T_\Delta = O(N^6 \log C_\varepsilon)$ and $T'_\Delta = O(\log C_\varepsilon)$. (Note that the DLT construction, upcast and broadcast algorithms are deterministic.) Hence, with probability at least $1 - \varepsilon/2n$, Lemma 3 implies the following four (deterministic) steps terminate within $T_\Delta$ rounds (and such that each node is awake during at most $T'_\Delta$ rounds, or in other words, is not bad). (1) Nodes in $V'_j$ correctly compute a DLT (within each maximal connected component). (2) Nodes correctly upcast all of their information (i.e., their ID and the IDs of their neighbors) to their DLT’s root. (3) That root correctly computes (a list of) which nodes are in the LFMIS (of that component). (4) The root correctly broadcasts that information over the DLT, and nodes set their state accordingly.

Finally, remember that at the end of the communication round of phase $j$, $V'_j = V_j \setminus N(M)$. By the induction hypothesis for $j - 1$, $M$ is the LFMIS on $G[\bigcup_{i=1}^{j-1} V_i]$ with probability at least $1 - (j - 1)(\varepsilon/2n)$. Moreover, among the nodes in $V'_j$ at the end of the communication round of phase $j$, let $M_j$ be the set of those nodes which have state “in MIS” when phase $j$ ends. Since no two nodes from two different components in $G[V_j]$ (and thus $G[V'_j]$) are adjacent, and from the correctness of the four steps above, $M_j$ is the LFMIS on $G[V'_j]$ with probability at least $1 - \varepsilon/2n$.

As a result, at the end of phase $j$, $M$ is indeed the LFMIS (with respect to the randomly chosen IDs) over $G[\bigcup_{i=1}^{j} V_i]$ with probability at least $1 - (j - 1)(\varepsilon/2n) - \varepsilon/2n = 1 - j(\varepsilon/2n)$.

For the remainder of the proof, $\varepsilon = \frac{1}{n}$. Consider an arbitrary node $v \in V$. Then, $v \in V_{z(v)}$ for some $z(v) \in [1, 2\Delta]$. Node $v$ participates “fully” in phase $z(v)$ only. In all other phases, $v$ can only participate in the communication rounds. During phase $z(v)$, node $v$ is awake during at most $T'_\Delta + 1 = O(\log \log n)$ rounds (by the algorithm’s description). Additionally, since $S_{z(v)}([1, 2\Delta]) \leq \lceil \log 2\Delta \rceil$ by Observation 1, $v$ participates in at most $O(\log \Delta)$ communication rounds in the other phases $j' \neq z(v)$. Therefore, Algorithm SLEEPY-SIMPLE-GREEDY-MIS has (worst-case) awake complexity $O(\log \Delta + \log \log n)$. The round complexity is easily obtained from the algorithm’s description and the choice of $T_\Delta$ above.

Finally, we show that $s = O(C_\varepsilon^2 \log N^3) = O(\log^3 n)$ bit messages are sufficient, or more precisely, that a node cannot be bad as a result of the message size constraint in the (correct) execution considered in the correctness proof. During the communication rounds, nodes send their (“in MIS”, “not in MIS”, or “undecided”) state, which only requires $O(1)$ bits. During the DLT construction, $O(\log n)$ bit messages are sufficient by Lemma 3. Remember that the maximal connected components are of size at most $C_\varepsilon$ in the correct execution. As a result, during the upcast, each node’s message (i.e., its local information) has size $O(C_\varepsilon \log N^3)$ bits — since its local information consists of its ID and its $C_\varepsilon$ neighbors’ IDs and each ID has size $\log N^3$ bits — and thus $O(C_\varepsilon^2 \log N^3)$ bit messages are sufficient by Lemma 3. Finally, for the broadcast, the DLT root (of a component) sends a list of size $O(C_\varepsilon \log N^3)$ bits, composed of the IDs of each node in that component, and for each such ID, whether the node is in the MIS. As such, $O(C_\varepsilon \log N^3)$ bit messages are sufficient by Lemma 3.

\[\square\]

Using Algorithm SLEEPY-SIMPLE-GREEDY-MIS as a sub-procedure. We present some conditions under which Algorithm SLEEPY-SIMPLE-GREEDY-MIS can used as a subprocedure by some
Algorithm $A$ being executed by all nodes in $V$.

First, we assume nodes in $V$ choose uniformly random unique IDs in $[1, N^3]$ within Algorithm $A$ (as described at the start of Section 4) and that these IDs are reused in Algorithm SLEEPY-SIMPLE-GREEDY-MIS. In other words, when used as a subprocedure, nodes do not choose unique IDs at the start of Algorithm SLEEPY-SIMPLE-GREEDY-MIS. (Note that the subset of nodes that execute the subprocedure within Algorithm $A$ may be arbitrary, and thus may not have uniformly random IDs (when considering that subset only).)

Second, when used as a subprocedure, Algorithm SLEEPY-SIMPLE-GREEDY-MIS takes three inputs. The first is a restricted ID range $[a, b]$, for two integers $a, b \in [1, N^3]$ such that $a < b$. The second is a (restricted) maximum degree upper bound $\Delta'$. The third and last input is the set of participating nodes $V' \subseteq V$. Importantly, $V'$ must satisfy the following two conditions. Let $W$ be the set of nodes with IDs in $[1, a-1]$ and $M$ be the LFMIS over $G[W]$. First, it must hold that $V'$ is the set of nodes with IDs in $[a, b]$ and with no neighbors in $M$. Second, it must hold that the maximum degree of $G[V']$ is at most $\Delta'$. We show that when these conditions are satisfied, Algorithm SLEEPY-SIMPLE-GREEDY-MIS — in which $[1, N^3]$ and $\Delta$ are respectively replaced by $[a, b]$ and $\Delta'$ — outputs the LFMIS of $G[V']$ with high probability and has $O(\log \Delta' + \log \log n)$ awake complexity. The main difficulty comes from the fact that nodes in $V'$ reuse the IDs from Algorithm $A$. Hence, it may be the case that nodes in $V'$ do not have IDs chosen uniformly at random in $[a, b]$. In which case, Lemma 5 is not guaranteed to hold. In the proof below, we show that due to the conditions on $V'$, an analogue of Lemma 5 can be shown to hold. As a result, the proof of Theorem 1 can be straightforwardly translated to a proof of Corollary 1.

**Corollary 1.** Under the conditions defined above, Algorithm SLEEPY-SIMPLE-GREEDY-MIS computes the LFMIS on $G[V']$ with high probability. It uses $O(\log^3 n)$ bit messages, has $O(\log \Delta' + \log \log n)$ awake complexity and $O(\Delta'N^6 \log \log n)$ round complexity.

**Proof.** The main difficulty in proving the corollary lies in obtaining an analogue of Lemma 5. For that purpose, we apply the principle of deferred decisions and initially consider only nodes with (randomly chosen) IDs in $[1, a-1]$ — this set is denoted by $W$ — are fixed. Let $M$ be the LFMIS on $G[W]$. Now that $N(W)$ is fixed (since being a neighbor of $W$ does not depend on one’s ID), we reveal nodes with randomly chosen IDs in $[a, b]$. Let this set of nodes be denoted by $U$, and let $V_j$ (for some integer $j \in [1, 2\Delta']$) denote the set of nodes with IDs in the $j^{th}$ $\frac{1}{2\Delta'}$ fraction of $[a, b]$. Since we reveal $U$ after $N(W)$, nodes in $U \cap N(W)$ and nodes in $V' = U \setminus N(W)$ have equal probability of being in $V_j$. Hence, $\Pr[v \in V_j] \leq \frac{1}{2\Delta'}$ for all nodes $v \in V'$. Given this last statement, the proof of Lemma 5 can straightforwardly be converted to a proof of its analogue. \hfill \square

## 5 Randomized Greedy MIS in $O(\log \log n)$ Awake Rounds

We now present our main result: an $O(\log \log n)$ awake complexity distributed MIS algorithm. More precisely, we present a distributed (Monte Carlo) randomized greedy MIS algorithm, Algorithm SLEEPY- Greedy-MIS, that achieves an $O(\log \log n)$ awake complexity by using Algorithm SLEEPY-SIMPLE- GREEDY-MIS as a building block.

**Assumptions.** Similarly to Algorithm SLEEPY-SIMPLE- GREEDY-MIS, we assume that nodes have (uniformly) random unique IDs in $[1, N^3]$. However, we do not assume that $\Delta$, or any upper bound on $\Delta$, is known.
**Algorithm Description.** Algorithm SLEEPY-GREEDY-MIS works in $\ell = \lceil \log N^3 \rceil$ super phases, each consisting of $T_N + 1$ rounds, for some $T_N = O(N^6 \log N \log \log n)$. Let $f_0 = 0$ and $f_i = \frac{2^i}{N^3}$ for any integer $i \in [1, \ell]$. Let $V_1, \ldots, V_{\ell}$ be a partition of the node set $V$ as follows: $V_i$ consists of all nodes with IDs in $[f_{i-1}N^3 + 1, f_iN^3]$ for any integer $i \in [1, \ell]$. Initially, all nodes start in the “undecided” state. A node is said to be decided if it sets its state to either “in MIS” or “not in MIS”. We denote by $M$ the set of nodes with state “in MIS”. For any integer $i \in [1, \ell]$, we denote by $V'_i$ the set of undecided nodes in $V_i$.

At a high level, the super phases in Algorithm SLEEPY-GREEDY-MIS are similar to phases in Algorithm SLEEPY-SIMPLE-GREEDY-MIS. A super phase starts with a communication round, used to keep the sets $V'_i$ updated, and the following $T_N$ rounds are used to compute the LFMIS $M_i$ on $G[V'_i]$. Crucially, $M_i$ is the LFMIS with respect to the uniformly at random chosen IDs at the start of Algorithm SLEEPY-GREEDY-MIS (rather than with respect to some IDs chosen in Algorithm SLEEPY-SIMPLE-GREEDY-MIS).

In more detail, consider super phase $i \in [1, \ell]$. First, note that for any node $v \in V$, there exists $z(v) \in [1, \ell]$ such that $v \in V_{z(v)}$. Then, in the communication round, node $v \in V$ is awake if $j \in S_{z(v)}([1, \ell])$, and asleep otherwise — see Section 3 for the definition of the communication set $S_{z(v)}([1, \ell])$. For the remaining $T_N$ rounds of super phase $i$, nodes in $V'_i$ stay awake and all other nodes sleep. During these $T_N$ rounds, nodes in $V'_i$ execute Algorithm SLEEPY-SIMPLE-GREEDY-MIS with restricted ID range $[f_{i-1}N^3 + 1, f_iN^3]$ and restricted maximum degree upper bound $\Delta'$ — where $\Delta' = O(\log n)$ is an upper bound on the maximum degree of $G[V'_i]$, see Lemma 7. Hence, by Corollary 1, nodes in $V'_i$ compute the LFMIS $M_i$ on $G[V'_i]$ (with respect to the uniformly at random chosen IDs at the start of Algorithm SLEEPY-GREEDY-MIS).

**Analysis.** The first two lemmas allow us to prove the correctness of Algorithm SLEEPY-GREEDY-MIS in Theorem 2.

**Lemma 6.** Let $0 < \varepsilon < 1$ be some constant and $\chi = \lfloor \ell - \log n + \log(10\ln(2/\varepsilon)) \rfloor$. Then:

- For any integer $i \in [1, \chi + 1]$, $|\bigcup_{k=1}^{j} V_k| \leq 120 \ln(2/\varepsilon)$ with probability at least $1 - \varepsilon$.
- For any integer $i \in [\chi + 1, \ell]$, $1/2f_in \leq |\bigcup_{k=1}^{j} V_k| \leq 3/2f_in$ with probability at least $1 - \varepsilon$.

**Proof.** Let $i \in [1, \ell]$. By definition, $\bigcup_{k=1}^{j} V_k$ consists of nodes with IDs in the first $f_i$ fraction of $[1, N^3]$. Moreover, remember that nodes choose IDs independently and uniformly at random in $[1, N^3]$. Hence, for any node $v \in V$, $\Pr[v \in \bigcup_{k=1}^{j} V_k] = f_i$. In other words, the indicator random variable $1_v$ (for some node $v \in V$), which is 1 if and only if $v \in \bigcup_{k=1}^{j} V_k$, is a Bernoulli random variable with parameter $f_i$. Moreover, the random variables $(1_v)_{v \in V}$ are independent. Finally, the linearity of expectation implies that $E[\sum_{v \in V} 1_v] = f_i n = \frac{2^i}{N^3} n$.

First, consider $i \in [1, \chi + 1]$. Since $\chi + 1 \leq \log \frac{N^3}{n} + \log(10\ln(2/\varepsilon)) + 2$, $E[\sum_{v \in V} 1_v] \leq 40 \ln(2/\varepsilon)$. We apply the (upper tail) Chernoff bound (see Lemma 4): $\Pr[\sum_{v \in V} 1_v \geq (1+\delta)f_in] \leq \exp\left(-\frac{\delta^2f_in}{2\cdot 3}\right)$ for any $\delta \geq 0$. Let $\delta = \frac{120 \ln(2/\varepsilon)}{f_in} - 1$, such that $(1 + \delta)f_in = 120 \ln(2/\varepsilon)$. Note that since
\[ f_in \leq 40\ln(2/\varepsilon), \delta \geq 0. \text{ Then,} \]

\[
\Pr \left[ \sum_{v \in V} 1_v \geq 120 \ln(2/\varepsilon) \right] \leq \exp \left( -\frac{(120 \ln(2/\varepsilon))}{f_in} - 1 \right)^2 f_in \]

\[
\leq \exp \left( -\frac{(3 \cdot 40 \ln(2/\varepsilon) - f_in)^2}{3 \cdot 40 \ln(2/\varepsilon) + f_in} \right) \]

\[
\leq \exp \left( -\frac{(2 \cdot 40 \ln(2/\varepsilon))^2}{4 \cdot 40 \ln(2/\varepsilon)} \right) \quad \text{since } f_in \leq 40\ln(2/\varepsilon)
\]

\[
\leq e^{-40\ln(2/\varepsilon)}
\]

To sum up, \( |\cup_{k=1}^i V_k| = \sum_{v \in V} 1_v \leq 120 \ln(2/\varepsilon) \) with probability at least \( 1 - \varepsilon \).

Now, consider \( i \in [x + 1, \ell] \). We apply the Chernoff bound (for the two tails, see Lemma 4) with \( \delta = 1/2 \): \( \Pr[\sum_{v \in V} 1_v \geq (3/2)f_in] \leq \exp \left( -\frac{f_in}{10} \right) \) and \( \Pr[\sum_{v \in V} 1_v \leq (1/2)f_in] \leq \exp \left( -\frac{f_in}{8} \right) \).

Since \( \mathbb{E}[\sum_{v \in V} 1_v] = f_in \geq 2^{x+1}n \geq 10\ln(2/\varepsilon) \), we obtain \( (1/2)f_in \leq |\cup_{k=1}^i V_k| \leq (3/2)f_in \) with probability at least \( 1 - \varepsilon/2 - \varepsilon/2 = 1 - \varepsilon \). \( \square \)

**Lemma 7.** For any constant \( 0 < \varepsilon < 1 \), with probability at least \( 1 - \varepsilon \), the following two statements hold for any super phase \( i \in [1, \ell] \):

- **At the start of super phase** \( i \), \( G[V_i'] \) has maximum degree at most \( \Delta' = 120 \ln(12n/\varepsilon) \).
- **At the end of super phase** \( i \), \( M \) is the LFMIS of \( G[\cup_{k=1}^i V_k] \) with respect to the randomly chosen IDs.

**Proof.** Let us show, by induction on the super phase \( i \in [1, \ell] \), that the lemma statement for \( i \) holds with probability at least \( 1 - i\varepsilon/n \).

For the base case of \( i = 1 \), note that \( V_1' = V_1 \) at the start of the first super phase. By Lemma 6 (for failure probability \( \varepsilon/(6n) \)), \( |V_1| \leq 120 \ln(12n/\varepsilon) \) with probability at least \( 1 - \varepsilon/(6n) \). Consequently, \( G[V_1'] \) has maximum degree at most \( \Delta' \) with probability at least \( 1 - \varepsilon/(6n) \). We adjust \( T_N = O(N^6 \log n \log \log(n/\varepsilon)) \) to be greater than the (worst-case) round complexity of Algorithm SLEEPY-SIMPLE-GREEDY-MIS with failure probability \( \varepsilon/(2n) \) and with inputs \( [1, f_1 N^3] \) and \( \Delta' \). Hence, by Corollary 1, Algorithm SLEEPY-SIMPLE-GREEDY-MIS terminates within the first super phase and its output \( M_1 \) is the LFMIS of \( G[V_1'] \) with probability at least \( 1 - \varepsilon/n \). Since all nodes in \( M_1 \) are in \( M \) by the end of super phase 1, \( M \) is the LFMIS of \( G[V_1] \) at the end of phase 1 with probability at least \( 1 - \varepsilon/n \), and thus the base case holds.

Next, consider the induction step for some integer \( i > 1 \). By the induction hypothesis for \( i - 1 \), \( M \) is the LFMIS of \( G[\cup_{k=1}^{i-1} V_k] \) (with respect to the randomly chosen IDs at the start of Algorithm SLEEPY-GREEDY-MIS) at the start of super phase \( i \), with probability at least \( 1 - (i - 1)\varepsilon/n \).

Let \( \chi = [\ell - \log n + \log(10 \ln(12n/\varepsilon))] \). (For this paragraph, we assume \( M \) is the LFMIS of \( G[\cup_{k=1}^{i-1} V_k] \) at the start of super phase \( i \).) If \( i \leq \chi + 1 \) then by Lemma 6 (for failure probability \( \varepsilon/(6n) \)), \( |\cup_{k=1}^i V_k| \leq 120 \ln(12n/\varepsilon) \) with probability at least \( 1 - \varepsilon/(6n) \). Consequently, \( G[V_i'] \) has maximum degree at most \( \Delta' \) with probability at least \( 1 - \varepsilon/(6n) \). Otherwise, if \( i > \chi + 1 \), then by Lemma 6 (for failure probability \( \varepsilon/(6n) \)), \( |\cup_{k=1}^i V_k| \leq (3/2)f_in \) and \( |\cup_{k=1}^{i-1} V_k| \geq (1/2)f_{i-1}n \) with probability at least \( 1 - 2\varepsilon/(6n) \). Moreover, by Lemma 1 (for failure probability \( \varepsilon/(6n) \)), the start of phase \( i \), \( G[V_i'] \) has maximum degree upper bounded by \( \frac{|\cup_{k=1}^i V_k|}{|\cup_{k=1}^{i-1} V_k|} \ln(6n^2/\varepsilon) \) with probability at least \( 1 - \varepsilon/(6n) \) by Lemma 1. Hence, \( G[V_i'] \) has maximum degree upper bounded by \( 6 \ln(6n^2/\varepsilon) \leq 12 \ln(6n/\varepsilon) \leq \Delta' \) with probability at least \( 1 - 2\varepsilon/(6n) - \varepsilon/(6n) \geq 1 - \varepsilon/(2n) \).
To sum up, at the start of super phase \( i \), with probability at least \( 1 - (i - 1)\varepsilon/n - \varepsilon/(2n) \), \( M \) is the LFMIS of \( G[\cup_{k=1}^{i-1} V_k] \) and \( G[V'] \) has maximum degree upper bounded by \( \Delta' \). We adjust \( T_N = O(N^6 \log n \log \log (n/\varepsilon)) \) to be greater than the (worst-case) round complexity of Algorithm SLEEPY-SIMPLE-GREEDY-MIS with failure probability \( \varepsilon/(2n) \) and with inputs \( [f_i N^3 + 1, f_i N^3] \) and \( \Delta' \). Hence, by Corollary 1, Algorithm SLEEPY-SIMPLE-GREEDY-MIS terminates within super phase \( i \) and its output \( M_i \) is the LFMIS of \( G[V'_i] \) with probability at least \( 1 - \varepsilon/n \). Since all nodes in \( M_i \) are in \( M \) by the end of super phase \( i \), \( M \) is the LFMIS of \( G[\cup_{k=1}^{i} V_k] \) at the end of super phase \( i \) with probability at least \( 1 - \varepsilon/n \), and thus the induction step holds.

**Theorem 2.** Algorithm SLEEPY-GREEDY-MIS computes the LFMIS with high probability. It uses \( O(\log^3 n) \) bit messages, has \( O(\log \log n) \) awake complexity and \( \tilde{O}(N^6) \) round complexity.

**Proof.** The correctness (with high probability) of Algorithm SLEEPY-GREEDY-MIS follows from Lemma 7 with failure probability \( \varepsilon = \frac{1}{n} \).

Consider an arbitrary node \( v \in V \). Then, \( v \in V_z(v) \) for some \( z(v) \in [1, \ell] \). Node \( v \) participates “fully” in super phase \( z(v) \) only. In all other super phases, \( v \) can only participate in the communication rounds. During super phase \( z(v) \), \( v \) executes Algorithm SLEEPY-SIMPLE-GREEDY-MIS with restricted maximum degree upper bound \( \Delta' = O(\log n) \). Thus, \( v \) is awake during at most \( O(\log \log n) \) rounds during super phase \( z(v) \), by Corollary 1. Additionally, since \( S_{z(v)}([1, \ell]) = O(\log \log n) \) by Observation 1, \( v \) participates in at most \( O(\log \log n) \) communication rounds in the other super phases \( \ell' \neq z(v) \). Hence, Algorithm SLEEPY-GREEDY-MIS has (worst-case) awake complexity \( O(\log \log n) \). The round complexity is easily obtained from the description of Algorithm SLEEPY-GREEDY-MIS and Corollary 1. Finally, since communication rounds require \( O(1) \) bit messages, then by Corollary 1, \( O(\log^3 n) \) bit messages are sufficient.

### 6 Improving the Round Complexity

The algorithm, as presented in Section 5, has a large round complexity. At the crux of this is an inherent limit when using DLTs, namely that the size of the schedules is proportional to the range from which IDs are taken, which can be much larger than the number of nodes over which we want to construct and use the DLT. There exists an alternative structure from [2], called Labeled Distance Trees (LDTs), which allows us to maintain a relatively small awake complexity while also using schedules whose size is proportional to a known upper bound on the number of nodes over which we want to construct and use the LDT.

In this section, we first describe an LDT along with a few useful procedures, and subsequently explain how to use an LDT to improve the round complexity of the algorithm in the previous section.

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 LDT. 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).

Once an LDT is constructed over a graph, it can be used to quickly broadcast and upcast information over the graph, assuming that there is sufficient bandwidth on the edges. More details about the structure and how to construct it can be found in Appendix A. The following lemma captures the relevant properties of an LDT.
Lemma 8. 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.

- Upcast over an LDT — in which each node $v \in V'$ has a message of size at most $m_v$ — can be executed deterministically with $O(\sum_{v \in V'} m_v)$ bit messages, $O(1)$ awake complexity and $O(n')$ 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, $O(1)$ awake complexity and $O(n')$ round complexity.

By constructing an LDT instead of a DLT in Algorithm Sleepy-Simple-Greedy-MIS, and subsequently upcasting then broadcasting over that LDT, the round complexity of Algorithm Sleepy-Simple-Greedy-MIS can be greatly improved at the cost of a small $O(\log^* n)$ factor in its awake complexity.$^{14}$

Theorem 3. Algorithm Sleepy-Simple-Greedy-MIS computes the LFMIS with high probability. It uses $O(\log^3 n)$ bit messages, has $O(\log \Delta + \log \log n \log^* n)$ awake complexity and $O(\Delta \log n \log \log n \log^* n)$ round complexity.

Proof. Let $\varepsilon$ be the failure probability of Algorithm Sleepy-Simple-Greedy-MIS and let $C_\varepsilon = 6 \ln(2n^2/\varepsilon)$. We adjust $T_\Delta$ and $T'_\Delta$ to be greater than, respectively, the (worst-case) round complexity and (worst-case) awake complexity of one execution of the LDT construction, upcast and broadcast algorithms (over that LDT) on a graph of size (at most) $C_\varepsilon$. By Lemma 8, we can choose $T_\Delta = O(C_\varepsilon \log C_\varepsilon \log^* N)$ and $T'_\Delta = O(\log C_\varepsilon \log^* N)$. Given this choice of $T_\Delta$ and $T'_\Delta$, the theorem statement’s proof follows that of Theorem 1.

When using this variant of Algorithm Sleepy-Simple-Greedy-MIS as the building block in Algorithm Sleepy-Greedy-MIS, it is straightforward to see that the following theorem holds.

Theorem 4. Algorithm Sleepy-Greedy-MIS computes the LFMIS with high probability. It uses $O(\log^3 n)$ bit messages, has $O(\log \log n \log^* n)$ awake complexity and $O(\log^3 n \log \log n \log^* n)$ round complexity.

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 in the sleeping model. A long-standing open question is whether a similar bound can be shown in the traditional model.

Several open problems remain. 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 important one is whether one can obtain a $O(\log \log n)$ awake complexity MIS algorithm that has $O(\log n)$ round complexity. More generally, can one obtain good trade-offs between awake and

$^{14}$Remember that $n$ is the number of nodes of the communication graph (i.e., $|V| = n$) and $N$ is a polynomial upper bound on $n$. Hence $\log^* N = O(\log^* n)$. 

19
round complexity of MIS? Furthermore, one can try to reduce the node-averaged awake complexity as well (which was not the focus of this paper).

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

[1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986.

[2] John Augustine, William K. Moses Jr., and Gopal Pandurangan. Distributed MST computation in the sleeping model: Wake-optimal algorithms and lower bounds, 2022. Manuscript.

[3] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikael Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *IEEE FOCS*, pages 481–497, November 2019.

[4] Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. 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] Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. *Distributed Comput.*, 22(5-6):363–379, 2010.

[6] Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *Journal of the ACM*, 63(3), June 2016. Conference version: *IEEE FOCS* 2012.

[7] Leonid Barenboim and Tzalik Maimon. 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] Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. *SPAA ’12*, pages 308–317, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2312005.2312058.

[9] Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, Qizheng He, Wenzheng Li, and Seth Pettie. The energy complexity of broadcast. In *ACM PODC*, pages 95–104, 2018.

[10] Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, and Seth Pettie. The energy complexity of BFS in radio networks. In *ACM PODC*, pages 273–282, 2020.

[11] Yi-Jun Chang, Tsvi Kopelowitz, Seth Pettie, Ruosong Wang, and Wei Zhan. Exponential separations in the energy complexity of leader election. *ACM Trans. Algorithms*, 15(4):49:1–49:31, 2019. Conference version: *ACM STOC* 2017.

[12] Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan. Sleeping is efficient: MIS in $O(1)$-rounds node-averaged awake complexity. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 99–108, 2020.

[13] Don Coppersmith, Prabhakar Raghavan, and Martin Tompa. Parallel graph algorithms that are efficient on average. *Information and Computation*, 81(3):318–333, 1989.

[14] Varsha Dani, Aayush Gupta, Thomas P. Hayes, and Seth Pettie. 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.
[15] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, volume 3, pages 1548–1557, 2001.

[16] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy MIS. In *SODA*, pages 2152–2160, 2018.

[17] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *SODA*, pages 270–277, 2016.

[18] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhon. Improved deterministic network decomposition. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2904–2923. SIAM, 2021.

[19] Mohsen Ghaffari and Julian Portmann. Average awake complexity of MIS and matching, 2021. Manuscript.

[20] Khalid Hourani, Gopal Pandurangan, and Peter Robinson. Awake-efficient distributed algorithms for maximal independent set, 2021. Manuscript.

[21] Valerie King, Cynthia A. Phillips, Jared Saia, and Maxwell Young. Sleeping on the job: Energy-efficient and robust broadcast for radio networks. *Algorithmica*, 61(3):518–554, 2011.

[22] Christian Konrad. MIS in the congested clique model in $O(\log \log \Delta)$ rounds. *arXiv preprint arXiv:1802.07647*, 2018.

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

[24] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM*, 63(2), 2016.

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

[26] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. Conference version: *ACM STOC 1985*.

[27] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[28] Tzalik Maimon. Sleeping model: Local and dynamic algorithms, 2021. *arXiv:2112.05344*.

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

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

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

[32] Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 350–363, 2020.
[33] Dorothea Wagner and Roger Wattenhofer, editors. *Algorithms for Sensor and Ad Hoc Networks*, volume 4621 of *Lecture Notes in Computer Science*. Springer, 2007.

[34] Qin Wang, Mark Hempstead, and Woodward Yang. 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.

[35] Roger Wattenhofer. Algorithms for ad hoc and sensor networks. *Comput. Commun.*, 28(13):1498–1504, 2005.

[36] Ou Yang and Wendi Heinzelman. An adaptive sensor sleeping solution based on sleeping multipath routing and duty-cycled mac protocols. *ACM Transactions on Sensor Networks*, 10(1), 2013.

[37] Rong Zheng and Robin Kravets. On-demand power management for ad hoc networks. *Ad Hoc Networks*, 3(1):51–68, 2005.
A 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 Section A.1. Subsequently, we give an informal description on how one might construct an LDT in Section A.2. Finally, in Section A.3 we give a few useful procedures that can be run once an LDT is constructed.

A.1 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, $\text{root}$, of the LDT it belongs to, $u$’s distance to $\text{root}$ 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 $\text{root}$ 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 $n$. The transmission schedule $\text{Transmission-Schedule(root, } u, n)$ 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 $\text{Transmission-Schedule(root, } u)$ 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, and Up-Send, respectively. $\text{Transmission-Schedule(root, root, } n)$ assigns to $\text{root}$ the set containing only the rounds $1$, $n + 1$, and $2n + 1$ with names Down-Send, Side-Send-Receive, and Up-Receive, respectively.

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 $\text{Transmission-Schedule(root, } u, n)$, 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.

---

\[15\] We assumed that $\text{Transmission-Schedule(·, ·, } n)$ was started in round 1 when assigning rounds. If $\text{Transmission-Schedule(·, ·, } n)$ is started in round $r$, then the correct round numbers can be obtained by adding $r - 1$ to the values mentioned in the description.
Broadcasting a message in an LDT. Procedure \textsc{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 \textsc{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 \textsc{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.

A.2 Construction of an LDT

In this section, we describe how to deterministically construct an LDT. 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. \textbf{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 \textsc{Upcast-Min}(n) to to collect the ID of “minimum” outgoing edge at the root of the LDT.

   (b) Each LDT runs procedure \textsc{Fragment-Broadcast}(n) to let all nodes in the LDT know the ID of the chosen outgoing edge.

   (c) Each LDT \( L \) runs \textsc{Transmit-Adjacent}(n) to let nodes from other LDTs know if they are part of the chosen outgoing edge from \( L \).

2. \textbf{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 \( C_1, C_2, \ldots, C_p \). Our final goal is to decompose the connected components in \( H \) in to a forest \( F \).
of small-depth trees $SDT_1$, $SDT_2$, \ldots, $SDT_k$.\textsuperscript{16} 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 $\text{Upcast-Min}(n)$ and $\text{Fragment-Broadcast}(n)$ 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 $\text{Upcast-Min}(n)$, $\text{Fragment-Broadcast}(n)$, and $\text{Transmit-Adjacent}(n)$.

(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 $\text{Upcast-Min}(n)$ 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 $\text{Upcast-Min}(n)$, $\text{Fragment-Broadcast}(n)$, and $\text{Transmit-Adjacent}(n)$.

(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 $\text{Upcast-Min}(n)$, $\text{Fragment-Broadcast}(n)$, and $\text{Transmit-Adjacent}(n)$.

\textsuperscript{16}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 $H$, $p$. 

26
(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 $\text{Upcast-Min}(n)$, $\text{Fragment-Broadcast}(n)$, and $\text{Transmit-Adjacent}(n)$.

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 most distance 4 apart).\(^{17}\) 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 propagated to every LDT in $SDT_i$ through a constant number of instances of $\text{Upcast-Min}(n)$, $\text{Fragment-Broadcast}(n)$, and $\text{Transmit-Adjacent}(n)$.

(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 $\text{Up-Send}$ and $\text{Up-Receive}$ rounds. In the second instance, the remaining nodes in $L'$ update their values in the $\text{Down-Send}$ and $\text{Down-Receive}$ 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

\(^{17}\)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.
fraction of the LDTs “disappearing” in the phase (but really just being merged into one another).\(^\text{18}\) 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 9.** There exists a deterministic algorithm in the CONGEST setting to construct an LDT over a given graph in \(O(\log n \log^* n)\) awake complexity and \(O(n \log n \log^* n)\) round complexity.

### A.3 Useful Procedures

Once an LDT is constructed, we can then leverage it to run fast awake complexity procedures. In the main paper, we allow more bandwidth than just \(O(\log n)\) bits per message. As such, we describe the following two procedures while also explicitly mentioning the bandwidth requirements.

**Broadcasting in an LDT.** Consider an LDT of size at most \(n'\), where the root of the LDT wants to broadcast a message of size at most \(m_v\) bits to all nodes in the LDT. Assume that messages of size \(O(m_v)\) 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.

**Upcasting in an LDT.** Consider an LDT consisting of \(V'\) nodes of size at most \(n'\) (i.e., \(n' = |V'|\)), where each node \(v\) of the LDT has a message of size \(m_v\) bits that needs to be sent to the root of the LDT. Assume that messages of size \(O(\sum_{v \in V'} m_v)\) bits can be sent over each edge. Upcast can be performed by having all nodes in the LDT participate in one instance of a transmission schedule parameterized by \(n'\), where each node listens for messages in its Up-Receive round, composes these messages along with its own into a larger message and then sends it to its parent during its Up-Send round. This 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 10.** 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.

- Upcast over an LDT — in which each node \(v \in V'\) has a message of size at most \(m_v\) — can be executed deterministically with \(O(\sum_{v \in V'} m_v)\) bit messages, \(O(1)\) awake complexity and \(O(n')\) round complexity.

\(^{18}\)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 it 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 \(T_i\)).
• 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.