Awake Complexity of Distributed Minimum Spanning Tree

John Augustine
Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India

William K. Moses Jr.
Department of Computer Science, Durham University, Durham, UK

Gopal Pandurangan
Department of Computer Science, University of Houston, Houston, TX, USA

Abstract

The *awake complexity* of a distributed algorithm measures the number of rounds in which a node is awake. When a node is not awake, it is *sleeping* and does not do any computation or communication and spends very little resources. Reducing the awake complexity of a distributed algorithm can be relevant in resource-constrained networks such as sensor networks, where saving energy of nodes is crucial. Awake complexity of many fundamental problems such as maximal independent set, maximal matching, coloring, and spanning trees have been studied recently.

In this work, we study the awake complexity of the fundamental distributed minimum spanning tree (MST) problem and present the following results.

1. Lower Bounds.

1. We show a lower bound of $\Omega(\log n)$ (where n is the number of nodes in the network) on the awake complexity for computing an MST that holds even for randomized algorithms.
2. To better understand the relationship between the awake complexity and the (traditional) round complexity (which counts both awake and sleeping rounds), we also prove a *trade-off* lower bound of $\tilde{\Omega}(n)$[1] on the product of round complexity and awake complexity for any distributed algorithm (even randomized) that outputs an MST.

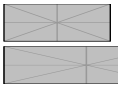
2. Awake-Optimal Algorithms.

1. We present a distributed randomized algorithm to find an MST that achieves the optimal awake complexity of $O(\log n)$ (with high probability). Its round complexity is $O(n \log n)$ and by our trade-off lower bound, this is the best round complexity (up to logarithmic factors) for an awake-optimal algorithm.
2. We also show that the $O(\log n)$ awake complexity bound can be achieved deterministically as well, by presenting a distributed *deterministic* algorithm that has $O(\log n)$ awake complexity and $O(nN \log n)$ round complexity, where N is an upper bound on the value of the maximum ID.

3. Trade-Off Algorithms. To complement our trade-off lower bound, we present a parameterized family of distributed algorithms that gives an essentially optimal trade-off (up to polylog n factors) between the awake complexity and the round complexity. Specifically we show a family of distributed algorithms that find an MST of the given graph with high probability in $\tilde{O}(D+2^k+n/2^k)$ round complexity and $\tilde{O}(n/2^k)$ awake complexity, where D is the network diameter and integer k is an input parameter to the algorithm. When $k \in [\max\{\lceil 0.5 \log n \rceil, \lceil \log D \rceil\}, \lceil \log n \rceil]$, we can obtain useful trade-offs.

Our work is a step towards understanding resource-efficient distributed algorithms for fundamental global problems such as MST. It shows that MST can be computed with any node being awake (and hence spending resources) for only $O(\log n)$ rounds which is significantly better than the fundamental lower bound of $\tilde{\Omega}(\text{Diameter}(G) + \sqrt{n})$ rounds for MST in the traditional CONGEST

[1] Throughout, the $\tilde{O}$ notation hides a polylog n factor and $\tilde{\Omega}$ hides a $1/(\text{polylog } n)$ factor.



© John Augustine, William K. Moses Jr., and Gopal Pandurangan;
licensed under Creative Commons License CC-BY 4.0

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

model (bandwidth per edge per round is limited to $O(\log n)$ bits), where nodes can be active for at least so many rounds.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Discrete mathematics

Keywords and phrases Minimum Spanning Tree, Sleeping model, energy-efficient, awake complexity, round complexity, trade-offs

Digital Object Identifier 10.4230/LIPIcs...

Funding *John Augustine*: J. Augustine was supported, in part, by DST/SERB MATRICS Grant MTR/2018/001198 and the Centre of Excellence in Cryptography Cybersecurity and Distributed Trust under the IIT Madras Institute of Eminence Scheme and by the VAJRA visiting faculty program of the Government of India.
William K. Moses Jr.: W. K. Moses Jr. was supported, in part, by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419.
Gopal Pandurangan: G. Pandurangan was supported, in part, by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075 and BSF grant 2016419 and by the VAJRA visiting faculty program of the Government of India.

1 Introduction

We study the distributed minimum spanning tree (MST) problem, a central problem in distributed computing. This problem has been studied extensively for several decades starting with the seminal work of Gallagher, Humblet, and Spira (GHS) in the early 1980s [14]; for example, we refer to the survey of [29] that traces the history of the problem till the state of the art. The round (time) complexity of the GHS algorithm is $O(n \log n)$ rounds, where n is the number of nodes in the network. The round complexity of the problem has been continuously improved since then and now tight optimal bounds are known.[2] It is now well-established that $\Theta(D + \sqrt{n})$ is essentially (up to logarithmic factor) a tight bound for the round complexity of distributed MST in the standard CONGEST model [31, 12, 24, 10]. The lower bound applies even to randomized Monte Carlo algorithms [10], while deterministic algorithms that match this bound (up to logarithmic factor) are now well-known (see e.g., [30, 27, 24, 28, 13]). Thus, the round complexity of the problem in the traditional CONGEST distributed model is settled (see also the recent works of [18, 16]). In the CONGEST model, any node can send, receive, or do local computation in any round and only $O(\log n)$-sized messages can be sent through any edge per round.

An MST serves as a basic primitive in many network applications including efficient broadcast [29]. For example, MST is useful for energy-efficient broadcast in wireless networks and has been extensively studied in this context, see e.g., [1, 21]. In resource-constrained networks such as sensor networks, where nodes spend a lot of energy or other resources over the course of an algorithm, a round complexity of $\tilde{O}(D + \sqrt{n})$ (which is essentially optimal) to construct an MST can be large. In particular, in such an algorithm, a node can be active over the entire course of the algorithm. It is worth studying whether MST can be constructed in such a way that each node is active only in a small number of rounds — much smaller than that taken over the (worst-case) number of rounds — and hence could use much less resources such as energy.

[2] Message complexity has also been well-studied, see e.g., [29], but this is not the focus of this paper, although our algorithms are also (essentially) message optimal — see Section 1.2.

Motivated by the above considerations, in recent years several works have studied energy-efficient distributed algorithms (see e.g., [5, 3, 4, 6, 2, 22]). This paper studies the distributed MST problem in the *sleeping model* [6]. In this model (see Section 1.1), nodes can operate in two modes: *awake* and *sleeping*. Each node can choose to enter the awake or asleep state at the start of any specified round. In the sleeping mode, a node cannot send, receive, or do any local computation; messages sent to it are also lost. The resources utilized in sleeping rounds are negligible and hence only awake rounds are counted. The goal in the sleeping model is to design distributed algorithms that solve problems in a small number of awake rounds, i.e., have small *awake complexity* (also called *awake time*), which is the (worst-case) number of awake rounds needed by any node until it terminates. This is motivated by the fact that, if the awake complexity is small, then every node takes only a small number of rounds during which it uses a significant amount of resources. For example, in ad hoc wireless or sensor networks, a node's energy consumption depends on the amount of time it is actively communicating with nodes. In fact, significant amount of energy is spent by a node even when it is just waiting to hear from a neighbor [6]. On the other hand, when a node is sleeping — when all of its radio devices are switched off — it spends little or no energy. While the main goal is to minimize awake complexity, we would also like to minimize the (traditional) *round complexity* (also called *time complexity or run time*) of the algorithm, which counts the (worst-case) total number of rounds taken by any node, including both awake and sleeping rounds.

The work of Barenboim and Maimon [2] shows that global problems such as broadcast and constructing a (arbitrary) spanning tree (but not an MST) can be accomplished in $O(\log n)$ awake rounds in the sleeping (CONGEST) model (see also the related result of [3] — Section 1.3.). This is significant because it shows that even such global problems can be accomplished in a very small number of awake rounds, bypassing the $\Omega(D)$ lower bound on the round complexity (in the traditional model). In this work, we focus on another fundamental global problem, namely MST, and study its awake complexity. We show that MST can be solved in $O(\log n)$ rounds awake complexity which we show is also *optimal*, by showing a matching *lower bound*. The upper bound is in contrast to the classical $\tilde{\Omega}(D + \sqrt{n})$ lower bound on the round complexity (in the traditional CONGEST model) where nodes can be awake for at least so many rounds. Another key issue we study is the relationship between awake complexity and round complexity of MST. It is intriguing whether one can obtain an algorithm that has both optimal awake and round complexity. We answer this in the negative by showing a *trade-off* lower bound (see Section 1.2).

1.1 Distributed Computing Model and Complexity Measures

Distributed Network Model. We are given a distributed network modeled as an arbitrary, undirected, connected, weighted graph $G(V, E, w)$, where the node set V ($|V| = n$) represent the processors, the edge set E ($|E| = m$) represents the communication links between them, and $w(e)$ is the weight of edge $e \in E$. The network diameter is denoted by D, also called as the hop-diameter (that is, the unweighted diameter) of G, and in this paper by diameter we always mean hop-diameter. We assume that the weights of the edges of the graph are all distinct. This implies that the MST of the graph is unique.

Each node hosts a processor with limited initial knowledge. We assume that nodes have unique `IDs` (of size $O(\log n)$ bits), and at the beginning of the computation each node is provided its `ID` as input and the weights of the edges incident to it. We also assume that nodes know n, the number of nodes in the network. For the deterministic algorithm, we make the additional assumption that nodes know the value of N, an upper bound on the

largest ID of any node. Nodes initially do not have any other global knowledge and have knowledge of only themselves.

Nodes communicate through the edges of the graph G and it is assumed that communication is *synchronous* and occurs in rounds. In particular, we assume that each node knows the current round number, starting from round 1. In each round, each node can perform some local computation (which happens instantaneously) including accessing a private source of randomness, and can exchange (possibly distinct) $O(\log n)$-bit messages with each of its neighboring nodes. This is the traditional *synchronous CONGEST* model.

As is standard, the goal, at the end of the distributed MST computation, is for every node to know which of its incident edges belong to the MST.

Sleeping Model and Complexity Measures. The *sleeping model* [6] is a generalization of the traditional model, where a node can be in either of the two states — sleeping or awake — before it finishes executing the algorithm (locally). (In the traditional model, each node is always awake till it finished the algorithm). Initially, we assume that all nodes are awake. That is, any node v, can decide to *sleep* starting at any (specified) round of its choice; we assume all nodes know the correct round number whenever they are awake. It can *wake up* again later at any specified round and enter the *awake* state. In the sleeping state, a node does not send or receive messages, nor does it do any local computation. Messages sent to it by other nodes when it was sleeping are lost. This aspect makes it especially challenging to design algorithms that have a small number of awake rounds, since one has to carefully coordinate the transmission of messages.

Let A_v denote the number of awake rounds for a node v before termination. We define the *(worst-case) awake complexity* as $\max_{v \in V} A_v$. For a randomized algorithm, A_v will be a random variable and our goal is to obtain high probability bounds on the awake complexity. Apart from minimizing the awake complexity, we also strive to minimize the overall (traditional) *round complexity* (also called *time complexity*), where both, sleeping and awake rounds, are counted.

1.2 Our Contributions and Techniques

We study the awake complexity of distributed MST and present the following results (see Table 1 for a summary).

Table 1 Summary of our Results.

Algorithm	Type	Awake Time (AT)	Run Time (RT)	AT Lower Bound	AT × RT Lower Bound
*Randomized-MST	Randomized	$O(\log n)$	$O(n \log n)$	$\Omega(\log n)$	$\tilde{\Omega}(n)$
Deterministic-MST	Deterministic	$O(\log n)$	$O(nN \log n)$	$\Omega(\log n)$	$\tilde{\Omega}(n)$
*♣Trade-Off-MST	Randomized	$\tilde{O}(n/2^k)$	$\tilde{O}(D + 2^k + n/2^k)$	-	-

*The algorithm outputs an MST with high probability.
♣This algorithm takes integer k as an input parameter.
Our lower bounds also apply to Monte Carlo randomized algorithms with constant success probability.
n is the number of nodes in the network and N is an upper bound on the the largest ID of a node.

1. Lower Bound on the Awake Complexity of MST. We show that $\Omega(\log n)$ is a lower bound on the awake complexity of constructing a MST, even for randomized Monte Carlo algorithms with constant success probability (Section 2.1). We note that showing lower bounds on the awake complexity is different from showing lower bounds for round complexity in the traditional LOCAL model. In the traditional LOCAL model, *locality* plays an important role in showing lower bounds. In particular, to obtain information from a r-hop neighborhood, one needs at least r rounds and lower bounds use indistinguishability arguments of identical r-hop neighborhoods to show a lower bound of r to solve a problem. For example, this approach is used to show a lower bound of $\Omega(D)$ for leader election or broadcast even for randomized algorithms [23]. Lower bounds in the CONGEST model are

more involved and exploit bandwidth restriction to show stronger lower bounds for certain problems, e.g., MST has a round complexity lower bound of $\tilde{\Omega}(D + \sqrt{n})$ rounds. In contrast, as we show in this paper, MST can be solved using only $O(\log n)$ awake rounds and requires a different type of lower bound argument for awake complexity.

We first show a randomized lower bound of $\Omega(\log n)$ awake complexity for *broadcast* on a line of n nodes. Our broadcast lower bound is an adaptation of a similar lower bound shown for the energy complexity model [3]. Our randomized lower bound is more general and subsumes a deterministic lower bound of $\Omega(\log n)$ shown in [2] for computing a spanning tree called the Distributed Layered Tree (by a reduction this lower bound applies to broadcast and leader election as well). The deterministic lower bound follows from the *deterministic message complexity* lower bound of $\Omega(n \log n)$ on leader election on rings (which also requires an additional condition that node IDs should be from a large range) to show that some node should be awake at least $\Omega(\log n)$ rounds. Note that this argument does not immediately apply for randomized algorithms, since such a message complexity lower bound does not hold for randomized (Monte Carlo) leader election [23]. On the other hand, our lower bound uses probabilistic arguments to show that some node has to be awake for at least $\Omega(\log n)$ rounds for accomplishing broadcast in a line. We then use a reduction to argue the same lower bound for MST problem on a *weighted ring*. We believe that the broadcast lower bound is fundamental to awake complexity and will have implications for several other problems as well.

2. Lower Bounds Relating Awake and Round Complexities. An important question is whether one can optimize awake complexity and round complexity *simultaneously* or whether one can only optimize one at the cost of the other. Our next result shows the latter is true by showing a lower bound on the *product* of awake and round complexities. Specifically, we construct a family of graphs with diameters ranging between $\tilde{\Omega}(\sqrt{n})$ to $\tilde{O}(n)$. Time-optimal MST algorithms for these graphs will have round complexity within polylog n factors of their diameters. We show that any algorithm that requires only $\tilde{O}(\text{Diameter}(G))$ rounds must have an awake complexity of at least $\tilde{\Omega}(n/\text{Diameter}(G))$ for any distributed algorithm. This holds even for Monte-Carlo randomized algorithms with constant success probability. The precise result in stated in Theorem 6 (see Section 2.2). In other words, the product of the round complexity and awake complexity is $\tilde{\Omega}(n)$.

Our lower bound technique for showing a conditional lower bound on the awake complexity (conditional on upper bounding the round complexity) can be of independent interest. We use a lower bound graph family that is similar to that used in prior work (e.g., [31, 10]), but our lower bound technique uses communication complexity to lower bound awake complexity by lower bounding the *congestion caused in some node*. This is different from the *Simulation Theorem* technique ([10]) used to show unconditional lower bounds for *round complexity* in the traditional setting. The main idea is showing that to solve the distributed set disjointness problem in less than c rounds, at least $\tilde{\Omega}(n/c)$ bits have to send through some node that has small (constant) degree. This means that the node has to be awake for essentially $\tilde{\Omega}(n/c)$ rounds. By using standard reductions, the same lower bound holds for MST. The technique is quite general and could be adapted to show similar conditional lower bounds on the awake complexity for other fundamental problems such as shortest paths, minimum cut, etc.

3. Awake-Optimal Algorithms and Techniques. We present a distributed randomized algorithm (Section 3.2) that has $O(\log n)$ awake complexity which is optimal since it matches the lower bound shown. The round complexity of our algorithm is $O(n \log n)$. We then show that the awake-optimal bound of $O(\log n)$ can be obtained deterministically by presenting a deterministic algorithm (see Section 3.3). However, the deterministic algorithm has a worse

round complexity of $O(nN \log n)$, assuming that the node IDs are in the range $[1, N]$ and N is known to all nodes. We also show that one can significantly reduce the deterministic round complexity to $O(n \log n \log^* n)$ at the cost of slightly increasing the awake complexity to $O(\log n \log^* n)$ rounds.

Our algorithms use several techniques for constructing an MST in an awake-efficient manner. Some of these can be of independent interest in designing such algorithms for other fundamental problems.[3] Our main idea is to construct a spanning tree called the *Labeled Distance Tree (LDT)*. An LDT is a rooted oriented spanning tree such that each node is labeled by its distance from the root and every node knows the labels of its parent and children (if any) and the label of the root of the tree. We show that an LDT can be constructed in $O(\log n)$ awake complexity and in a weighted graph, the LDT can be constructed so that it is an MST. While LDT construction is akin to the classic GHS/Boruvka algorithm [14], it is technically challenging to construct an LDT that is also an MST in $O(\log n)$ awake rounds (Section 3). As in GHS algorithm, starting with n singleton fragments we merge fragments via minimum outgoing edge (MOE) in each phase. The LDT distance property allows for finding an MOE in $O(1)$ awake rounds, since broadcast and convergecast can be accomplished in $O(1)$ rounds. On the other hand, maintaining the property of LDT when fragments are merged is non-trivial. A key idea is that we show that merging can be accomplished in $O(1)$ awake rounds by merging only fragment chains of *constant* length (details under "Technical challenges" in Sections 3.2 and 3.3).[4] We develop a technical lemma that shows that this merging restriction still reduces the number of fragments by a constant factor in every phase and hence the algorithm takes overall $O(\log n)$ awake rounds.

Our tree construction is different compared to the construction of trees in [2, 3]. In particular, a tree structure called as Distributed Layered Tree (DLT) is used in Barenboim and Maimon [2]. A DLT is a rooted oriented spanning tree where the vertices are labeled, such that each vertex has a greater label than that of its parent, according to a given order and each vertex knows its own label and the label of its parent. Another similar tree structure is used in [3]. A crucial difference between the LDT construction and the others is that it allows fragments to be merged via desired edges (MOEs), unlike the construction of DLT, for example, where one has to merge along edges that connect a higher label to a lower label. This is not useful for MST construction. Another important difference is that the labels used in LDTs scale with the number of nodes, whereas the labels in DLTs scale with the maximum ID assigned to any node. As the running time for both constructions are proportional to the maximum possible labels and the ID range is usually polynomially larger than the number of nodes, the running time to construct a DLT is much larger than the running time to construct an LDT.

4. Trade-Off Algorithms. We present a parameterized family of distributed algorithms that show a trade-off between the awake complexity and the round complexity and essentially (up to a polylog n factor) matches our product lower bound of $\tilde{\Omega}(n)$.[5] Specifically we show a family of distributed algorithms that find an MST of the given graph with high probability in $\tilde{O}(D + 2^k + n/2^k)$ running time and $\tilde{O}(n/2^k)$ awake time, where D is the network diameter

[3] As an example, the $O(n \log n \log^* n)$ deterministic algorithm is useful in designing an MIS algorithm with small awake and round complexities [11] — see Section 1.3.

[4] Consider the supergraph where the fragments are nodes and the MOEs are edges. A fragment chain is one such supergraph that forms a path. The exact details of the supergraphs formed are slightly different and explained in the relevant section, but this idea is beneficial to understanding.

[5] The product lower bound of $\tilde{\Omega}(n)$ is shown for graphs with diameter at least $\tilde{\Omega}(\sqrt{n})$. Hence, the near tightness claim holds for graphs in this diameter range.

and integer $k \in [\max\{\lceil 0.5 \log n \rceil, \lceil \log D \rceil\}, \lceil \log n \rceil]$ is an input parameter to the algorithm. Notice that when $D = O(\sqrt{n})$, the round complexity can vary from $\tilde{O}(\sqrt{n})$ to $\tilde{O}(n)$, and we can choose (integer) $k \in [\lceil 0.5 \log n \rceil, \lceil \log n \rceil]$ from to get the $\tilde{O}(n)$ product bound for this entire range. On the other hand, when $D = \omega(\sqrt{n})$, the round complexity can vary from $\tilde{O}(D)$ to $\tilde{O}(n)$, and we can choose $k \in [\lceil \log D \rceil, \lceil \log n \rceil]$ and get a family of algorithms with (essentially) optimal round complexities from $\tilde{O}(D)$ to $\tilde{O}(n)$.

Due to lack of space, we refer the reader to the full paper (in Appendix) for the detailed descriptions of algorithms, analysis, and omitted proofs.

1.3 Related Work and Comparison

The sleeping model and the awake complexity measure was introduced in a paper by Chatterjee, Gmyr and Pandurangan [6] who 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. 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. Subsequently, Ghaffari and Portmann [17] developed a randomized MIS algorithm that has worst-case complexity of $O(\log n)$, while having $O(1)$ node-averaged awake complexity (both bounds hold with high probability). They studied approximate maximum matching and vertex cover and presented algorithms that have similar node-averaged and worst-case awake complexities. These results show that the above fundamental local symmetry breaking problems have $O(\log n)$ (worst-case) awake complexity as is shown for global problems such as spanning tree [2] and MST (this paper). In a recent result, Dufoulon, Moses Jr., and Pandurangan [11] show that MIS can be solved in $O(\log \log n)$ (worst-case) awake complexity which is exponentially better than previous results. But the round complexity is $O(poly(n))$. It then uses the *deterministic* LDT construction algorithm of this paper to obtain an MIS algorithm that has a slightly larger awake complexity of $O(\log \log n \log^* n)$, but significantly better round complexity of $O(\text{polylog}\, n)$. The existence of a deterministic LDT algorithm is crucial to obtaining their result.

Barenboim and Maimon [2] showed that many problems, including broadcast, construction of a spanning tree, and leader election can be solved deterministically in $O(\log n)$ awake complexity. They also showed that fundamental symmetry breaking problems such as MIS and $(\Delta + 1)$-coloring can be solved deterministically in $O(\log \Delta + \log^* n)$ awake rounds in the LOCAL model, where Δ is the maximum degree. More generally, they also define the class of *O-LOCAL* problems (that includes MIS and coloring) and showed that problems in this class admit a deterministic algorithm that runs in $O(\log \Delta + \log^* n)$ awake time and $O(\Delta^2)$ round complexity. Maimon [25] presents trade-offs between awake and round complexity for O-LOCAL problems.

While there is significant amount of work on energy-efficient distributed algorithms over the years we discuss those that are most relevant to this paper. A recent line of relevant work is that Chang, Kopelowitz, Pettie, Wang, and Zhan and their follow ups [5, 3, 4, 7, 8] (see also the references therein and its follow up papers mentioned below and also much earlier work on energy complexity in radio networks e.g., [26, 19, 20]). This work defines the measure of *energy complexity* which is the same as (worst-case) awake complexity (i.e., both measures count only the rounds that a node is awake). While the awake complexity used here and several other papers [6, 17, 2] assumes the usual CONGEST(or LOCAL) communication model (and hence the model can be called *SLEEPING-CONGEST* (or *SLEEPING-LOCAL*)), the energy complexity measure used in [5] (and also papers mentioned above) has some additional communication restrictions that pertain to radio networks (and can be called

SLEEPING-RADIO model). The most important being that nodes can only broadcast messages (hence the same message is sent to all neighbors) and when a node transmits, no other neighboring node can. (Also a node cannot transmit and listen in the same round.) The energy model has a few variants depending on how collisions are handled. There is a version of the SLEEPING-RADIO model called "Local" where collisions are ignored and nodes can transmit messages at the same time; this is essentially same as SLEEPING-LOCAL model, apart from the notion that in a given round a node can transmit only the same message to its neighbors. In particular, upper bounds in the radio model apply directly to the sleeping model. In particular, we use a recent result due to Dani and Hayes [8] that computes breadth-first search (BFS) tree with $O(\text{polylog}(n))$ energy complexity in the radio model as a subroutine in our MST tradeoff algorithm. Also, algorithms in the SLEEPING-CONGEST model can be made to work in the SLEEPING-RADIO model yielding similar bounds (with possibly a $O(\text{polylog}(n))$ multiplicative factor) to the energy/awake complexity.

Lower bounds shown in the local version of the SLEEPING-RADIO model apply to other models including SLEEPING-LOCAL (and SLEEPING-CONGEST). For example, Chang, Dani, Hayes, He, Li, and Pettie [3] show a lower bound $\Omega(\log n)$ on the energy complexity of broadcast which applies also to randomized algorithms. This lower bound is shown for the local version of their model, and this result holds also for the awake complexity in the sleeping model. We adapt this lower bound result to show a $\Omega(\log n)$ lower bound on the awake complexity of MST even for randomized algorithms.

2 Lower Bounds

We first show that $\Omega(\log n)$ is an *unconditional* lower bound on the awake time for MST. This shows that our algorithms presented in Section 3 achieve optimal awake complexity. We then show a lower bound of $\tilde{\Omega}(n)$ on the product of the awake and round complexities. This can be considered as a conditional lower bound on awake complexity, conditioned on an upper bound on the round complexity. This conditional lower bound shows that our randomized awake optimal algorithm (see Section 3.2) has essentially the best possible round complexity (up to a polylog n factor).

2.1 Unconditional Lower bound on Awake Complexity of MST

We consider a ring of $\Theta(n)$ nodes with random weights and edges. The two largest weighted edges will be apart by a hop distance of $\Omega(n)$ with constant probability and any MST algorithm must detect which one has the lower weight. Clearly, this will require communication over either one of the $\Omega(n)$ length paths between the two edges. Under this setting, we get the following theorem.

▶ **Theorem 1.** *Any algorithm to solve MST with probability exceeding* $1/8$ *on a ring network comprising* $\Theta(n)$ *nodes requires* $\Omega(\log n)$ *awake time even when message sizes are unbounded.*

2.2 Lower Bound on the Product of Awake and Round Complexity

We adapt the lower bound technique from [9] to the sleeping model and show a lower bound on the product of the awake and round complexities, thus exposing an inherent trade off between them.

Note that both endpoints of an edge must be awake in a round for $O(\log n)$ bits to be transmitted across in that round. Thus, if an edge $e = (u, v)$ must transmit B bits when executing an algorithm in the CONGEST model, then, both u and v must be awake for at least $\Omega(B/\log n)$ rounds. Thus congestion increases awake time. Our goal is to exploit this intuition to prove the lower bound.

We use a communication complexity based reduction to reduce *set disjointness* (SD) in the classical communication complexity model to the *distributed set disjointness* (DSD) problem in the sleeping model and then extend DSD to the minimum spanning tree problem via an intermediate *connected spanning subgraph* (CSS), both to be solved in the sleeping model.

In the SD problem, two players, Alice and Bob, possess two k-bit strings $a = (a_i)_{1 \leqslant i \leqslant k}$ and $b = (b_i)_{1 \leqslant i \leqslant k}$, respectively. They are required to compute an output bit $\mathsf{Disj}(a, b)$ that is 1 iff there is no $i \in [k]$ such that $a_i = b_i = 1$ (i.e., the inner product $\langle a, b \rangle = 0$), and 0 otherwise. Alice and Bob must compute $\mathsf{Disj}(a, b)$ while exchanging the least number of bits. It is well-known that any protocol that solves SD requires $\Omega(k)$ bits (on expectation) to be exchanged between Alice and Bob even if they employ a randomized protocol [32] that can fail with a small fixed probability $\varepsilon > 0$.

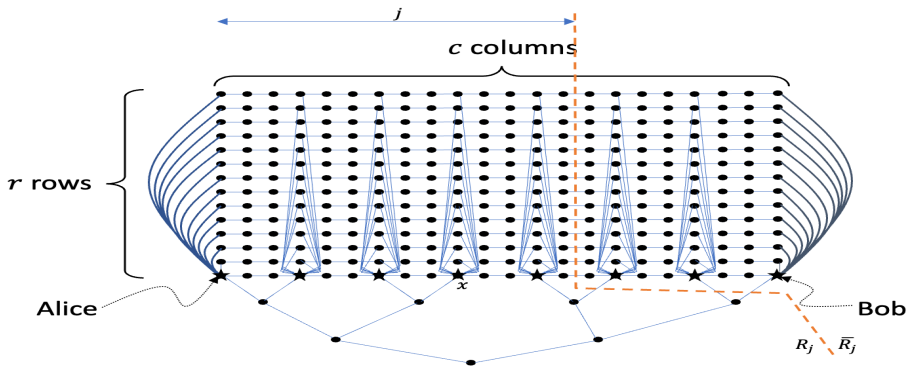


Figure 1 Construction of network graph G_{rc} for proving lower bound. The vertices in X are shown as stars (there is a binary tree at the bottom having the nodes in X as its leaves). One such $x \in X$ is labeled. The cut induced by an R_j is shown in dotted lines.

The DSD problem is defined on a graph G_{rc} that is schematically shown in Figure 1. Let r and c be two positive integers such that $rc + \Theta(\log n) = n$ (the network size). We focus on the regime where $c \in \omega(\sqrt{n} \log^2 n)$ and $r \in o(\sqrt{n}/\log^2 n)$. The graph comprises r rows (or parallel paths) p_ℓ, $1 \leqslant \ell \leqslant r$, with p_1 referring to the parallel path at the bottom. Each parallel path comprises c nodes arranged from left to right with the first node referring to the leftmost node and the last node referring to the rightmost node. The first and last nodes in p_1 are designated Alice and Bob because they are controlled by the players Alice and Bob in our reduction. Alice (resp., Bob) is connected to first node (resp., last node) of each p_ℓ, $2 \leqslant \ell \leqslant r$. Additionally, we pick $\Theta(\log n)$ equally spaced nodes X (of cardinality that is a power of two) from p_1 such that the first and last nodes in p_1 are included in X. For each $x \in X$, say at position j in p_1, we add edges from x to the jth node in each p_ℓ, $2 \leqslant \ell \leqslant r$. Using X as leaves, we construct a balanced binary tree. We will use I to denote the internal nodes of this tree. Alice is in possession of bit string a and Bob is in possession of b and, to solve DSD, they must compute $\mathsf{Disj}(a, b)$ in the sleeping model over the network G_{rc}. The outputs of all other nodes don't matter; for concreteness, we specify that their outputs should be empty .

In the CSS problem defined again on G_{rc}, some edges in G_{rc} are marked and at least one node in the network must determine whether the marked edges form a connected spanning subgraph of G_{rc}. For the MST problem, we require edges in G_{rc} to be weighted and the goal is to construct a minimum spanning tree of G_{rc} such that the endpoints of each MST edge e are aware that e is an MST edge. Both CSS and MST must be solved in the sleeping model.

G_{rc} is constructed such that any node can reach some $x \in X$ within $O(c/\log n)$ steps and any pair of nodes in X are within $O(\log\log n)$ steps (through the tree). Recall that $c \in \omega(\sqrt{n}\log^2 n)$. Thus, we have the following observation.

▶ **Observation 2.** *The network graph G_{rc} has diameter $D \in \Theta(c/\log n)$. Moreover, $D \in \omega(\sqrt{n}\log^* n)$. Therefore, DSD, CSS, and MST (if edges in G_{rc} are assigned weights) can be computed in $O(D) = O(c/\log n)$ rounds [15].*

2.2.0.1 Reduction from SD → DSD

Recall that in DSD, Alice is in possession of a and Bob is in possession of b. They must compute $\mathsf{Disj}(a, b)$ in the sleeping model.

▶ **Lemma 3.** *Consider an algorithm P in the sleeping model that solves DSD on G_{rc} with $c \in \omega(\sqrt{n}\log^2 n)$ and $r \in o(\sqrt{n}/\log^2 n)$ in T (worst-case) rounds such that $T \in o(c)$ (and we know such an algorithm exists from Observation 2, in particular because $D \in O(c/\log n)$). Then, the awake time of P must be at least $\Omega(r/\log^2 n)$. This holds even if P is randomized and has an error probability that is bounded by a small constant $\varepsilon > 0$.*

Proof. Suppose for the sake of contradiction P runs in time T and has an awake complexity of $o(r/\log^2 n)$. Then, we can show that Alice and Bob can simulate P in the classical communication complexity model and solve SD on r bits by exchanging only $o(r)$ bits which will yield a contradiction to the SD lower bound. We establish this by showing that Alice and Bob can simulate P to solve SD in the classical communication complexity model.

We show this simulation from Alice's perspective. Bob's perspective will be symmetric. Recall that p_ℓ is the ℓth parallel path. Let p_ℓ^j, $1 \leqslant j \leqslant c$, denote the first j vertices of path p_ℓ. We define R_j to be the union of all p_ℓ^j and I (recall that I is the set of the internal nodes of the binary tree), i.e., $R_j = (\bigcup_{\ell=1}^{r} p_\ell^j) \cup I$. Note that R_j induces a cut $(R_j, \bar{R}_j)$ that is shown in Figure 1. Alice begins by simulating R_{c-1} in round 1 as she knows the state of all nodes in R_{c-1}. At each subsequent round t, Alice simulates R_{c-t}. Initially, all the information needed for the simulation is available for Alice because the structure of G_{rc} is fully known (except for Bob's input).

As the simulation progresses, in each round $t > 1$, $t \leqslant T \in o(c)$, all inputs will be available except for the new bits that may enter I through nodes in $\bar{R}_{c-t}$. Alice will not need to ask for the bits needed by p_ℓ^{c-t} because she simulated all nodes in p_ℓ^{c-t+1}, $1 \leqslant \ell \leqslant r$, in the previous round. Note that the portion simulated by Bob will encompass the portion from which Alice may need bits from Bob, so Bob will indeed have the bits requested by Alice. In order to continue the simulation, Alice borrows bits that P transmitted from $\bar{R}_{c-t}$ to $I \cap R_{c-t}$ from Bob. Suppose during the course of the simulation in the communication complexity model, B bits are borrowed from Bob. Then nodes in I must have been awake for a collective total of at least $\Omega(B/\log n)$ rounds (because each message of $O(\log n)$ bits must be received by a node that is awake in P).[6] This implies that at least one node in I must have been awake for $\Omega(B/\log^2 n)$ rounds because $|I| \in O(\log n)$ and the number of edges incident to nodes in I is also $O(\log n)$ (since nodes in I are of constant degree).

Since the node awake time is $o(r/\log^2 n)$ for P, B must be $o(r)$. But this contradicts the fact that SD requires $\Omega(r)$ bits in the communication complexity model. ◀

[6] Note that all the B bits cannot solely come through a row path of length c, since we are restricting $T \in o(c)$. In other words, each of the bits has to go through at least one node in I.

2.2.0.2 Reduction from DSD $\rightarrow$ CSS

We now show a reduction from DSD $\rightarrow$ CSS by encoding a given DSD problem instance as a CSS instance in the following manner. Recall that in DSD, Alice and Bob have bit strings a and b, respectively, of length r each. Furthermore, recall that Alice (resp., Bob) is connected to first node (resp., last node) of each p_ℓ, $2 \leqslant \ell \leqslant r$.

▶ **Lemma 4.** *Suppose there is a protocol Q in the sleeping model that solves CSS on G_{rc} with $c \in \omega(\sqrt{n}\log^2 n)$ and $r \in o(\sqrt{n}/\log^2 n)$ in T rounds such that $T \in o(c)$. Then, the node awake time of Q must be at least $\Omega(r/\log^2 n)$. This holds even if Q is randomized and has an error probability that is bounded by a small constant $\varepsilon > 0$.*

2.2.0.3 Reduction from CSS $\rightarrow$ MST

Recall that CSS is a decision problem that requires a subset of the edges in the network graph G_{rc} to be marked; we are to report whether the marked edges form a spanning subgraph of G_{rc}. MST on the other hand is a construction problem. It takes a weighted network graph and computes the minimum spanning tree. A reduction from CSS to MST can be constructed by assigning a weight of 1 for marked edges in the CSS instance and n for all other edges and asking if any edge of weight n is included in the MST. This leads to the following lemma.

▶ **Lemma 5.** *Suppose there is a protocol M in the sleeping model that solves MST on G_{rc} with $c \in \omega(\sqrt{n}\log^2 n)$ and $r \in o(\sqrt{n}/\log^2 n)$ in T rounds such that $T \in o(c)$. Then, the node awake time of M must be at least $\Omega(r/\log^2 n)$. This holds even if M is randomized and has an error probability that is bounded by a small constant $\varepsilon > 0$.*

Generalizing to arbitrary graphs, we can conclude with the following theorem.

▶ **Theorem 6.** *Consider positive integers r and c such that $rc + \Theta(\log n) = n$ (the network size) and $c \in \omega(\sqrt{n}\log^2 n)$. (Thus, c can range between $\tilde{\Omega}(\sqrt{n})$ to $\tilde{O}(n)$ with $r \in \Theta(n/c)$.) Suppose there exists a randomized algorithm M for MST in the sleeping model that runs in time $T \in o(c)$ rounds and guaranteed to compute the MST with probability at least $1 - \varepsilon$ for any small fixed $\varepsilon > 0$. Then, the (worst case) awake complexity of M must be at least $\Omega(r/\log^2 n)$.*

3 MST Algorithms with Optimal Awake Complexity

In this section, we present our algorithms to construct an MST that take optimal awake complexity. Due to space constraints, we only cover the most salient points of the algorithms as well as the technical challenges overcome. Interested readers may refer to the full version in the Appendix for detailed descriptions of the algorithms and their analysis.

3.1 Main Ideas

Both of the algorithms we develop in this section can be seen as variations of the classic GHS algorithm to find the MST, adapted to optimize the awake time of the algorithm. Recall that each phase of the GHS algorithm consists of two steps. Step (i) corresponds to finding the minimum outgoing edges (MOEs) for the current fragments and step (ii) involves merging these fragments.

Our algorithms work in phases where at the end of each phase, we ensure that the original graph has been partitioned into a forest of node-disjoint trees that satisfy the following property. For each such tree, all nodes within the tree know the ID of the root of the tree (called *fragment ID*), the IDs of their parents and children in the tree, if any, and their

distance from the root (note that it is the hop distance, ignoring the weights) of that tree. We call each such tree a *Labeled Distance Tree (LDT)* and a forest of such trees a *Forest of Labeled Distance Trees (FLDT)*. By the end of the algorithms we design, our goal is to have the FLDT reduced to just one LDT which corresponds to the MST of the original graph. The challenge is to construct an LDT (which will also be an MST) in an awake-optimal manner.

The purpose of maintaining such a structure is that we know how to design fast awake procedures to propagate information within an LDT. By making use of blocks of $2n + 1$ rounds, we can design schedules for nodes to wake up so that procedures associated with GHS, such as broadcast, upcast-min, etc. can be implemented in $O(1)$ awake complexity and $O(n)$ round complexity. In the course of our algorithms, we ensure that nodes stay synchronized, i.e., time can be viewed in blocks of $2n + 1$ rounds such that all nodes start their first schedule at the same time (and end them at the same time) and continue to start (and end) future schedules at the same time.

3.2 Awake-Optimal Randomized Algorithm

Technical Challenges. As mentioned above, one of the key changes we make is to restrict the MOEs to a subset of "valid" ones. This is to address a key technical challenge. When we merge two fragments together, one of those fragments must internally re-orient itself and update its internal values (including distance to the root). This re-alignment and updation takes $O(1)$ awake time. If we have a chain of fragments, say of diameter d, we may have to perform this re-alignment procedure $d - 1$ times since the re-alignment of fragments is sequential in nature.[7] As a result, if we do not control the length of chains of connected components formed by the fragments and their MOEs, we risk blowing up the awake time of the algorithm. We use randomness to ensure the diameter of any such connected component is a constant.

As a result of the above change, we have a second technical challenge. Because we reduce the number of valid MOEs, we have to be careful to argue that a sufficient number of fragments are merged together in each phase so that after $O(\log n)$ phases, we end up with exactly one fragment with high probability. We provide such a careful argument, which is somewhat different from the usual argument used in GHS style algorithms.

Detailed Algorithm. Algorithm Randomized-MST consists of nodes participating in $4\lceil \log_{4/3} n \rceil + 1$ phases of the following computations. Recall that between phases we want to maintain an FLDT that is eventually converted into a single LDT. In each phase, there are three steps. The first step corresponds to finding the MOE of each fragment, the second step relates to finding valid MOEs, and the third step corresponds to merging fragments along valid MOEs.

Step (i): Finding MOE of each fragment. Consider a single fragment. All nodes in it participate in a combination of broadcasts and convergecasts to send the smallest MOE to the fragment root, which in turn transmits this info to all nodes in the fragment.

Step (ii): Finding "valid" MOEs. In step (ii), each fragment's root flips a coin and only MOEs from fragments whose roots flipped tails to those that flipped heads are "valid".[8]

[7] To observe the sequential nature of the re-alignment, consider a chain with three fragments, say $A \leftarrow B \leftarrow C$. Suppose A maintains its orientation. The nodes in B must be processed first and must update their distance to A. Only then can the nodes of C accurately update their distance to A (after the node u in C connected to the node v in B learns v's updated distance to A).

[8] Intuitively, stars are formed by fragments and MOEs as a result of this process. In each such star, the center fragment is one whose root flipped heads and the leaves, if any, are fragments whose roots flipped

Henceforth, if a fragment's root flipped a heads (tails), then we say that the fragment flipped a heads (tails). Alternatively, we can say that the fragment is considered a heads (tails) fragment. Each fragment root flips an unbiased coin and broadcasts the result to its fragment nodes. Each node communicates its fragment's coin flip result to its neighbors. As a result, each node adjacent to an MOE knows if that MOE is a valid one or not. This information can be transmitted to all nodes in the fragment via an upcast and broadcast in the fragment.

Step (iii): Merging fragments. In step (iii), we merge each subgraph formed by fragments and valid MOEs into a single fragment. Consider a subgraph consisting of several tails fragments whose MOEs lead to a single heads fragment. The heads fragment retains its fragment ID while the remaining fragments take on the ID of the heads fragment. Furthermore, these other fragments also re-orient themselves such that they form subtrees of the heads fragment. Specifically, consider a tails fragment T with root $root_T$ and an MOE to a heads fragment H where nodes u_T and u_H are the nodes of the MOE belonging to T and H, respectively. The nodes in fragment T re-orient themselves such that u_T is the new root of the tree. Additionally, u_T considers u_H its parent in the merged graph. The process is similar to that in [2] and is illustrated in Figures 2, 3, 4, and 5 which are found in the Appendix.

Analysis. We now prove that the algorithm correctly outputs the MST of the original graph with the desired running time and awake time. Recall that the number of fragments can never increase from one phase to the next. Let phase $\mathcal{P}$ correspond to the last phase in which there is more than one fragment at the beginning of the phase. We will show that $\mathcal{P} = 4\lceil \log_{4/3} n \rceil$. The following lemma shows that for the first $\mathcal{P}$ phases of the algorithm, the number of fragments is reduced by a constant factor in each phase with high probability.

▶ **Lemma 7.** *For each phase of Algorithm* Randomized-MST *where there are initially at least two fragments at the start of that phase, the number of fragments is reduced by at least a factor of* $4/3$ *in that phase on expectation. Furthermore, by phase* $4\lceil \log_{4/3} n \rceil + 1$*, there is at most one fragment in the graph.*

We are now ready to argue that the algorithm is correct.

▶ **Lemma 8.** *Algorithm* Randomized-MST *results in each node of the initial graph knowing which of its edges are in the MST with high probability.*

We also bound the running time and awake time of the algorithm below.

▶ **Lemma 9.** *Algorithm* Randomized-MST *takes* $O(n \log n)$ *running time and* $O(\log n)$ *awake time.*

▶ **Theorem 10.** *Algorithm* Randomized-MST *is a randomized algorithm to find the MST of a graph with high probability in* $O(n \log n)$ *running time and* $O(\log n)$ *awake time.*

3.3 Awake-Optimal Deterministic Algorithm

Technical Challenges. We experience similar technical challenges as those faced when designing Algorithm Randomized-MST. However, we resolve those issues quite differently here. As before, when we construct connected components of fragments and their MOEs, we want that the diameter of each of these components is a constant, so that we can re-orient fragments quickly during the merging process. Since we do not have access to randomness,

tails.

we rely on the somewhat standard approach of using a deterministic maximal matching over these components to reduce the diameter. However, while the approach is standard, the execution is not. In order to maintain a small awake time, we first reduce every component to a set of bounded degree components and then run a tailored algorithm to color all fragments in $O(1)$ awake time. Subsequently, converting this coloring to a maximal matching is done as usual.

Once again, due to the above changes, we must deal with a second technical challenge. Because we reduce the number of valid MOEs, we have to be careful to argue that a sufficient number of fragments are merged together in each phase so that after $O(\log n)$ phases, we end up with exactly one fragment. We utilize an interesting combinatorial approach to argue this.

Detailed Algorithm. We now give a detailed break up of each phase of the algorithm. Let $c = 240000$. Recall that there are $\lceil \log_{c/(c-1)} n \rceil + c$ phases and in each phase, there are three steps.[9] We describe each in detail separately.

Step (i): Finding MOE of each fragment. Each fragment finds it MOE in the same way as step (i) of Algorithm Randomized-MST.

Step (ii): Finding "valid" MOEs. Each node in a given fragment knows, for each of its edges adjacent to it, whether that edge is an MOE from some other fragment to the given fragment. Let us differentiate these MOEs *to* the fragment from the MOE *from* the fragment by calling the former INCOMING-MOEs. Now, we have each fragment select up to 3 "valid" MOEs from its INCOMING-MOEs, chosen arbitrarily. This is in contrast to how the valid MOEs were chosen during Algorithm Randomized-MST, where we used coin flips to determine valid MOEs. Define an *incoming MOE node v of fragment f* as a node v belonging to fragment f such that v is adjacent to an edge that is an MOE from some other fragment to f. In the context of a given fragment f, define a *valid MOE child node v of a node u* as a child node of u such that in the subtree rooted at v, there exists an incoming MOE node of f. At a high level, the total number of INCOMING-MOEs is communicated to the root of the fragment. The root then allots up to 3 virtual "tokens" (i) to its valid MOE child nodes to be used to select INCOMING-MOEs and (ii) to itself if the root is an incoming MOE node. Any node that receives one or more such tokens distributes them among its valid MOE child nodes and itself if it is an incoming MOE node. This process is repeated until all tokens are transmitted to incoming MOE nodes of the fragment.

Step (iii): Merging fragments. We first make each fragment f's nodes aware of the fragment IDs from and to which it has valid MOEs. We first collect information about valid MOEs (both incoming & outgoing) at the root of the fragment f and then broadcast this information to all nodes in the MOE. Subsequently, we color the fragments and then selectively merge them. Consider a color palette consisting of colors Blue, Red, Orange, Black, and Green. Furthermore, let there exist a total ordering on this palette based on the relation of priority, where we say that a color A has a higher priority than color B and denote the relation by $A > B$, such that Blue $>$ Red $>$ Orange $>$ Black $>$ Green. Let us consider the supergraph G' where the nodes are the set of fragments present at the beginning of the phase and the edges are the valid MOEs, as computed from step (ii). Recall that the maximum degree of any node in G' is 4, so 5 colors are sufficient for coloring (there always exists a $\Delta + 1$ coloring of a graph with maximum degree Δ). At a high level, we wake fragments up in order of the fragment IDs and color them with the first available color, i.e., the highest priority color not chosen by any of its neighbors. Neighboring fragments become

[9] We have not chosen to optimize the constants in our analysis.

aware of a given fragment F's chosen color by having F *AND* its neighboring fragments wake up in the rounds that F was assigned. As a given fragment has $O(1)$ neighbors, it does not have to stay awake for too long.

Now, we describe the selective merging in more detail. We first identify the set of fragments that will merge into other fragments and will thus "not survive" the phase. These are all fragments that were colored Blue. Recall that there are two types of fragments that are colored Blue. Those with neighbors in G' and those without, which we call *singleton fragments*.

Those Blue fragments with neighbors pick one of their neighbors in G' arbitrarily (which is of course a non-Blue fragment) and then merge into them. This can be achieved by using the merging process of Algorithm Randomized-MST where we consider Blue fragments as Tails fragments and all non-Blue fragments as Heads fragments. The merged fragment takes on the fragment ID of the fragment that acted as the Heads fragment.

Now we have singleton Blue fragments merge into the fragments at the end of their MOEs in a manner similar to above. But before doing that, each node in these singleton fragments becomes aware of any changes to fragment IDs and level numbers of neighboring nodes (and corresponding neighboring fragments) by having all nodes in the graph swap information with their neighbors.

Analysis. We now prove that the algorithm correctly outputs the MST of the original graph with the desired running time and awake time.

Recall that we use the notation $c = 240000$. We show that the number of phases needed to reduce the number of fragments to one is at most $\lceil \log_{c/(c-1)} n \rceil + c$. Correctness immediately follows as we implement GHS. We first argue that in each phase of the algorithm where there is initially a sufficient number of fragments c, the number of fragments is reduced by a constant factor of $c/(c-1)$. We then show that in an additional c phases, we can reduce the number of fragments to one. As we only add MST edges and all nodes will be in this one fragment, the final fragment represents the MST of original graph.

Let $\mathcal{P}$ represent the phase by which the number of fragments at the beginning of the phase is less than c. We eventually show that $\mathcal{P} = \lceil \log_{c/(c-1)} n \rceil$. We first argue that in every phase up to $\mathcal{P}$, the number of fragments is reduced by a constant factor. We do this by considering an arbitrary phase i and identifying a set of fragments in that phase that are guaranteed to merge into other fragments, thus "being lost" or "not surviving" in that phase. We show that this set is at least a constant fraction of the total set of fragments that existed at the start of the phase.

Consider an arbitrary phase i such that at the beginning of the phase there exists a set $\mathcal{F}_i$ of fragments and define $F_i = |\mathcal{F}_i|$. Furthermore, define the supergraph H as the undirected graph where the nodes are the set $\mathcal{F}_i$ and the edges are the valid MOEs between the different fragments, i.e., the graph obtained after pruning MOEs in step (ii) of the phase. In the subsequent analysis we use nodes and fragments interchangeably in the context of graph H. We now show that the number of Blue fragments (which by the algorithm are all merged into other fragments) constitute a sufficiently large constant fraction of $\mathcal{F}_i$.

▶ **Lemma 11.** *Let H' be a connected subgraph of H. If $|H'| \geqslant 342$, then at least $\lfloor |H'|/342 \rfloor$ of the fragments are Blue.*

The above lemma by itself is insufficient to show that the required number of fragments are removed in each phase. The reason is that H may consist of a set of disjoint connected subgraphs. Let us assume that $|H| \geqslant c$. Let S denote the set of all disjoint connected subgraphs (i.e., connected components) in H. Now, either $|S| \geqslant |H|/342^2$ or $|S| < |H|/342^2$.

We show that in either case, the number of fragments that survive the current phase is $(c-1)|H|/c$.

If $|S| \geqslant |H|/342^2$, then since each subgraph in S contains at least one Blue fragment which disappears in the phase, the total number of fragments that survive the phase is at most $|H| - |H|/342^2 \leqslant (c-1)|H|/c$.[10]

Now let us look at the situation where $|S| < |H|/342^2$. Divide S into the sets S_1 and S_2 which contain the disjoint connected subgraphs of H which have < 342 fragments and $\geqslant 342$ fragments, respectively. Observe that $S = S_1 \bigcup S_2$. (It is easy to see that $|S_2| \geqslant 1$ since otherwise if all subgraphs belonged to S_1, there would be less than $|H|$ total fragments.) We now show that a sufficient number of fragments in the subgraphs in set S_2 are Blue fragments, thus resulting in a sufficient number of fragments being removed in the phase. Let us lower bound how many fragments are present in the subgraphs in set S_2. Recall that $|S_1| \leqslant |S|$, we are considering the situation where $|S| < |H|/342^2$, and each subgraph in $|S_1|$ can have < 342 fragments. We lower bound the fragments in S_2 by pessimistically ignoring the less than 342 fragments from each of the $|S_1|$ subgraphs (recall that $|S_1| \leqslant |S|$), i.e., the number of fragments in S_2 is $\geqslant |H| - (|H|/342^2) \cdot 342$ (since the total number of subgraphs is at most $|H|/342^2$) $= 341|H|/342$.

We now lower bound the number of Blue fragments in subgraphs in S_2. Let the subgraphs in S_2 be denoted by $H_1, H_2, \ldots, H_{|S_2|}$. Since each subgraph in S_2 is of size at least 342, we can use Lemma 4. Now, the number of Blue fragments in S_2 is $= \sum_{i=1}^{|S_2|} \lfloor |H_i|/342 \rfloor \geqslant \sum_{i=1}^{|S_2|} (|H_i|/342 - 1) = 1/342 \cdot (\sum_{i=1}^{|S_2|} |H_i|) - |S_2| \geqslant 1/342 \cdot (341|H_i|/342) - |S_2| \geqslant 1/342 \cdot (341|H_i|/342) - |H|/342^2$ (since $|S_2| \leqslant |S| < |H|/342^2$) $= 340|H|/342^2$.

Recall that all Blue fragments do not survive a phase. Thus, the number of fragments that survive the current phase is $\leqslant |H| - 340|H|/342^2 \leqslant (c-1)|H|/c$.

Thus, in both situations, we see that the number of fragments that survive the present phase is upper bounded as desired.

▶ **Lemma 12.** *After $\lceil \log_{c/(c-1)} n \rceil$ phases, there are at most c fragments at the beginning of the phase.*

An easy observation is that whenever $F_i \geqslant 2$, at least one fragment is Blue and will not survive the phase. Thus, if there are at most c fragments at the beginning of the phase, then running an additional c phases guarantees that only one fragment will remain. Initially, each node is a fragment by itself and over the course of the algorithm, only possible MST edges are added to any fragment. Thus, we have the following lemma.

▶ **Lemma 13.** *Algorithm Deterministic-MST correctly outputs an MST after $\lceil \log_{c/(c-1)} n \rceil + c$ phases.*

The running time and awake time are analyzed in the Appendix, leading to the following lemma.

▶ **Lemma 14.** *Each phase of Algorithm Deterministic-MST takes $O(1)$ awake time and $O(nN)$ running time.*

Thus, combining Lemma 13 and Lemma 14, we get the following theorem.

▶ **Theorem 15.** *Algorithm Deterministic-MST is a deterministic algorithm to find the MST of a given graph in $O(\log n)$ awake time and $O(nN \log n)$ running time.*

[10] It is easy to see that each subgraph contains a Blue fragment because Blue is the highest priority color and so the first fragment that colors itself in any given subgraph colors itself Blue.

Remark. The coloring procedure, Fast-Awake-Coloring, is the main reason for the large run time. As we noted near the beginning of this section, we can replace this procedure with one that can accomplish this deterministically in $O(\log^* n)$ run time even in the traditional model (see e.g., [28]). However, we suffer an overhead of $O(\log^* n)$ factor in the awake time. As a result, using this modified procedure would allow us to get the following corollary.

▶ **Corollary 16.** *There exists a deterministic algorithm to find the MST of a given graph in $O(\log n \log^* n)$ awake time and $O(n \log n \log^* n)$ run time.*

4 MST Algorithm with a Trade-off

In this section, we present an algorithm to create an MST that shows a trade-off between its running time and awake time. We only give a high-level overview of the algorithm and defer the detailed explanation of the algorithm along with analysis to the full paper.

High-level Overview. We present algorithm Trade-Off-MST, which finds the MST of the given graph with high probability in $\tilde{O}(D + 2^k + n/2^k)$ running time and $\tilde{O}(n/2^k)$ awake time, where integer k is an input parameter to the algorithm. It is an awake time efficient version of the Controlled-GHS algorithm from Chapter 7 of [27] (itself a version of the optimized version of the algorithm from [15]) adapted to reduce the awake complexity. We describe it as a 3 stage algorithm.

In stage one, we elect a leader node among all the nodes. We then construct a Breadth First Search (BFS) tree rooted at this leader (using the algorithm of [8]), which will be used later on. In stage two, we switch gears and have all nodes perform the controlled version of the GHS algorithm for $k-1$ phases until $\leqslant n/2^k$ fragments are formed, each of size $\geqslant 2^k$ and diameter $\leqslant 5 \cdot 2^k$. In stage three, each node now uses the BFS tree formed in stage one to send its MOE (inter-fragment MOE) for each of the at most $n/2^k$ fragments with corresponding node IDs and fragments IDs to the leader using pipelining. Using the red rule to prevent cycles, we ensure that this is done quickly. The leader then locally computes which $O(n/2^k)$ edges between the $O(n/2^k)$ fragments are a part of the MST and sends messages about those edges down to the respective nodes.

▶ **Theorem 17.** *Algorithm Trade-Off-MST is a randomized algorithm to find the MST of a graph with high probability and takes $\tilde{O}(D + 2^k + n/2^k)$ running time and $\tilde{O}(n/2^k)$ awake time, where k is an input to the algorithm.*

5 Conclusion

We presented distributed algorithms for the fundamental MST problem that are optimal with respect to awake complexity. We also showed that there is an inherent trade-off bottleneck between awake and round complexities of MST. In other words, one cannot attain optimal complexities simultaneously under both measures. We also presented an algorithm that shows a trade-off between awake complexity and round complexity, complementing our trade-off lower bound. Interesting lines of future work including designing awake-efficient algorithms for other fundamental global problems such as shortest path and minimum cut.

References

1 Christoph Ambühl. An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1139–1150. Springer, 2005.

2 Leonid Barenboim and Tzalik Maimon. Deterministic logarithmic completeness in the distributed sleeping model. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.DISC.2021.10.

3 Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, Qizheng He, Wenzheng Li, and Seth Pettie. The energy complexity of broadcast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 95–104, 2018.

4 Yi-Jun Chang, Varsha Dani, Thomas P. Hayes, and Seth Pettie. The energy complexity of bfs in radio networks. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 273–282, 2020.

5 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: STOC 2017.

6 Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan. Sleeping is efficient: MIS in $O(1)$-rounds node-averaged awake complexity. In Yuval Emek and Christian Cachin, editors, *PODC '20: ACM Symposium on Principles of Distributed Computing*, pages 99–108, 2020.

7 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.

8 Varsha Dani and Thomas P. Hayes. How to wake up your neighbors: Safe and nearly optimal generic energy conservation in radio networks. In Christian Scheideler, editor, *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA*, volume 246 of *LIPIcs*, pages 16:1–16:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.DISC.2022.16.

9 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 363–372, New York, NY, USA, 2011. Association for Computing Machinery.

10 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

11 Fabien Dufoulon, William K. Moses Jr., and Gopal Pandurangan. Distributed MIS in o(log log n) awake complexity. In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*, 2023.

12 Michael Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proc. of the ACM Symposium on Theory of Computing*, pages 331 – 340, 2004.

13 Michael Elkin. A simple deterministic distributed MST algorithm, with near-optimal time and message complexities. In *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 157–163, 2017.

14 R. Gallager, P. Humblet, and P. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, January 1983.

15 Juan A. Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, 27(1):302–316, 1998.

16 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 202–219. SIAM, 2016.

17 Mohsen Ghaffari and Julian Portmann. Average awake complexity of MIS and matching. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 45–55, 2022.

18 Bernhard Haeupler, David Wajc, and Goran Zuzic. Universally-optimal distributed algorithms for known topologies. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1166–1179. ACM, 2021.

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

20 Marcin Kardas, Marek Klonowski, and Dominik Pajak. Energy-efficient leader election protocols for single-hop radio networks. In *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013*, pages 399–408. IEEE Computer Society, 2013.

21 Maleq Khan, Gopal Pandurangan, and V. S. Anil Kumar. Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. *IEEE Trans. Parallel Distributed Syst.*, 20(1):124–139, 2009.

22 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. doi:10.1007/s00453-010-9422-0.

23 Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. On the complexity of universal leader election. *J. ACM*, 62(1), 2015.

24 Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *J. Algorithms*, 28(1):40–66, 1998.

25 Tzalik Maimon. Sleeping model: Local and dynamic algorithms, 2021. arXiv:2112.05344.

26 Koji Nakano and Stephan Olariu. Randomized leader election protocols in radio networks with no collision detection. In D. T. Lee and Shang-Hua Teng, editors, *Algorithms and Computation, 11th International Conference, ISAAC 2000, Taipei, Taiwan, December 18-20, 2000, Proceedings*, volume 1969 of *Lecture Notes in Computer Science*, pages 362–373. Springer, 2000.

27 Gopal Pandurangan. Distributed network algorithms. In *Distributed Network Algorithms*, 2021. URL: https://sites.google.com/site/gopalpandurangan/home/distributed-network-algorithms.

28 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time- and message-optimal distributed algorithm for minimum spanning trees. In *Proceedings of the 49th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 743–756, 2017.

29 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. The distributed minimum spanning tree problem. *Bull. EATCS*, 125, 2018.

30 David Peleg. *Distributed Computing: A Locality Sensitive Approach*. SIAM, 2000.

31 David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM J. Comput.*, 30(5):1427–1442, 2000.

32 Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

Appendix: Full version of paper