Distributed MST and Broadcast with Fewer Messages, and Faster Gossiping

Mohsen Ghaffari
ETH Zurich, Switzerland
ghaffari@inf.ethz.ch

Fabian Kuhn
University of Freiburg, Germany
kuhn@cs.uni-freiburg.de

Abstract

We present a distributed minimum spanning tree algorithm with near-optimal round complexity of $\tilde{O}(D + \sqrt{n})$ and message complexity $\tilde{O}(\min\{n^{3/2}, m\})$. This is the first algorithm with sub-linear message complexity and near-optimal round complexity and it improves over the recent algorithms of Elkin [PODC'17] and Pandurangan et al. [STOC'17], which have the same round complexity but message complexity $\tilde{O}(m)$. Our method also gives the first broadcast algorithm with $o(n)$ time complexity – when that is possible at all, i.e., when $D = o(n)$ – and $o(m)$ messages. Moreover, our method leads to an $\tilde{O}(\sqrt{nD})$-round GOSSIP algorithm with bounded-size messages. This is the first such algorithm with a sublinear round complexity.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed Algorithms, Minimum Spanning Tree, Round Complexity, Message Complexity, Gossiping, Broadcast

Digital Object Identifier 10.4230/LIPIcs.DISC.2018.30

Acknowledgements We are grateful to Merav Parter for helpful discussions.

1 Introduction

This paper presents a distributed algorithm for computing a Minimum Spanning Tree (MST) with a nearly optimal round complexity and an improved message complexity. Our method also leads to improvements for two other basic problems, namely broadcast and gossiping. Let us start with briefly recalling the CONGEST model, which is the standard synchronous message-passing model of distributed computing with small messages:

The CONGEST Model [37]. The network is abstracted as a weighted graph $G = (V, E, w)$ where $n = |V|$, $m = |E|$. Moreover, we use D to denote the diameter of the graph. Initially, each node has a unique $\Theta(\log n)$-bit identifier and knows its own edges – i.e., the identifier of the other endpoint of the edges – as well as the weight of these edges. At the end, each node should know its own part of the output, e.g., which of its edges are in the computed minimum spanning tree. Per round, each node can send one $O(\log n)$-bit message to each of its neighbors. The round complexity of an algorithm is the number of rounds until all nodes are done with their computation, and the message complexity of the algorithm is the total number of messages sent throughout the algorithm.

History and Significance of MST in Distributed Algorithms. Minimum Spanning Tree (MST) is one of the central problems in the study of distributed graph algorithms, and it has been studied extensively since 1980s [12, 6, 11, 2, 13, 30, 39, 38, 8, 10, 26, 7, 28, 27, 16, 18, 36, 9, 32]. One can argue that much of the developments in the CONGEST model of distributed computing have been centered around (MST); the algorithmic or impossibility techniques developed for MST led to important results for other fundamental graph problems.

The work on distributed MST started with the algorithm of Gallager, Humblet, and Spira [12], which has time complexity $O(n \log n)$, and can be seen as a variant of the 1926 algorithm of Boruvka [35]. The time complexity was gradually improved [6, 11], eventually leading to the "*optimal*" $O(n)$-round algorithm of Awerbuch [2]. At the time, a round complexity of $O(n)$ was regarded as being optimal, because there are graphs of diameter $D = \Omega(n)$ on which one cannot do better (e.g., the n-node cycle).

A pioneering work of Garay, Kutten and Peleg [30, 13] shifted the are toward sublinear time algorithms in graphs of sublinear diameter, i.e., where this excuse of graphs with $D = \Omega(n)$ is ruled out. In particular, Garay et al. [13] presented an $O(D + n^{0.61})$-round MST algorithm, which was subsequently improved by Kutten and Peleg [30] to $O(D + \sqrt{n} \log^* n)$. Shortly after, Rubinovich and Peleg [38] proved a lower bound of $\tilde{\Omega}(D + \sqrt{n})$ for the round complexity of any distributed MST algorithm. By now, we understand that a wide range of other fundamental graph problems can be solved or approximated in time (close to) $\tilde{O}(D + \sqrt{n})$: the list includes minimum cut [14, 34], single-source shortest path [31, 33, 24, 4], tree embedding [19], maximum s-t flow [17], and minimum connected dominating set [15]. Moreover, this time complexity is optimal for essentially all of these problems [7]. Many of these results build on the methods and results developed initially for MST.

Message Complexity Comes Back. In the above line of work, the primary focus has been the round complexity. However, now that we understand the round complexity aspect of MST rather well, there has been a revived interest in getting algorithms with improved message complexity. The time-optimal $\tilde{O}(D + \sqrt{n})$-round algorithm of Kutten and Peleg has message complexity $\tilde{O}(m + n^{3/2})$. In a recent work, Pandurangan et al. [36] provided the first (randomized) MST algorithm with round complexity $\tilde{O}(D + \sqrt{n})$ and message complexity $\tilde{O}(m)$. Shortly after, Elkin [9] presented a simpler deterministic algorithm with similar time and message complexities (in fact with improvements in the logarithmic factors).

By a result of Awerbuch et al. [3], this $\tilde{O}(m)$ message complexity is known to be optimal for deterministic algorithms. Moreover, it is the best possible if either we restrict ourselves to randomized *comparison-based* algorithms [3] or to the variant of the distributed model where at the beginning, each node does not know its neighbors [3, 29] (this is sometimes called the KT0 variant of the model, as opposed to the KT1 version where at the beginning nodes know their neighbors). However, if at the beginning, each node knows all its neighbors and we are allowed to use general randomized algorithms, this $\tilde{\Omega}(m)$ message complexity lower bound does not apply. In fact, a beautiful result of King, Kutten, and Thorup [27] presents an algorithm with message complexity $\tilde{O}(n)$, however at the expense of having time complexity $\tilde{O}(n)$.

Open Question. This state of the art exhibits one clear open question:

Can we find a time-optimal MST algorithm with message complexity $o(m)$?

1.1 Our Results

MST. Our primary result in this paper is to provide a positive answer to the above open question. In particular, we prove that

▶ **Theorem 1.** *There is a distributed randomized MST algorithm with round complexity* $\tilde{O}(D + \sqrt{n})$ *and message complexity* $\tilde{O}(\min\{n^{3/2}, m\})$.

Broadcast. Along the way to this MST algorithm, we find a single-message broadcast algorithm – which can deliver a message from any source to all nodes – with an improved message complexity. We find this to be important on its own, given the centrality of the broadcast problem throughout distributed computing.

▶ **Theorem 2.** *There is a distributed randomized algorithm that broadcasts one message to all nodes in* $\tilde{O}(D + \sqrt{n})$ *using* $\tilde{O}(\min\{n^{3/2}, m\})$ *messages.*

We note that prior to this result, all known broadcast algorithms needed to use at least $\Omega(m)$ messages (in which case a simple flooding delivers the message to all in $O(D)$ rounds), with only one exception: the only known broadcast algorithm with message complexity $o(m)$ would need to use $\Omega(n)$ rounds [27].

Gossiping. The method that we develop for a message-efficient MST algorithm has one more significant consequence: it leads to the first sublinear-time GOSSIP algorithm in general graphs with bounded size messages [21, 5, 22]. In this problem, initially one node knows an $O(\log n)$-bit message, which should be delivered to all nodes. Per round each node can either PUSH an $O(\log n)$-bit message to an arbitrary neighbor or PULL an $O(\log n)$-bit message from an arbitrary neighbor.

▶ **Theorem 3.** *There is a distributed randomized algorithm that broadcasts one message to all nodes in* $\tilde{O}(\sqrt{nD})$ *rounds of the* GOSSIP *model.*

2 Preliminaries

In this section, we introduce two tools that will be used throughout our algorithms: one is an adaptation of the well-known 1926 algorithm of Boruvka for computing a minimum spanning tree [35], and the other is a *randomized linear sketching* tool.

Boruvka's algorithm. The algorithm is made of $\Theta(\log n)$ phases, where we gradually merge fragments (i.e., subtrees) of the MST with each other until we have exactly one fragment, i.e., the MST. Initially, each node is its own fragment. Per phase, we do as follows: Each fragment C picks the lightest edge that connects C to nodes outside C. It is well-known and easy to see that all these lighest edges, one per fragment, belong to the MST. We will add only some of these edges to the MST, in a way that ensures that the corresponding fragment merges have a small depth (in a sense that will become clear soon); this allows us to perform these merges efficiently. In particular, each fragment throws a Head/Tails fair coin. Then, for each edge $e = (v, u)$ which is the lightest edge of the fragment $C \ni v$ who connects to fragment $C' \ni u$, we *accept* edge e for a merge if and only if C has a Tail coin toss and C' has a Head coin toss. The accepted merge edges get added to the output MST. As a result, we perform a merge along each accepted merge edge by unifying the fragments connected via accepted merge edges, all at the same time. Notice that each new fragment is formed by merging a number of previous fragments in a star shape. That is, the new fragment is made of a number Tail old fragments which merge with one central Head old fragment. Then, we proceed to the next phase. It is known that after $\Theta(\log n)$ phases, with high probability, we have exactly one fragment which is the final MST. See e.g. [16] for the correctness proof of this method.

We also note that the algorithm can be extended directly to disconnected graphs, in which case it computes a minimum spanning tree in each connected component. Thus, overall, it provides a maximal forest with minimum weight.

Linear Sketching. The linear sketching tool that we will use was first used by Ahn et al. [1] in streaming algorithms. By now, variants of it have been used in various distributed algorithms [23, 27, 20, 25]. In a rough sense, this tool creates a "sketch" of a large set of elements (particularly, edges in our applications) which has only a few bits, and such that out of these few bits, we can still decode one element of the set. Since the sketch is linear (concretely, a bit-wise XOR), adding the sketches of two sets gives a linear sketch of their symmetric set difference. This last property will be important for our application. We can abstract this sketching as the following lemma.

▶ **Lemma 4** (Linear Sketching). *Consider a set E of elements, each with a unique $\Theta(\log n)$ bit identifier. There is a family $\mathcal{F}$ of encoder-decoder function pairs, where $|F| = 2^{\Theta(\log^3 n)}$, with the following properties: Here, in each pair in $\mathcal{F}$, the encoder is a function $ENC : E \to \{0,1\}^{\Theta(\log^3 n)}$ and the decoder is a function $DEC : \{0,1\}^{\Theta(\log^3 n)} \to E$. The family $\mathcal{F}$ is such that if we pick one pair of encoder function ENC and the corresponding decoder function DEC at random from $\mathcal{F}$, then for each set $S \in E$, we have $DEC(\oplus_{e \in S} ENC(e)) \in S$, with probability at least $1 - 1/n^5$.*

3 Minimum Spanning Tree

In this section, we prove the following result, which is a more detailed version of Theorem 1 and provides a time-optimal MST algorithm, up to logarithmic factors, with an improved message complexity.

▶ **Theorem 5** (Sublinear Messages & Near-Optimal Time). *There is a randomized distributed algorithm that in any n-node m-edge weighted graph $G = (V, E, w)$ with diameter D computes an MST with round complexity $\tilde{O}(D + \sqrt{n})$ and message complexity $\widetilde{O}(\min\{n^{3/2}, m\})$, with high probability.*

Roadmap. The algorithm for computing this MST is made of two parts. We first compute a sparser subgraph which has at most $\tilde{O}(n^{3/2})$ edges and still the diameter of it does not exceed $\tilde{O}(D + \sqrt{n})$. Moreover, this subgraph is such that we are able to compute it in $\tilde{O}(D + \sqrt{n})$ rounds and using $\tilde{O}(n^{3/2})$ messages. Then, we can easily compute a breadth first tree T of this sparse subgraph, which has depth at most $\tilde{O}(D + \sqrt{n})$, in $\tilde{O}(D + \sqrt{n})$ rounds and using $\tilde{O}(\min\{n^{3/2}, m\})$ messages. This part is captured by Lemma 6. This lemma itself has the direct corollary for the boardcast problem, it proves Theorem 2, showing that we can perform a global broadcast in $\tilde{O}(D + \sqrt{n})$ rounds and using $\widetilde{O}(n^{3/2})$ messages. To finish the proof of Theorem 5, we then explain how to use the low-depth tree T constructed by Lemma 6 to compute the MST, in $\tilde{O}(D + \sqrt{n})$ rounds and using $\tilde{O}(n)$ messages. This second part is captured by Lemma 7. Putting Lemma 6 and Lemma 7 together proves Theorem 5.

Finally, we comment that one can generalize Lemma 7 and prove that for every $\epsilon \in [0, 1/2]$, there exists a randomized algorithm for constructing MST using $\widetilde{O}(D + n^{1-\epsilon})$ rounds and $\widetilde{O}(\min\{n^{1+\epsilon}, m\})$ messages.

▶ **Lemma 6** (Sparsification Lemma). *There is a distributed algorithm that in any n-node m-edge weighted graph $G = (V, E, w)$ with diameter D, computes a spanning subgraph that has diameter $\tilde{O}(D + \sqrt{n})$ and $\tilde{O}(n^{3/2})$ edges, in $\tilde{O}(D + \sqrt{n})$ rounds and using $\widetilde{O}(\min\{n^{3/2}, m\})$ messages. Then, we can also compute a spanning tree of diameter $\tilde{O}(D+\sqrt{n})$, in $\tilde{O}(D+\sqrt{n})$ rounds and using $\widetilde{O}(\min\{n^{3/2}, m\})$ messages.*

Proof. Before describing the algorithm, we comment that the algorithm is described with focus on graphs that have $\Omega(n^{3/2})$ edges; for these, we show that our algorithm will use $\tilde{O}(n^{3/2})$ messages. The algorithm is such that it passes at most poly$(\log n)$ messages through each edge and therefore, it automatically has message complexity at most $\tilde{O}(m)$. Hence, we can write the message complexity as $\widetilde{O}(\min\{n^{3/2}, m\})$.

Heavy and Light Nodes and the Heavy Subgraph G'. Call a node *heavy* if its degree exceeds $\sqrt{n}$ and *light* otherwise. Let G' be the subgraph of G induced by heavy vertices, i.e., the subgraph made of all edges that both of their endpoints are heavy. We call this the Heavy Subgraph. Notice that $(G \setminus G')$ has at most $n^{3/2}$ edges.

Step 1 – Forming Stars. Define S to be a random subset of all vertices where each node is included in S with probability $\frac{10 \log n}{\sqrt{n}}$. We add all these vertices and their edges to the so-called heavy subgraph G'. It is easy to see that, w.h.p., $|S| = O(\sqrt{n} \log n)$ and moreover, each heavy node has at least one neighbor in S. Make each node in S send a message to each of its neighbors. This uses $O(n^{3/2} \log n)$ messages overall. Then, make each heavy node $v \notin S$ that receives a message from a neighbor in S pick one such neighbor as its center. For each node $s \in S$, this defines a *star* subgraph centered at s along with some of its neighbors. These stars are disjoint for different centers and they satisfy the following guarantees: We have $O(n^{3/2} \log n)$ stars and each heavy node is in exactly one star.

Step 2 – Boruvka on the Heavy Subgraph, and Starting From the Stars. Now, we run Boruvka's algorithm – as explained in Section 2 – for computing a certain maximal spanning forest F of G', with the initial configuration that each star is one fragment of F. We remark that here we can ignore the edge weights as we are interested in simply one maximal spanning forest, with no regard for the weights. Hence, per phase, for each component, it suffices to find one outgoing edge for each fragment C of F. Notice that the naive way for detecting whether an edge $e = u, v$ incident on a node $v \in C$ is outgoing or not, i.e., whether $u \in C$ or not, would require communicating through this edge e, which overall can lead to $O(m)$ message complexity. To circumvent that, we follow a *linear sketching* method of King, Kutten, and Thorup [27]. The key aspect of this linear sketching is abstracted by Lemma 4.

We make the center of fragment C pick one sketch (i.e., a pair of encoder and decoder functions) from the family $\mathcal{F}$ of Lemma 4 at random and we deliver the $\Theta(\log^3 n)$ bits of the description of $\mathcal{F}$ to all nodes of C, via a simple broadcast along the tree of fragment C. Then, each node v computes $ENC(e)$ for each edge e incident on v. Then, we aggregate a bit-wise XOR of these values $END(e)$, through a convergecast, at the root of the fragment. Notice that for each internal edge $e = u, v$ which has both u and v in fragment C, the related encoding $ENC(e)$ gets added to the computed bit-wise XOR twice, once from each endpoint. Hence, it gets canceled out. On the other hand, $ENC(e)$ for each outgoing edge e is added exactly once to the bit-wise XOR and therefore, it does not get canceled out. By Lemma 4, once the fragment center receives this bit-wise XOR, that is $\oplus_{e \in O} ENC(e)$ where O denotes the set of edges going out of fragment C, it can decode it using the function $DEC()$. Thus, the fragment center then obtains one outgoing edge $e \in O$. We then broadcast this outgoing edge to all vertices of the fragment. In particular the endpoint of e in C knows that its edge e is chosen as a potential edge for a merge.

The above process explains how we compute one outgoing edge for each fragment (unless the fragment already fully spans its component and no edge can be added to it, in which case the sketch shows an empty set of outgoing edges). Moreover, it uses $\tilde{O}(n)$ messages overall because we are simply performing $O(1)$ convergecasts and broadcasts in the tree of each fragment, each of them carrying poly$(\log n)$ bit messages, and all the trees together have at most $n-1$ edges. Furthermore, this procedure has round complexity equal to the maximum fragment diameter, up to logarithmic factors. Since we have $\tilde{O}(n^{1/2})$ stars and each edge of G' is incident on one of the stars, and because in each fragment all nodes of each star also have all of their star edges as a part of the fragment, the maximum fragment diameter is $\tilde{O}(n^{1/2})$. This completes the process of selecting outgoing edges.

Then, selecting the accepted outgoing edges for merge, using the related Head/Tail coins of the fragments, and performing the corresponding merges can be done similarly with simple broadcasts and convergecasts, same as described in the description of Boruvka's algorithm in Section 2. This again takes $\tilde{O}(n^{1/2})$ rounds and $\tilde{O}(n)$ messages. After going through all the $\Theta(\log n)$ phases of Boruvka, we get a maximal spanning forest F of G'. throughout step 2, we have used $\tilde{O}(n)$ messages and $\tilde{O}(n^{1/2})$ rounds.

Arguing that $(G \setminus G') \cup F$ is our desired sparse low-diameter spanning subgraph. Consider the spanning subgraph H with all edges of $(G \setminus G') \cup F$. This subgraph clearly has at most $\tilde{O}(n^{3/2})$ edges. We next argue that it has diameter at most $\tilde{O}(D + \sqrt{n})$. Consider two arbitrary nodes $v, u \in G$. We prove that there is a path of length at most $\tilde{O}(D + \sqrt{n})$ in H between u and v. Suppose that in G, we contract each connected component of F into a single node, and call the resulting graph H'. There must still be a path of length at most D between u and v in this contracted graph H'. Let $P_{u,v}$ be the shortest path in H' between u and v. Now we can transform $P_{u,v}$ to a path of length $D + O(\sqrt{n}\log n)$ in H as follows: any contracted part of the path $P_{u,v}$ can be "un-contracted" – essentially undoing the process of contraction – and replaced with the portion of the maximal forest F that spans that component of F. This step increases the length of the path by at most a constant factor of the number of stars in that connected component of F. Since the total number of stars is $O(\sqrt{n}\log n)$, over all the components, this "un-contracting" process increase the path length from D to at most $D + O(\sqrt{n}\log n)$. This path now exists in $H = (G \setminus G') \cup F$. Since this argument holds for any two arbitrary nodes $v, u \in G$, we get that the spanning subgraph H has diameter at most $\tilde{O}(D + \sqrt{n})$.

Computing a Low-Diameter Spanning Tree. Once we have H, we can also perform a breadth first search (BFS) in H, which finds a spanning tree (of G) that has diameter $\tilde{O}(D + \sqrt{n})$. Since H has at most $\tilde{O}(\min\{n^{3/2}, m\})$ edges, the construction of this BFS tree uses at most $\tilde{O}(\min\{n^{3/2}, m\})$ edges, thus proving the second part of the lemma. ◀

We next describe our MST computation method which we shall use with the help of the computed low-diameter spanning tree provided by Lemma 6.

▶ **Lemma 7** (MST Computation). *Suppose that we are given a spanning tree with depth $d = \tilde{O}(D + \sqrt{n})$ of the graph. Then, we can compute the MST in $\tilde{O}(D + \sqrt{n})$ rounds and using $\tilde{O}(n)$ messages.*

Proof Sketch. We will use Boruvka's MST algorithm, combined with the linear sketching method of King et al. [27] for finding the lightest outgoing edge of each fragment, as well as an idea of Elkin [9]. Roughly speaking, the latter will allow us to primarily work on low-depth fragments, except for switching to computation on the global spanning tree, at the end. The algorithm is made of two parts, which we explain next.

Part 1 – MST Growth With Low-Diameter Fragments. During the first part, we grow a partial forest T of the MST. This will be done such that at the end, each fragment of T has diameter in $[d, 5d]$. We always make sure not to have any fragment of diameter exceeding $5d$. Initially, T is the trivial forest made of each node as its own fragment. Then, throughout $\Theta(\log n)$ phases, we merge some fragments with each other similar to Boruvka's algorithm explained in Section 2, but with some modification: In each phase, we do as follows. Any fragment C of diameter at most d is called *active* and each other fragment is called *inactive*. Each inactive fragment C will not initiate a merge, that is, it will not compute its lightest outgoing edge to propose it to be added to MST. But C is still receptive to merges, i.e., if an active fragment C' proposes a merge with C (and C has a Head coin toss and C' has a Tail coin toss), we accept this merge.

Let us discuss how we compute the lightest outgoing edge for each active component. This is done via a binary search through the range of the weights. Each time if the active search range for fragment C is $[L, U]$, we sketch all edges incident on C with weight in $[L, \frac{L+U}{2}]$ and deliver this sketch to the root of fragment C using a convergecast of the XOR of sketches (similar to proof of Lemma 6). If this sketch indicates an empty edge-set, then the binary search zooms into search range $[\frac{L+U}{2}, U]$. Otherwise, it zooms into $[L, \frac{L+U}{2}]$ as the search range. In either case, the upper and lower bound of the search range are then broadcast to all nodes of the fragment C. After $O(\log n)$ steps of the binary search, each of which takes $\tilde{O}(d)$ rounds, we find the lightest outgoing edge of C and this edge is known to all nodes of C. Then, the merge operations are similar to Section 2, along edges that go from Tail coin active fragments to Head coin fragments.

Once these merges happen, the diameter of the resulting fragment might exceed $5d$. However, it still will be at most $7d + 2 = O(d)$. This is because the new fragment is made of a star merge centered at one fragment of diameter at most $5d$ with a number of side fragments, each of diameter at most d. We then spend $\Theta(d)$ rounds to truncated the fragment to diameter at most $5d$. In particular, for each fragment that has diameter exceeding $5d$, we select some edges of it to be discarded from the current forest such that each remaining fragment has diameter in $[d, 5d]$. This can be done in $O(d)$ rounds as the next paragraph explains.

To cut this component tree into fragments with diameter in $[d, 5d]$, we do as follows: first, perform a broadcast on the fragment tree so that each node knows its distance from the root of the fragment. Then, for each node v with distance an integer multiple of $(d + 1)$, the edge of that node to its parent will be discarded from the forest. This ensures that each fragment of the remaining forest has diameter at most $2d$. However, an unfortunate side-effect is that there might be some nodes v whose edge to its parent gets discarded and such that the subtree rooted at v has a height smaller than d. We can detect such nodes using a simple convergecast from the leaves toward the root. For every such node v, we undo the step of discarding the edge connecting v to its parent and thus effectively we merge back its subtree with its parent fragment. This might increase the depth of its parent fragment but the corresponding diameter cannot exceed $4d$.

Now that the fragments are truncated to have diameter in $[d, 5d]$, we proceed to the next phase. After $\Theta(\log n)$ phases, the first part of the algorithm is done. At that point, we have reached a setting where we have a forest F made of $O(n/d)$ fragments, each with diameter in $[d, 5d]$. Moreover, all edges of this forest belong to the minimum spanning tree. Overall, this first part uses $\tilde{O}(d)$ rounds and $\tilde{O}(n)$ messages.

Part 2 – MST Growth Beyond Low-Diameter Fragments. Now, the second part of the algorithm starts. Here, we grow the eventual MST T, but using a different method. Initially, we set $T = F$. The forest T will grow over time while F is maintained as is, permanently. We again will have $\Theta(\log n)$ Boruvka-style phases of growing the fragments of T. At each time, each fragment of T is made of a number of fragments of F.

To compute the lightest outgoing edge of each fragment t of T, we do the corresponding binary search differently. Suppose the current weight range of binary search for t is some range $[L, U]$, which is known to all nodes of t. We then make each of the fragments $f \in F$ which are a part of t compute its own sketch of edges incident on f with weight in $[L, \frac{L+U}{2}]$ using a convegecast inside f. This results in one $\Theta(\log^3 n)$-bit sketch for the whole fragment f. Then, we broadcast all these sketch, one for each fragment of F, to the root of the whole graph using the given spanning tree of depth d. Since each fragment of F sends one $\Theta(\log^3 n)$-bit message, F has $O(n/d)$ fragments, and each message travels d hops to reach the root of the spanning tree, the overall message complexity is $\tilde{O}(n/d \cdot d) = \tilde{O}(n)$. Once the root receives all these sketches, it can compute one outgoing edge for each fragment t of T (if there was some edge in the corresponding search range and the sketch was not empty). Then, the root can broadcast these edges back to the nodes of t by simply putting the identifier and weight of that found edge in the message that arrived from each subfragment $f \in F$ of t and reversing the schedule of the convergecast, now delivering messages from the root to all nodes. This completes one step of the binary search. After $O(\log n)$ such steps of the binary search, the root knows the lightest outgoing edge of each fragment t of T. Thus, the root can perform one phase of Boruvka-style merges on T. It then updates T accordingly, by merging the related fragments, and reports the new fragment IDs to all nodes. This is again done by putting this T-fragment ID back in the message that came from the corresponding subfragment $f \in F$ of t and reversing the schedule of the convergecast, after which a broadcast inside fragment $f \in F$ delivers it to all nodes of that fragment. This finishes the description for one phase of Boruvka.

Repeating a similar process for $\Theta(\log n)$ phases finishes the computation of the MST. The round complexity of the process is $\tilde{O}(d)$, since each phase is made of poly$(\log n)$ many broadcasts and convergecasts, each it trees of diameter d. Moreover, the message complexity is $\tilde{O}(n)$, again because the convergecasts and broadcasts inside fragments of F clearly use at most $\tilde{O}(n)$ messages, and the broadcast and converecasts in the overall spanning tree use $\tilde{O}(n/d \cdot d) = \tilde{O}(n)$ messages. ◀

4 Fast Gossiping with Bounded-Size Messages

In this section, we prove the following results, which is a more detailed version of Theorem 3.

▶ **Theorem 8.** *There is a distributed algorithm that spreads a rumor from one node to all in $\tilde{O}(\sqrt{nD})$ rounds of the* GOSSIP *model, where per round each node initiates one* PUSH *or* PULL *contact with one of its neighbors, communicating an $O(\log n)$ bit message.*

Proof. We first find a sparse spanning subgraph and then perform a gossiping broadcast on this subgraph. The sparsity of the subgraph allows us to have a fast algorithm in the GOSSIP model, as it enables each node to contact only a few other nodes. To build the sparse subgraph, the general method is mostly similar to the one used in Lemma 6. However, we now need to adapt the construction method to the GOSSIP model and analyze the resulting round complexity. Here, we explain the changes.

Step 1 – Forming Stars. Now, we call a node heavy if its degree exceeds $\sqrt{n/D}$ and light otherwise. The subgraph G' made of all edges induced by heavy vertices is called the heavy subgraph. We pick $10\sqrt{nD}\log n$ nodes at random in expectation to be the set S of centers of the stars – i.e., we include each node of the graph in S randomly with probability $10\sqrt{D/n}\log n$. To form the stars, each (heavy) node goes through its neighbors and pulls from them, one by one, until finding the first node who is chosen to be a star. Notice that each heavy node will find a star within $O(\sqrt{n/D})$ pulls, with high probability. Moreover, each contacted node can always respond whether it is star center or not.

Step 2 – Boruvka on the Heavy Subgraph, and Starting From the Stars. Having formed these stars, the algorithm continues essentially the same as in Lemma 6 to build a maximal forest F of the graph G' made of all edges with at least one heavy endpoint (and the corresponding nodes). There is only one comment in order: We should explain that we can perform the convergecast and broadcasts of Boruvka's algorithm in the GOSSIP model, because these trees do not necessarily have small degrees. Suppose that we have a rooted tree of depth d for each fragment of the forest and each node knows its depth in this tree. Then, we can perform a broadcast (sending a single-message from the root to all) or convergecast (e.g., computing one aggregate function such as sum by sending messages from the leaves up towards the root) in this tree in time $O(d)$ also in the GOSSIP model. In the case of broadcast, in the i^{th} round, each node v at depth i makes a PULL contact to its parent. A simple induction shows that each node at depth i receives the message in round i. A convergecast is similar; in the i^{th} round each node v at depth $d-i$ makes a PUSH contact to its parent, sending the message of the convergecast to its parent. Again, via a simple induction, we can prove that after d rounds the root receives the result of the convegecast. Having this broadcast and convergecast, we can build the maximal forest F of the subgraph G' similar to Lemma 6. At the end, each component of this maximal forest F is rooted in one of its nodes and each node of that component knows its deoth in the rooted tree of that component. Since we now have $O(\sqrt{nD}\log n)$ stars, the diameter of each fragment is at most $O(\sqrt{nD}\log n)$. Therefore, also the construction of F finishes in $O(\sqrt{nD}\log n)$ rounds of the GOSSIP model.

Gossiping on the Sparse Subgraph. Now, we use the spanning subgraph defined by $(G \setminus G') \cup F$ to perform the gossiping of one message to all nodes. For that, we divide time into phases, each made of two rounds. In odd rounds, each informed node – who already has the gossiping message – picks one of its edges at random – if it has any – and pushes the message through that edge and each uninformed node pulls from one of its neighbors chosen at random. In even rounds, each uninformed node pulls from its parent in the forest F and each informed node pushes to its parent.

We now argue that the message reaches all nodes in $O(\sqrt{nD}\log n)$ rounds, with high probability. Let v be the source of the gossiping, i.e., the node that initially holds the gossip message. As in the proof of Lemma 6, we can see that for every vertex u, there is a path $P_{v,u}$ of length $D + O(\sqrt{nD}\log n)$ in $(G \setminus G') \cup F$ connecting v to u. Let us say edge $\{w_1, w_2\} \in P_{v,u}$ is in the *waiting mode* during all the rounds that w_1 is the closest informed node on $P_{v,u}$ to u. In articular, during these times w_1 is informed and w_2 is not informed. We analyze the time that the edges of $P_{v,u}$ spend in the waiting mode, in two parts, one for $(G \setminus G')$ and one for F. We show that the summation of the time that different edges of $P_{u,v}$ spend in the waiting mode is at most $O(\sqrt{nD}\log n)$ rounds. Hence, the message reaches from v to u in $O(\sqrt{nD}\log n)$ rounds. Notice that by definition, per round at most one edge of $P_{v,u}$ is in the waiting mode.

Similar to the proof of the last part of Lemma 6, we can see that at most D hops of the path $P_{v,u}$ are edges of $(G \setminus G')$ and the rest are edges of F. Now, because of the odd rounds, whenever a low-degree node w_1 gets informed, each of its neighbors w_2 in the subgraph $G \setminus G'$ gets informed with high probability in $O(\sqrt{n/D})$ rounds. Hence, the message travels one hop through $G \setminus G'$ per $O(\sqrt{n/D})$ rounds. Let us say edge $\{w_1, w_2\} \in (G \setminus G')$ is in the *waiting mode* during all the rounds that w_1 is informed but w_2 is not informed (or vice verse). Thus, each edge is in the waiting mode at most $O(\sqrt{n/D})$ rounds, w.h.p. Overall, the time that the message spends waiting to travel through the edges of $P_{v,u} \cap (G \setminus G')$ is at most $D \cdot O(\sqrt{n/D})$. Now, we focus on the waiting time of edges of F.

Because of the even rounds, when the message reaches some node of a fragment of F, it gets pushed to the root and then pulled to all the nodes of the fragment, in a number of phases asymptotically equal to the depth of that forest fragment. Notice that this depth is upper bounded asymptotically by the number of stars in that fragment. Hence, the total waiting time of the edges of $P_{v,u}$ in that fragment is upper bounded asymptotically by the number of stars in that fragment. Since overall the number of stars is at most $O(\sqrt{nD}\log n)$, the message spends at most $O(\sqrt{nD}\log n)$ rounds going through the edges of $P_{u,v} \cap F$. Hence, overall the time the message takes to reach from v to u is at most $D \cdot O(\sqrt{n/D}) + O(\sqrt{nD}\log n) = O(\sqrt{nD}\log n)$. ◀

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 5–14. ACM, 2012.

2 Baruch Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 230–240. ACM, 1987.

3 Baruch Awerbuch, Oded Goldreich, Ronen Vainish, and David Peleg. A trade-off between information and communication in broadcast protocols. *Journal of the ACM (JACM)*, 37(2):238–256, 1990.

4 Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 91. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

5 Keren Censor-Hillel, Bernhard Haeupler, Jonathan Kelner, and Petar Maymounkov. Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 961–970. ACM, 2012.

6 F Chin and HF Ting. An almost linear time and O(nlogn+ e) messages distributed algorithm for minimum-weight spanning trees. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages 257–266. IEEE, 1985.

7 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 *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 363–372, 2011.

8 Michael Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 331–340, 2004.

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

10 Michalis Faloutsos and Mart Molle. A linear-time optimal-message distributed algorithm for minimum spanning trees. *Distributed Computing*, 17(2):151–170, 2004.

11 Eli Gafni. Improvements in the time complexity of two message-optimal election algorithms. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 175–185. ACM, 1985.

12 Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and systems (TOPLAS)*, 5(1):66–77, 1983.

13 J.A. Garay, S. Kutten, and D. Peleg. A sub-linear time distributed algorithm for minimum-weight spanning trees. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, 1993.

14 M. Ghaffari and F. Kuhn. Distributed minimum cut approximation. In *Proc. of the Int'l Symp. on Dist. Comp. (DISC)*, pages 1–15, 2013.

15 Mohsen Ghaffari. Near-optimal distributed approximation of minimum-weight connected dominating set. In *International Colloquium on Automata, Languages, and Programming*, pages 483–494. Springer, 2014.

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*, pages 202–219. SIAM, 2016.

17 Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 81–90. ACM, 2015.

18 Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed mst and routing in almost mixing time. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 131–140. ACM, 2017.

19 Mohsen Ghaffari and Christoph Lenzen. Near-optimal distributed tree embedding. In *International Symposium on Distributed Computing*, pages 197–211. Springer, 2014.

20 Mohsen Ghaffari and Merav Parter. Mst in log-star rounds of congested clique. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 19–28. ACM, 2016.

21 George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *Symposium on Theoretical Aspects of Computer Science (STACS2011)*, volume 9, pages 57–68, 2011.

22 Bernhard Haeupler. Simple, fast and deterministic gossip and rumor spreading. *Journal of the ACM (JACM)*, 62(6):47, 2015.

23 James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 91–100. ACM, 2015.

24 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings*

of the forty-eighth annual ACM symposium on Theory of Computing, pages 489–498. ACM, 2016.

25 Tomasz Jurdziński and Krzysztof Nowicki. Mst in o (1) rounds of congested clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2620–2632. SIAM, 2018.

26 Maleq Khan and Gopal Pandurangan. A fast distributed approximation algorithm for minimum spanning trees. *Distributed Computing*, 20(6):391–402, 2008.

27 Valerie King, Shay Kutten, and Mikkel Thorup. Construction and impromptu repair of an mst in a distributed network with o (m) communication. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 71–80. ACM, 2015.

28 Liah Kor, Amos Korman, and David Peleg. Tight bounds for distributed mst verification. In *Symposium on Theoretical Aspects of Computer Science (STACS2011)*, volume 9, pages 57–68, 2011.

29 Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. On the complexity of universal leader election. *Journal of the ACM (JACM)*, 62(1):7, 2015.

30 Shay Kutten and David Peleg. Fast distributed construction of k-dominating sets and applications. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 238–251, 1995.

31 Christoph Lenzen and Boaz Patt-Shamir. Fast routing table construction using small messages: Extended abstract. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 381–390, 2013.

32 Ali Mashreghi and Valerie King. Time-communication trade-offs for minimum spanning tree construction. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*, page 8. ACM, 2017.

33 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proc. of the Symp. on Theory of Comp. (STOC)*, 2014.

34 Danupon Nanongkai and Hsin-Hao Su. Almost-tight distributed minimum cut algorithms. In *International Symposium on Distributed Computing*, pages 439–453. Springer, 2014.

35 Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1):3–36, 2001.

36 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 SIGACT Symposium on Theory of Computing*, pages 743–756. ACM, 2017.

37 David Peleg. *Distributed Computing: A Locality-sensitive Approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

38 David Peleg and Vitaly Rubinovich. A near-tight lower bound on the time complexity of distributed MST construction. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages 253–, 1999.

39 Gurdip Singh and Arthur J Bernstein. A highly asynchronous minimum spanning tree protocol. *Distributed Computing*, 8(3):151–161, 1995.