A Simple Deterministic Distributed MST Algorithm, with Near-Optimal Time and Message Complexities

Michael Elkin[*1]

[1]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
Email: {elkinm}@cs.bgu.ac.il

Abstract

Distributed *minimum spanning tree* (MST) problem is one of the most central and fundamental problems in distributed graph algorithms. Garay et al. [GKP98, KP98] devised an algorithm with running time $O(D + \sqrt{n} \cdot \log^* n)$, where D is the hop-diameter of the input n-vertex m-edge graph, and with message complexity $O(m + n^{3/2})$. Peleg and Rubinovich [PR99] showed that the running time of the algorithm of [KP98] is essentially tight, and asked if one can achieve near-optimal running time *together with near-optimal message complexity*.

In a recent breakthrough, Pandurangan et al. [PRS16] answered this question in the affirmative, and devised a *randomized* algorithm with time $\tilde{O}(D + \sqrt{n})$ and message complexity $\tilde{O}(m)$. They asked if such a simultaneous time- and message-optimality can be achieved by a *deterministic* algorithm.

In this paper, building upon the work of [PRS16], we answer this question in the affirmative, and devise a *deterministic* algorithm that computes MST in time $O((D + \sqrt{n}) \cdot \log n)$, using $O(m \cdot \log n + n \log n \cdot \log^* n)$ messages. The polylogarithmic factors in the time and message complexities of our algorithm are significantly smaller than the respective factors in the result of [PRS16]. Also, our algorithm and its analysis are very *simple* and self-contained, as opposed to rather complicated previous sublinear-time algorithms [GKP98, KP98, Elk04b, PRS16].

1   Introduction

1.1   Background and New Results

Distributed *minimum-weight spanning tree* (henceforth, MST) problem is one of the most funda-
mental and extensively studied problems in distributed graph algorithms [GHS83, CT85, Gaf85,
Awe87, SB95, GKP98, KP98, PR99, Elk04a, Elk04b, FM04, KP08, KKP11, KKT15, PRS16,
MK17]. The seminal work of Gallager et al. [GHS83] gave an algorithm with running time
$O(n \log n)$ and message complexity $O(m + n \log n)$ for the problem, where $n = |V|$ and $m = |E|$
are the number of vertices and edges of the input graph $G = (V, E)$, respectively. The time com-
plexity was then improved to $O(n)$ [CT85, Gaf85, Awe87, FM04], while still retaining the bound
of $O(m + n \log n)$ on the number of messages.

Garay et al. [GKP98, KP98] devised an algorithm with running time $O(D + \sqrt{n} \cdot \log^* n)$, where
D is the hop-diameter (equivalently, unweighted diameter) of G, albeit with message complexity
$O(m + n^{3/2})$. Peleg and Rubinovich [PR99] showed a lower bound of $\tilde{\Omega}(\sqrt{n})$ for the problem,[1] even
when $D = O(\log n)$. In the open problems section of their groundbreaking paper they raised the
question of devising a nearly time- and message-optimal algorithm:

"Another research direction is to try to reduce the communication *complexity of nearly time optimal
algorithm of [KP98] from $O(|E| + n^{3/2})$ towards the lower bound of $O(|E| + n \log n)$. "*

In a recent breakthrough, Pandurangan et al. [PRS16] devised a randomized algorithm with time
complexity $\tilde{O}(D + \sqrt{n})$ and message complexity $\tilde{O}(m)$. In the Conclusion section of their paper
they write:

*"An intriguing open question is whether randomization is necessary to simultaneously achieve time
and message optimality. "*

In this paper we answer this question, and devise a *deterministic* algorithm with running time
$O((D + \sqrt{n}) \cdot \log n)$, and message complexity $O(|E| \cdot \log n + n \log n \cdot \log^* n)$. In addition to being
deterministic, our algorithm is also drastically simpler than that of [PRS16].[2] Also, the polylog-
arithmic factors in the time and message complexities of our algorithm are significantly smaller
than the respective factors in [PRS16]. (Pandurangan et al. [PRS16] do not specify explicitly
these factors. However, since they are using an algorithm for constructing neighborhood covers
from [Elk04b], and the latter algorithm has running time $O(D \log^3 n)$ and message complexity
$O(m \cdot \log^2 n)$, these factors are definitely incurred by the algorithm of [PRS16]. Also, it is apparent
from their analysis that the $\sqrt{n}$ term in their time complexity is multiplied by at least $\log^2 n$.)

We also generalize our result to the $CONGEST(b \log n)$ model, for any positive integer pa-
rameter b. In this model the bandwidth of every edge is b edge weights and/or vertex identi-
ties. (See Section 2 for a formal definition.) We show that our algorithm can be implemented in
$O((D + \sqrt{\frac{n}{b}}) \cdot \log n)$ time, using $O(|E| + n \log n \cdot \log^* n)$ messages.

The lower bound for the time required to compute MST in the $CONGEST(b \log n)$ model is
$\Omega(D + \sqrt{\frac{n}{b \log n}})$ [Elk04a, PR99], i.e., our upper bound is $\Theta(\log n)$-off the lower bound in the first

---

[1]$\tilde{O}$, $\tilde{\Omega}$ and $\tilde{\Theta}$ notations hide factors polylogarithmic in n.

[2]Though we stress that it heavily builds upon several crucial ideas from [PRS16]; see more details below.

term and $\Theta(\log^{3/2} n)$-off in the second term, for all values of b. (In particular, this is also the gap in the standard $CONGEST$ model, i.e., when $b = 1$.)

The lower bound on message complexity, due to Awerbuch et al. [AGPV90], is $\Omega(|E|)$. The lower bound of [AGPV90] applies to deterministic algorithms, and also even to randomized comparison-based algorithms. It also applies to randomized not comparison-based ones, as long as they apply to the so-called *clean network model*. In the latter model, at the beginning of the computation every vertex v knows only its own identity number. On the other hand, if a vertex knows also (at the beginning of the computation) identities of all its neighbors, then a not comparison-based randomized algorithm of King et al. [KKT15] achieves message complexity of $\tilde{O}(n)$ (though their time complexity is not sublinear in n).

Our algorithm is deterministic, comparison-based, and applies to the clean network model. (Any one of these three properties makes the lower bound of [AGPV90] applicable.) Hence its message complexity is $O(|E| \log n + n \log n \cdot \log^* n)$ is optimal up to a $\log n$ factor in the first term, and a $\log n \cdot \log^* n$ factor in the second.

1.2 Technical Overview

The sublinear-time MST algorithm of [KP98, GKP98] consists of two phases. In the first phase one constructs an *MST forest*, i.e., a collection of vertex-disjoint subtrees of the same fixed MST, that cover all vertices of the input graph $G = (V, E)$. These subtrees are called *fragments*. Moreover, each of these fragments in the algorithm of [KP98] has diameter $O(\sqrt{n})$, and there are $O(\sqrt{n})$ such fragments in the forest. The computation of this MST forest requires $\tilde{O}(\sqrt{n})$ time: it is done by an ingenious variant of Boruvka's algorithm.

At this stage there are only $O(\sqrt{n})$ MST edges missing. These are computed by a procedure, called Pipeline-MST [GKP98]. In this procedure one uses an auxiliary BFS tree τ of the input graph G. All candidate edges (i.e., crossing between different fragments) are pipelined towards the root rt of τ, but the key to efficiency is that every intermediate vertex v of τ filters out all candidate edges e that are discovered to be heaviest in some cycle. This (second) phase of the algorithm of [KP98] is responsible for its large message complexity, and it also involves heavy local computations.

The recent nearly message-optimal algorithm of [PRS16] also consists of two phases, where the first phase is the same as in the algorithm of [KP98]. However, on the second phase, the algorithm of [PRS16] employs a different strategy than that of [KP98]. Rather than using a communication-heavy Pipeline-MST procedure, they continue merging fragments via a Boruvka-type algorithm.

The problem with merging large-diameter fragments via Boruvka's algorithm is that a naive implementation of this merging requires time proportional to the diameter of these fragments. When $D = O(\sqrt{n})$, Pandurangan et al. [PRS16] overcome this problem by maintaining two MST forests at all times: one is the *base forest* $\mathcal{F}$ (and its fragments are referred to as *base fragments*), which was computed at the first phase of the algorithm. Recall that $\mathcal{F}$ consists of $O(\sqrt{n})$ fragments of size $O(\sqrt{n})$ each. (For a pair of parameters α, β, an (α, β)-*MST forest* is an MST forest with at most α fragments, each of diameter at most β. The base forest is an $(O(\sqrt{n}), O(\sqrt{n}))$-MST forest.)

The second MST forest $\hat{\mathcal{F}}$ that the algorithm of [PRS16] maintains is obtained by merging some of the base fragments into fragments of $\hat{\mathcal{F}}$ via Boruvka's algorithm. To compute the *minimum weight outgoing edge* (henceforth, MWOE) of a fragment $\hat{F} \in \hat{\mathcal{F}}$, the algorithm computes in each base fragment $F \in \mathcal{F}$ a minimum-weight edge e_F crossing between $V(F)$ and $V \setminus V(\hat{F})$, where $F \subseteq \hat{F}$, and $\hat{F} \in \hat{\mathcal{F}}$. Then the algorithm upcasts these edges e_F to the root rt of the auxiliary BFS tree τ.

The root rt uses this information to compute the MWOE $e_{\hat{F}}$ of every fragment $\hat{F} \in \hat{\mathcal{F}}$, and then to compute a new MST forest $\hat{\mathcal{F}}'$. The fragments of the latter forest are obtained by merging some of the fragments of $\hat{\mathcal{F}}$, via Boruvka's algorithm.

When $D \leq \sqrt{n}$, the procedure described above is both time- and message-efficient. However, generally, its message complexity is $\tilde{\Theta}(D\sqrt{n}+n)$, and this is super-linear for $D = \omega(\sqrt{n})$. To resolve this issue, [PRS16] employ hierarchies of sparse neighborhood covers [ABCP93, Coh93, Elk04b], and use them build what they call "communication-efficient fragments and paths" within large-diameter fragments. This results in a sophisticated and complicated algorithm, with an elaborate analysis, which incurs quite a few polylogarithmic factors in both time and message complexities, and requires storing certain non-trivial local data structures in every vertex. Moreover, since there are currently no known *deterministic* distributed time- and message-efficient algorithms for constructing neighborhood covers, the algorithm of [PRS16] resorts to using a *randomized* algorithm of [Elk04b]. As a result, the solution of [PRS16] becomes randomized as well.

In this paper we propose a different, and a much simpler solution, for the situation when $D \geq \sqrt{n}$. Instead of constructing an $(O(\sqrt{n}), O(\sqrt{n}))$-MST forest $\mathcal{F}$ as a base forest, we construct an $(O(n/D), O(D))$-MST forest. By slightly generalizing and refining the analysis of [KP98, Len16, PRS16], we show that this can be done in $O(D \cdot \log^* n)$ time, and with $O(|E| \cdot \log D + n \cdot \log D \cdot \log^* n)$ messages. By doing so we spend more time on the first phase than the algorithms of [KP98, PRS16]; this is however still well within our desired time bounds.

Then we use the algorithm of [PRS16] on top of this base forest, as opposed to using it on top of an $(O(\sqrt{n}), O(\sqrt{n}))$-MST forest. (Recall that, as was argued above, the latter would have not been message-efficient.) Now computing a minimum-weight edge e_F crossing between the vertex set $V(F)$ of a base fragment F and $V \setminus V(\hat{F})$, where $\hat{F} \in \hat{\mathcal{F}}$ is the fragment that contains F, can be done in $O(D + n/D) = O(D)$ time. Even more importantly, upcasting all these edges e_F to the root rt of the auxiliary tree τ requires now just $O(D \cdot n/D) = O(n)$ messages. As a result, the entire message complexity of our algorithm is near-linear.

As opposed to previous solutions, our entire algorithm and its analysis are ultimately very simple. In fact, we essentially provide all the details (including those which originate from previous work) in this extended abstract.

1.3 Related Work

Singh and Bernstein [SB95] devised an MST algorithm with near-optimal message complexity, and with running time $O((\Delta + Diam(MST)) \cdot \log n)$, where Δ is the maximum degree of the input graph $G = (V, E, \omega)$, [3] and $Diam(MST)$ is the hop-diameter of the computed MST of G. The latter parameter is always greater or equal to $D = Diam(G)$, but for many instances it is smaller than n.

The current author [Elk04b] devised an MST algorithm with running time $\tilde{O}(\mu(G, \omega) + \sqrt{n})$, where $\mu(G, \omega)$ is a parameter which is never greater than D, and for many instances it is much smaller than D. There is also a lower bound of $\Omega(\mu(G, \omega)) + \tilde{\Omega}(\sqrt{n})$ for the MST computation on an input graph (G, ω) [Elk04b]. Albeit, the algorithm of [Elk04b] does not detect termination (unless it is given an estimate of $\mu(G, \omega)$ as a part of the input).

Khan and Pandurangan [KP08] devised an $O(\log n)$-approximate MST algorithm with running time $\tilde{O}(D + L(G, \omega))$, where $L(G, \omega)$ is yet another parameter, called *local shortest path diameter*.

---

[3]$\omega : E \to \mathrm{R}^+$ is a weight function on edges of G.

It may be smaller or larger than D.

Lower bounds on the time required to compute an approximate MST were shown in [Elk04a, SHK+12, EKNP14]. In particular, [EKNP14] showed such lower bounds even when quantum distributed communication is allowed. Lower bounds for MST on graphs with constant hop-diameter D were shown in [LPP06, Elk04a]. MST on graphs with $D = 1$ (the *Congested Clique* model) was studied in [LPPP05, HPP+15, GP16]. In particular, [HPP+15] devised a message-optimal and time-efficient MST algorithm for this model.

Mahreghi and King [MK17] devised randomized, not comparison-based MST algorithm with running time $\tilde{O}(Diam(MST))$, and with $\tilde{O}(n)$ messages. (This algorithm assumes that at the beginning of the computation, every vertex knows the identities of all its neighbors, i.e., the so-called KT_1 model.)

Efficient MST algorithms for planar graphs, and more generally, graphs of bounded genus, were given in [GH16, HIZ16].

2   Preliminaries

We consider the synchronous $CONGEST$ model of distributed communication. Every vertex v of an input graph $G = (V, E)$ hosts a processor, and these processors communicate with one another via $O(\log n)$-size messages in synchronous rounds. All edge weights are assumed to be at most polynomial in n, or alternatively, the message size can be restricted to $O(1)$ edge weights or/and identity numbers. In a more general $CONGEST(b \log n)$ model, for a parameter $b \geq 1$, on every round every vertex is allowed to send messages of size $O(b \log n)$ bits, or alternatively, $O(b)$ edge weights and/or vertex identities via every edge incident on it.

At the beginning of the communication every vertex v knows its own unique identity number, denoted $Id(v)$. The *running time* of an algorithm in this model is the worst-case number of rounds that it runs. The *message complexity* of an algorithm is the worst-case overall number of messages sent throughout an execution of the algorithm. At the end of an execution, every vertex v is required to know which among the edges incident on it belong to the MST.

We assume that the MST is unique. This assumption is without loss of generality, see, e.g., [Pel00], Ch. 5. A connected subtree of the unique MST is called an *MST fragment*, or simply a *fragment*.

We say that a collection $\{F_1, F_2, \ldots, F_h\}$, for some positive integer h, is an *MST forest*, if for each $i \in [h]$, F_i is an MST fragment, these fragments are vertex-disjoint, and $\bigcup_{i=1}^{h} V(F_i) = V$. For a pair of positive parameters α and β, we say that an MST forest $\mathcal{F}$ is an (α, β)-*MST-forest*, if it contains at most α fragments, each with strong diameter at most β. (*Strong diameter* of a subgraph F is the maximum distance in F between a pair of vertices $u, v \in V(F)$.) A *diameter* of an MST forest $\mathcal{F}$ is the maximum diameter of one of its fragments.

We say that an MST forest $\mathcal{F}'$ *coarsens* MST forest $\mathcal{F}$, if for every fragment $F \in \mathcal{F}$, there exists a fragment $F' \in \mathcal{F}'$ that contains it, i.e., $V(F) \subseteq V(F')$ (and, as a result, also $E(F) \subseteq E(F')$, because F and F' are subtrees of the same spanning tree).

Boruvka's algorithm starts from a collection of MST fragments. On each phase it computes the MWOE of every fragment, and computes the fragments' graph, whose vertices are the fragments, and edges are the MWOEs. It then merges each connected component of the fragments' graph into a greater fragment, and obtains an MST forest with fewer fragments. In fact, the number of fragments decreases at least by a factor of 2, and so the number of phases is $O(\log n)$. See [Pel00],

Ch. 5, for further details.

For a rooted tree T and a non-root vertex v in T, we denote by $\pi_T(v)$ the *parent* of v in T. For a vertex v, we denote by $Id(v)$ the identity of the vertex v. For each fragment F, there is a designated root vertex rt_F, and the identity $Id(F)$ of F is set to be the identity $Id(rt)$ of the root rt.

3   The Algorithm and its Analysis

Based on [GKP98, KP98] (see also [PRS16], Algorithm 1, called Controlled-GHS, and Lemma 1, and Lenzen's lecture notes [Len16], the chapter about MST, Lemmas 6.15-6.17), we show in Section 4 that for any positive parameter k, an $(n/k, O(k))$-MST forest $\mathcal{F} = \mathcal{F}_0$ can be computed in $O(k \cdot \log^* n)$ time, and using $O(|E| \log k + n \log k \cdot \log^* n)$ messages. [4] We refer to $\mathcal{F}_0$ as the *base* MST forest, and call its fragments *base fragments*.

The algorithm starts with constructing an auxiliary BFS tree τ for the entire graph G rooted at a root vertex rt. This step requires $O(D)$ time and $O(|E|)$ messages.

Every base fragment F has its designated root vertex r_F. We need every vertex v of τ to be able to route messages from the root rt of τ to each of the roots r_F of base fragments $F \in \mathcal{F}$, which belong to the subtree τ_v of τ rooted at v. For this end, we compute intervals I_v for each vertex $v \in V(\tau)$, such that for every pair u, v of vertices in V, their intervals are either disjoint (if they belong to different branches of τ), or nested if the vertex with a larger interval is an ancestor in τ of the vertex with a smaller interval. Given these intervals, when a vertex v needs to route a message to a root r_F of a base fragment F which belongs to $V(\tau_v)$, it finds a child u of v whose interval $I(u)$ contains $I(r_F)$, and sends the message to this child.

To compute the intervals, we first conduct a convergecast in τ. As a result of this convergecast, every vertex v knows the size $|V(\tau_v)|$ of its subtree. Then the root rt of τ assigns itself the interval $I(rt) = [1, n]$, $n = |V(\tau)| = |V|$, and assigns its children $u_1, \ldots, u_d$, for $d = deg(rt)$, disjoint intervals $I(u_1), \ldots, I(u_d) \subseteq I(rt)$, with $|I(u_i)| = |V(\tau_{u_i})|$, for every $i \in [d]$. (This is possible because $\sum_{i=1}^{d} |V(\tau_{u_i})| = n - 1 = |I(rt)| - 1$. Observe also that rt can learn $|V(\tau_{u_i})|$, for all $i \in [d]$, within one round.) Next, each of the children u_i assigns (in parallel) disjoint intervals to their children, etc. Finally, at the end of this process, we conduct a pipelined convergecast during which the root rt learns the $|\mathcal{F}|$ intervals of all the base fragments.

The entire process of computing the intervals requires $O(D)$ time and $O(n)$ messages, while the final pipelined convergecast requires $O(D + |\mathcal{F}|) = O(D + n/k)$ time, and $O(D \cdot n/k)$ messages.

Consider first the case $D \leq \sqrt{n}$. We set $k = \sqrt{n}$. Suppose we have already conducted j phases of the Boruvka's algorithm, starting from $\mathcal{F}_0$, and obtained a coarsening forest $\mathcal{F}_j$, for some $j = 0, 1, 2, \ldots$, of $\mathcal{F}$. We now show how to implement the next phase of Boruvka's algorithm, and to construct a coarsening MST forest $\mathcal{F}_{j+1}$ of $\mathcal{F}_j$ (and, consequently, of $\mathcal{F}$ too).

We assume that every vertex v knows the identities of both the base fragment F_v and the fragment $\hat{F}_v$ of $\mathcal{F}_j$ that it belongs to. Also, for every neighbor u of v, we assume that v knows the identities of F_u and $\hat{F}_u$. We also assume that the root rt knows the identities of all base fragments, and at the beginning of phase j, $j = 0, 1, \ldots$, it knows the identities of all fragments of $\mathcal{F}_j$, and for

---

[4]In fact, Lemma 6.17 of [Len16] applies this only for $k \leq \sqrt{n}$, but inspecting its proof reveals that it holds for larger values of k as well. The message complexity of this procedure is not analyzed in [KP98, Len16], while its analysis in [PRS16] provides a slightly weaker bound. For the sake of completeness, we provide a self-contained proof of this result in Section 4.

each base fragment $F \in \mathcal{F}$, the root knows the identity of the fragment $\hat{F} \in \mathcal{F}_j$ that coarsens it. This is argued by induction on j.

To guarantee that the induction base $j = 0$ holds, after the base MST forest is constructed, every vertex v updates its neighbors with the identity of F_v. This requires $O(1)$ time and $O(|E|)$ messages. Also, an upcast of $|\mathcal{F}_0| \leq n/k$ identities of base fragments is conducted over the BFS tree τ at this stage. This step requires $O(D + n/k)$ time, and $O(D \cdot n/k)$ messages.

In every base fragment $F \in \mathcal{F}_0$ we compute (in parallel in all base fragments) the edge $e = (u, v)$ of minimum weight that crosses between $u \in V(F)$ and $v \in V \setminus V(\hat{F})$, where $\hat{F} \in \mathcal{F}_j$ is the fragment that coarsens the base fragment F. This computation requires $O(k) = O(\sqrt{n})$ time, and $O(n)$ messages.

Once this is done, we upcast all these $O(n/k) = O(\sqrt{n})$ pieces of information over the auxiliary BFS tree τ to the root vertex rt of τ. This is done via a pipelined convergecast procedure, in which every intermediate vertex u of τ forwards to his parent $\pi_\tau(u)$ in τ only the lightest edge for each fragment $\hat{F} \in \mathcal{F}_j$, among edges that were initially stored at one of the vertices z of the subtree τ_u of τ, rooted at u. This step requires $O(D + |\mathcal{F}_j|)$ time, and $O(D \cdot |\mathcal{F}_j|)$ messages. (See [Pel00], ch. 3.)

The root rt locally computes the MWOE $e_{\hat{F}}$ for every fragment $\hat{F} \in \mathcal{F}_j$. It then locally computes the fragments' graph whose vertices are fragments of $\mathcal{F}_j$, and edges are the MWOEs, and computes the MST forest $\mathcal{F}_{j+1}$. Specifically, for every base fragment $F \in \mathcal{F}$, the root knew the identity of a fragment $\hat{F} \in \mathcal{F}_j$ that coarsens it. As a result of the computation that rt conducts, it now knows the identity of a fragment $\hat{F}' \in \mathcal{F}_{j+1}$ that coarsens $\hat{F}$. (Consequently, $\hat{F}'$ also coarsens F.) The root rt then sends $|\mathcal{F}|$ messages over τ, each message is of the form $(F, \hat{F}')$, where $F \in \mathcal{F}$, $\hat{F}' \in \mathcal{F}_{j+1}$, $\hat{F}'$ coarsens F. Each such a message $(F, \hat{F}')$ has the destination interval $I(rt_F)$ attached to it, and it is routed along the unique $rt - rt_F$ path in τ. The root rt_F of the base fragment F receives this message, and writes down to itself that it belongs to $\hat{F}'$. This (pipelined) downcast requires $O(D + |\mathcal{F}|)$ time, and $O(D \cdot |\mathcal{F}|) = O(D \cdot n/k)$ messages. (This is because every one of the $|\mathcal{F}|$ messages is routed to its destination along a path with at most D edges.)

Next, every root vertex r_F of a base fragment $F \in \mathcal{F}$ broadcasts the identity $Id(\hat{F}')$ of their new $(j + 1)$st level fragment $\hat{F}' \in \mathcal{F}_{j+1}$ to all vertices of F. This requires $O(k)$ time and $O(n)$ messages. Finally, every vertex v updates its neighbors in G with its new $(j + 1)$st level's fragment identity. This requires $O(1)$ time, and $O(|E|)$ messages. This completes the description of a single phase of Boruvka's algorithm.

To analyze the running time and message complexity, observe that for every $j = 0, 1, 2, \ldots$, we have $|\mathcal{F}_{j+1}| \leq \frac{1}{2} \cdot |\mathcal{F}_j|$, and so the number of phases ℓ is $O(\log n)$ phases. Hence the overall time is

$$O(D + n/k) + O(k \cdot \log^* n) + O((D + k + |\mathcal{F}|) \cdot \log n) = O((D + k + n/k) \cdot \log n) = O(\sqrt{n} \cdot \log n). \quad (1)$$

Similarly, the message complexity is $O(|E| \log n + n \log n \cdot \log^* n)$ for constructing $\mathcal{F}$, $O(D \cdot n/k + n)$ for computing the intervals, and $O(D \cdot n/k + |E| + n)$ on each consequent phase. As $D \leq k$, the overall message complexity is $O(|E| \log n + n \log n \cdot \log^* n)$.

For $D > \sqrt{n}$, we compute the $(n/k, O(k))$-MST forest $\mathcal{F} = \mathcal{F}_0$ with parameter $k = D$ in $O(D \cdot \log^* n)$ time, and $O(|E| \log n + n \log n \cdot \log^* n)$ messages. From this point on, the algorithm is identical to the one that we have just described. For every $j = 0, 1, 2, \ldots$, the jth phase of it requires $O(D + k + |\mathcal{F}|) = O(D + k + n/k) = O(D)$ time, i.e., all phases altogether require $O(D \log n)$ time.

The number of messages is $O(|E| + n + D \cdot |\mathcal{F}|)$ on every phase, i.e., $O((|E| + n) \cdot \log n)$ messages in all the ℓ phases. Hence the total message complexity is $O(|E| \log n + n \log n \cdot \log^* n)$.

We summarize this result below.

Theorem 3.1 *The deterministic algorithm that was described above computes the minimum spanning tree in the CONGEST model, in $O((D + \sqrt{n}) \cdot \log n)$ time, using $O(|E| \log n + n \log n \cdot \log^* n)$ messages.*

Next, we extend the algorithm to the $CONGEST(b \log n)$ model, for a positive integer parameter b. We first discuss the case of small diameter, i.e., $D \le \sqrt{\frac{n}{b}}$, and then proceed to discussing the complementary case.

In the small-diameter regime, we set $k = \sqrt{\frac{n}{b}}$, i.e., $D \le k$. We construct an $(n/k, O(k))$-MST forest $\mathcal{F}_0$ in $O(k \log^* n)$ time, using $O(|E| \log n + n \log n \cdot \log^* n)$ messages. The upcast of $|\mathcal{F}_0| \le n/k$ identities of base fragments requires $O(D + \frac{n}{k \cdot b})$ time and $O(D \cdot n/k)$ messages. Now consider the jth phase of the algorithm, for some $j = 0, 1, 2, \ldots$. Computing minimum weight crossing edges in parallel in all base fragments $\{e = (u, v) \mid u \in V(F), v \in V \setminus V(\hat{F}), F \in \mathcal{F}, \hat{F} \in \mathcal{F}_j\}$ requires $O(k)$ time and $O(n)$ messages. Pipelined convergecast of $|\mathcal{F}_j|$ items requires $O(D + \frac{|\mathcal{F}_j|}{b})$ time and $O(D \cdot |\mathcal{F}_j|)$ messages. The pipelined downcast of $|\mathcal{F}| \le n/k$ messages requires $O(D + |\mathcal{F}|/b) = O(D + \frac{n}{kb}) = O(D + \sqrt{n/b})$ time, and $O(D \cdot |\mathcal{F}|) = O(D \cdot n/k)$ messages. (Note that this downcast sends each message only along its own root-destination path, rather than broadcasting it to the entire graph.) Updating neighbors with new fragments' identities requires $O(1)$ time and $O(|E|)$ messages. The overall running time of the ℓ phases is

$$O\left(D + \frac{n}{k \cdot b}\right) + O(k \log n + D \log n + |\mathcal{F}| \cdot \log n) = O\left(\left(D + k + \frac{n}{k \cdot b}\right) \cdot \log n\right)$$
$$= O\left(\sqrt{\frac{n}{b}} \cdot \log n\right) .$$

This is also the upper bound on the total running time. The overall number of messages used in the ℓ phases is $O((D \cdot n/k + n + |E|) \log n) = O(|E| \cdot \log n)$. Hence the total message complexity is $O(|E| \log n + n \log n \cdot \log^* n)$.

In the large-diameter regime, i.e., when $D > \sqrt{\frac{n}{b}}$, we set $k = D$. Constructing $\mathcal{F} = \mathcal{F}_0$ requires $O(D \log^* n)$ time and $O(|E| \log n + n \log n \cdot \log^* n)$ messages. Computing minimum weight crossing edges in all base fragments in parallel requires (on each phase) $O(D)$ time and $O(n)$ messages. Other than that on phase j, for $j = 0, 1, \ldots, \ell - 1$, we have time $O(D + \frac{|\mathcal{F}|}{b})$ time and $O(D \cdot |\mathcal{F}|)$ messages. Overall, this sums up to

$$O((D + |\mathcal{F}|/b) \cdot \log n) = O\left(\left(D + \frac{n}{Db}\right) \cdot \log n\right) = O(D \log n)$$

time, and $O((D \cdot n/k + n + |E|) \cdot \log n) = O(|E| \cdot \log n)$ messages. Hence the total running time of the entire algorithm in this case is $O(D \log n)$, and its message complexity is $O(|E| \log n + n \log n \cdot \log^* n)$.

Theorem 3.2 *For any $b \ge 1$, the deterministic algorithm that was described above computes the minimum spanning tree in $CONGEST(b \log n)$ model, in $O((D + \sqrt{n/b}) \cdot \log n$ time, using $O(|E| \log n + n \log n \cdot \log^* n)$ messages.*

4 Constructing an MST Forest

For the sake of completeness, we next describe the algorithm (due to [GKP98, KP98, Len16]) for constructing an $(n/k, O(k))$-MST forest, for an integer parameter $k \le n/10$. (The constant 10 is

7

quite arbitrary.) Our version of the algorithm is slightly more general than that in [Len16], and our bounds on its time and message complexities are slightly better than the respective bounds in [PRS16].

The algorithm runs for $t = \lceil \log k \rceil$ phases. At the beginning of a phase $i = 0, 1, 2, \ldots, t-1$, the algorithm has already computed $(n/2^{i-1}, 6 \cdot 2^i)$-MST forest $\mathcal{F}_i$. The induction base $i = 0$ holds for the MST forest of singletons. We next describe a single phase of the algorithm, and show that the resulting collection $\mathcal{F}_{i+1}$ is an $(n/2^i, 6 \cdot 2^{i+1})$-MST forest.

At the beginning of a phase i, every fragment $F \in \mathcal{F}_i$ with diameter at most 2^i computes the edge $e_F = MWOE(F)$. We denote the set of fragments $\mathcal{F}'_i$. This step requires $O(2^i)$ time and $O(n)$ messages. (Also, at the beginning of each phase, every vertex updates its neighbors with the identity of its fragment. This requires $O(1)$ time and $O(|E|)$ messages.) Then, for every $e_F = (u, v)$, $u \in V(F)$, $v \in V \setminus V(F)$, a message is sent over e_F, and the receiver v writes down u as a "foreign-fragment" child of itself. (In the special case when $(u, v) = MWOE(F_u) = MWOE(F_v)$, with $u \in V(F_u)$, $v \in V(F_v)$, the endpoint belonging to a higher-identity fragment becomes the parent of the other endpoint.)

This defines a *candidate fragment graph* $\mathcal{G}'_i = (\mathcal{F}'_i, \mathcal{E}_i)$, whose vertices are the fragments of $\mathcal{F}'_i$, and edges are the $MWOE$ edges of these fragments. We then compute a maximal matching (henceforth, MM) M in $\mathcal{G}'_i$. (We will soon elaborate on this.) For every pair $(F, F') \in M$, the two fragments merge into a single fragment, along the $MWOE$ edge that connects them. Every fragment $F'' \in \mathcal{F}_i \setminus \mathcal{F}'_i(M)$ is necessarily connected via its $MWOE(F'') = e''$ to either a matched fragment $F \in \mathcal{F}'_i(M)$, or to a fragment $F \in \mathcal{F}_i \setminus \mathcal{F}'_i$ of diameter larger than 2^i. In either case, it now merges with F along the edge e''. Except for the procedure that computes an MM, this completes the description of the algorithm. The MST forest $\mathcal{F}_{i+1}$ consists now of the resulting merged fragments, and of those fragments of $\mathcal{F}_i \setminus \mathcal{F}'_i$ that did not participate in the merging process described above. (These are the fragments F with diameter $Diam(F) > 2^i$, and such that no unmatched fragment $F' \in \mathcal{F}'_i$ has its $MWOE$ (u, v) with an endpoint in F.)

Remark: The next two lemmas and their proofs are closely related to that of Lemmas 6.15 and 6.17 in [Len16].

Lemma 4.1 $Diam(\mathcal{F}_{i+1}) \leq 6 \cdot 2^{i+1}$.

Proof: Each new fragment $\hat{F} \in \mathcal{F}_{i+1}$ can be viewed as a subtree of diameter at most 3 in the fragment graph $\mathcal{G}_i = (\mathcal{F}_i, \mathcal{E}_i)$, whose edge set $\mathcal{E}_i$ is the set of all the $MWOE$s of fragments of $\mathcal{F}_i$. Moreover, at most one of the fragments in this subtree may have diameter greater than 2^i. (But, by induction hypothesis, its diameter is, nevertheless, at most $6 \cdot 2^i$.) Hence the diameter of $\hat{F}$ in G is at most $6 \cdot 2^i + 3 \cdot 2^i + 3 \leq 12 \cdot 2^i = 6 \cdot 2^{i+1}$. ∎

Lemma 4.2 *For $i = 0, 1, \ldots, t-2$, each fragment $\hat{F} \in \mathcal{F}_{i+1}$ contains at least 2^i vertices.*

Proof: The proof is by induction on i. The base $i = 0$ holds, as every fragment of $\hat{F}_1$ contains at least one vertex.

By induction hypothesis, every fragment $F \in \mathcal{F}_i$ contains at least 2^{i-1} vertices. Consider a fragment $F \in \mathcal{F}_i$ with $|F| < 2^i$. Then $Diam(F) < 2^i$ too, and hence $F \in \mathcal{F}'_i$. Thus F merges with at least one other fragment F' of $\mathcal{F}_i$. (As $|F| < 2^i \leq n$, F has outgoing edges, and thus has an $MWOE$.) Since, by induction hypothesis, $|F|, |F'| \geq 2^{i-1}$, it follows that the merged fragment has size at least 2^i. ∎

8

Hence $|\mathcal{F}_i| \leq n/2^{i-1}$. By substituting $i = t - 1 = \lceil \log k \rceil - 1$, we get $|\mathcal{F}_{t-1}| = O(n/k)$, and $Diam(F_{t-1}) = O(k)$. By rescaling (setting $k' = c \cdot k$, for an appropriate constant c), we obtain the desired $(n/k, O(k))$-MST forest.

Next, we sketch the procedure that computes an MM in the forest $\mathcal{G}_i' = (\mathcal{F}_i', \mathcal{E}_i')$, and analyze its time and message complexities. Recall that, by Lemma 4.1, each fragment $F \in \mathcal{F}_i'$ has diameter $O(2^i)$.

The first step is to simulate Cole-Vishkin's 3-vertex-coloring algorithm [CV86] in $\mathcal{G}_i'$. For this end, on every step every internal fragment $F \in \mathcal{F}_i'$ needs to send a message with its current color to its children in $\mathcal{G}_i'$. (Initial colors are set as fragments' identities.) This is implemented in $O(2^i)$ time, using $O(n)$ messages, in a straightforward manner. Since there are $\log^* n$ such steps, overall this computation requires $O(2^i \cdot \log^* n)$ time and $O(n \cdot \log^* n)$ messages.

Given a 3-vertex-coloring of $\mathcal{G}_i'$, there are 3 steps. On each step $j \in \{1, 2, 3\}$, fragments F' of color j that have at least one of their children F'' unmatched, insert an edge (F', F'') connecting them to such an unmatched child into the matching, and update their parents F that they became matched. The second step involves a convergecast in the parent fragment F, during which the root of F learns if it still has an unmatched child. Also, every internal vertex $v \in F$ learns if one of its descendents leads to an unmatched child of F or not.

This entire part of the algorithm can also be implemented in $O(2^i)$ time and $O(n)$ messages, in a straightforward manner. Hence the entire computation of MM on phase i of the algorithm requires $O(2^i \cdot \log^* n)$ time, and $O(n \cdot \log^* n)$ messages. Thus, the total running time of phase i is $O(2^i \cdot \log^* n)$, and the number of messages if $O(|E| + n \cdot \log^* n)$. Summing up over all the $\lceil \log k \rceil$ phases, we obtain running time $O(\log^* n \sum_{i=0}^{\lceil \log k \rceil} 2^i) = O(k \cdot \log^* n)$, and message complexity $O(|E| \cdot \log k + n \log k \cdot \log^* n)$.

We summarize this section in the following theorem.

Theorem 4.3 *For an integer parameter $k \leq n/10$, the deterministic algorithm described above computes an $(n/k, O(k))$-MST forest in time $O(k \cdot \log^* n)$, using $O(|E| \cdot \log k + n \cdot \log k \cdot \log^* n)$ messages.*

References

[ABCP93] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear cost sequential and distribured constructions of sparse neighborhood covers. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 638–647, 1993.

[AGPV90] Baruch Awerbuch, Oded Goldreich, David Peleg, and Ronen Vainish. A trade-off between information and communication in broadcast protocols. *J. ACM*, 37(2):238–256, 1990.

[Awe87] Baruch Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary). In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 230–240, 1987.

[Coh93] Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 648–658, 1993.

[CT85] Francis Y. L. Chin and H. F. Ting. An almost linear time and o(n log n + e) messages distributed algorithm for minimum-weight spanning trees. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 257–266, 1985.

[CV86] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[EKNP14] Michael Elkin, Hartmut Klauck, Danupon Nanongkai, and Gopal Pandurangan. Can quantum communication speed up distributed computation? In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 166–175, 2014.

[Elk04a] M. Elkin. An unconditional lower bound on the time-approximation tradeoff of the minimum spanning tree problem. In *Proc. of the 36th ACM Symp. on Theory of Comput. (STOC 2004)*, pages 331–340, 2004.

[Elk04b] Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 359–368, 2004.

[FM04] Michalis Faloutsos and Mart Molle. A linear-time optimal-message distributed algorithm for minimum spanning trees. *Distributed Computing*, 17(2):151–170, 2004.

[Gaf85] Eli Gafni. Improvements in the time complexity of two message-optimal election algorithms. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing, Minaki, Ontario, Canada, August 5-7, 1985*, pages 175–185, 1985.

[GH16]   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 2016, Arlington, VA, USA, January 10-12, 2016*, pages 202–219, 2016.

[GHS83]   Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, 1983.

[GKP98]   Juan A. Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM J. Comput.*, 27(1):302–316, 1998.

[GP16]   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, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 19–28, 2016.

[HIZ16]   Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Low-congestion shortcuts without embedding. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 451–460, 2016.

[HPP+15]   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 *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, San Sebastian, Spain*, pages 91–100, 2015.

[KKP11]   Liah Kor, Amos Korman, and David Peleg. Tight bounds for distributed MST verification. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 69–80, 2011.

[KKT15]   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, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 71–80, 2015.

[KP98]   Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *J. Algorithms*, 28(1):40–66, 1998.

[KP08]   Maleq Khan and Gopal Pandurangan. A fast distributed approximation algorithm for minimum spanning trees. *Distributed Computing*, 20(6):391–402, 2008.

[Len16]   C. Lenzen. Lecture notes on theory of distributed systems. https://www.mpi-inf.mpg.de/fileadmin/inf/d1/teaching/winter15/tods/ToDS.pdf, 2016.

[LPP06]   Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed MST for constant diameter graphs. *Distributed Computing*, 18(6):453–460, 2006.

[LPPP05]   Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in $O(\log \log n)$ communication rounds. *SIAM J. Comput.*, 35(1):120–131, 2005.

[MK17]   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, Hyderabad, India, January 5-7, 2017*, page 8, 2017.

[Pel00]   D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[PR99]   D. Peleg and V. Rubinovich. A near-tight lower bound on the time complexity of distributed mst construction. In *Proc. 40th IEEE Symp. on Foundations of Computer Science*, pages 253–261, 1999.

[PRS16]   Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time- and message-optimal distributed algorithm for minimum spanning trees. *CoRR, accepted to STOC'17*, abs/1607.06883, 2016.

[SB95]   Gurdip Singh and Arthur J. Bernstein. A highly asynchronous minimum spanning tree protocol. *Distributed Computing*, 8(3):151–161, 1995.

[SHK+12]   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.