Distributed MST for Constant Diameter Graphs

EXTENDED ABSTRACT

Zvi Lotker
zvilo@eng.tau.ac.il
Dept. of Electrical Engineering
Tel Aviv University
Tel Aviv 69978
Israel

Boaz Patt-Shamir
boaz@eng.tau.ac.il
Dept. of Electrical Engineering
Tel Aviv University
Tel Aviv 69978
Israel

David Peleg
peleg@wisdom.weizmann.ac.il
Dept. of Computer Science
Weizmann Institute
Rehovot 76100
Israel

Abstract

This paper considers the problem of distributively constructing a minimum-weight spanning tree (MST) for graphs of constant diameter in the bounded-messages model, where each message can contain at most B bits for some parameter B. It is shown that the time required to compute an MST for graphs of diameter 4 or 3 can be as high as $\Omega(\sqrt[3]{n}/B)$ and $\Omega(\sqrt[4]{n}/\sqrt[2]{B})$, respectively. The lower bound holds even if the algorithm is allowed to be randomized. On the other hand, it is shown that $O(\log n)$ time units suffice to compute an MST deterministically for graphs with diameter 2, when $B = O(\log n)$. These results complement a previously known lower bound of $\Omega(\sqrt[2]{n}/B)$ for graphs of diameter $\Omega(\log n)$.

1 Introduction

The communication complexity of distributed network algorithms is often determined by *size* parameters such as the number of vertices or edges in the network. In contrast, the *time* efficiency of distributed solutions for various problems on networks is sometimes largely affected by other types of network parameters. For some natural distributed network problems, the time complexity is controlled by *locality* or *sparsity* measures such as the network's diameter D (namely, the maximum distance between any two vertices, measured in hops). This holds, for example, for leader election and related problems [7]. Yet other problems are independent of (or sublinearly dependent on) global parameters. Maximal independent set computations and related problems fall into that category [6].

This paper concerns the time complexity of the MST problem. It is clear that $\Omega(D)$ time is necessary for distributed MST construction in the worst case, and more formally, for every $n \geq 2$ and $1 \leq D \leq \lfloor n/2 \rfloor$ there exist weighted n-vertex networks of diameter D (e.g., formed as a $2D$-vertex ring with $n - 2D$ vertices attached to it as leaves) on which any distributed MST algorithm requires $\Omega(D)$ time.

However, it is natural to ask whether $O(D)$-time algorithms exist for MST construction, or alternatively, the problem depends also on other parameters of the network. In particular, the need for information flow among vertices residing on common cycles, in order to resolve the fate of the links of that cycle, suggests that the capacity of various cuts in the network may come into play.

It is obvious that in the extreme model allowing the transmission of an unbounded-size message on a link in a single time unit (cf. [6]), the problem can be solved in $O(D)$ time; simply collect the graph's topology and the edge weights into a root vertex, compute an MST locally and broadcast it to all the vertices. The problem thus becomes interesting in the more realistic, and rather common, *B-bounded-message* model (henceforth, the B model), in which message size is bounded by some value B, and a vertex may send at most one message on each edge at each time unit. It is customary to assume that $B \geq \log n$, so as to allow messages to contain node IDs; another standard assumption is that an edge weight can be contained in a single message. Our lower bounds do

PODC 01 Newport Rhode Island USA

not rely on any of these assumptions; our upper bounds (algorithms) do.

In this B model (with $B = O(\log n)$), the classical distributed MST construction algorithms of [2] and [1] were communication optimal and required $O(n \log n)$ and $O(n)$ time, respectively. Subsequently, the distributed MST construction algorithms of [3] and [4] had time complexity $O(D+n^{0.613} \log^* n)$ and $O(D+\sqrt{n} \log^* n)$, respectively.

Recently, a near-tight lower bound has been proved on the time complexity of distributed MST construction [9]. Specifically, the time complexity of any deterministic algorithm for MST was shown to be $\Omega(\sqrt{n}/(B \log n))$. However, this result was shown only for graphs of sufficiently large diameter, specifically, $D = \Omega(\log n)$. The time complexity of distributed MST computation for graphs of smaller diameter (i.e., $D = o(\log n)$) was left as an open problem.

In this paper we present negative and positive results for constant-diameter graphs. On the negtaive side, we extend the lower bound of [9] in two ways. First, we present constructions that show that the time complexity of distributed MST is $\Omega(\sqrt[3]{n}/B)$ even for graphs of diameter 4, and $\Omega(\sqrt[4]{n}/\sqrt[2]{B})$ for graphs of diameter 3. Secondly, we show that all lower bounds hold even for randomized algorithms. On the positive side, we show that if the diameter of the graph is 2, then MST can be constructed deterministically in $O(\log n)$ time units. This result demonstrates a sharp threshold in the time complexity, when the diameter changes from 2 to 3.

The remainder of this paper is organized as follows. In Section 2 we define the basic terms we use. In Section 3 we present our lower bounds. In Section 4 we present the algorithm for diameter 2. We conclude in Section 5.

2 Preliminaries

In this section we define the model and the MST problem. we also define the mailing problem [9] which is key to our lower bounds.

2.1 The model

A point-to-point communication network is modeled as an undirected graph $G = (V, E)$, where the vertices of V represent the network processors and the edges of E represent the communication links connecting them. Vertices are allowed to have unique identifiers. The vertices do not know the topology or the edge weights of the entire network, but they may know the ID's of their neighbors and the weights of the corresponding edges.

A weight function $\omega : E \to \mathbb{R}^+$ associated with the graph assigns a nonnegative integer weight $\omega(e)$ to each edge $e = (u, v) \in E$. The weight $\omega(e)$ is known to the adjacent vertices, u and v. The vertices can communicate only by sending and receiving messages over the communication links. Communication is carried out in a synchronous manner, i.e., all the vertices are driven by a global clock. Messages are sent at the beginning of each round, and are received at the end of the round. (Clearly, our lower bounds hold for asynchronous networks as well.) At most one B-bit message can be sent on each link in one direction on every round. Our algorithms assume that B is large enough to allow the transmission of an edge weight and a node ID in a single message (this assumption is not used for the lower bounds). To avoid subtleties that do not affect the asymptotic time complexity, we require that each node transmits at each round exactly B bits. (Otherwise, silence can be interpreted as useful information, reducing the time complexity by a costant factor.)

The length of a path p in the network is the number of edges it contains. The *distance* between two vertices u and v is defined as the length of the shortest path connecting them in G. The *diameter* of G, denoted D, is the maximum distance between any two vertices of G.

2.2 The mailing problem

The mailing problem is defined in the following situation. We are given a graph G with two distinguished vertices s and r, referred to as the *sender* and the *receiver*, respectively. Both the sender s and the receiver r store b boolean variables each, $X_1^s, \ldots, X_b^s$ and $X_1^r, \ldots, X_b^r$ respectively, for some integer $b \geq 1$. An instance of the problem consists of an initial assignment $X = \{x_i : 1 \leq i \leq b\}$, where $x_i \in \{0, 1\}$, to the variables of s, such that $X_i^s = x_i$. Given such an instance, the mailing problem requires s to deliver the string X to the receiver r, i.e., upon termination, the variables of r should contain the output $X_i^r = x_i$ for every $1 \leq i \leq b$. Henceforth, we refer to this problem instance as $\texttt{Mail}(G, s, r, X)$, and to

the class of problem instances containing all strings X of length b as $\texttt{Mail}(G, s, r, b)$. Let $T_A(G, s, r, X)$ denote the time required by algorithm A for solving the problem instance $\texttt{Mail}(G, s, r, X)$, and let $T_A(G, s, r, b)$ denote the worst-case time complexity of algorithm A over the class of problem instances $\texttt{Mail}(G, s, r, b)$.

Remark 1: One important technical restriction we impose on algorithms solving the mailing problem is the following: *In this paper we require that bits are communicated explicitly*, i.e., bits are viewed as opaque objects that can only be stored and moved by shipping them over communication links. In other words, we rule out all sophisticated compression schemes. This restriction may seem severe at first glance, but we argue that no generality is lost if we impose it. To see this, note that by information-theoretic considerations, we have that under any (possibly randomized) correct coding scheme, at least a half of the bit-strings of length b have representations of at least $b-1$ bits long [5]. Since we consider asymptotic worst-cases in this paper, we can disallow all non-trivial codings without loss of generality.

2.3 The distributed MST problem

Formally, the *minimum spanning tree (MST)* problem can be stated as follows. Given a graph $G(V, E)$ and a weight function ω on the edges, it is required to find a spanning tree $MST(G) \subseteq E$ whose total weight, $\omega(MST(G)) = \sum_{e \in MST(G)} \omega(e)$, is minimal. In the distributed model, the input and output of the MST problem are represented as follows. Each vertex knows the IDs of its closest neighbors and the weights of the corresponding edges. A degree-d vertex $v \in V$ with neighbors $u_1, \ldots, u_d$ has d *weight variables* $W_1^v, \ldots, W_d^v$, with W_i^v containing the weight of the edge connecting v to u_i, i.e. $W_i^v = \omega(v, u_i)$. The output of the MST problem at each vertex v is an assignment to the (boolean) output variables $Y_1^v, \ldots, Y_d^v$, assigning 1 to Y_i^v if $(u_i, v) \in MST(G)$ and 0 otherwise.

3 Lower bounds on distributed MST construction

In this section we prove lower bounds on distributed MST construction. The idea is to prove a lower bound on the time complexity of the mailing problem in certain topologies, and then argue that for each topology there is a collection of edge weight assignments such that under these assignments, the mailing problem can be reduced to MST. This reduction shows that any lower bound on the mailing problem applies to the MST problem. In the remainder of this section, we concentrate on proving lower bounds on the mailing problem; the reduction from MST is similar to the one used in [9].

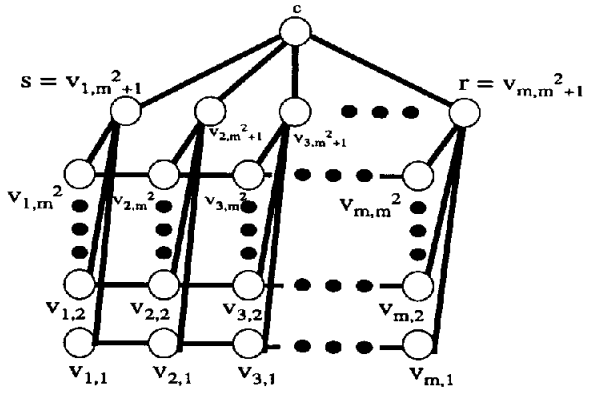

Figure 1: The graph F_m used in the lower bound for diameter 4.

3.1 Diameter 4 graphs

Fix a parameter $m \geq 2$. We start by defining a generic graph F_m. F_m is defined using two basic building blocks. The first is a path $\mathcal{P}$ on m nodes, defined by

$$\begin{aligned} V(\mathcal{P}) &= \{v_1, v_2, \ldots, v_m\} \\ E(\mathcal{P}) &= \{(v_i, v_{i+1}) : i = 1, 2, \ldots, m-1\}. \end{aligned}$$

The graph F_m contains m^2 copies of the path $\mathcal{P}_1, \ldots, \mathcal{P}_{m^2}$, where the kth copy $\mathcal{P}_k$ is defined by

$$\begin{aligned} V(\mathcal{P}_k) &= \{v_{1,k}, v_{2,k}, \ldots, v_{m,k}\} \\ E(\mathcal{P}_k) &= \{(v_{i,k}, v_{i+1,k}) : i = 1, 2, \ldots, m\}. \end{aligned}$$

The second building block is a star graph. There are two kinds of stars in F_m. The first star is called a *level star*: for each $i = 1, 2, \ldots, m$, we add a node v_{i,m^2+1} called *level i center*. For each path $\mathcal{P}_k$ we connect $v_{i,k}$ to v_{i,m^2+1}. Formally, we define

$$E(C_i) = \{(v_{i,m^2+1}, v_{i,k}) : k = 1, 2, \ldots, m^2\}.$$

The second kind of star is the *center star*: We add a single node c and connect it to all the level center nodes.

Formally, we define

$$E(C) = \{(c, v_{i,m^2+1}) : i = 1, 2, \ldots, m\} .$$

Overall, the graph F_m is defined by

$$\begin{aligned} V(F_m) &= \{v_{1,1}, v_{1,2}, \ldots, v_{m,m^2+1}, c\} \\ E(F_m) &= \left(\cup_{i=1}^{m^2} E(\mathcal{P}_i)\right) \cup \left(\cup_{i=1}^{m} E(C_i)\right) \cup E(C) . \end{aligned}$$

We denote the node v_{1,m^2+1} by s and the node v_{m,m^2+1} by r.

The following properties are obvious.

Lemma 3.1 *For all $m \geq 2$, the number of nodes in F_m is $m^3 + m + 1$ and its diameter is 4.*

Proof: The node count is immediate from the construction. The diameter follows from the fact that the distance between any node and c is at most 2. ∎

We now show that solving the mailing problem on F_m requires $\Omega(\sqrt[3]{n}/B)$ time units. We use the following observation.

Lemma 3.2 *For any time $t \leq m$, the number of bits delivered at r by time t is at most tmB.*

Proof: Observe that the distance between s and r in $F_m - \{c\}$ is $m+1$. It therefore follows that each bit delivered at r by time m must have traversed the node c. The bound follows from the fact that since the degree of c is m, at most mB bits can cross c in each time unit. ∎

Lemma 3.3 *For any correct mailing algorithm A, any $m \geq 2$ and any m^2-bit input string X, we have $T_A(F_m, s, r, X) \geq m/B$.*

Proof: Let $t \leq m$. Using Lemma 3.2 we get that the number of bits that move from s to r in t time units is at most tmB. Hence in the first $m/B - 1$ time units, fewer than m^2 bits from s can be delivered at r. The lemma follows. ∎

As $m = \Omega(\sqrt[3]{n})$, we can now prove the lower bound for the mailing problem.

Lemma 3.4 *For any correct mailing algorithm A and $m \geq 2$, $T_A(F_m, s, r, m^2) = \Omega(n^{1/3}/B)$.*

Remark 2: Dropping the restriction of "explicit communication" made in Remark 1 above, Lemma 3.3 no longer holds, as clever encodings may succeed in mailing certain strings X faster. Nevertheless, Lemma 3.4 still holds in the worst case, as explained above. Details are deferred to the full paper.

The lower bound for the MST construction problem can now be established by a reduction from MST construction to mailing. Essentially, the reduction shows that for the graph F_m, it is possible to construct a corresponding family of weighted graphs $\mathcal{F}_m$, such that if the mailing problem requires $\Omega(t)$ time on F_m in the B model, then any distributed MST construction algorithm requires $\Omega(t)$ time on some graphs of $\mathcal{F}_m$ in the B model. More specifically, the weight assignment is as follows. All graphs in $\mathcal{F}_m$ have the following edge weights.

$$\omega(e) = \begin{cases} 0, & \text{if } e \in E(\mathcal{P}_k) \\ 0, & \text{if } e \in E(C) \\ 1, & \text{if } e \in E(C_m) \\ 10, & \text{if } e \in E(C_i) \text{ for } 2 \leq i \leq m-1 \end{cases}$$

The only difference between different graphs is the weights of edges in $E(C_1)$, i.e., edges of the type $(v_{1,m^2+1}, v_{1,i})$, where $1 \leq i \leq m^2$. These edges take as weights either 0 or 2, with one graph in $\mathcal{F}_m$ for each of the 2^{m^2} combinations. It is straightforward to verify that any MST algorithm that works for all graphs in the family solves the mailing problem where $X_i^s = 1$ iff $\omega(v_{1,m^2+1}, v_{1,i}) = 2$. We conclude with the following result.

Theorem 3.5 *For every $m \geq 2$, there exists a family $\mathcal{F}_m$ of diameter 4 graphs such that every distributed algorithm MST construction algorithm requires $\Omega(n^{1/3}/B)$ time on some graph of $\mathcal{F}_m$ in the B model.*

3.2 Diameter 3 graphs

In this section we prove a weaker lower bound for graphs with diameter 3. Fix $m \geq 2$. We define a collection of graphs denoted $\mathcal{H}_m$. We remark that $\mathcal{H}_m$ resembles $\mathcal{F}_m$: essentially, the idea is to replace the center star with a clique, and optimize the parameters for the new topology.

Specifically, the graph H_m is constructed as follows (see Figure 2). First, we have m^3 copies of a path $\mathcal{P}$ of length m. The kth copy is denoted $\mathcal{P}_k$, and is formally defined as follows.

$$\begin{aligned} V(\mathcal{P}_k) &= \{v_{1,k}, v_{2,k}, \ldots, v_{m,k}\} \\ E(\mathcal{P}_k) &= \{(v_{i,k}, v_{i+1,k}) : i = 1, 2, \ldots, m-1\} . \end{aligned}$$

Next, we add a node z_i for $1 \leq i \leq m$. We call these

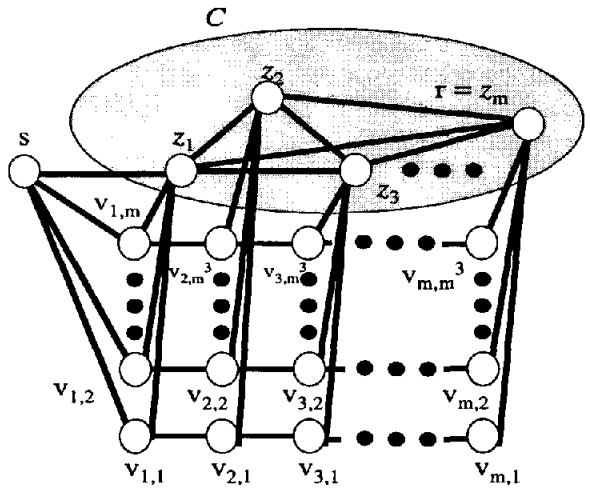

Figure 2: The graph H_m used in the lower bound for diameter 3.

nodes the *level center nodes*. For each path $\mathcal{P}_k$ we connect the node $v_{i,k}$ to z_i, i.e.,

$$E(C_i) = \{(z_i, v_{i,k}) : k = 1, 2, \ldots, m^3\} .$$

Let $C \stackrel{\text{def}}{=} \{z_1, z_2, \ldots, z_m\}$. We make C fully connected, i.e., we add the edges

$$E(C) = \{(z_i, z_j) : 1 \leq i < j \leq m\} .$$

Finally, we add a sender node s, and connect it to the beginning of each path, and to the first level center node z_1. Formally, we add the edges

$$E(S) = \{(s, v_{1,j}) : j = 1, 2, \ldots, m^3\} \cup (s, z_1) .$$

Now we can define the graph H_m formally:

$$\begin{aligned} V(H_m) &= \{v_{1,1}, v_{1,2}, \ldots, v_{m,m^3}, z_1, \ldots, z_m, s\} \\ E(H_m) &= \left(\cup_{i=1}^{m^3} E(\mathcal{P}_i)\right) \cup E(C) \cup \left(\cup_{i=1}^{m} E(C_i)\right) \\ &\quad \cup E(S) . \end{aligned}$$

We denote the node z_m by r.

Directly from the construction, we have the following properties.

Lemma 3.6 *For all $m \geq 1$, the number of nodes in H_m is $m^4 + m + 1$ and its diameter is 3.*

Proof: The number of nodes is immediate from the construction. The bound on the diameter follows from the fact that every node is at distance at most one from a node in C, and that the distance between any two nodes in C is at most 1. ∎

The lower bound we prove relies on counting the number of bits that cross edges in $E(C)$. The following lemma facilitates this counting.

Lemma 3.7 *For any $t \leq m$, new bits that arrive at C enter only at nodes z_i with $1 \leq i \leq t$.*

Proof: We prove, by induction on t, that by time t, a bit which did not go through C may only be in the sender or the nodes $\{v_{i,k} : i \leq t\}$. The base $t = 0$ follows from the fact that at time 0, all bits reside in s. For the induction step, it is sufficient to note that all edges incident to $\{v_{i,k} : 1 \leq i \leq t\}$ are eiher incident to nodes in $V(C)$ or to nodes in $\{v_{i,k} : 1 \leq i \leq t+1\} \cup \{s\}$. ∎

Lemma 3.8 *For $t < m$, the number of bits arriving at r by time t is less than $t^2 mB/2$.*

Proof: First, observe that the distance from s to r in $H_m - E(C)$ is $m + 1$. It follows that any bit arriving at r by time m must have traversed an edge of $E(C)$. At each time step t, we distinguish between bits crossing edges of $E(C)$ at step t for the first time and bits that have already crossed an internal edge of C earlier. Now, by Lemma 3.7, at each time step t, bits crossing an edge of $E(C)$ for the first time at step t can originate only from nodes $z_1, \ldots, z_t$. It therefore follows that the number of new bits that cross edges of $E(C)$ at time t is at most $(m-1)tB$: this is true since the number of $E(C)$ edges adjacent to a node of $V(C)$ is exactly $m - 1$. Noting that each bit delivered at r at time $t < m$ must have been a new bit crossing an edge of $E(C)$ at some time before t, we can bound the number of bits delivered at r by time t by

$$\sum_{i=0}^{t-1} (m-1)iB = \frac{(m-1)t(t-1)B}{2} < \frac{t^2 mB}{2} . \quad ∎$$

We can therefore conclude the lower bound on the mailing problem.

Lemma 3.9 *For any correct mailing algorithm A we have that $T_A(H_m, s, r, m^3) = \Omega(n^{1/4}/B^{1/2})$ for $m \geq 2$.*

Proof: By Lemma 3.8, we have that the maximal number of bits that A can deliver at r in the first $m/2B$ time units is less than m^3. It follows that

$$\begin{aligned} T_A(H_m, s, r, m^3) &\geq \frac{m}{2B} \\ &= \Omega(\sqrt[4]{n}/\sqrt[2]{B}) . \quad ∎ \end{aligned}$$

The lower bound on the time complexity of algorithms for the MST problem follows by a direct reduction to the mailing problem, similarly to Section 3.1. We omit details here.

Theorem 3.10 *For every $m \geq 2$, there exists a family $\mathcal{H}_m$ of diameter 3 graphs such that every distributed algorithm MST construction algorithm requires $\Omega(n^{1/4}/B)$ time on some graph of $\mathcal{H}_m$ in the B model.*

3.3 A lower bound for randomized algorithms

A randomized algorithm is a deterministic algorithm that has access to a tape of random bits (in addition to the standard inputs). A randomized Las-Vegas algorithm is said to solve a given problem if for any instance of the problem, the algorithm produces the correct output in finite expected time (where the expectation is taken over the space of random tapes). In this section we show that all Las-Vegas algorithms for MST admit the same asymptotic bounds we proved for deterministic algorithms.

Let us concentrate on the diameter 4 graph F_m of Section 3.1; the proof for diameter 3 graphs is similar. We first prove a lower bound on the mailing problem over the graph F_m.

The tool we use for establishing the lower bound on the expected time complexity of any randomized (Las-Vegas) distributed algorithm for the mailing problem is Yao's method [11]. By this method, in order to bound this complexity (over any distribution D_1 on random input strings X), it suffices to find some "difficult" distribution D_2 on the inputs, and prove that on D_2, every *deterministic* mailing algorithm requires $\Omega(n^{1/3}/B)$ *expected* time.

Fix a deterministic distributed MST construction algorithm A. Lemma 3.4 states that for algorithm A, there exists an m^2-bit string X for which $T_A(F_m, s, r, X) = \Omega(m/B)$. Note, however, that Lemma 3.4 makes a considerably stronger claim, guaranteeing the same bound for *every* string X. This implies that, assuming the input string is taken randomly (from any distribution over all possible strings), the expected time required by the algorithm A is at least $\Omega(m/B)$.

Remark 3: Again, discarding the restriction to "direct communication" made in Remark 1 requires us to be slightly more formal. We can show that for every $m \geq 2$, and for at least half of the possible m^2-bit input strings X of the mailing problem, $T_A(F_m, s, r, X) \geq m/(2B)$. To show this, we need to consider the set of possible states a node v may be in at any given stage t of the execution of a mailing algorithm on some m^2-bit input X. (The state of a node consists of all its local data, hence it is affected by its input, topological knowledge, and history, namely, all incoming messages.) Each node starts at some initial state, and as the computation progresses, its set of possible states becomes larger. In particular, when the execution starts at round 0, each of the nodes is in one specific initial local state, except for the sender s, which may be in any one of 2^{m^2} states, determined by the value of the input string X. Upon termination, the string X should be known to the receiver r, hence r should be in one of 2^{m^2} states. Our argument is based on analyzing the growth rate of the sets of possible states, and showing that at least $\Omega(m^2/B)$ time must elapse until r's set of possible states is of size 2^{m^2}. More formally, let φ_X denote the execution of A on an m^2-bit input X in the graph F_m. Denote the *state* of the vertex v at the beginning of round t during the execution φ_X on the input X by $\sigma(v, t, X)$. In two different executions φ_X and $\varphi_{X'}$, a vertex reaches the same state at time t, i.e., $\sigma(v, t, X) = \sigma(v, t, X')$, iff it receives the same sequence of messages on each of its incoming links; for different sequences, the states are distinguishable. By Lemma 3.2, after time $t = m/B - 1$, the number of bits arriving at r from s by time t is at most $m^2 - mB$. Messages arriving at r from other sources do not affect the growth of r's set of possible states, as these nodes start at the same initial state in all executions. (This intuitive argument will be made formal in the full paper.) Therefore, after time $t = m/B - 1$, the number of different states r may be in is at most 2^{m^2-mB}. As each input string must lead to a distinct final state at r, it follows that for at least $2^{m^2} - 2^{m^2-mB}$ of the input strings, the computation has not terminated by time $t = m/B - 1$. Hence, assuming the input string is randomly taken from the uniform distribution over all possible strings, the expected time required by the algorithm A is at least $(2^{m^2} - 2^{m^2-mB}) \cdot m/B + 2^{m^2-mB} \cdot 1 \geq m/(2B)$.

The reduction from MST to mailing applies without change, hence we get the following result.

Theorem 3.11 *There exists a family of n-vertex graphs of diameter 4 on which any randomized Las-Vegas distributed algorithm for the MST problem in the B model requires $\Omega(n^{1/3}/B)$ expected time.*

4 Fast algorithms for diameter 2 graphs

In this section we demonstrate an exponential gap in the time complexity of MST: graphs with diameter 2 can always be solved in $O(\log n)$ steps, whereas, as we showed above, graphs with diameter 3 may require up to $n^{\Omega(1)}$ steps. The main idea is to use Boruvka's algorithm, combined with a simple technique for quickly disseminating multiple minima in graphs of diameter 2. The algorithm is somewhat tricky, so for the sake of exposition we first present an algorithm with running time $O(\log^2 n)$, and then explain how to change this algorithm into one taking only $O(\log n)$ time. We omit most proofs of the algorithm from this extended abstract.

4.1 Segmented minima

The idea of the algorithm is that since the graph has diameter 2, all the necessary information can be relayed by neighbors; the trick is that relaying must be done judiciously, since in the B model, a node cannot transmit *all* information it received in the previous rounds to *all* its neighbors.

The new algorithmic tool we introduce is *segmented minima*, defined as follows. Let $G = (V, E)$ be a graph of diameter 2, and suppose that for each $v \in V$ we have a *segment number* $f(v)$ and a value $a(v)$. The task of segmented minima is to compute, for each $v \in V$, the value $b(v) = \min\{a(u) : f(u) = f(v)\}$. In words, each node should find the minimum of all a values in its segment. Note that the segments are not necessarily connected, and even if they are connected, their diameter may be much larger than 2.

Lemma 4.1 *Segmented minima can be computed in 2 steps in the B model, assuming that $(f(v), a(v))$ can be represented by less than B bits together.*

Proof: The algorithm to solve the problem is as follows, at each node v.

1. Send $(f(v), a(v))$ to all neighbors.
2. Send to each neighbor u the minimal value received from u's segment. Formally, send $\min\{a(w) : f(w) = f(u)\}$, where the minimum is from the values received in the first step.
3. Set $b(v)$ to be the minimum of all values received in the second step.

The time complexity claim is obvious. To see that the algorithm is correct, consider any node v, and let v_0 be the node such that $a(v_0) = \min\{a(w) : f(w) = f(v)\}$. We need to prove that $a(v_0) = b(v)$. To see that, first note that any value received at v in the second step is indeed an a value of a node in v's segment, and hence $b(v) \geq a(v_0)$. Finally, observe that since the diameter of the graph is 2, there must exist a node u such that $(u, v_0) \in E$ and $(v, u) \in E$. By the algorithm, this node u receives $a(v_0)$ in the first step, and it sends $a(v_0)$ to v in the second step, and hence $b(v) \leq a(v_0)$. ∎

4.2 A simple $O(\log^2 n)$ time algorithm

We need to define some terms (derived from Boruvka's Algorithm). A *fragment* is a collection of nodes connected by edges already added to the MST. The *leader* of a fragment is the node whose ID is minimal in the fragment. At any given time step, we use $f(v)$ to denote the leader of the fragment that contains a node v. The algorithm works in phases. Each phase takes $O(\log n)$ time units, and there are $\log n + 1$ phases, for a total of $O(\log^2 n)$ time complexity.

The key to the algorithm is the following invariant, satisfied at the beginning of each phase: *all nodes in a fragment know the ID of their leader.* Initially, each node forms a singleton fragment with itself being the leader and the invariant holds trivially. We now describe the steps taken in phase k for all $0 \leq k \leq \log n$.

Phase k:

1. Each node v sends the ID of its current leader $f(v)$ to all its neighbors. Note that after this step, each node also knows which of its incident edges is connected to another fragment.
2. The nodes execute segmented minima, with the a values being the weight of their lightest incident edge outgoing to to another fragment. After this is done, each node v knows which is the lightest edge outgoing from its fragment. We call these edges *chosen edges*. The chosen edges are added to the MST.
3. Each node adjacent to a chosen edge informs the node at the other side that this edge was chosen, so

that each fragment knows (distributively) all the chosen edges it is incident to, and, more importantly, the IDs of the leaders of all fragments that will merge with it.

4. Each node computes locally the minimal ID of a leader of a fragment it is incident to by a chosen edge (including its own fragment leader ID). These values are used as the a values for a segmented minima computation. When this computation ends, each node knows what is the smallest ID of the leader among all fragments its fragment is joined to with a chosen edge. Each node assigns its b value as its new leader ID.

5. $\log n + 1$ *indirection resolution* rounds are taken (see below).

The situation after Step 4 is done is that all nodes know what is the smallest leader ID among all leaders of fragments adjacent to their fragment. This is not sufficient. The problem is that a single round of leader update may lead to inconsistencies in case there are long "learning chains." For instance, it may happen for three nodes v, u, w, that while v assigns $f(u)$ to be its leader, u may assign $f(w)$ to be its own leader. The solution to this difficulty is to use $\log n + 1$ rounds of leader updates, a process called "pointer jumping" [10]. In our context, this is done as follows.

Phase k pointer jumping: The pointer jumping part of the phase consists of $\log n + 1$ *indirection resolutions* (abbreviated IR henceforth). The idea in an IR is that each node records locally, for each of its neighbors, the *previous* leader ID it announced. This allows it to inform other nodes if that neighbor changes its leader. Specifically, IR works as follows at a node v. In the first step of an IR, v informs all its neighbors of the ID of its current leader, and records, for each of its neighbors u, the value of $f(u)$ as received by v. In the second step of IR, for each neighbor u, v informs u of the value of $f(f(u))$, if there is a neighbor w of v such that its previous $f(w)$ is the current $f(u)$; in this case, v sends to u the last value of $f(w)$. The correctness of this technique relies on the fact that if a node u decides to change its leader to $f(w)$, then since the diameter of the graph is 2, there must exists a node v at distance 1 from a node in the fragment led by $f(w)$, which can inform u about the ID of the *current* leader of w.

4.3 From $O(\log^2 n)$ to $O(\log n)$ time

We now explain how to modify the algorithm above to run in logarithmic time. The aim is to reduce the amount of time consumed by the pointer jumping part. Let us examine the pointer jumping part more closely. Pointer jumping consists of a series of IR steps, where each indirection resolution is a pair of time steps, ending with a re-computation of the leader. Call an IR step at a fragment *idle* if the fragment does not update its pointers, and no pointers are updated to point at the fragment. It is easy to verify that in a sequence of i non-idle of IR steps, the size of fragment grows by a factor of at least 2^{i-1}. This property is used in the algorithm above: clearly, $\log n + 1$ pointer updates are sufficient to resolve all indirections. The crucial observation we use to reduce the time complexity in the new algorithm is that in fact, the total number of non-idle indirection resolution steps for each node is $\log n + 1$ *throughout the the execution of the algorithm.* In other words, we amortize the pointer jumping part of the algorithm over all phases, thus reducing its time complexity from $O(\log^2 n)$ to $O(\log n)$.

More specifically, the algorithm works as follows. Each node has an additional state bit, which we call "passive" or "active." The overall algorithm is very similar to the one above, except that not all nodes take part in all steps, and that the pointer jumping part takes only a constant number of steps. Each phase k proceeds as follows. Initially, all nodes are active.

1. Each active node broadcasts its leader ID to all its neighbors, along with its state bit.

2. Segmented minima is now executed, but only *active* nodes have a values, set to be the weight of the lightest edge outgoing from the node to a different *active* fragment. Passive nodes take part only in the second step of the segmented minima. Active nodes thus compute their chosen edges as before.

3. Each *active* node v incident to a chosen edge sends $f(v)$ over the chosen edge.

4. Each *active* node computes locally the smallest leader ID among all fragments incident to it by a chosen edge, including its own leader ID. Then the active nodes execute segmented minima over these values (again, passive nodes help in the second step of the segmented minima). The value obtained in this step is set to be the new leader ID.

5. In the pointer jumping part, *all* nodes participate.

   - First, two rounds of indirection resolution are executed by all nodes. (Each indirection resolution consists of each node v sending out $f(v)$ to all neighbors, and then, sending to each node u, the identity of $f(f(u))$ as sent in the previous step by a node w whose previous leader was u; then each node sets its new leader to be the maximum of its old leader and the new leader it heard about.)
   - Then, another round of indirection resolution is performed, this time *without* the assignment of a new leader. In an additional step, each node v informs each of its neighbors u, whether there exists a node w with $f(w) \neq f(u)$ that wishes to assign $f(w) \leftarrow f(u)$, i.e., w wishes to joins the fragment that u belongs to. All nodes that wish to join another fragment, as well as all node that belong to a fragment that some node wishes to join, set their state to *passive*. All other nodes set their state to *active*.

The key argument in the analysis of the algorithm is the following.

Lemma 4.2 *At the beginning of phase k the number of nodes in each fragment is at least* 2^k. *Moreover, if the fragment is passive then the number of nodes in the fragment is at least* 2^{k+1}.

5 Conclusion

In this paper we have proved the somewhat surprising result that the complexity of distributed MST computation changes exponentially when the diameter changes from 2 to 3. The significance of diameter 2 may be explained by the fact that the only bottlenecks in graphs of diameter 2 may be articulation *nodes*: nodes, however, may relay linear number of bits in a single time step. This property does not hold for graphs of diameter 3, where the bottleneck may be an edge, that can relay only B bits in a single time step.

We do not know whether the lower bounds on the time complexity we have demonstrated are the best possible. We leave this question open.

References

[1] B. Awerbuch, Optimal distributed algorithms for minimum-weight spanning tree, counting, leader election and related problems, *Proc. 19th Symp. on Theory of Computing*, pp. 230–240, May 1987.

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

[3] J. Garay, S. Kutten and D. Peleg, A sub-linear time distributed algorithm for minimum-weight spanning trees, *SIAM J. on Computing* Vol. 27, (1998), 302–316.

[4] S. Kutten and D. Peleg, Fast distributed construction of small k-dominating sets and applications, *J. of Algorithms*, Vol. 28, (1998), 40–66.

[5] M. Li and P.M.B. Vitanyi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 1993.

[6] N. Linial, Locality in distributed graph algorithms, *SIAM J. on Computing* Vol. 21, (1992), 193–201.

[7] D. Peleg, Time-optimal leader election in general networks, *Journal of Parallel and Distributed Computing*, Vol. 8, (1990), 96–99.

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

[9] D. Peleg and V. Rubinovich, A near-tight lower bound on the time complexity of distributed MST construction, *SIAM J. on Computing* Vol. 30, (2000), 1427–1442.

[10] Y. Shiloach and U. Vishkin, An $O(\log n)$ parallel connectivity algorithm, *J. of Algorithms* Vol. 3, (1982), 57–67.

[11] A. Yao, Probabilistic computations: Towards a unified measure of complexity, *Proc. 17th IEEE Symp. on Foundations of Computer Science*, pages 222–227. Comp. Soc. IEEE, April 1977.