Message Lower Bounds via Efficient Network Synchronization[*]

Gopal Pandurangan[†] David Peleg[‡] Michele Scquizzato[§]

Abstract

We present a uniform approach to derive message-time tradeoffs and message lower bounds for synchronous distributed computations using results from communication complexity theory.

Since the models used in the classical theory of communication complexity are inherently asynchronous, lower bounds do not directly apply in a synchronous setting. To address this issue, we show a general result called *Synchronous Simulation Theorem (SST)* which allows to obtain message lower bounds for synchronous distributed computations by leveraging lower bounds on communication complexity. The SST is a by-product of a new efficient synchronizer for complete networks, called σ, which has simulation overheads that are only logarithmic in the number of synchronous rounds with respect to both time and message complexity, even in networks with limited bandwidth. Synchronizer σ is particularly efficient in simulating synchronous algorithms which employ *silence*, a situation that occurs when in some round no processor sends any message. In particular, a curious property of this synchronizer, which sets it apart from its predecessors, is that it is *time-compressing*, and hence in some cases it may result in a simulation that is faster than the original execution.

While the SST gives near-optimal message lower bounds up to large values of the number of allowed synchronous rounds r (usually polynomial in the size of the input), it fails to provide meaningful bounds when the synchronous algorithm to be simulated may comprise a very large number of rounds. To complement the bounds provided by the SST, we then derive message lower bounds for the synchronous message-passing model that are *unconditional*, that is, independent of r, by establishing novel lower bounds for multi-party synchronous communication complexity.

We apply our approach to show (almost) tight message-time tradeoffs and message lower bounds for several fundamental problems in the synchronous message-passing model of distributed computation. These include sorting, matrix multiplication, and several graph problems. All these lower bounds hold for any distributed algorithms, including randomized Monte Carlo algorithms.

Key words: distributed algorithms; synchronous message-passing; communication complexity; lower bounds; synchronizers

[*]A preliminary version of this work [41] was presented at the *23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO 2016)*.

[†]Department of Computer Science, University of Houston, Houston, TX 77204, USA. E-mail: gopalpandurangan@gmail.com. Supported, in part, by National Science Foundation (NSF) grants CCF-1527867, CCF-1540512, IIS-1633720, CCF-BSF-1717075, and by US-Israel Binational Science Foundation (BSF) grant 2016419.

[‡]Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel. E-mail: david.peleg@weizmann.ac.il. Supported, in part, by a BSF-NSF grant (2016732).

[§]Department of Mathematics, University of Padova, 35121 Padova, Italy. E-mail: michele.scquizzato@unipd.it. Most of the work was done while at the University of Houston.

1 Introduction

Message complexity, which refers to the total number of messages exchanged during the execution of a distributed algorithm, is one of the two fundamental complexity measures used to evaluate the performance of algorithms in distributed computing [46]. Although time complexity is the most important measure in practice, message complexity is also significant. In fact, in practice the performance of the underlying communication subsystem is influenced by the load on the message queues at the various sites, especially when many distributed algorithms run simultaneously. Consequently, as discussed e.g. in [22], optimizing the message (as well as the time) complexity in some models for distributed computing has direct consequences on the time complexity in other models. Moreover, message complexity has also a considerable impact on the auxiliary resources used by an algorithm, such as energy. This is especially crucial in contexts, such as wireless sensor networks, where processors are powered by batteries with limited capacity. Besides, from a physical standpoint, it can be argued that energy leads to more stringent constraints than time does since, according to a popular quote by Peter M. Kogge, "You can hide the latency, but you can't hide the energy."

Investigating the message complexity of distributed computations is therefore a fundamental task. In particular, proving lower bounds on the message complexity has been a major focus in the theory of distributed computing for decades (see, e.g., [36, 50, 46, 52]). Tight message lower bounds for several fundamental problems such as leader election [28, 2, 30, 31], broadcast [5, 30], selection [17], sorting [35, 58], spanning trees [27, 36, 52, 30], minimum spanning trees [28, 52, 36, 30, 22, 43], graph connectivity [22], maximal independent set [40], and triangle enumeration [45] have been derived in various models for distributed computing.

One of the most important distinctions among message passing systems is whether the mode of communication is synchronous or asynchronous. In this paper we focus on proving lower bounds on the message complexity of distributed algorithms in the synchronous communication setting. Most of the synchronous message lower bounds mentioned above (e.g., [27, 28, 5, 30, 22]) use *ad hoc* (typically combinatorial) arguments, which usually apply only to the problem at hand. In this paper, on the other hand, the approach is to use *communication complexity* [29] as a uniform tool to derive message lower bounds for a variety of problems in the synchronous setting.

Communication complexity, originally introduced by Yao [57], is a subfield of complexity theory with numerous applications in several markedly different branches of computer science (see, e.g., [29] for a comprehensive treatment). In the basic two-party model, there are two distinct parties, usually referred to as Alice and Bob, each of whom holds an n-bit input, say x and y. Neither knows the other's input, and they wish to collaboratively compute a function $f(x, y)$ by following an agreed-upon protocol. The cost of this protocol is the number of bits communicated by the two players for the worst-case choice of inputs x and y. It is important to notice that this simple model is inherently asynchronous, since it does not provide the two parties with a common clock. Synchronicity, however, makes the model subtly different, in a way highlighted by the following simple example (see, e.g., [51]). If equipped with a common clock, the two parties could agree upon a protocol in which time-coding is used to convey information: for instance, an n-bit message can be sent from one party to the other by encoding it with a single bit sent in one of 2^n possible synchronous rounds (keeping silent throughout all the other rounds). Hence, in a synchronous setting, *any* problem can be solved (deterministically) with communication complexity of one bit. This is a big difference compared to the classical (asynchronous) model! Likewise, as observed in [22], $k - 1$ bits of communication suffice to solve *any* problem in a complete network of k parties that initially agree upon a leader (e.g., the node with smallest identifier) to whom they each send the bit that encodes their own input. However, the low message complexity comes at the price of a high number of synchronous rounds, which has to be at least exponential in the size of the input that has to be encoded, as a single bit within time t can encode at most $\log t$ bits of information. The above observation raises some intriguing questions: if one allows only a small number of rounds (e.g., polynomial in n) can such a low message complexity be achieved? More generally, how can one show message lower bounds in the synchronous distributed computing model vis-a-vis the time (round) complexity? This paper tries to answer these questions in a general and comprehensive way.

As discussed before, in the synchronous setting, for message lower bounds it does not suffice to appeal directly to lower bounds from communication complexity theory. This is unlike the situation when we are interested in showing time lower bounds. In particular, the Simulation Theorem of [11], which is useful to show time lower bounds, does not apply if we want to show message lower bounds in a synchronous setting. Informally speaking, the reason is that "silence" (i.e., when in some round no party sends any message) does not help to save time, as it consumes rounds anyway.

Our approach is based on the design of a new and efficient *synchronizer* that can efficiently simulate synchronous algorithms that use (a lot of) silence, unlike previous synchronizers. Recall that a synchronizer ν transforms an algorithm S designed for a synchronous system into an algorithm $A = \nu(S)$ that can be executed on an asynchronous system. The goal is to keep T_A and C_A, the time and communication complexities of the resulting asynchronous algorithm A, respectively, close to T_S and C_S, the corresponding complexities of the original synchronous algorithm S. The synchronizers appearing in the literature follow a methodology (described, e.g., in [46]) which resulted in bounding the complexities T_A and C_A of the asynchronous algorithm A for every input instance $\mathcal{I}$ as

$$T_A(\mathcal{I}) \leq T_{\text{init}}(\nu) + \Psi_T(\nu) \cdot T_S(\mathcal{I}),$$
$$C_A(\mathcal{I}) \leq C_{\text{init}}(\nu) + C_S(\mathcal{I}) + \Psi_C(\nu) \cdot T_S(\mathcal{I}),$$

where $\Psi_T(\nu)$ (resp., $\Psi_C(\nu)$) is the time (resp., communication) overhead coefficient of the synchronizer ν, and $T_{\text{init}}(\nu)$ (resp., $C_{\text{init}}(\nu)$) is the time (resp., communication) initialization cost. In particular, the early synchronizers, historically named α [4], β [4], γ [4], and δ [48] (see also [46]), handled each synchronous round separately, and incurred a communication overhead of at least $O(k)$ bits per synchronous round, where k is the number of processors in the system. The synchronizer μ of [6] remedies this limitation by taking a more global approach, and its time and communication overheads $\Psi_T(\mu)$ and $\Psi_C(\mu)$ are both $O(\log^3 k)$, which is at most a polylogarithmic factor away from optimal under the described methodology.

Note, however, that the dependency of the asynchronous communication complexity $C_A(\mathcal{I})$ on the synchronous time complexity $T_S(\mathcal{I})$ might be problematic in situations where the synchronous algorithm takes advantage of synchronicity in order to exploit *silence*, using time-coding to convey information while transmitting fewer messages (e.g., see [22, 24]). Such an algorithm, typically having low communication complexity but high time complexity, translates into an asynchronous algorithm with high time *and* communication complexities. In this situation it is preferable that the asynchronous communication complexity $C_A(\mathcal{I})$ be a function of only the communication complexity $C_S(\mathcal{I})$, and not of the time complexity $T_S(\mathcal{I})$ (provided that the overhead coefficient $\Psi_C(\nu)$ is a slow-growing function of $T_S(\mathcal{I})$). Hence, in this situation we may prefer a simulation methodology that results in a communication dependency of the form

$$C_A(\mathcal{I}) \leq C_{\text{init}}(\nu) + \Psi_C(\nu) \cdot C_S(\mathcal{I}),$$

and where $\Psi_C(\nu)$ is a slow-growing function of the number of rounds $T_S(\mathcal{I})$ of the synchronous algorithm.

1.1 Our Contributions

We present a uniform approach to derive message lower bounds for synchronous distributed computations by leveraging results from the theory of communication complexity. In this sense, this can be seen a companion paper of [11], which leverages the connection between communication complexity and distributed computing to prove lower bounds on the time complexity of synchronous distributed computations.

A New and Efficient Synchronizer. Our approach, developed in Section 3, is based on the design of a new and efficient synchronizer for complete networks, which we call synchronizer σ,[1] and which is of independent interest. The new attractive feature of synchronizer σ, compared to existing ones, is that it is *time-compressing*. To define this property, we say that a synchronous round is *active* (or *communicative*) when at least one node of the network sends a message, and *inactive* (or *quiet*) otherwise. Synchronizer σ compresses the execution time of the simulation by discarding the inactive rounds, and remaining only with the active ones. This is in sharp contrast to all previous synchronizers, whereby every single round of the synchronous execution is simulated in the asynchronous network.

The distinction between active and inactive rounds is achieved through the concept of *tentative time*, a value that indicates, for each node, the next synchronous round on which that node plans to send a message. These values are maintained by one designated node of the network, which orchestrates the whole simulation by determining the next synchronous round to be simulated and by coordinating the start of such a round by sending the corresponding announcements only to those nodes that will be active in that round. These values are maintained and updated throughout the whole simulation, and cause an overhead of at most $O(\log T_S)$ bits for each message sent in the synchronous execution.

Table 1 compares the complexities of various synchronizers when used for complete networks.

Synchronizer	Time Complexity T_A	Message Complexity C_A
α [4]	$O(T_S)$	$O(k^2) + O(C_S) + O(T_S\, k^2)$
β [4]	$O(k) + O(T_S)$	$O(k \log k) + O(C_S) + O(T_S\, k)$
μ [6]	$O(k \log k) + O(T_S \log^3 k)$	$O(k \log k) + O(C_S) + O(T_S \log^3 k)$
σ [this paper]	$O(T_S^c \log_k T_S)$	$O(C_S \log_k T_S)$

Table 1: Comparison among different synchronizers for k-node complete networks. The message size is assumed to be $O(\log k)$ bits, and C_A is expressed in number of messages. T_S^c denotes the number of active synchronous rounds. (Note that on a complete network, synchronizers γ [4] and δ [48] are out-performed by β, hence their complexities are omitted from this table.)

Synchronous Simulation Theorem. As a by-product of synchronizer σ we have the *Synchronous Simulation Theorem (SST)*, which shows how message lower bounds for synchronous distributed computations can be derived by leveraging communication complexity lower bounds for the asynchronous model. Specifically, the SST provides a tradeoff between the communication complexity of a synchronous computation and the maximum number of synchronous rounds T_S allowed to the computation. This tradeoff reveals that communication complexity in the synchronous model can differ from the communication complexity in its asynchronous counterpart by a factor at most logarithmic in T_S, and thus that a communication lower bound for the asynchronous model can be applied, modulo such a factor, to its synchronous counterpart.

[1]We use σ as it is the first letter of the Greek word σιωπή, which means "silence".

Application: Message-Time Tradeoffs. In Section 4 we apply the SST to obtain message-time tradeoffs for several fundamental problems, such as sorting, Boolean matrix multiplication, graph connectivity, diameter computation, and approximate maximum matching. These results assume that the underlying communication network is complete, a case of interest in many settings (e.g., [34, 25]). Most of the resulting message lower bounds are tight (up to factors polylogarithmic in the input size of the problem) when the number of synchronous rounds T_S is at most polynomial in the input size. All these lower bounds hold even for randomized algorithms whose output may be incorrect with a small constant probability.

Unconditional Lower Bounds. While the SST gives essentially tight communication lower bounds up to large values of T_S, thees lower bounds vanish for larger values of T_S. To complement the bounds provided by the SST, in Section 5 we derive message lower bounds in the synchronous message-passing model which are *unconditional*, that is, independent of time, by establishing novel lower bounds for multi-party synchronous communication complexity. These are all of the form $\tilde{\Omega}(k)$,[2] Resorting to a multi-party case, as opposed to two-party, is crucial since, as observed earlier, two-party problems can be solved, exploiting clocks, with only one bit of communication.

1.2 Further Related Work

Peleg and Rubinovich [47] were the first to derive time lower bounds in a synchronous distributed setting by applying communication complexity techniques (see, e.g., [29]). Specifically, by applying such techniques to a new suitably constructed family of graphs, they proved a near-tight lower bound on the time complexity of distributed minimum spanning tree construction in the CONGEST model. Elkin [15] extended this result to approximation algorithms. The same technique was then used to prove a tight lower bound for minimum spanning tree verification [26]. Das Sarma et al. [11] explored this connection between the theory of communication complexity and distributed computing further by presenting almost tight time lower bounds for a long list of problems, including inapproximability results. Leveraging communication complexity lower bounds to derive time lower bounds for synchronous distributed computations has now become a standard technique, and has been used in a number of subsequent papers, see e.g. [38, 19, 21, 14, 25, 42, 44, 8, 1], as well as [39] and references therein.

Researchers have also investigated how to leverage results in communication complexity to establish lower bounds on the message complexity of distributed computations. For example, Tiwari [53] shows communication complexity lower bounds for deterministic computations over networks with some specific topologies. Woodruff and Zhang [55] study the message complexity of several graph and statistical problems in complete networks. Their lower bounds are derived through a common approach that reduces those problems from a new meta-problem whose communication complexity is established. However, the models considered in these two papers are inherently asynchronous, and therefore these lower bounds do not apply when synchronized discrete clocks are additionally assumed.

To the best of our knowledge, the first rigorous connection between the classical communication complexity theory and synchronized communication complexity has been established by Impagliazzo and Williams [24], who consider the natural extension of the standard two-party communication complexity model where the two parties are equipped with synchronized clocks. They show that for deterministic protocols these two models are equivalent up to a factor roughly logarithmic in the number of synchronous rounds. Specifically, they show that the communication complexity of deterministic protocols can be reduced by such a factor in

[2]Throughout this paper, notation $\tilde{\Omega}(\cdot)$ hides polylogarithmic factors in the input size $|\mathcal{I}|$ and in k, i.e., $\tilde{\Omega}(f(|\mathcal{I}|, k))$ denotes $\Omega(f(|\mathcal{I}|, k)/(\text{polylog}|\mathcal{I}| \text{ polylog } k))$.

the synchronous model, but no more than such a factor. The latter result follows from a general simulation of a synchronous protocol in the asynchronous model which shares some ideas with the simulation of this paper (which, however, is more involved since it involves k parties instead of just two.) Ellen et al. [16] claim simulation results among models for multi-party communication complexity. Their results are similar to our synchronous simulation theorem. However, their simulations do not consider time, whereas ours is time-efficient as well.

Interactive communication in a model where parties are allowed to remain silent is investigated in [12], which considers the communication complexity of computing symmetric functions in the multiparty setting. Other examples of distributed algorithms that make effective use of synchrony to reduce communication can be found, e.g., in [10, 18, 33, 30].

2 Preliminaries: Models, Assumptions, and Notation

The *message-passing model* is one of the fundamental models in the theory of distributed computing, and many variants thereof have been studied. It consists of a complete network of k nodes, represented as a complete undirected simple graph where nodes correspond to the processors of the network and edges represent bidirectional communication channels. Each node initially holds some portion of the input instance $\mathcal{I}$, and this portion is known only to itself and not to the other nodes. All nodes are initially awake, and simultaneously start executing the algorithm. Each node can communicate directly with any other node by exchanging messages. The goal is to jointly solve some given computational problem.

Nodes have a unique identifier of $O(\log k)$ bits. Before the computation starts, each node knows its own identifier and also the identifiers of its neighbors. (This is usually referred to as the KT_1 assumption [46].) Each link incident to a node has a unique representation in that node. All messages received at a node are stamped with the identification of the link through which they arrived. By the number of its incident edges, every node knows the value of k before the computation starts. All the local computation performed by the processors of the network happens instantaneously, and each processor has an unbounded amount of local memory. It is also assumed that both the computing entities and the communication links are fault-free.

A key distinction among message-passing systems is whether the mode of communication is synchronous or asynchronous. In the *synchronous* mode of communication, a global clock is connected to all the nodes of the network. The time interval between two consecutive pulses of the clock is called a *round*. The computation proceeds in rounds, as follows. At the beginning of each synchronous round, each node sends (possibly different) messages to its neighbors. Each node then receives all the messages sent to it in that round, and performs some local computation, which will determine what messages to send in the next round. In the *asynchronous* mode of communication, there is no global clock. Messages over a link incur finite but arbitrary delays (see, e.g., [20]). This can be modeled as each node of the network having a queue where to place outgoing messages, with an adversarial global scheduler responsible of dequeuing messages, which are then instantly delivered to their respective recipients. Communication complexity, the subfield of complexity theory introduced by Yao [57], studies the asynchronous message-passing model.

We now formally define the complexity measures studied in this paper. Most of these definitions can be found in [29]. The *communication complexity of a computation* is the total number of bits exchanged across all the links of the network during the computation (or, equivalently, the total number of bits sent by all parties). The *communication complexity of a distributed algorithm A* is the maximum number of bits exchanged during the execution of A over all possible inputs of a particular size. The *communication complexity of a problem* Π is the minimum communication complexity of any algorithm that solves Π. *Message complexity* refers to the total number of messages exchanged, where the message size is bounded by some value B of bits. Clearly, a lower bound on the communication complexity becomes a valid lower bound on the message complexity when divided by B.

In this paper we are interested in lower bounds for *Monte Carlo* distributed algorithms. A Monte Carlo algorithm is a randomized algorithm whose output may be incorrect with some probability. Formally, *algorithm A solves a problem Π with ϵ-error* if, for every input I, A outputs $\Pi(I)$ with probability at least $1-\epsilon$, where the probability is taken only over the random strings of the players. The *communication complexity of an ϵ-error randomized algorithm A on input I* is the maximum number of bits exchanged for any choice of the random strings of the parties. The *communication complexity of an ϵ-error randomized algorithm A* is the maximum, over all possible inputs I, of the communication complexity of A of input I. The *randomized ϵ-error communication complexity of a problem* Π is the minimum communication complexity of any ϵ-error randomized algorithm that solves Π. In a model with $k \geq 2$ parties, this is denoted with $R_{k,\epsilon}(\Pi)$. The same quantity can be defined likewise for a synchronous model, in which case it is denoted with $SR_{k,\epsilon}(\Pi)$. Throughout the paper we assume ϵ to be a small constant and therefore, for notational convenience, we will drop the ϵ in the notation defined heretofore.

We say that a randomized distributed algorithm uses a *public coin* if all parties have access to an infinitely long common random string. In this paper we are interested in lower bounds for public-coin randomized distributed algorithms. Clearly, lower bounds of this kind also hold for *private-coin* algorithms, in which parties do not share a common random string, since allowing a public coin only gives more power to the algorithms.

Another complexity measure of interest in distributed computing is the *time complexity*. In the synchronous mode of communication, it is defined as the (worst-case) number of synchronous rounds until all the required outputs are produced, or until the processes all halt. It is additionally referred to as *round complexity*. Following [11], we define the *randomized r-round ϵ-error communication complexity of a problem* Π in a synchronous model to be the minimum communication complexity of any algorithm that solves Π with error probability ϵ when it runs in at most r rounds.[3] We denote this quantity with $SR_{k,\epsilon,r}(\Pi)$. In the asynchronous case, the time complexity of a computation is the (worst-case) number of time units that it comprises, assuming that each message incurs a delay of at most one time unit [46, Definition 2.2.2]. Thus, in arguing about time complexity, a message is allowed to traverse an edge in any fraction of the time unit. This assumption is used only for the purpose of time complexity analysis, and does not imply that there is a bound on the message transmission delay in asynchronous networks.

Throughout this paper, we shall use interchangeably node, party, or processor to refer to elements of the network, whereas we will use vertex to refer to a node of the input graph when the problem Π is specified on a graph.

3 Efficient Network Synchronization and the Synchronous Simulation Theorem

3.1 The Simulation

We present synchronizer σ, an efficient (deterministic) simulation of a synchronous algorithm S, designed for a complete network of k nodes, in the corresponding asynchronous counterpart. The main idea underlying the simulation is to differentiate between active and inactive synchronous rounds, via the use of the concept of *tentative time*, in conjunction with the use of acknowledgments as a method to avoid congestion

[3]In this paper we interchangeably denote the number of synchronous rounds with T_S and r.

and thus reduce the time overhead in networks whose links have limited bandwidth. All messages are required to be *self-delimiting*, i.e., no one is a prefix of another, so that it is possible to uniquely determine when one message has been sent or received completely.

One of the k nodes is designated to be a *coordinator*, denoted with $\mathcal{C}$, which organizes and synchronizes the operations of all the processors.[4] Since, by assumption, nodes have initial knowledge of the identity of their neighbors, the coordinator can be agreed upon (e.g., the node with smallest identifier) before the simulation begins. The coordinator starts the simulation by sending to each node a message $\mathtt{START}(1)$, instructing them to start the simulation of round 1.

At any given time each node v maintains a *tentative time* estimate $\mathsf{TT}(v)$, defined as the next synchronous round in which v plans to send a message to one (or more) of its neighbors. This estimate may change at a later point, i.e., v may send out messages earlier than time $\mathsf{TT}(v)$, for example in case v receives a message from one of its neighbors, prompting it to act. However, assuming no such event happens, v will send its next message on round $\mathsf{TT}(v)$. (In case v currently has no plans to send any messages in the future, it sets its estimate to $\mathsf{TT}(v) = \infty$.) The coordinator $\mathcal{C}$ maintains k local variables, which store, at any given time, the tentative times of all the nodes of the network.

We now describe the execution of phase t of the simulation, which simulates the actions of the processors in round t of the execution ξ_S of algorithm S in the synchronous network. Phase t of the simulation starts when the coordinator realizes that the current phase, simulating some round t' where $t' < t$, is completed, in the sense that all messages that were supposed to be sent and received by the processors on round t' of ξ_S were sent and received in the simulation on the asynchronous network. The phase proceeds as follows.

(1) The coordinator $\mathcal{C}$ determines the minimum value of $\mathsf{TT}(v)$ over all processors v, and sets t to that value. (In the first phase, the coordinator sets $t = 1$ directly.) If $t = \infty$ then the simulation is completed and it is possible to halt. By definition of tentative time, if $t > t'+1$ then the synchronous rounds $t'+1, \ldots, t-1$ are inactive rounds, that is, in which all processors are silent. Thus, the system conceptually skips all rounds $t'+1, \ldots, t-1$, and goes straight to simulating round t. Since only the coordinator can detect the halting condition, it is also responsible for informing the remaining $k-1$ nodes by sending, in one time unit, $k-1$ additional $\mathtt{HALT}$ messages to each of them.

(2) The coordinator (locally) determines the set of *active* nodes, defined as the set of nodes whose tentative time is t, that is,

$$\mathcal{A}(t) = \{v \mid \mathsf{TT}(v) = t\},$$

and sends to each of them a message $\mathtt{START}(t)$ instructing them to start round t. (In the first phase, all nodes are viewed as active, i.e., $\mathcal{A}(1) = V$).

(3) Upon the receipt of this message, each active node v sends all the messages it is required by the synchronous algorithm to send on round t, to the appropriate subset $\mathcal{N}(v,t)$ of its neighbors. This subset of neighbors is hereafter referred to as v's *clan* on round t, and we refer to v itself as the *clan leader*. We stress that these messages are sent directly to their destination; they must not be routed from v to its clan via the coordinator, as this might cause congestion on the links from the coordinator to the members of $\mathcal{N}(v,t)$.

(4) Each neighbor $w \in \mathcal{N}(v,t)$ receiving such a message immediately sends back an acknowledgment directly to v. Note that receiving a message from v may cause w to want to change its tentative time $\mathsf{TT}(w)$. However, w must wait for now with determining the new value of $\mathsf{TT}(w)$, for the following reason. Note that w may belong to more than one clan. Let $\mathcal{A}_w(t) \subseteq \mathcal{A}(t)$ denote the set of active nodes which are

[4]The coordinator should not be confused with the notion of coordinator in the variant of the message-passing model introduced in [13]. In the latter, (1) the coordinator is an additional party, which has no input at the beginning of the computation, and which must hold the result of the computation at the end of the computation, and (2) nodes of the network cannot communicate directly among themselves, and therefore they can communicate only with the coordinator.

required to send a message to w on round t (namely, the clan leaders to whose clans w belongs). At this time, w does not know the set $\mathcal{A}_w(t)$, and therefore it cannot be certain that no additional messages have been sent to it from other neighbors on round t. Such messages might cause additional changes in $\mathsf{TT}(w)$.

(5) Once an active node v has received acknowledgments from each of its clan members $w \in \mathcal{N}(v,t)$, v sends a message $\texttt{SAFE}(v,t)$ to the coordinator $\mathcal{C}$.

(6) Once the coordinator $\mathcal{C}$ has received messages $\texttt{SAFE}(v,t)$ from all the active nodes in $\mathcal{A}(t)$, it knows that all the original messages of round t have reached their destinations. What remains is to require all the nodes that were involved in the above activities (namely, all clan members and leaders) to recalculate their tentative time estimate. Subsequently, the coordinator $\mathcal{C}$ sends out a message `ReCalcT` to all the active nodes of $\mathcal{A}(t)$ (which are the only ones $\mathcal{C}$ knows about directly), and each $v \in \mathcal{A}(t)$ forwards this message to its entire clan, namely, its $\mathcal{N}(v,t)$ neighbors, as well.

(7) Every clan leader or member $x \in \mathcal{A}(t) \cup \bigcup_{v \in \mathcal{A}(t)} \mathcal{N}(v,t)$ now recalculates its new tentative time estimate $\mathsf{TT}(x)$, and sends it directly to the coordinator $\mathcal{C}$. (These messages must not be forwarded from the clan members to the coordinator via their clan leaders, as this might cause congestion on the links from the clan leaders to the coordinator.) The coordinator immediately replies each such message by sending an acknowledgment directly back to x.

(8) Once a (non-active) clan member w has received such an acknowledgment, it sends all its clan leaders in $\mathcal{A}_w(t)$ a message `DoneReCalcT`. (Note that at this stage, w already knows the set $\mathcal{A}_w(t)$ of its clan leaders—it is precisely the set of nodes from which it received messages in step (3).)

(9) Once an active node v has received an acknowledgment from $\mathcal{C}$ as well as messages `DoneReCalcT` from every member $w \in \mathcal{N}(v,t)$ of its clan, it sends the coordinator $\mathcal{C}$ a message `DoneReCalcT`, representing itself along with all its clan.

(10) Once the coordinator $\mathcal{C}$ has received an acknowledgment from every active node, it knows that the simulation of round t is completed.

3.2 Analysis of Complexity

Theorem 1. *Synchronizer σ is a synchronizer for complete networks such that*

$$T_A = O\left(\left(1 + \frac{\log T_S}{B}\right) T_S^c\right), \qquad (1)$$

$$C_A = O(C_S \log T_S), \qquad (2)$$

where T_S^c is the number of synchronous rounds in which at least one node of the network sends a message, k is the number of nodes of the network, and B is the message size of the network, in which at most one message can cross each edge at each time unit.

Proof. For any bit sent in the synchronous execution ξ_S, the simulation uses $\lceil \log_2 T_S \rceil$ additional bits to encode the values of the tentative times, and a constant number of bits for the acknowledgments and for the special messages $\texttt{START}(t)$, $\texttt{SAFE}(v,t)$, `ReCalcT`, and `DoneReCalcT`. Observe, finally, that no congestion is created by the simulation, meaning that in each synchronous round being simulated each node sends and receives at most $O(1 + \lceil \log_2 T_S \rceil)$ bits in addition to any bit sent and received in the synchronous execution ξ_S. □

Observe that a somewhat surprising consequence of its time-compressing nature is that synchronizer σ may in certain situations result in a simulation algorithm *faster* than the original synchronous algorithm. Specifically, T_A can be strictly smaller than T_S when the number of synchronous rounds in which no node communicates is sufficiently high. (In fact, the time compression may occur even when simulating the original synchronous algorithm on another *synchronous* network, in which case the resulting simulation algorithm may yield faster, albeit more communication-intensive, synchronous executions.)

3.3 The Synchronous Simulation Theorem

Theorem 2 (Synchronous Simulation Theorem (SST))**.** *Let $SCC^{\mathcal{D}}_{k,r}(\Pi)$ be the r-round communication complexity of problem Π in the synchronous message-passing complete network model with k nodes, where $\mathcal{D}$ is the initial distribution of the input bits among the nodes. Let $CC^{\mathcal{D}'}_{k'}(\Pi)$ be the communication complexity of problem Π in the asynchronous message-passing complete network model with $k' \leq k$ nodes where, given some partition of the nodes of a complete network of size k into sets $S_1, S_2, \ldots, S_{k'}$, $\mathcal{D}'$ is the initial distribution of the input bits whereby, for each $i \in \{1, 2, \ldots, k'\}$, node i holds all the input bits held by nodes in S_i under the distribution $\mathcal{D}$. Then,*

$$SCC^{\mathcal{D}}_{k,r}(\Pi) = \Omega\left(\frac{CC^{\mathcal{D}'}_{k'}(\Pi)}{1 + \log r + \lceil (k-k')/k \rceil \log k}\right).$$

Proof. We leverage the communication complexity bound for synchronizer σ in Theorem 1. More precisely, we can use synchronizer σ to simulate any synchronous algorithm for problem Π to obtain an asynchronous algorithm for Π whose message complexity satisfies Equation (2) of Theorem 1. We first consider the case when $k' = k$. Rearranging Equation (2), and by substituting T_S with r, C_A with $CC^{\mathcal{D}}_{k}(\Pi)$, and C_S with $SCC^{\mathcal{D}}_{k}(\Pi)$, and by setting $B = 1$ (since $(S)CC$ is expressed in number of bits), we obtain the claimed lower bound on $SCC^{\mathcal{D}}_{k,r}(\Pi)$.

Now we consider the case $k' < k$. In this case, we need to make a minor modification to the simulation of Section 3.1: since we do not assume that messages contain the identifiers of sender and receiver, when the network carrying the simulation has fewer nodes than the network to be simulated, in each message the identifiers of both the source and the destination of that message (which, by the KT_1 assumption of Section 2, are initially known to the sender of the message) have to be appended to it. This is the sole alteration needed for the simulation to handle this case. This entails $\lceil (k-k')/k \rceil \cdot 2\lceil \log k \rceil$ additional bits to be added to each message. In this case, the communication complexity of synchronizer σ is increased by a factor of $O(\lceil (k-k')/k \rceil \log k)$. This gives the claimed result. $\square$

Observe that CC and SCC can be either both deterministic or both randomized. In the latter case, such quantities can be plugged in Theorem 2 according to the definition of ϵ-error r-round algorithm given in Section 2.

4 Message-Time Tradeoffs for Synchronous Distributed Computations

In this section we apply the Synchronous Simulation Theorem to obtain lower bounds on the communication complexity of several fundamental problems in the synchronous message-passing model.

4.1 Sorting

We now derive a lower bound on the communication complexity of comparison-based sorting algorithms. Initially, each of the k parties holds n elements of $O(\log n)$ bits each. At the end, the i-th party must hold the $(i-1)n+1, (i-1)n+2, \ldots, (in)$-th order statistics. We have the following result.

Theorem 3. *For $k = \Omega(\log n)$, the randomized r-round ϵ-error communication complexity of sorting in the synchronous message-passing model with k parties all connected to each other is $\Omega(nk/\log k \log r)$ bits.*

Proof. We use a simple reduction from k-party set disjointness. Given an instance of k-party DISJ(n) $= \{X_i = x_{i,1}, x_{i,2}, \ldots, x_{i,n}$ s.t. $i \in [k] = \{1, 2, \ldots, k\}\}$, for any of such nk bits $x_{i,j}$, let $I = \{(j, x_{i,j})$ s.t. $i \in [k], j \in [n]\}$ be the set of nk inputs for the sorting problem. Once these nk pairs are ordered with respect to the first of the two elements, then $X_1, X_2, \ldots, X_k$ are disjoint if and only if there exists one party $i \in [k]$ whose n output pairs are all $(i, 1)$. Then, with $k - 1$ additional bits of communication all the k parties get to know the output to the k-party DISJ problem. For $k = \Omega(\log n)$, the k-party communication complexity of DISJ(n) in the coordinator model [13] is $\Omega(nk)$ bits [7], and this implies a lower bound of $\Omega(nk/\log k)$ in the classical message passing-model where all nodes can communicate directly with each other [7, 16]. The theorem follows by applying Theorem 2. □

This result implies that Lenzen's $O(1)$-round sorting algorithm for the Congested Clique model [32] has optimal message complexity (to within small logarithmic factors).

4.2 Matrix Multiplication

We now consider Boolean matrix multiplication, that is, the problem of multiplying two matrices over the semiring $(\{0, 1\}, \wedge, \vee)$. We will use the following result [54, Theorem 4]: suppose Alice holds a Boolean $m \times n$ matrix A, Bob holds a Boolean $n \times m$ matrix B, and the Boolean product of these matrices has at most z non-zero entries; then the randomized communication complexity of matrix multiplication is $\tilde{\Omega}(n\sqrt{z})$. To apply Theorem 2 we then just need to identify a partition of the k parties into two disjoint sets, one initially holding matrix A and one initially holding matrix B, immediately obtaining the following.

Theorem 4. *Let A and B be two Boolean $m \times n$ and $n \times m$ matrices distributed across k parties in such a way that no party holds entries from both matrices. The randomized r-round ϵ-error communication complexity of Boolean matrix multiplication in the synchronous message-passing model with k parties all connected to each other is $\tilde{\Omega}(n\sqrt{z}/\log rk)$ bits.*

Proof. Since, by assumption, it is possible to identify a partition of the k parties into two disjoint sets, one initially holding matrix A and one initially holding matrix B, the claim follows by applying Theorem 4 of [54] and Theorem 2. □

4.3 Problems on Graphs

In the asynchronous setting, good sources of communication lower bounds for problems on graphs are [55] and [23]. These papers present nearly-tight lower bounds on the communication complexity of a number of fundamental graph problems in the asynchronous message-passing model with coordinator [13], assuming that the input graph is given as edges initially (and adversarially) distributed, possibly with duplication, among the k parties. We shall seamlessly apply the SST to all of their results, obtaining the following.

Theorem 5. *Let G be an undirected graph with n vertices and m edges, and let its edges be initially distributed, possibly with duplication, among k parties, where $\Omega(\log n) \leq k \leq \min\{n, m\}$. Then the randomized r-round ϵ-error communication complexity of the following problems in the synchronous message-passing model where the k parties are all connected to each other satisfies the following bounds:*

- *testing cycle-freeness, testing connectivity, testing bipartiteness: $\Omega(nk/\log^2 k \log r)$ bits;*

- *testing triangle-freeness:* $\Omega(mk/\log^2 k \log r)$ *bits;*
- *diameter computation:* $\Omega(mk/\log k \log r)$ *bits;*
- *α-approximate maximum matching:* $\Omega(\alpha^2 nk/\log k \log r)$ *bits.*

Proof. These bounds follow by applying Corollaries 5, 7, 9, 11, Theorem 5 of [55], Theorem 1 of [23], respectively, and Theorem 2, along with the fact that the message-passing model can be simulated by the variant with coordinator with an overhead of $O(\log k)$ bits [16]. □

Since all the communication lower bounds in [55, 23] have matching (up to polylogarithmic factors) upper bounds in the asynchronous message-passing model, and since an upper bound which applies to the asynchronous model applies also to the synchronous model, it follows that when r is polynomial in the size of the input all the communication lower bounds in Theorem 5 are tight up to polylogarithmic factors.

We stress that the bounds in Theorem 5 hold under the edge-partitioning assumption, that is, when the input graph is encoded in edges which are initially distributed among the parties; instead, with the vertex-partitioning assumption, whereby each vertex of the graph is initially placed at one party along with all its incident edges, some problems such as connectivity can be solved with only $\tilde{O}(k+n)$ bits of communication, even in the asynchronous setting [55, Remark 3]. We conclude this section by presenting a communication lower bound for a graph problem under the vertex-partitioning assumption.

Graph Diameter, with Vertex-Partitioning. We consider the problem of determining the diameter of an n-vertex graph which initially is distributed among the k parties with respect to its vertices, that is, each party initially stores a subset of the vertices together with all their incident edges. Working in the regime $n = k$ where at each party corresponds exactly one vertex, Frischknecht et al. [19] show a reduction for this problem from the 2-party set disjointness problem of size $\Theta(n^2)$ bits. This implies that the communication complexity of this problem is $\Omega(n^2)$ bits in the asynchronous setting. Our SST immediately gives the following result.

Theorem 6. *The randomized r-round ϵ-error communication complexity of computing the diameter of an n-vertex graph in the synchronous message-passing model with n parties all connected to each other, with vertex-partitioning, is $\Omega(n^2/\log rn)$ bits.*

5 Unconditional Message Lower Bounds for Synchronous Distributed Computations

All the lower bounds in Section 4 become vanishing as r increases. Hence it is natural to ask whether there are problems that can be solved by exchanging, say, only a constant number of bits when a very large number of rounds is used. In this section we discuss problems for which this cannot happen. Specifically, we show that $\tilde{\Omega}(k)$ bits is a lower bound for several important problems in a synchronous complete network of k nodes, assuming that initially each node knows its own identifier but not the identifiers of its neighbors (this is usually referred to as the *clean network model*, or KT_0 assumption [46]). This lower bound is unconditional, that is, it holds irrespective of the number of rounds r of the computation.

When the communication mode is asynchronous, for an algorithm to meaningfully compute a function on k inputs, its communication complexity must be $\Omega(k)$ bits, for otherwise not all the input elements would be examined. This argument does not extend to the case of synchronous communication, where players can convey information by *not* sending any bits in a given round. In the synchronous case, the key idea to establish an $\Omega(k)$ lower bound is to argue that otherwise, for every input instance, there is always a constant fraction of the nodes that remain silent during the computation *independently of their own input*. Hence these nodes cannot convey any information about their own inputs to the rest of the network.

5.1 Selection

In the distributed selection problem the input is a set of $n \geq k$ elements, drawn from a totally ordered domain and initially distributed among the k nodes of a network, and an integer $1 \leq i \leq n$. The goal is to return the i-th smallest element in the set.

If the communication mode is asynchronous, it is straightforward to argue that $\Omega(k)$ bits must be exchanged. Frederickson [17, Theorem 6] presented an adversarial argument to derive a tight $\Omega(k \log(2n/k))$ message lower bound for deterministic algorithms. However, these arguments do not extend to the case of synchronous communication.

To prove a lower bound for randomized ϵ-error algorithms, we use Yao's Lemma [56]. We first prove a bound for deterministic algorithms, showing that any algorithm that does not exchange enough bits fails with probability at least ϵ over inputs chosen randomly from a distribution μ. This implies, by Yao's Lemma,[5] that any randomized algorithm with roughly the same communication bound fails on some input in the support of μ with probability at least $\epsilon/2$ over the algorithm's random choices.

We now define the distribution of the inputs. All the possible input elements are integer numbers. Let x be an integer, and call an integer *big* if it is greater than x, and *small* otherwise. Consider the input distribution μ that chooses index $1 \leq i \leq k$ uniformly at random, and that chooses integer x uniformly at random assigning, one to each node,

- k random big integers with probability $1/3$,
- k random small integers with probability $1/3$,
- $k/2$ random big integers and $k/2$ random small integers with probability $1/3$.

Lemma 1. *The expected deterministic* $1/24$*-error communication complexity of selection in the synchronous message-passing model for input distribution* μ *is at least* $k/48$ *bits.*

Proof. Let A be any deterministic algorithm for the distributed selection problem whose expected communication complexity on input distribution μ is less than $k/48$ bits. Hence the expected communication complexity of algorithm A on input integers that are half big and half small is less than $k/16$ bits. In turn, this implies that there are at least one sixth of all possible inputs from distribution μ which consist of half big integers and half small integers, and for which the communication complexity of algorithm A is less than $k/8$ bits. Therefore, with probability $1/6$ there are at least $k/4$ nodes that initially hold both big and small integers and that do not send or receive any bit.

Let X be a set of $k/4$ of such nodes, and let Y be the set of the remaining $3k/4$ nodes of the network. Let $U \subset Y$ be a set of $k/4$ nodes in Y initially holding both big and small integers. By construction, no node in X sends any bit to any node in Y, and vice-versa. This means that in this case the k nodes of the network can be divided in two sets, X and Y, both of which contain $k/4$ nodes that do not send any bit to nodes in the other set and that initially hold both big and small integers. Hence nodes in X or U do not send anything to nodes in the other set independently of their input elements, and thus, because of the KT_0 assumption, their

[5]Yao's Lemma for randomized Monte Carlo algorithms can also be found in [37, Proposition 2.6].

silence cannot convey any information. Thus, in this case, algorithm A errs with probability at least $1/4$. We conclude that algorithm A errs with probability at least $1/24$ on distribution μ. Thus, in the synchronous message-passing model, the expected communication complexity of any deterministic algorithm for the selection problem that errs with probability at most $1/24$ on input distribution μ is at least $k/48$ bits. $\square$

We can now apply Yao's Lemma to the distribution μ to obtain a lower bound for randomized Monte Carlo algorithms.

Theorem 7. *The randomized $1/48$-error communication complexity of selection in the synchronous message-passing model is at least $k/96$ bits.*

Proof. The claim follows from Lemma 1 and Yao's Lemma [56, Theorem 3]. $\square$

5.2 Graph Problems, with Vertex-Partitioning

To show unconditional lower bounds for graph problems in the vertex-partitioning model, we use a reduction from a novel multiparty problem, called *input-distributed disjointness* (ID-DISJ) and defined as follows.

Definition 1. *Given k parties, each holding one input bit, partitioned in two distinct subsets $S_1 = \{1, 2, \ldots, k/2\}$ and $S_2 = \{k/2+1, k/2+2, \ldots, k\}$ of $k/2$ parties each, the* input-distributed disjointness *function* ID-DISJ(k) *is* 0 *if there exists some index $i \in \{1, 2, \ldots, k/2\}$ such that both the input bits held by parties i and $i + k/2$ are* 1*, and* 1 *otherwise.*

Notice that this problem is, loosely speaking, "in between" the classical 2-party set disjointness and the k-party set disjointness: as in the latter, there are k distinct parties, and as in the former, the input can be seen as two vectors of $(k/2)$ bits.

We now prove that the communication complexity of ID-DISJ(k) in the synchronous message-passing model is $\Omega(k)$. As before in this section, we will argue that the expected cost of any deterministic algorithm over an adversarially chosen distribution μ of inputs is $\Omega(k)$, and then apply Yao's Lemma. The argument is similar to the one used for the selection problem.

Theorem 8. *The randomized $1/16$-error communication complexity of* ID-DISJ(k) *in the synchronous message-passing model is at least $k/32$ bits.*

Proof. Given any initial partition of the k parties between S_1 and S_2, the values of the k bits in S_1 and S_2 (that is, the two $k/2$-bit vectors associated with S_1 and S_2) are fixed using the following input distribution μ, which was first used in [49]. With probability $1/4$, the two vectors are chosen uniformly at random subject to (i) both have exactly $k/4$ 1's, and (ii) there is exactly one index $i \in \{1, 2, \ldots, k/2\}$ such that both party i and party $i + k/2$ hold a 1 (hence ID-DISJ = 0); and with probability $3/4$, the two vectors are chosen uniformly at random subject to (i) both have exactly $k/4$ 1's, and (ii) there is no index $i \in \{1, 2, \ldots, k/2\}$ such that both party i and party $i + k/2$ hold a 1 (hence ID-DISJ = 1).

Let A be any deterministic algorithm for ID-DISJ(k) whose expected communication complexity on input distribution μ is less than $k/16$ bits. This means that there are at least one half of all possible inputs from distribution μ for which the communication complexity of algorithm A is less than $k/8$ bits. Therefore, since under input distribution μ half of the input bits are 1's and half are 0's, with probability $1/2$ there are at least $k/4$ nodes that do not send or receive any bit and that initially hold half 1's and half 0's.

Let X be a set of $k/4$ of such nodes, and let Y be the set of the remaining $3k/4$ nodes of the network. Let $U \subset Y$ be a set of $k/4$ nodes in Y initially holding half 1's and half 0's. By construction, no node in X

sends any bit to any node in Y, and vice-versa. This means that the k nodes of the network can be divided in two sets, X and Y, both of which contain $k/4$ nodes that do not send any bit to nodes in the other set and that initially hold half 1's and half 0's. Hence nodes in X or U do not send anything to nodes in the other set independently of their input bits, and thus, because of the KT_0 assumption, their silence cannot convey any information. Therefore, in both cases under μ, algorithm A errs with probability at least $1/4$. We conclude that algorithm A errs with probability at least $1/8$ on distribution μ. Thus, in the synchronous message-passing model, the expected communication complexity of any deterministic algorithm for ID-DISJ(k) that errs with probability at most $1/8$ on input distribution μ is at least $k/16$ bits.

The claim then follows by applying Yao's Lemma [56, Theorem 3] to the distribution μ. □

In the rest of this section we assume $k = n$, that is each party is assigned exactly one vertex, along with all its incident edges.

5.2.1 Connectivity

We now leverage Theorem 8 to show an $\Omega(n)$ lower bound for graph connectivity, and thus for all the graph problems that can be reduced from it.

Theorem 9. *The randomized ϵ-error communication complexity of graph connectivity in the synchronous message-passing model, with vertex-partitioning, is $\Omega(n)$ bits.*

Proof. We reduce the connectivity problem from ID-DISJ, as follows.[6] Given a generic instance of ID-DISJ($n/2$), denoted $S_1 = \{s_1, s_2, \ldots, s_{n/4}\}$ and $S_2 = \{t_1, t_2, \ldots, t_{n/4}\}$, with party s_i holding bit x_i and party t_i holding bit y_i, we shall define the n-vertex graph G as shown in Figure 1, where each vertex corresponds to a distinct party, and parties s_i's and t_i's hold the bit associated with the instance of ID-DISJ($n/2$). For any $i \in [n/4]$ there is an edge between s_i and u_i (resp., between t_i and v_i) if and only if $x_i = 0$ (resp., $y_i = 0$). Additionally, there is an edge between $s_{n/4}$ and $t_{n/4}$, and for every $i \in [n/4]$ there is an edge between u_i and v_i.

The key property of G is that it is connected if and only if ID-DISJ($n/2$) = 1. Parties s_i's and t_i's simulate the distributed algorithm for their respective nodes. Then the claim follows from Theorem 8. □

5.2.2 Diameter

Here we consider the problem of determining the diameter of an unweighted graph in the synchronous message-passing model, where the vertices of the graph are initially partitioned across the parties. We show that determining a $(10/9 - \delta)$-approximation of the diameter requires $\Omega(n/\log n)$ bits of communication.

Theorem 10. *For any constant $\delta > 0$, the randomized ϵ-error communication complexity of computing a $(10/9 - \delta)$-approximation of the diameter in the synchronous message-passing model, with vertex-partitioning, is $\Omega(n/\log n)$ bits.*

Proof. We shall reduce the task of computing the diameter of a given graph from ID-DISJ, as follows. Assume, without loss of generality, that $n/4$ is a power of two. Given a generic instance of ID-DISJ($n/2$), denoted $S_1 = \{s_1, s_2, \ldots, s_{n/4}\}$ and $S_2 = \{t_1, t_2, \ldots, t_{n/4}\}$, with party s_i holding bit x_i and party t_i holding bit y_i, we shall define a graph similar to the one in Figure 1, with the following changes. All the edges connecting the s_i's nodes with each other and the t_i's nodes with each other are removed. A complete

[6] This reduction is inspired by the reduction from 2-party set disjointness to connectivity in [25].

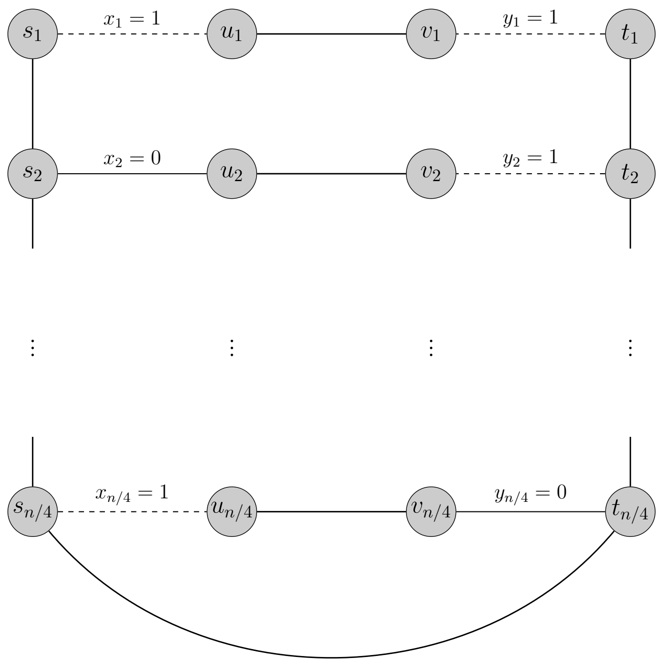

Figure 1: The reduction used in the proof of Theorem 9. Dashed edges represent missing edges.

binary tree with the s_i's as leaf nodes is added. Observe that the height of this binary tree is $\ell = \log_2(n/4)$, and thus any node s_i is always reachable from any other node s_j through a path of length at most 2ℓ. The same is done on the t side of the graph. If $x_i = 0$ then an edge connecting s_i to u_i is added; similarly if $y_i = 0$ on the t side. Conversely, if $x_i = 1$ then a path of length $5\ell - 1$ that connects s_i to u_i is added; similarly if $y_i = 1$ on the t side. The resulting graph is depicted in Figure 2. This graph has $\Omega(n)$ and $O(n \log n)$ vertices.

The key property of this graph is that its diameter is at most $(4\ell + 3 + 5\ell - 1)/2$ if ID-DISJ($n/2$) = 1, where $4\ell + 3 + 5\ell - 1$ is the maximum length of a shortest path from one vertex to itself in this case, and at least 5ℓ otherwise. Parties s_i's and t_i's simulate the distributed algorithm for their respective nodes. Then the claim follows from Theorem 8 by observing that, for any constant $\delta > 0$, $(9/10 + \delta)5\ell > 4.5\ell + 1$. □

6 Conclusions

In this paper we have presented a uniform approach to derive lower bounds on the number of messages exchanged in synchronous distributed computations. The most interesting avenue for further research is to explore the possibility of deriving tight message lower bounds in the synchronous message-passing model where the underlying topology is *arbitrary*. In fact, although the number of messages required by an algorithm in a general network is no better than the number of messages required by the same computation on a network with complete interconnection (a fact that makes a message lower bound for complete networks also valid in arbitrary networks), lower bounds for complete networks might not be tight in other topologies (see, e.g., [9]).

It is easy to show that in an arbitrary network any problem can be solved by exchanging only $O(m)$

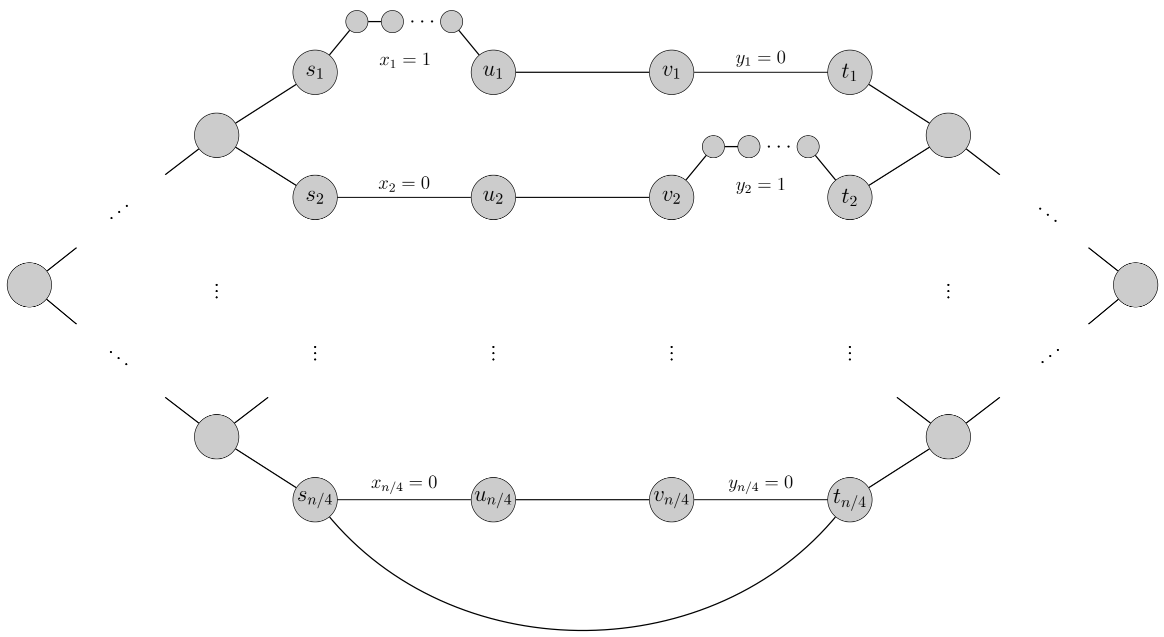

Figure 2: The reduction used in the proof of Theorem 10.

messages, where m is the number of edges of the network, by building a spanning tree and then sending all the information to the root of the tree, through its edges, by using time encoding. However, this can take time at least exponential in the size of the input. It would be interesting to explore such message-time tradeoffs, at least for some specific networks or classes of networks of practical interest such as core-periphery networks [3].

References

[1] A. Abboud, K. Censor-Hillel, and S. Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Proceedings of the 30th International Symposium on Distributed Computing (DISC)*, pages 29–42, 2016.

[2] Y. Afek and E. Gafni. Time and message bounds for election in synchronous and asynchronous complete networks. *SIAM J. Comput.*, 20(2):376–394, 1991.

[3] C. Avin, M. Borokhovich, Z. Lotker, and D. Peleg. Distributed computing on core-periphery networks: Axiom-based design. *J. Parallel Distrib. Comput.*, 99:51–67, 2017.

[4] B. Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[5] B. Awerbuch, O. Goldreich, D. Peleg, and R. Vainish. A trade-off between information and communication in broadcast protocols. *J. ACM*, 37(2):238–256, 1990.

[6] B. Awerbuch and D. Peleg. Network synchronization with polylogarithmic overhead. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 514–522, 1990.

[7] M. Braverman, F. Ellen, R. Oshman, T. Pitassi, and V. Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 668–677, 2013.

[8] K. Censor-Hillel, T. Kavitha, A. Paz, and A. Yehudayoff. Distributed construction of purely additive spanners. In *Proceedings of the 30th International Symposium on Distributed Computing (DISC)*, pages 129–142, 2016.

[9] A. Chattopadhyay, J. Radhakrishnan, and A. Rudra. Topology matters in communication. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 631–640, 2014.

[10] S. A. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, 1986.

[11] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

[12] A. K. Dhulipala, C. Fragouli, and A. Orlitsky. Silence-based communication. *IEEE Transactions on Information Theory*, 56(1):350–366, 2010.

[13] D. Dolev and T. Feder. Determinism vs. nondeterminism in multiparty communication complexity. *SIAM J. Comput.*, 21(5):889–895, 1992.

[14] A. Drucker, F. Kuhn, and R. Oshman. On the power of the congested clique model. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 367–376, 2014.

[15] M. Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM J. Comput.*, 36(2):433–456, 2006.

[16] F. Ellen, R. Oshman, T. Pitassi, and V. Vaikuntanathan. Brief announcement: Private channel models in multi-party communication complexity. In *Proceedings of the 27th International Symposium on Distributed Computing (DISC)*, pages 575–576, 2013.

[17] G. N. Frederickson. Distributed algorithms for selection in sets. *J. Comput. Syst. Sci.*, 37(3):337–348, 1988.

[18] G. N. Frederickson and N. A. Lynch. Electing a leader in a synchronous ring. *J. ACM*, 34(1):98–115, 1987.

[19] S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1150–1162, 2012.

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

[21] M. Ghaffari and F. Kuhn. Distributed minimum cut approximation. In *Proceedings of the 27th International Symposium on Distributed Computing (DISC)*, pages 1–15, 2013.

[22] J. W. Hegeman, G. Pandurangan, S. V. Pemmaraju, V. B. Sardeshmukh, and M. 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)*, pages 91–100, 2015.

[23] Z. Huang, B. Radunovic, M. Vojnovic, and Q. Zhang. Communication complexity of approximate matching in distributed graphs. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 460–473, 2015.

[24] R. Impagliazzo and R. Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC)*, pages 259–269, 2010.

[25] H. Klauck, D. Nanongkai, G. Pandurangan, and P. Robinson. Distributed computation of large-scale graph problems. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 391–410, 2015.

[26] L. Kor, A. Korman, and D. Peleg. Tight bounds for distributed minimum-weight spanning tree verification. *Theory Comput. Syst.*, 53(2):318–340, 2013.

[27] E. Korach, S. Moran, and S. Zaks. The optimality of distributive constructions of minimum weight and degree restricted spanning trees in a complete network of processors. *SIAM J. Comput.*, 16(2):231–236, 1987.

[28] E. Korach, S. Moran, and S. Zaks. Optimal lower bounds for some distributed algorithms for a complete network of processors. *Theor. Comput. Sci.*, 64(1):125–132, 1989.

[29] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[30] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. On the complexity of universal leader election. *J. ACM*, 62(1):7:1–7:27, 2015.

[31] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. Sublinear bounds for randomized leader election. *Theor. Comput. Sci.*, 561:134–143, 2015.

[32] C. Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 42–50, 2013.

[33] B. Liskov. Practical uses of synchronized clocks in distributed systems. *Distrib. Comput.*, 6(4):211–219, 1993.

[34] Z. Lotker, B. Patt-Shamir, E. Pavlov, and D. Peleg. Minimum-weight spanning tree construction in $O(\log \log n)$ communication rounds. *SIAM J. Comput.*, 35(1):120–131, 2005.

[35] M. C. Loui. The complexity of sorting on distributed systems. *Information and Control*, 60(1-3):70–85, 1984.

[36] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.

[37] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[38] D. Nanongkai, A. D. Sarma, and G. Pandurangan. A tight unconditional lower bound on distributed randomwalk computation. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 257–266, 2011.

[39] R. Oshman. Communication complexity lower bounds in distributed message-passing. In *Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 14–17, 2014.

[40] S. Pai, G. Pandurangan, S. V. Pemmaraju, T. Riaz, and P. Robinson. Symmetry breaking in the congest model: time- and message-efficient algorithms for ruling sets. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC)*, pages 38:1–38:16, 2017.

[41] G. Pandurangan, D. Peleg, and M. Scquizzato. Message lower bounds via efficient network synchronization. In *Proceedings of the 23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 75–91, 2016.

[42] G. Pandurangan, P. Robinson, and M. Scquizzato. Fast distributed algorithms for connectivity and MST in large graphs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 429–438, 2016.

[43] G. Pandurangan, P. Robinson, and M. Scquizzato. A time- and message-optimal distributed algorithm for minimum spanning trees. In *Proceedings of the 49th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 743–756, 2017.

[44] G. Pandurangan, P. Robinson, and M. Scquizzato. Fast distributed algorithms for connectivity and MST in large graphs. *ACM Trans. Parallel Comput.*, 5(1), 2018.

[45] G. Pandurangan, P. Robinson, and M. Scquizzato. On the distributed complexity of large-scale graph computations. In *Proceedings of the 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 405–414, 2018.

[46] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000.

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

[48] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.

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

[50] N. Santoro. *Design and Analysis of Distributed Algorithms*. Wiley-Interscience, 2006.

[51] J. Schneider and R. Wattenhofer. Trading bit, message, and time complexity of distributed algorithms. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC)*, pages 51–65, 2011.

[52] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2nd edition, 2001.

[53] P. Tiwari. Lower bounds on communication complexity in distributed computer networks. *J. ACM*, 34(4):921–938, 1987.

[54] D. Van Gucht, R. Williams, D. P. Woodruff, and Q. Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS)*, pages 199–212, 2015.

[55] D. P. Woodruff and Q. Zhang. When distributed computation is communication expensive. *Distrib. Comput.*, 30(5):309–323, 2017.

[56] A. C. Yao. Probabilistic computations: toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[57] A. C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

[58] S. Zaks. Optimal distributed algorithms for sorting and ranking. *IEEE Trans. Computers*, 34(4):376–379, 1985.