Message Lower Bounds via Efficient Network Synchronization

Gopal Pandurangan[1,⋆], David Peleg[2], and Michele Scquizzato[1]

[1] University of Houston, Houston, USA
[2] The Weizmann Institute of Science, Rehovot, Israel

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 in the CONGEST model. The σ synchronizer is particularly efficient in simulating synchronous algorithms that employ silence. 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 network), it fails to provide meaningful bounds when a very large number of rounds is allowed. 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, via direct reductions from multi-party 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 many graph problems. All these lower bounds hold for any distributed algorithms, including randomized Monte Carlo algorithms.

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

⋆ Supported, in part, by NSF grants CCF-1527867 and CCF-1540512.

complexity measures used to evaluate the performance of algorithms in distributed computing [28]. Even when time complexity is the primary consideration, message complexity is 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 [12], 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 for various problems has been a major focus in the theory of distributed computing for decades (see, e.g., [23, 31, 28, 33]). Tight message lower bounds for several fundamental problems such as leader election [19, 20], broadcast [3, 19], spanning tree [16, 23, 33, 19], minimum spanning tree [17, 33, 23, 19, 12, 27], and graph connectivity [12], 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. Many of the message lower bounds mentioned above (e.g., [16, 17, 3, 19, 12]) 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* [18] as a uniform tool to derive message lower bounds for a variety of problems in the synchronous setting.

Communication complexity, originally introduced by Yao [37], is a subfield of complexity theory with numerous applications in several, and very different, branches of computer science (see, e.g., [18] 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., [32]). If endowed 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 [12], k 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 ID) to whom they each send the bit that encodes their 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 many intriguing questions: (1) If one allows only a small number of rounds (e.g., polynomial in n) can such a low message complexity be achieved? (2) 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.

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 [28]) which resulted in bounding the complexities T_A and C_A of the asynchronous algorithm A for every input instance $\mathcal{I}$ as

$$\begin{aligned} T_A(\mathcal{I}) &\le T_{\text{init}}(\nu) + \Psi_T(\nu) \cdot T_S(\mathcal{I}), \\ C_A(\mathcal{I}) &\le C_{\text{init}}(\nu) + C_S(\mathcal{I}) + \Psi_C(\nu) \cdot T_S(\mathcal{I}), \end{aligned}$$

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 [2], β [2], γ [2], and δ [30] (see also [28]), 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 [4] 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*, and uses time-coding for conveying information while transmitting fewer messages (e.g., see [12, 13]). Such an algorithm, typically having low communication complexity but high time complexity, translates into an asynchronous algorithm with high time and communication complexities. Hence, 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 at most polylogarithmic in the number of rounds T_S 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 [5], 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,[3] 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, let us denote by T_S^c the number of *communicative* (or *active*) synchronous rounds, in which at least one node of the network sends a message. Analogously, let T_S^q denote the number of *quiet* (or *inactive*) synchronous rounds, in which all processors are silent. Clearly, $T_S = T_S^c + T_S^q$. Synchronizer σ compresses the execution time of the simulation by essentially 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. A somewhat surprising consequence of this feature is that synchronizer σ may in certain situations result in a simulation algorithm whose execution time is *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, we observe that 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.)

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

Synchronous Simulation Theorem. As a by-product of synchronizer σ, we show a general theorem, the *Synchronous Simulation Theorem (SST)*, which shows how message lower bounds for synchronous distributed computations can be derived by leveraging communication complexity results obtained in the asynchronous

[3] We use σ as it is the first letter in the Greek word σιωπή which means "silence".

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

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. (Note that on a complete network, synchronizers γ [2] and δ [30] are out-performed by β, hence their complexities are omitted from this table.)

model. More precisely, the SST provides a tradeoff between the message complexity of the synchronous computation and the maximum number of synchronous rounds T_S allowed to the computation. This tradeoff reveals that message lower bounds in the synchronous model are worse by at most logarithmic factors (in T_S and k, where T_S is the number of rounds taken in the synchronous model, and k is the network size) compared to the corresponding lower bounds in the asynchronous model.

Applications: Message-Time Tradeoffs. In Section 4 we apply the SST to obtain message-time tradeoffs for several fundamental problems. These lower bounds assume that the underlying communication network is complete, which is the case in many computational models [22, 14]; however, the same lower bounds clearly apply also when the network topology is arbitrary. The corresponding bounds on communication complexity are tight up to polylogarithmic factors (in the input size n of the problem and network size k) when the number of rounds is at most *polynomial* in the input size. This is because a naive algorithm that sends all the bits to a leader via the time encoding approach of [13, Theorem 3.1] is optimal up to polylogarithmic factors. We next summarize our lower bound results for various problems for a precise statement of these results. All the lower bounds in this paper hold even for randomized protocols that can return the wrong answer with a small constant probability.

Our lower bounds assume that the underlying topology is complete and the input is partitioned (in an adversarial way) among the k nodes. We assume that at most T_S rounds of synchronous computation are allowed. (We will interchangeably denote the number of rounds with T_S and r.) For sorting, where each of the k nodes have $n \geq 1$ input numbers, we show a message lower bound[4] of $\tilde{\Omega}(nk/\log r)$. This result immediately implies that Lenzen's $O(1)$-round sorting algorithm for the Congested Clique model [21] has also optimal (to within log factors) message complexity. For the Boolean matrix multiplication of two Boolean $n \times n$ matrices we show a lower bound of $\Omega(n^2/\log rk)$. For graph problems, there is an important distinction that influences the lower bounds: whether the input graph is initially partitioned across the nodes in an *edge-*

[4] Throughout this paper, the notation $\tilde{\Omega}$ hides polylogarithmic factors in k and n, i.e., $\tilde{\Omega}(f(n, k))$ denotes $\Omega(f(n, k)/(\text{polylog}\, n\, \text{polylog}\, k))$.

partitioning fashion or in *vertex-partitioning* fashion. In the former, the edges of the graph are arbitrarily distributed across the k parties, while in the latter each vertex of the graph is initially held by one party, together with the set of its incident edges. In the edge-partitioning setting, using the results of [36] in conjunction with the SST yields non-trivial lower bounds of $\tilde{\Omega}(kn/\log rk)$, where n is the number of vertices of the input graph, for several graph problems such as graph connectivity, testing cycle-freeness, and testing bipartiteness. For testing triangle-freeness and diameter the respective bounds are $\tilde{\Omega}(km)$ and $\tilde{\Omega}(m)$. (In the vertex-partitioning setting, on the other hand, many graph problems such as graph connectivity can be solved with $O(n \operatorname{polylog} n)$ message complexity [36].)

Unconditional Lower Bounds. While the SST gives essentially tight lower bounds up to very large values of T_S (e.g., polynomial in n), they become trivial for larger values of T_S (in particular, when T_S is exponential in n). 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. These lower bounds are established via direct reductions from *multi-party* communication complexity. They are of the form $\tilde{\Omega}(k)$, and this is almost tight since every problem can be solved with $O(k)$ bits of communication by letting each party encode its input in just one bit via time encoding. We point out that the unconditional lower bounds cannot be shown by reductions from 2-party case, as typically done for many reductions for these problems. A case in point are the reductions to establish the lower bounds for connectivity and diameter in the vertex-partitioning model. To show unconditional lower bounds for connectivity and diameter we define a new multi-party problem called *input-distributed disjointness (ID-DISJ)* (see Section 5) and establish a lower bound for it. We note that, unlike in the asynchronous setting, reduction from a 2-party setting will not yield the desired lower bound of $\Omega(k)$ in the synchronous setting (since 2-party problems can be solved trivially, exploiting clocks, using only one bit, as observed earlier).

1.2 Further Related Work

The first paper that showed how to leverage lower bounds on communication complexity in a synchronous distributed setting is [29], which proves a near-tight lower bound on the time complexity of distributed minimum spanning tree construction in the CONGEST model [28]. Elkin [8] extended this result to approximation algorithms. The same technique was then used to prove a tight lower bound for minimum spanning tree verification [15]. Later, Das Sarma et al. [5] explored the connection between the theory of communication complexity and distributed computing further by presenting time lower bounds for a long list of problems, including inapproximability results. For further work on time lower bounds via communication complexity see, e.g., [24, 10, 7, 14, 26], as well as [25] and references therein.

Researchers also investigated how to leverage results in communication complexity to establish lower bounds on the message complexity of distributed computations. Tiwari [34] shows communication complexity lower bounds for deterministic computations over networks with some specific topologies. Woodruff and Zhang [36] 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 problem from a new meta-problem whose communication complexity is established. However, the models considered in these two papers are inherently asynchronous, hence their lower bounds do not hold if a common clock is additionally assumed.

Hegeman et al. [12] study the message complexity of connectivity and MST in the (synchronous) congested clique. However, their lower bounds are derived using ad hoc arguments. To the best of our knowledge, the first connection between the classical communication complexity theory and the message complexity in a synchronous setting has been established by Impagliazzo and Williams [13]. They show almost tight bounds for the case with two parties for deterministic algorithms, by efficiently simulating a synchronous protocol in an asynchronous model (like we do). Ellen et al. [9] claim a simulation result for $k \geq 2$ parties. Their results are similar to our synchronous simulation theorem. However, their simulation does not consider time, whereas ours is time-efficient as well.

2 Preliminaries: Models, Assumptions, and Notation

The *message-passing model* is one of the fundamental models in the theory of distributed computing, and many variations of it have been studied. We are given a complete network of k nodes, which can be viewed 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 I, and this portion is known only to itself and not to the other nodes. Each node can communicate directly with any other node by exchanging messages. Nodes wake up spontaneously at arbitrary times. The goal is to jointly solve some given problem Π on input instance I.

Nodes have a unique identifier of $O(\log k)$ bits. Before the computation starts, each node knows its own identifier but not the identifiers of any other node. 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., [11]). 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 [37], studies the asynchronous message-passing model.

We now formally define the complexity measure studied in this paper. Most of these definitions can be found in [18]. 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.

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 protocol/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 protocol/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 protocol 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 a 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.

We now define a second complexity measure for a distributed computation, the *time complexity.* In the synchronous mode of communication, it is defined as the (worst-case) number of synchronous rounds that it comprises. It is additionally referred to as *round complexity.* Following [5], we define the *randomized ϵ-error r-round randomized communication complexity of a problem Π* in a synchronous model to be the minimum communication complexity of any protocol that solves Π with error probability ϵ when it runs in at most r rounds. We denote this quantity with $SR_{k,\epsilon,r}(\Pi)$. A lower bound on $SR_{k,\epsilon,r}(\Pi)$ holds also for Las Vegas randomized algorithms as well as for deterministic algorithms. 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 [28, 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, while 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 ideas underlying the simulation are the exploitation of inactive nodes and inactive rounds, via the use of the concept of *tentative time*, in conjunction with the use of acknowledgments as a method to avoid congestion and thus reduce the time overhead in networks whose links have limited bandwidth. It is required that all the possible communication sequences between any two nodes of the network are self-determining, i.e., no one is a prefix of another.

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. The coordinator is determined before the actual simulation begins, and this can be done by executing a leader election algorithm for asynchronous complete networks, such as the one in [1]. (The coordinator should not be confused with the notion of coordinator in the variant of the message-passing model introduced in [6]. 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 are not allowed to communicate directly among themselves, and therefore they can communicate only with the coordinator.) After its election, the coordinator sends to each node a message `START`(1) instructing them to start the simulation of round 1.

At any given time each node v maintains a *tentative time* estimate $\mathsf{TT}(v)$, representing the next synchronous round on 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. Its starting point is when the coordinator realizes that the current phase, simulating some round $t' < t$, is completed, in the sense that all messages that were supposed to be sent and received by 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. If $t' + 1 < t$, 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 `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 $\texttt{START}(t)$ 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 acknowledgement 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 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 `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 acknowledgement directly back to x.

(8) Once a (non-active) clan member w has received such an acknowledgement, 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 acknowledgement 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 acknowledgement 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*

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

$$C_A = O\big(k \log^2 k + C_S \log T_S\big), \tag{2}$$

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 `START`(t), `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.

The $O(k \log k)$ and $O(k \log^2 k)$ additive factors in the first and second equation are, respectively, the time and message complexity of the asynchronous leader election algorithm in [1] run in the initialization phase. This algorithm exchanges a total of $O(k \log k)$ messages of size $O(\log k)$ bits each, and takes $O(k)$ time. □

3.3 Message Lower Bound for Synchronous Distributed Computations

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,*

$$SCC^{\mathcal{D}}_{k,r}(\Pi) = \Omega\left(\frac{CC^{\mathcal{D}'}_{k'}(\Pi) - k \log^2 k}{1 + \log r + \lceil (k - k')/k \rceil \log k}\right).$$

Proof. We leverage the communication complexity bound of the σ synchronizer result to prove a lower bound on $SCC^{\mathcal{D}}_{k,r}(\Pi)$, synchronous communication complexity, by relating it to $CC^{\mathcal{D}'}_{k'}(\Pi)$, the communication complexity in the asynchronous setting. More precisely, we can use the σ synchronizer to simulate any synchronous algorithm for the 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 $SCC^{\mathcal{D}}_{k,r}(\Pi)$.

Next we consider $k' < k$. In this case, we need to do a minor modification to the σ synchronizer. Since we assume that messages do not contain the ID of the receiver and of the sender, when the network carrying the simulation has fewer nodes than the network to be simulated the ID of both source and the destination of any message has to be appended to the latter. 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 the σ synchronizer is increased by a factor of $O(\lceil (k-k')/k \rceil \log k)$. This gives the claimed result. □

Clearly, a corresponding lower bound on the total number of messages follows by dividing the communication complexity by the message size B. 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 protocols given in Section 2.

4 Message-Time Tradeoffs for Synchronous Distributed Computations

We now apply the Synchronous Simulation Theorem to get lower bounds on the communication complexity of some fundamental problems in the synchronous message-passing model.

4.1 Sorting

In this section we give a lower bound to the communication complexity of comparison-based sorting algorithms. At the beginning each of the k parties holds n elements of $O(\log n)$ bits each. At the end, the i-th party must hold the $(i-1)k+1, (i-1)k+2, \ldots, i \cdot k$-th order statistics. We have the following result.

Theorem 3. *The randomized r-round ϵ-error communication complexity of sorting in the synchronous message-passing model with k parties is $\Omega(nk/\log k \log r)$.*

4.2 Matrix Multiplication

We now show a synchronous message lower bound for Boolean matrix multiplication, that is, the problem of multiplying two $n \times n$ matrices over the semiring $(\{0,1\}, \wedge, \vee)$.

In [35, Theorem 4] it is shown the following: 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 nonzeroes. Then the randomized communication complexity of matrix multiplication is $\tilde{\Omega}(\sqrt{z} \cdot n)$. To apply Theorem 2 we then just have to consider an initial partition of the $2mn$ input elements among k parties such that there exists a cut in the network that divides the elements of A from those of B. Given such a partition, we immediately obtain the following.

Theorem 4. *The randomized r-round ϵ-error communication complexity of Boolean matrix multiplication in the synchronous message-passing model with k parties is $\Omega(\sqrt{z} \cdot n/\log rk)$.*

4.3 Statistical and Graph Problems

The generality of the SSTs allows us to directly apply any previous result derived for the asynchronous message-passing model. As an example, of particular interest are the results of Woodruff and Zhang [36], who present lower bounds on the communication complexity of a number of fundamental statistical and graph problems in the (asynchronous) message-passing model with k parties, all connected to each other. We shall seamlessly apply the SST for complete networks to all of their results, obtaining the following.

Theorem 5. *The randomized r-round ϵ-error communication complexity of graph connectivity, testing cycle-freeness, testing bipartiteness, testing triangle-freeness, and diameter of graphs with n nodes in the synchronous message-passing model with k parties, where the input graph is encoded in edges (u, v) which are initially (adversarially) distributed among the parties, is $\tilde{\Omega}(nk/\log rk)$.*

5 Unconditional Message Lower Bounds for Synchronous Distributed Computations

The bounds resulting from the application of the synchronous simulation theorem of Section 3 become vanishing as r increases, independently of the problem Π at hand. Hence it is natural to ask whether there are problems that can be solved by exchanging, e.g., only a constant number of bits when a very large number of rounds is allowed. In this section we discuss problems for which this cannot happen.

Specifically, we show that $\tilde{\Omega}(k)$ bits is an unconditional lower bound for several important problems in a synchronous complete network of k nodes. The key idea to prove unconditional $\tilde{\Omega}(k)$ bounds via communication complexity is to resort to multiparty communication complexity (rather than just to classical 2-party communication complexity), where by a simple and direct information-theoretic argument (i.e., without reducing from the asynchronous setting, as in Section 3) we can show that many problems in such a setting satisfy an $\Omega(k)$-bit lower bound, no matter how many synchronous rounds are allowed.

5.1 Graph Problems in the Vertex-Partitioning Model

To show unconditional lower bounds for graph problems in the vertex-partitioning model, we use a reduction from a new multiparty problem, called *input-distributed disjointness* (ID-DISJ) defined as follows. For the rest of this section, we assume $k = n$ and thus each party is assigned one vertex (and all its incident edges).

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

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

Theorem 6. *The randomized ϵ-error communication complexity of* ID-DISJ(n) *in the synchronous message-passing model is $\Omega(n)$.*

We now leverage the preceding result to prove lower bounds on the communication complexity of graph connectivity and graph diameter.

Theorem 7. *The randomized ϵ-error communication complexity of graph connectivity in the synchronous message-passing model, with vertex-partitioning, is $\Omega(n)$.*

Theorem 8. *The randomized ϵ-error communication complexity of computing the diameter in the synchronous message-passing model, with vertex-partitioning, is $\Omega(n/\log n)$.*

References

1. 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.
2. B. Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.
3. 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.
4. 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.
5. 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.
6. D. Dolev and T. Feder. Determinism vs. nondeterminism in multiparty communication complexity. *SIAM J. Comput.*, 21(5):889–895, 1992.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. L. Kor, A. Korman, and D. Peleg. Tight bounds for distributed minimum-weight spanning tree verification. *Theory Comput. Syst.*, 53(2):318–340, 2013.
16. 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.
17. 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.

18. E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.
19. 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.
20. S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. Sublinear bounds for randomized leader election. *Theor. Comput. Sci.*, 561:134–143, 2015.
21. 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.
22. 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.
23. N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., 1996.
24. D. Nanongkai, A. D. Sarma, and G. Pandurangan. A tight unconditional lower bound on distributed randomwalk computation. In *Proceedings of the 30th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 257–266, 2011.
25. R. Oshman. Communication complexity lower bounds in distributed message-passing. In *Proceedings of the 21th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 14–17, 2014.
26. 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.
27. G. Pandurangan, P. Robinson, and M. Scquizzato. A time- and message-optimal distributed algorithm for minimum spanning trees. *CoRR*, abs/1607.06883, 2016.
28. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* Society for Industrial and Applied Mathematics, 2000.
29. 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.
30. D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.
31. N. Santoro. *Design and Analysis of Distributed Algorithms.* Wiley, 2006.
32. 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.
33. G. Tel. *Introduction to Distributed Algorithms.* Cambridge University Press, 2nd edition, 2001.
34. P. Tiwari. Lower bounds on communication complexity in distributed computer networks. *J. ACM*, 34(4):921–938, 1987.
35. 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.
36. D. P. Woodruff and Q. Zhang. When distributed computation is communication expensive. *Distrib. Comput.*, to appear.
37. A. C.-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.