Improved Tradeoffs for Leader Election

SHAY KUTTEN, Technion - Israel Institute of Technology, Israel
PETER ROBINSON, Augusta University, Georgia, USA
MING MING TAN, Augusta University, Georgia, USA
XIANBIN ZHU, City University of Hong Kong, Hong Kong SAR

We consider leader election in clique networks, where n nodes are connected by point-to-point communication links. For the *synchronous clique under simultaneous wake-up*, i.e., where all nodes start executing the algorithm in round 1, we show a tradeoff between the number of messages and the amount of time. The previous lower bound side of such a tradeoff, in the seminal paper of Afek and Gafni (1991), was shown only assuming adversarial wake-up. Interestingly, our new tradeoff also improves the previous lower bounds for a large part of the spectrum, even under simultaneous wake-up. More specifically, we show that any deterministic algorithm with a message complexity of $n f(n)$ requires $\Omega\left(\frac{\log n}{\log f(n)+1}\right)$ rounds, for $f(n) > 1$. Our result holds even if the node IDs are chosen from a relatively small set of size $\Theta(n \log n)$, as we are able to avoid using Ramsey's theorem, in contrast to many existing lower bounds for deterministic algorithms. We also give an upper bound that improves over the previously-best tradeoff achieved by the algorithm of Afek and Gafni. Our second contribution for the synchronous clique under simultaneous wake-up is to show that $\Omega(n \log n)$ is in fact a lower bound on the message complexity that holds for any deterministic algorithm with a termination time $T(n)$ (i.e., any function of n), for a sufficiently large ID space. We complement this result by giving a simple deterministic algorithm that achieves leader election in sublinear time while sending only $o(n \log n)$ messages, if the ID space is of at most linear size. We also show that Las Vegas algorithms (that never fail) require $\Theta(n)$ messages. This exhibits a gap between Las Vegas and Monte Carlo algorithms.

For the *synchronous clique under adversarial wake-up*, we show that $\Omega(n^{3/2})$ is a lower bound for 2-round algorithms. Our result is the first superlinear lower bound for *randomized* leader election algorithms in the clique. We also give a simple algorithm that matches this bound.

Finally, we turn our attention to the *asynchronous clique*: Assuming *adversarial wake-up*, we give a randomized algorithm that, for any $k \in [2, O(\log n/\log \log n)]$, achieves a message complexity of $O(n^{1+1/k})$ and an asynchronous time complexity of $k + 8$. Our algorithm achieves the first tradeoff between messages and time in the asynchronous model. For *simultaneous wake-up*, we translate the deterministic tradeoff algorithm of Afek and Gafni to the asynchronous model, thus partially answering an open problem they pose.

CCS Concepts: • **Theory of computation** → **Distributed algorithms**.

Additional Key Words and Phrases: Distributed Algorithm; Leader Election; Lower Bound

1 INTRODUCTION

We address one of the most fundamental problems in the area of distributed computing—breaking symmetry by electing a leader among the n nodes of a network. This problem was introduced by Le Lann [17] and serves as a crucial building block in numerous applications such as resource allocation, load balancing, etc. The literature on leader election is too rich to survey here even for models close to the one used here, and the study of leader election has had a significant influence on other areas including e.g., peer-to-peer networks, sensor networks, population protocols, programmable matter, etc. [3, 5, 8, 19, 23]. The main complexity measures studied have been message and time complexities as studied here. Note that for some of the other above mentioned kinds of networks (e.g., sensor networks),

saving in energy is also very important. However, saving in messages and time can also help save energy.

In this work, we focus on resolving the complexity bounds of leader election in the synchronous clique. However, let us note that clique networks have been studied intensively for their own sake. This includes a wide literature on leader election in clique networks. Some such papers are mentioned below. Clique networks have also been studied for other problems and other models of distributed computing. One reason may be the fact that they capture the network (or the application) layer of most networks, where essentially, every node is able to reach every other node.

1.1   Our Contributions

Our main contributions are new tradeoff and lower bound results that aim at resolving the message and time complexity of leader election in clique networks under various assumptions. Table 1 summarizes our main results.

1.1.1   *Results for the Synchronous Clique under Simultaneous Wake-up.*

Improved Tradeoff between Messages and Time. In Section 3.2, we focus on the setting where every node is awake at the start of round 1. We show that there is an inherent tradeoff between messages and time by proving that any deterministic leader election algorithm takes at least $\frac{\log_2 n - 1}{\log_2 f(n) + 1}$ rounds, if it sends at most $n \cdot f(n)$ messages, for $f(n) > 1$, and therefore any ℓ-round algorithm requires at least $\Omega\left(n^{1+1/(\ell-1)}\right)$ messages. We point out that, in contrast to previous lower bounds for general[1] deterministic algorithms under simultaneous wake-up, such as the celebrated message complexity lower bound of $\Omega(n \log n)$ for leader election in rings by Frederickson and Lynch [7], our result does not use Ramsey's theorem, and thus holds even for moderately large ID spaces, i.e., of size $\Theta(n \log n)$. This difference is meaningful in the CONGEST model [20], where assuming a large ID space implies a large communication complexity trivially.

In Section 3.3, we give a nearly matching upper bound via a simple modification of the deterministic algorithm by Afek and Gafni [1], obtaining an ℓ-round algorithm, for any odd $\ell = 2k - 3 \geqslant 3$ that sends $O\left(\ell \, n^{1+2/(\ell+1)}\right)$ messages.

An $\Omega(n \log n)$ Lower Bound for Time-Bounded Algorithms. In Section 3.4, we prove a lower bound of $\Omega(n \log n)$ messages that holds for any time-bounded algorithm, i.e., for algorithms with a time complexity that is a function of n. We also show that the assumption of a sufficiently large ID set (though significantly smaller than the one used in [7]) is indeed crucial for a message complexity lower bound of $\Omega(n \log n)$ to hold, by giving a deterministic algorithm that terminates in sublinear time and sends $o(n \log n)$ messages when the ID space has linear size.

An $\Omega(n)$ Lower Bound for Las Vegas Algorithms. In Section 3.5, we show that any randomized Las Vegas leader election algorithm has a message complexity of $\Omega(n)$ and argue that this bound is tight. This reveals a polynomial gap in the message complexity compared to the Monte Carlo algorithm of Kutten et al. [16] which succeeds with high probability while terminating in only 2 rounds and sending $O(\sqrt{n} \log^{3/2} n)$ messages.

---

[1]that is, not necessarily comparison-based algorithm

Result	Remarks	Time		Messages
Synchronous, Deterministic, Simultaneous Wake-up				
Lower Bound, Theorem 3.8	for any $f(n) > 1$ *	$\leqslant \frac{\log_2 n - 1}{\log f(n)+1} + 1$	$\Rightarrow$	$\geqslant n f(n)$
Lower Bound, Theorem 3.11	$T(n)$ is any function of n $	$\leqslant T(n)$	$\Rightarrow$	$\Omega(n \log n)$
Algorithm, Theorem 3.10	any odd $\ell \geqslant 3$	ℓ		$O\left(\ell \, n^{1+\frac{2}{\ell+1}}\right)$
Synchronous, Deterministic, Adversarial Wake-up				
Algorithm [1]	any $\ell \geqslant 2$	ℓ		$O\left(\ell \, n^{1+\frac{2}{\ell}}\right)$
Lower Bound [1]	for any $c \geqslant 2$	$\leqslant \frac{1}{2}\log_c n$	$\Rightarrow$	$\geqslant \frac{c-1}{2} n \log_c n$
Lower Bound [1]	$T(n)$ is any function of n	$\leqslant T(n)$	$\Rightarrow$	$\Omega(n \log n)$
Synchronous, Randomized, Simultaneous Wake-up				
Algorithm, Theorem 3.16	Las Vegas	3 (w.h.p.)		$O(n)$ (w.h.p.)
Lower Bound, Theorem 3.16	Las Vegas			$\Omega(n)$
Algorithm [16]	succeeds w.h.p.	2		$O\left(\sqrt{n}\log^{3/2}(n)\right)$
Lower Bound [16]	for small constant error prob.			$\Omega(\sqrt{n})$
Synchronous, Randomized, Adversarial Wake-up				
Algorithm, Theorem 4.1	succeeds w.p. $1 - \epsilon$	2		$O\left(n^{3/2}\log\left(\frac{1}{\epsilon}\right)\right)$ [†,††]
Lower Bound, Theorem 4.2	for any constant error prob.	$\leqslant 2$	$\Rightarrow$	$\Omega\left(n^{3/2}\right)$ [†]
Algorithm [14]	succeeds w.h.p.	9		$O(n)$
Asynchronous, Randomized				
Algorithm, Theorem 5.1	succeeds w.h.p.; $k \in [2, O\left(\frac{\log n}{\log\log n}\right)]$	$k + 8$		$O\left(n^{1+1/k}\right)$
Algorithm [14]	succeeds w.h.p.	$O(\log^2 n)$		$O(n)$

* Assuming an ID space of size at least $\Omega(n \log n)$.

$ Assuming an ID space of size at least $\Omega\left(n^{\log_2 n}(T(n))^{\log_2 n - 1}\right)$.

† In expectation.

†† With high probability, the message complexity does not exceed $O(n^{3/2}\log n)$.

Table 1. Results for Leader Election in Clique Networks. Note that the "$\Rightarrow$" symbol for the entries of Theorems 3.8, 3.11, and 4.2 indicates that a sufficiently small time complexity implies the stated bound on the message complexity.

1.1.2 *A Tight Bound for the Synchronous Clique under Adversarial Wake-up*. In Section 4, we turn our attention to the setting where the adversary wakes up an arbitrary set of nodes, while all other nodes are asleep initially, and awake only upon receiving a message. We show that any randomized 2-round algorithm that succeeds with constant probability must send $\Omega(n^{3/2})$ messages in expectation, and give a simple algorithm that tightly matches this bound. To the best of our knowledge, this is the first lower bound that holds for *randomized* leader election algorithms under adversarial wake-up.

*1.1.3   **Results for the Asynchronous Clique**.* In Section 5, we give the first algorithm that achieves a tradeoff between messages and time in the asynchronous model, again, under the assumption that some subset of nodes is awoken by the adversary. In particular, we show that, for any parameter choice of $k \in [2, O(\log n / \log \log n)]$, there is an algorithm that terminates in $k + 8$ rounds while sending $O(n^{1+1/k})$ messages.

The above tradeoff algorithm is randomized. As a second asynchronous tradeoff, in Section 5.4 we translated the original deterministic tradeoff algorithm (and tradeoff upper bounds) of Afek and Gafni [1] to the limited asynchronous model. That is, for the sake of this (deterministic) algorithm only, we assume that indeed, the delay of each message is still chosen by the adversary, however, the time complexity for this algorithm is counted from the last spontaneous wake up of a node (or, alternatively, assuming all the nodes wake up simultaneously). This partially answers an old open problem posed by Afek and Gafni.

1.2   Related Work

A result that is closely related to our tradeoff lower bound is the seminal work of Afek and Gafni [1], who showed (see Theorem 4.5 in [1]) that if a deterministic algorithm elects a leader in at most $\frac{1}{2} \log_c(n)$ rounds of the synchronous clique under adversarial wake-up, then it must send at least $\frac{c-1}{2} n \log_c n$ messages, for any $c \geqslant 2$. However, their lower bound uses an adversarially constructed execution that crucially relies on the fact that nodes do *not* wake up simultaneously. Hence, their techniques do not apply to the setting that we consider in Section 3.2, where all the nodes are assumed to start the execution *simultaneously*.

For the case of adversarial wake up, where the lower bounds of [1] apply too, their tradeoff is not directly comparable to the one obtained in in our Theorem 3.8. In more detail, consider any k-round deterministic algorithm, where $2 \leqslant k = O(\log n)$. (Note that any 1-round algorithm must trivially send $\Theta(n^2)$ messages.) Our Theorem 3.8 shows that any k-round deterministic algorithm must send $\Omega((n/2)^{1+\frac{1}{k-1}})$ messages, whereas the message complexity lower bound of [1] is $\Omega(k\, n^{1+1/2k})$. Thus, for constant time algorithms, our result improves by a polynomial factor over the previously best lower bound. On the other hand, for $k = \Theta(\log n)$, the result of [1] yields a bound that is greater by a factor of $\Theta(\log n)$.

Afek and Gafni [1] also provide an $\Omega(n \log n)$ lower bound on the message complexity that was not conditioned by the bound on the time. However, again, their proof relies heavily on the assumption that the adversary can choose which nodes to wake up and when, which is not admissible in our setting; the $\Omega(n \log n)$ lower bound we provide in Theorem 3.11 holds even when assuming simultaneous wake-up. An earlier $\Omega(n \log n)$ lower bound for *asynchronous* networks was shown by Korach, Moran, and Zaks [13]. Again, the wake-up was adversarial (since the network was asynchronous). Moreover, a communication lower bound for asynchronous networks does not translate easily to one in synchronous networks, since in synchronous networks, it is possible to convey information using the passage of time.

Afek and Gafni also pose the open problem whether a sublinear time message optimal asynchronous algorithm exists. In particular, when they translate their synchronous tradeoff algorithm into the asynchronous model, they are forced to increase its time complexity to linear. The also explain and demonstrate that "The arbitrary delay of messages (and not the absence of the clock) is the source of

the increase in the time complexity of the algorithm." We managed to translate their algorithm to the asynchronous model, preserving the very same tradeoff they obtained between time and number of messages. The translated algorithm does cope well with the arbitrary delay of messages, thus overcoming the hurdle they posed. However, for this algorithm only, we assume that all the nodes wake up simultaneously. Alternatively, the translated algorithm exhibit that tradeoff (in the asynchronous model) if one counts the time only starting from the time of the last spontaneous wake up.

Previous work on leader election also considered the trade-off between message complexity and time complexity on networks with a regular structure. Frederickson [6] gave trade-offs on special network topologies, including rings, meshes and trees.

For clique networks, it was shown by Korach, Moran, and Zaks [13] that the existential lower bound of $\Omega(m)$ (eventually established in [15]) does not hold, since they presented an algorithm with $O(n \log n)$ message complexity, even though in a clique, $m = \Theta(n^2)$. Such deterministic algorithms were also presented by others, e.g., Humblet [9]. One explanation was a graph property that complete networks have (linear traversability) as demonstrated in [12].

Afek and Matias [2] presented a randomized algorithm for the *asynchronous* model, but assuming that all the nodes wake up at the same time. (Their algorithm requires each node to take actions initially, to create a sparse graph.) On the average, the message complexity is $O(n)$ and the time complexity is $O(\log n)$. Ramanathan et al. [21] proposed a randomized *synchronous* leader election algorithm with error probability $O(1/\log^{\Omega(1)} n)$ that has $O(\log n)$ time complexity and a linear message complexity. Subsequently, Kutten et al. [16] gave a randomized *synchronous* algorithm that terminates in just 2 rounds, while using $O(\sqrt{n} \log^{3/2} n)$ messages with high probability, and elects a unique leader with high probability. Recently, Kutten et al. [14] presented a randomized *asynchronous* algorithm that uses $O(n)$ messages and $O(\log^2 n)$ time under adversarial wake up. For the same model, Singh [22] presented a lower bound for deterministic algorithms, which shows that achieving $O(n)$ messages comes at the price of having a time complexity of $\Omega(n/\log n)$.

2 PRELIMINARIES

To solve leader election, exactly one of the nodes in the network needs to be distinguished as the leader. In the *explicit* variant of leader election, every node must output the ID of the leader when it terminates, whereas in *implicit leader election* (known to be in some cases easier [16]), each node must irrevocably output a single bit indicating whether it is the leader, and the goal is that exactly one of the bits is set to 1. Since any algorithm that achieves explicit leader election also solves the implicit version, we focus on implicit leader election whenever showing a lower bound.

We consider a clique network of n nodes that communicate via synchronous message passing using point-to-point links. The computation proceeds in rounds and in each round $r \geqslant 1$, a node can send (possibly distinct) messages to every other node. When we are mainly concerned with showing lower bounds, we consider the LOCAL model [20] which means that we do not restrict the size of these messages. However, our algorithms have their claimed complexities also under the CONGEST model. Each node has a unique ID and we assume the most common model which is that of the *clean network model* [20] (sometimes called KT_0) that restricts the knowledge of each node as follows: initially, a node is aware only of its own ID, and the total number of nodes n, but it does not know the IDs of the other nodes. The nodes are connected via bidirectional links, each connecting two nodes. To address the

links, each node has a set of $n - 1$ ports, over which it receives and sends messages, respectively. The assignment of port numbers to destinations is arbitrary and may differ for different nodes. Formally, a *port mapping p* is a function that maps each pair (u, i) to some pair (v, j), i.e., $p((u, i)) = (v, j)$ means that a message sent by node u over port i is received by node v over port j. We note that port mapping is bijective, that is, if $p((u, i) = (v, j))$ then $p((v, j)) = (u, i)$. Note that neither u nor v are aware of how their ports are connected until they send/receive the first message over these ports. We say that a port i is *unused* if no message was sent or received over i and we say that a node u *opens a port i in round r* if u sends the first message over i in r. When analyzing the execution of the algorithm for a limited subset of nodes S and a restricted number rounds up to some number r, it may not be necessary for us to fully specify all port mappings, but rather only the mappings of the ports over which actual messages were sent/received by nodes in S until round r, while leaving the remaining assignments undefined. We call such a partially defined function a *partial port mapping*. If a port mapping q extends a given partial port mapping p by assigning some previously undefined ports, we say that q is *compatible* with p.

3 THE SYNCHRONOUS CLIQUE UNDER SIMULTANEOUS WAKE-UP

3.1 Communication Graphs

For deterministic algorithms, we assume that each node is equipped with a unique ID chosen from an *ID universe U*, which is a "sufficiently large" set of integers. An adversary chooses an n-subset from U, called *(valid) ID assignment*, which, together with the port mapping, fully determines the execution of any given deterministic leader election algorithm (since the network is synchronous and all the nodes wake up simultaneously). In some parts of our analysis, we are only interested in the behavior of the algorithm up to some specific round r, and we define the *round r execution prefix* to consist of the first r rounds of the execution.

We say that two executions E_1 and E_2 are *indistinguishable* for a node u up to round r if u has the same ID in both executions and u receives the exact same set of messages in each round until the start of round r. A simple consequence of indistinguishability is that node u will behave the same in both E_1 and E_2 and also output the same value in both executions, assuming r is sufficiently large. Our lower bounds in this section assume that n is a power of 2; we do not need this restrictions for our algorithms, unless stated otherwise.

We need to reason about which parts of the network have communicated with each other, which is captured by the communication graph:

Definition 3.1 (Communication Graph). Consider a round $r \geqslant 1$, an ID assignment I, and a partial port mapping p. In the round r communication graph, we have the same node set as the clique network, and there is a directed edge (u, v), if node u sent a message over a port that is connected to v in some round $r' < r$. The behavior of the algorithm is fully determined by I and p, and hence we write $\mathcal{G}_r^{I,p}$ to denote the resulting *round r communication graph*.

For brevity, we sometimes omit the superscripts I and p when they are irrelevant or clear from the context. Note that $\mathcal{G}_1$ is the empty graph that consists only of singleton nodes without any edges.

Consider some weakly connected components $C_1, \ldots, C_k \subseteq \mathcal{G}_r^{I,p}$, and let $S = \bigcup_{i=1}^{k} V(C_i)$, where $V(C_i)$ denotes the nodes in C_i. By definition, the nodes in S do not have any edges to nodes outside of S in graph $\mathcal{G}_r^{I,p}$. Therefore, the behavior of the nodes in S up to round r only depends on their $|S|$ IDs

and port mapping and is, in particular, independent of the IDs assigned to the remaining nodes. We say that S is *isolated up to round r under I and p*. For the remainder of this section, whenever we use the term "component", we mean a weakly connected component.

Definition 3.2 (Capacity of Components). We say that a *component C has capacity λ* if, for each node $u \in C$, it holds that u has no incoming and outgoing edge to at least λ nodes in C.

In other words, a capacity of λ means that each node has at least λ other nodes in C to which it has not communicated yet.

Since any deterministic algorithms must correctly work on all ID assignments and any port mapping between the node IDs, it is admissible for us to choose the mapping of the unused ports of nodes "adaptively", i.e., depending on the current state of the nodes. The proof of the following lemma is immediate:

LEMMA 3.3. *Let $C \subseteq \mathcal{G}_r^{I,p}$ be a component with capacity λ and suppose that the nodes in C send (in total) at most $t \leqslant \lambda$ messages $m_1, \ldots, m_t$ during the interval of rounds $[r, r']$, for some round $r' \geqslant r$. Then, there exists a port mapping that is compatible with p such that messages $m_1, \ldots, m_t$ are received by nodes in C.*

Definition 3.4 (Restricted Execution Prefix). Consider a set X of at most $\frac{n}{2}$ IDs and suppose that we execute the given algorithm for exactly r rounds on a set V_X of $|X|$ nodes with IDs in X that are connected according to some partial port mapping p. The resulting execution prefix will depend on the partial port mapping p, and we point out that the algorithm assumes that there are n nodes and hence each node in V_X has $n-1$ ports. We define the set $\mathsf{Exec}_r(X)$ to contain every round r execution prefix of V_X, in which the port mapping p is such that every message sent by a node $u \in X$ is received by some $u' \in X$.

Intuitively speaking, $\mathsf{Exec}_r(X)$ consists of all of the execution prefixes where the nodes with IDs in X operate "undisturbed", i.e., in isolation, from the rest of the network. Notice that in every execution prefix $E \in \mathsf{Exec}_r(X)$, every node sends messages over less than $|X|$ distinct ports by round r, since otherwise, it would have sent a message to at least one node outside X and thus $E \notin \mathsf{Exec}_r(X)$.

The next definition captures the scenario that nodes with IDs in a certain set X can terminate "on their own", i.e., without sending messages to nodes with IDs not in X. Consequently, there must exist some round r after which the components formed by these nodes in $\mathcal{G}_r$ do not add any more outgoing edges.

Definition 3.5 (Terminating and Expanding Components). Consider a set of IDs X of size at most $n/2$. We say that X *forms terminating components* if there exists a round r such that, for every execution prefix $E \in \mathsf{Exec}_r(X)$, it holds that all nodes with IDs in X have terminated in E by round r. Otherwise, we say that X *forms expanding components*.

We point out that when we say a set of IDs X forms terminating components, it is for each partial port mapping that results in an execution prefix in $\mathsf{Exec}_r(X)$. Throughout this section, all logarithms are assumed to be of base 2 unless stated otherwise.

LEMMA 3.6. *Consider any ID universe U' and any integer $\ell \leqslant \log_2 n - 1$. There exist at most $2^{\log_2 n - \ell}$ disjoint subsets $X_i \in \binom{U'}{2^\ell}$ of size 2^ℓ such that X_i form terminating components[2].*

PROOF. Fix some integer ℓ that satisfies the premise and assume towards a contradiction that there are at least $2^{\log_2 n - \ell} + 1$ disjoint subsets $X_1, \ldots X_{2^{\log_2 n - \ell} + 1}$, each of which results in terminating components (see Definition 3.5).

That is, for each i, there exists a partial port mapping p_i such that the nodes with IDs in X_i only send messages to other nodes with IDs in X_i. Consider the set of IDs $X_a = \bigcup_{i=1}^{2^{\log_2 n - \ell}} X_i$. Since each X_i has size exactly 2^ℓ, and these sets are pairwise disjoint, X_a consists of n unique IDs. Let E_a be the execution of the algorithm on X_a with the port mapping p such that, for each node u with ID in X_i and all $k \in [n-1]$, we define $p(u, k) := p_i(u, k)$. By the correctness of the algorithm, this yields a leader with some ID $a \in X_a$. Without loss of generality, assume that $a \in X_1$. Now consider the set $X_b = \bigcup_{i=2}^{2^{\log_2 n - \ell} + 1} X_i$. Since X_b is disjoint union of $2^{\log_2 n - \ell}$ sets of size 2^ℓ we know that $|X_b| = |X_a|$, and thus X_b is also a valid ID assignment for n nodes. Moreover, by combining the port mappings associated with $X_2, \ldots, X_{2^{\log_2 n - \ell} + 1}$ in a similar manner as for E_a, we obtain an execution E_b and a leader with some ID b when executing the algorithm on X_b. As we have assumed that each X_i forms terminating components, it follows that, for each u with ID in X_i ($i \in [2, \log_2 n - \ell]$), some execution prefix in $\text{Exec}_r(X_i)$ is indistinguishable from E_a for u as well as from E_b, for any round r. Thus, all the components formed by the sets $X_2, \ldots, X_{\log_2 n - \ell}$ must behave the same in both executions E_a and E_b until the nodes have terminated, which implies that $b \in X_{2^{\log_2 n - \ell} + 1}$. Now consider the execution E on $X_1 \cup X_{2^{\log_2 n - \ell} + 1} \cup \bigcup_{i=2}^{2^{\log_2 n - \ell} - 1} X_i$ with port mapping p. It follows that we obtain leaders with IDs $a \in X_1$ and $b \in X_{2^{\log_2 n - \ell} + 1}$ in E, yielding a contradiction. □

An immediate consequence of Lemma 3.6 is that a sufficiently large ID universe must contain a large subset of IDs without any terminating component:

COROLLARY 3.7. *Consider any ID universe U'. There exists a set of IDs $U \subseteq U'$ of size $|U'| - n \log_2 n$ such that, for any $\ell \leqslant \log_2 n - 1$ and any subset $X \subseteq U$ of size 2^ℓ, it holds that X does not form terminating components.*

3.2 A Lower Bound on the Communication-Time Tradeoff in the Synchronous Clique

THEOREM 3.8. *Suppose that n is a sufficiently large power of two. Consider any deterministic algorithm that elects a leader in the synchronous clique of n nodes and terminates in $T(n)$ rounds while sending at most $n \cdot f(n)$ messages, where $f(n) > 1$ is any increasing function of n. If the IDs of the nodes are chosen from a set of size at least $2n \log_2 n + n$, then it must be that $T(n) > \frac{\log_2 n - 1}{\log_2 f(n) + 1} + 1$.*

Consequently, any deterministic k-round algorithm requires at least $\Omega\left(\left(\frac{n}{2}\right)^{1 + 1/(k-1)}\right)$ messages.

In the remainder of this section, we prove Theorem 3.8. Let U' be the given ID universe. Corollary 3.7 ensures that, after removing $n \log_2 n$ IDs from U', the remaining IDs do not produce any terminating components of certain sizes (up to $n/2$).

---

[2]We will follow the standard notation from extremal combinatorics where $\binom{U'}{k}$ denotes the family of all the k-element subsets of U'.

Define $U \subseteq U'$ to be this set of the remaining IDs and observe that

$$|U| \geqslant n \log_2 n + n.$$

Note that any execution of the algorithm on an ID assignment from U results in a communication graph where the largest component contains a majority of the nodes upon the termination of the algorithm.

Assume towards contradiction that $T(n) \leqslant \frac{\log_2 n - 1}{\log_2 f(n) + 1} + 1$. The following lemma is the key technical component for proving Theorem 3.8. Intuitively speaking, it shows that by selectively excluding certain ID assignments, we can prevent components from growing too quickly.

LEMMA 3.9. *Suppose that n is a power of two. For every round $r \leqslant T(n)$, there is a set $U_r \subseteq U$ of size at least $|U| - n\,(r-1)$ such that, for every ID assignment $I \subseteq U_r$, there exists a partial port mapping p, and a decomposition of $\mathcal{G}_r^{I,p}$ into sets of nodes $X_1, \dots, X_{n/2^{\sigma_r}}$, where*

$$\sigma_r := \left(\lceil \log_2 f(n) \rceil + 1\right)(r-1) \tag{1}$$

and the following properties hold for every set X_i:

(A) *For some integer $k_I \geqslant 1$ (depending on I), there exist components $C_1, \dots, C_{k_I} \subseteq \mathcal{G}_r^{I,p}$ such that $X_i = \bigcup_{j=1}^{k_I} V(C_j)$, i.e., there are no edges between nodes in X_i and nodes not in X_i.*

(B) *X_i contains exactly 2^{σ_r} nodes.*

PROOF. We proceed by induction on r. For the basis $r = 1$, we define $U_1 := U$. Properties (A) and (B) are immediate since $\mathcal{G}_1^{I,p}$ is the empty graph that consists only of singleton nodes without any edges, for every ID assignment $I \subseteq U_1$ and port mappings p. Moreover, $\sigma_1 = 0$ and hence $2^{\sigma_1} = 1$, as required. Now suppose that the lemma holds for some $r \geqslant 1$, i.e., there exists a set U_r of size $|U| - n\,(r-1)$ such that all ID assignments from U_r satisfy the inductive hypothesis with respect to r. We define

$$\mu_{r+1} := 2^{\sigma_r}\,(2\,f(n) - 1). \tag{2}$$

Consider any ID assignment $I \subseteq U_r$. Let p be a partial port mapping and $X_1, \dots, X_{n/2^{\sigma_r}}$ be the decomposition of $\mathcal{G}_r^{I,p}$ induced by I as guaranteed by the inductive hypothesis. We say that I is *costly*, if, when executing the algorithm on I and p, there exists some $X_{I,p} \in \{X_1, \dots, X_{n/2^{\sigma_r}}\}$ such that the nodes in $X_{I,p}$ send at least μ_{r+1} messages during round $r + 1$. We call the subset $X_{I,p}$ a *costly subset* of $\mathcal{G}_r^{I,p}$. We will iteratively prune the IDs of costly subsets from U_r as follows: Let I_1 be any costly ID assignment from U_r. Then, there exists a costly subset $X_{I_1,p_1} \subset \mathcal{G}_r^{I_1,p_1}$ (where p_1 is the partial port mapping associated with I_1 according to the inductive hypothesis), which contains 2^{σ_r} nodes that send (in total) at least μ_{r+1} messages in round $r + 1$. We remove the IDs of X_{I_1,p_1} from U_r and obtain a slightly smaller ID universe denoted by $U_r^{(1)} \subseteq U_r$. Again, we check whether there exists an ID assignment $I_2 \subseteq U_r^{(1)}$ with a costly subset X_{I_2,p_2} and proceed by removing the offending set of 2^{σ_r} IDs in the decomposition induced by I_2 from $U_r^{(1)}$, and so on.

This pruning process stops after ℓ iterations, for some integer ℓ, once we can no longer find any costly ID assignment in the ID universe $U_r^{(\ell)}$. We define $U_{r+1} := U_r^{(\ell)}$.

CLAIM 1. *The pruning process stops after at most $\ell \leqslant \frac{n}{2^{\sigma_r}} - 1$ iterations, and*

$$|U_{r+1}| \geqslant |U_r| - n \geqslant |U| - n\,r.$$

PROOF OF CLAIM 1. By a slight abuse of notation, the same variable may refer to a set of nodes as well as to their IDs. Assume towards a contradiction that we remove at least $\ell := n/2^{\sigma_r}$ sets $X_{I_1,p_1}, \ldots, X_{I_\ell,p_\ell}$, each of size 2^{σ_r}, due to having identified some costly ID assignments $I_1, \ldots, I_\ell$. Recall that p_i is the partial port mapping associated with I_i according to the inductive hypothesis of Lemma 3.9.

According to the pruning process described above, we identify the next costly ID assignment I_i after having removed the set $X_{I_{i-1},p_{i-1}} \subseteq I_{i-1}$ of 2^{σ_r} IDs. It follows that the ID sets $X_{I_1,p_1}, \ldots, X_{I_\ell,p_\ell}$ are pairwise disjoint, which tells us that the set $J := \bigcup_{i=1}^{\ell} X_{I_i,p_i}$ is a valid ID assignment for n nodes. By combining the port mappings $p_1, \ldots, p_\ell$, we obtain a partial port mapping p in a natural way: for $u \in X_{I_i,p_i}$, we define $p(u, k) := p_i(u, k)$.

Let E be the round $r + 1$ execution prefix of the algorithm on J and p, and let E_{I_i} be the round $r + 1$ execution prefix with IDs in I_i and port mapping p_i. Since X_{I_i,p_i} was one of the subsets of the decomposition induced by I_i, we know by the inductive hypothesis of Lemma 3.9 that X_{I_i,p_i} results in a set of components in execution E_{I_i}, i.e., there is no communication between the nodes with IDs in X_{I_i,p_i} and the rest of the network. Consequently, $E_{I_i} \in \text{Exec}_{r+1}(X_{I_i})$ and, by construction, E and E_{I_i} are indistinguishable for all nodes in X_{I_i,p_i}, for all i. It follows that the nodes in each set X_{I_i,p_i} will also jointly send at least μ_{r+1} messages in round $r + 1$ of E. Recalling (2), this yields

$$\ell \cdot \mu_{r+1} = \frac{n}{2^{\sigma_r}} (2^{\sigma_r} (2 f(n) - 1)) = 2 n f(n) - n > n f(n)$$

messages in E, where in the last inequality, we use the assumption that $f(n) > 1$, contradicting the assumed bound on the message complexity of the algorithm. □

Next, we describe the strategy of the adversary to ensure that, for any ID set chosen from U_{r+1}, the existing components do not expand too much. For any ID assignment $I \subseteq U_{r+1} \subseteq U_r$, consider the decomposition of $G_r^{I,p}$ into $X_1, \ldots, X_{n/2^{\sigma_r}}$ sets for a partial port mapping p guaranteed by the inductive hypothesis. Since I is not a costly ID assignment, we know that at most μ_{r+1}, messages are sent by the nodes in any X_i in round $r + 1$. Any message that is sent over an already-used port by a node in X_i will reach some other node in X_i by construction, and thus we focus only on messages that are sent over previously unused ports in round $r + 1$.

Let $m_1 \leqslant \mu_{r+1}$ be the actual number of messages sent by nodes in X_1 during round $r + 1$ over m_1 unused ports. We define

$$t = 1 + \lceil \log_2 f(n) \rceil, \tag{3}$$

and connect these m_1 ports to arbitrary nodes in $\bigcup_{i=1}^{2^t} X_i$. We will argue below that there are sufficiently available ports in this set. Note that it is admissible for us to adaptively choose the port connections, as we are considering deterministic algorithms that works for all IDs assignments and, moreover, in the clean network model (see Section 3.1), a node $u \in X_1$ with an unused port q does not know to which one of the nodes q connects to until u sends or receives a message across q. In the following, we show that there are sufficiently many nodes in $\bigcup_{i=2}^{2^t} X_i$ to which the m_1 ports opened by nodes in X_1 can be

connected to. Since $|X_i| = 2^{\sigma_r}$ by the inductive hypothesis, it follows from (2) that

$$\left| \bigcup_{i=2}^{2^t} X_i \right| = \sum_{i=2}^{2^t} |X_i|$$

$$= (2^t - 1) \, 2^{\sigma_r}$$

$$\text{(by (3))} \quad \geqslant (2 f(n) - 1) \, 2^{\sigma_r}$$

$$\text{(by (2))} \quad = \mu_{r+1}. \tag{4}$$

As there are no prior connections between the nodes in X_1 and $\bigcup_{i=2}^{2^t} X_i$ in $\mathcal{G}_r^{I,p}$ by assumption, the above bound tells us that there are sufficiently many nodes in $\bigcup_{i=2}^{2^t} X_i$ to which the m_1 ports opened by nodes in X_1 can be connected to. We define $X_1' = \bigcup_{i=1}^{2^t} X_i$, and we make this set X_1' part of the decomposition of $\mathcal{G}_{r+1}^{I,p'}$ for some partial port mapping p' that is compatible with p. Analogously, we can direct the $m_j \leqslant \mu_{r+1}$ messages that are sent over previously unused ports by nodes in X_j, for $j \in [2, 2^t]$, to nodes in $\bigcup_{i=1, i \neq j}^{2^t} X_i \subseteq X_1'$.

We proceed similarly for the remaining sets: That is, we define $X_2' = \bigcup_{2^t+1}^{2 \cdot 2^t} X_i$ and connect the newly opened ports of the nodes in X_{2^t+1} to nodes in $X_{2^t+2}, \ldots, X_{2^{2t}}$, and so forth. As a result, we obtain the required decomposition of $\mathcal{G}_{r+1}^{I,p'}$ into sets $X_1', \ldots, X_{n/2^{t+\sigma_r}}'$. Note that it is possible to process all sets in this way since we assumed that n is a power of 2, and hence $n/2^t$ is an integer.

To complete the proof, we need to show that each obtained X_j' set satisfies (A) and (B). Property (A) follows readily from the assumption that each set X_i consists of components in $\mathcal{G}_r^{I,p}$ together with our strategy for connecting the newly opened ports in round $r + 1$. In more detail, this ensures that a node in X_j' does not have any edges to $\mathcal{G}_{r+1}^{I,p'} \setminus X_j'$. For Property (B), we need to show that $|X_j'|$ contain exactly $2^{\sigma_{r+1}}$ nodes: According to the inductive hypothesis, $|X_i| = 2^{\sigma_r}$ and, by the fact that the sets X_i are pairwise disjoint, for any j, we have that

$$|X_j'| = 2^{\sigma_r + t}$$

$$\text{(by (3))} \quad = 2^{\sigma_r + \lceil \log_2 f(n) \rceil + 1}$$

$$\text{(by (1))} \quad = 2^{(\lceil \log_2 f(n) \rceil + 1)(r-1) + \lceil \log_2 f(n) \rceil + 1}$$

$$= 2^{(\lceil \log_2 f(n) \rceil + 1)(r)}$$

$$\text{(by (1))} \quad = 2^{\sigma_{r+1}}.$$

This concludes the proof of Lemma 3.9. □

We now complete the proof of Theorem 3.8. Recall that the algorithm terminates by round $T(n)$ and that we assume towards contradiction that $T \leqslant \frac{\log_2 n - 1}{\log_2 f(n) + 1} + 1$. Lemma 3.9 implies that all components have size $2^{\sigma_T} \leqslant 2^{(\log_2 f(n) + 1)(T-1)} = 2^{\log_2 n - 1} \leqslant n/2$. From Corollary 3.7, we know that the algorithm cannot terminate unless one of the components has a size of more than $n/2$. Therefore, we have arrived at a contradiction.

3.3 An Improved Deterministic Algorithm

We now describe a simple algorithm that can be viewed as an optimized variant of the deterministic synchronous algorithm in [1] assuming simultaneous wake-up.

For a given integer parameter $k \geqslant 3$, the algorithm starts by executing $k - 2$ iterations, each of which consists of two rounds. Initially, every node is a *survivor*. In round 1 of iteration i, each survivor sends its ID to $\lceil n^{\frac{i}{k-1}} \rceil$ other nodes that become *referees*. Then, in round 2, each referee responds to the survivor with the highest ID and discards all other messages. A node remains a survivor for iteration $i + 1$ if and only if it received a response from every referee; otherwise it is *eliminated*. Finally, at the end of iteration $k - 2$, we perform one iteration that consists only of the first round, in which all remaining survivors send a message to all other nodes (i.e., every node becomes a referee), and a survivor terminates as leader if its own ID is greater than all other IDs that it received.

To bound the remaining survivors at the end of the i-th iteration, observe that at most $\frac{n}{n^{i/(k-1)}} = n^{1-i/(k-1)}$ of the survivors who sent out a message in this iteration could have received a response from all of their referees. Thus, it follows that there remains only a single survivor at the end of the $(k - 1)$-th iteration, who becomes leader. Moreover, each survivor of iteration i contacts $\lceil n^{(i+1)/(k-1)} \rceil$ referees in iteration $i+1$, which means that we send $O(n^{1-i/(k-1)+(i+1)/(k-1)}) = O(n^{1+1/(k-1)})$ messages per iteration, for a total message complexity of $O(k\, n^{1+1/(k-1)})$. Since every iteration except the final one require two rounds, the time complexity is $2(k-2)+1 = 2k-3$. In particular, for a time complexity of $\ell = 2k - 3$ rounds, we obtain a message complexity of

$$O\left(k\, n^{1+1/(k-1)}\right) = O\left(\ell\, n^{1+\frac{1}{(\ell+3)/2-1}}\right) = O\left(\ell\, n^{1+\frac{2}{\ell+1}}\right).$$

Thus we have shown the following:

THEOREM 3.10. *Consider the synchronous clique under simultaneous wake-up. For any odd integer $\ell \geqslant 3$, there exists a deterministic algorithm that terminates in ℓ rounds and sends $O\left(\ell\, n^{1+\frac{2}{\ell+1}}\right)$ messages.*

We point out that Theorem 3.10 improves over the algorithm of Afek and Gafni [1], which, for ℓ rounds, achieves a message complexity of $O(\ell\, n^{1+2/\ell})$. In particular, for constant-time algorithms, we obtain a polynomial improvement in the message complexity.

3.4 An Extreme Point of the Trade-off

By how much can we reduce the message complexity if we increase the time complexity? We answer this question by showing that any leader election algorithm must send at least $\Omega(n \log_2 n)$ messages, even if all the nodes wake up spontaneously at the same time. This holds for all time-bounded algorithms where the termination time $T(n)$ is a function of n.

THEOREM 3.11. *Consider any deterministic algorithm that elects a leader in the synchronous clique in $T(n)$ rounds where the number of nodes n is a sufficiently large power of two. If the IDs of the nodes are chosen from a set of size at least $n^{\log_2 n}(T(n))^{\log_2 n - 1}$, then it must send at least $\Omega(n \log n)$ messages.*

The main technical argument of our proof will focus on the restricted class of algorithms where each node sends at most one message per round, which we call *single-send algorithms*. The following lemma shows that single-send algorithms are equivalent to standard multicast algorithms in terms of their message complexity.

LEMMA 3.12. *If there exists a multicast leader election algorithm that sends $M(n)$ messages and terminates in $T(n)$ rounds, then there exists a single-send leader election algorithm that sends at most $M(n)$ messages and terminates in at most $n \cdot T(n)$ rounds.*

PROOF. We show how to simulate a given multicast algorithm $\mathcal{A}$ to obtain a new algorithm, called $\mathcal{S}$, that is a single-send algorithm. The idea behind the simulation is simple: We simulate each round $r \geqslant 1$ of the multicast algorithm $\mathcal{A}$, in an interval consisting of the sequence of rounds $(r-1)n + 1, \ldots, rn$ as follows: If, given a node u's state at the start of round r, algorithm $\mathcal{A}$ requires u to send messages $m_1, \ldots, m_k$ during round r, the new algorithm $\mathcal{S}$ sends message m_i in round $(r-1)n + i$. Since a node can send at most one message to any other node in a given round, we know that $k \leqslant n - 1$, which ensures that u sends at most one message per round in algorithm $\mathcal{S}$. Moreover, any node v that receives messages in some round $(r-1)n + j$, simply adds this message to a buffer and, at the end of round rn, it uses $\mathcal{A}$ to empty its buffer and process all messages accordingly.

By a straightforward inductive argument, it follows that every node u executing $\mathcal{S}$ processes the exact same set of messages at the end of round rn that it receives in round r when executing $\mathcal{A}$ and hence also performs the same state transitions. This implies the claimed time complexity of $\mathcal{S}$. The message complexity bound is immediate since $\mathcal{S}$ does not send any messages in addition to the ones produced by $\mathcal{A}$. □

The next lemma shows that the message complexity of leader election is high for single-send algorithms, which, together with Lemma 3.12, completes the proof of Theorem 3.11. The proof of Lemma 3.13 is technically more involved and postponed to Section 3.4.1.

LEMMA 3.13. *Consider a $T(n)$ time-bounded single-send algorithm $\mathcal{S}$ and an ID universe U of size at least $n T(n)^{\log_2 n - 1}$. There exists an ID assignment $I \subseteq U$ such that $\mathcal{S}$ sends at least $\Omega(n \log n)$ messages under I.*

3.4.1 Proof of Lemma 3.13.

LEMMA 3.14. *Consider an ID universe U' of size at least $n T(n)^{\log_2 n - 1} + n \log(n)$ and let $U \subseteq U'$ be a set of IDs of the kind guaranteed by Corollary 3.7. For every integer $i = 0, \ldots, \log_2 n - 1$, there exists a round r_i and a set family $U_i \subseteq \binom{U}{2^i}$ of 2^i-element subsets of U such that, for every set $I \in U_i$, every execution prefix E in $\mathrm{Exec}_{r_i}(I)$, there exists a partial port mapping p such that E forms a component C of 2^i nodes in $G_{r_i+1}^{\tilde{I},p}$, for all ID assignment $\tilde{I}$ that contains I, with the following properties:*

(a) C has capacity at least $\frac{|C|}{2} - 1 = 2^{i-1} - 1$ at the start of round $r_i + 1$, if $i \geqslant 1$;

(b) the nodes in C have opened $\frac{|C|}{2} = 2^{i-1}$ previously unused ports during rounds $[r_{i-1}+1, r_i]$, if $i \geqslant 1$;

(c) U_i contains at least $\frac{n}{2^i} T(n)^{\log_2 n - i - 1}$ pairwise disjoint sets.

PROOF. For the base case, we define $r_0 := 0$ and $U_0 := U$. Note that the "end of round 0" refers to the start of the (first) round 1 of the algorithm, and hence $\mathcal{G}_1$ is the empty graph consisting of n isolated vertices, each of which forms a singleton component. Thus Properties (a)-(c) hold.

For the inductive step, suppose that (a)-(c) hold for i, for some $0 \leqslant i \leqslant \log_2 n - 2$, which means that rounds $r_0, \ldots, r_i$ are already defined. Consider any $I \in U_i$ and any execution $E_I \in \text{Exec}_{r_i}(I)$. Let C be the component formed in E_I in round r_i. By Property (a), C has a capacity of at least $\frac{|C|}{2} - 1$ at the start of round $r_i + 1$, and Lemma 3.3 guarantees that it will be possible to connect the next $\frac{|C|}{2} - 1$ previously unused ports that the nodes in C open during rounds $[r_i+1, r_{i+1}]$ to other nodes within C. Conceptually, opening these ports corresponds to adding $\frac{|C|}{2} - 1$ directed edges to C in the communication graph. Since $2^i \leqslant \frac{n}{2}$, Corollary 3.7 tells us that I does not form a terminating component and hence the nodes in C cannot terminate without opening at least one port that is not connected to another node in C. It follows that there exists some earliest round r_I such that, for all $r \in [r_i + 1, r_I - 1]$, the total number of newly opened ports used by the nodes in C is some $W \leqslant \frac{|C|}{2} - 1$ and each node in some nonempty subset $S' \subseteq V(C)$ such that $|S'| \geqslant \frac{|C|}{2} - W$, opens one new port in round r_I. In other words, r_I is the earliest round in which the nodes in C have (collectively) opened at least $\frac{|C|}{2}$ ports since r_i.

Let $f : U_i \to [T(n)]$ be the mapping that assigns r_I to each 2^i-element subset $I \in U_i$, and define $g : [T(n)] \to [|U_i|]$ be the mapping such that $g(r)$ is equal to the number of 2^i-element subsets $I \in U_i$ such that $f(I) = r$. We define

$$r_{i+1} := \arg\max_{r \leqslant T(n)} \{g(r)\}.$$

Intuitively speaking, when considering all possible rounds in which the algorithm opens the $\frac{|C|}{2}$-th port since the start of round $r_i + 1$, round r_{i+1} is the one that occurs most frequently, breaking ties arbitrarily. Define $U_i' \subseteq U_i$ to be the family of all 2^i-subsets that open their $\frac{|C|}{2}$-th new port since round r_i exactly in round r_{i+1}.

We now show how to construct isolated executions that yield components of size 2^{i+1} by merging the 2^i components in U_i' in round r_{i+1} as follows: Let U_{i+1} be the resulting family of 2^{i+1}-element subsets obtained by taking the union of all possible pairs of disjoint 2^i-subsets of U_i'.

For any $I \in U_i'$, let E_I be an execution in $\text{Exec}_{r_i}(I)$, and S_I be the set of nodes that open a new port in round r_{i+1} in E_I. Now consider disjoint sets $I, I' \in U_i'$ and let C and C' be their components in E_I and $E_{I'}$, respectively. As we focus on single-send algorithms, we know that, in round r_{i+1}, at most $k \leqslant 2^i$ nodes $u_1, \ldots, u_k$ each open a new port in execution E_I and, similarly, at most $k' \leqslant 2^i$ nodes $v_1, \ldots, v_{k'} \in S_{I'}$, each open a new port in execution $E_{I'}$. Fix any one-to-one mappings $g : \{u_1, \ldots, u_k\} \to I'$ and $h : \{v_1, \ldots, v_{k'}\} \to I$. We connect the port opened by u_i to the node with ID $g(u_i)$ and, similarly, connect the port opened by v_i to $h(v_i)$, yielding the (merged) component $D = C \cup C'$ of size 2^{i+1} at the end of round r_{i+1} with IDs $I \cup I'$. As a result, we have extended the r_i-round executions E_I and $E_{I'}$ to an r_{i+1}-round execution in $\text{Exec}_{r_{i+1}}(I \cup I')$.

To see why Property (a) holds for the merged component D with IDs in $(I \cup I') \in U_{i+1}$, recall that we add at most 1 incident edge per node in the communication graph between components C and C'

in round r_{i+1} and these are the first edges interconnecting C and C'. In particular, any node $u \in C$ has at least $2^i - 1$ unused ports that can be connected to nodes in C' and the same is true for nodes in C'. This means that every node in D must have at least

$$|C| - 1 = |C'| - 1 = 2^i - 1 = \frac{|D|}{2} - 1$$

unused ports, which proves the claimed capacity bound for the merged component D.

Property (b) follows since the nodes in C as well as the ones in C' have opened $\frac{|C|}{2}$ new ports during $[r_i + 1, r_{i+1}]$ according to our port mapping described above.

Finally, to show that Property (c) holds for U_{i+1}, a counting argument reveals that

$$|U_i'| \geqslant \frac{|U_i|}{T(n)}.$$

Since U_i contains at least $\frac{n}{2^i} T(n)^{\log_2 n - i - 1}$ pairwise disjoint sets by induction, the lower bound on $|U_i'|$ shows that U_i' contains at least $\frac{n}{2^i} T(n)^{\log_2 n - i - 2}$ pairwise disjoint sets. Note that any set of four pairwise disjoint subsets X_1, X_2, X_3, X_4 in U_i' will give a pair of disjoint subsets $X_1 \cup X_2, X_3 \cup X_4$ in U_{i+1}. Hence, U_{i+1} contains at least $\frac{n}{2^{i+1}} T(n)^{\log_2 n - i - 2}$ pairwise disjoint sets of size 2^{i+1}. □

We now complete the proof of Lemma 3.13: According to Lemma 3.14, the family $U_{\log_2 n - 1}$ contains at least 2 disjoint sets I' and I'', each of size $\frac{n}{2}$. Consequently, $I^* = I' \cup I''$ is a valid ID assignment. By construction, I^* consists of the union of two sets in $U_{\log_2 n - 1}$, which in turn each consist of the union of sets in $U_{\log_2 n - 2}$ and so forth. This means that, for every round r_j ($j \in [0, \log_2 n - 1]$) guaranteed by Lemma 3.14, there exist $\frac{n}{2^j}$ sets $I_1^*, \ldots, I_{n/2^j}^*$ of size 2^j such that $I^* = \bigcup_{k=1}^{n/2^j} I_k^*$, and, for all k, any restricted execution prefix in $\text{Exec}_{r_j}(I_k^*)$ will satisfy Property (b). Now, for any restricted execution prefix $E_{I_1^*}, \ldots, E_{I_{n/2^j}^*}$ in $\text{Exec}(I_1^*), \ldots, \text{Exec}(I_{n/2^j}^*)$ respectively, consider the n-node execution E^* obtained by running the executions $E_{I_1^*}, \ldots, E_{I_{n/2^j}^*}$ in parallel up to round r_j. By construction, there will be no messages sent between the components and hence a node u with some ID $x \in I_k^*$ will behave exactly in the same way as it does in $E_{I_k^*}$, because $E_{I_k^*}$ and E^* are indistinguishable for u until round r_j. Consequently, we obtain $n/2^j$ components $C_1, \ldots, C_{n/2^j}$ at the end of round r_j, and Property (b) is guaranteed to hold for each of these components even in execution E^*. This shows that $2^{j-1} \frac{n}{2^j} = \frac{n}{2}$ new ports are opened in the interval $[r_{j-1} + 1, r_j]$, for every $j \geqslant 1$. Summing up over all $j = 1, \ldots, \log_2 n - 1$, it follows that $\Omega(n \log n)$ ports are opened in total throughout execution E^*.

3.4.2 An Algorithm with $o(n \log_2 n)$ Message Complexity and Sublinear Time for Small ID Universes.

We now show that requiring a sufficiently large ID space in Theorem 3.11 is indeed necessary, by giving a deterministic algorithm for the case where the range of IDs of the nodes is restricted to the set $\{1, \ldots, n\,g(n)\}$, where $g(n) \geqslant 1$ is any integer-valued function of n. In the trivial case where the IDs are from 1 to n, then the algorithm that chooses 1 to be the ID of the leader is optimal. In the following algorithm, we generalize this observation. We use $id(u)$ to denote the ID of a node u. Parameter $d \leqslant n$ controls the trade-off between the time and message complexity.

Algorithm 1 Deterministic Algorithm for Small ID Universes

1: **for** each $u \in V$ execute the following steps in parallel **do**
2:      **for** round $i = 1$ to $i = \lceil \frac{n}{d} \rceil$ **do**
3:          **if** $id(u) \in [(i-1)\, d\, g(n) + 1, i\, d\, g(n)]$ **then**
4:              Node u sends its ID to all other nodes.
5:              If a node receives messages in round i, then it selects the smallest ID from all received IDs in round i (including its own value, if it sent one) as the leader.

THEOREM 3.15. *Suppose that nodes are chosen from the set $\{1, \ldots, n\, g(n)\}$, where $g(n) \geqslant 1$ is any integer-valued function of n. For any $d \leqslant n$, Algorithm 1 outputs a leader within $\lceil \frac{n}{d} \rceil$ rounds with message complexity $n\, d\, g(n)$. In particular, for an ID universe of size $O(n)$, we obtain a sublinear time algorithm that sends $o(n \log n)$ messages.*

PROOF. Let u be the node with the minimum ID and i^* is such that $id(u) \in [(i^*-1)\, d\, g(n)+1, i^*\, d\, g(n)]$. Then at round i^*, each node with ID in $[(i^*-1)\, d\, g(n) + 1, i^*\, d\, g(n)]$ will send its ID to all other nodes. At the end of round i^*, all nodes will receive the same set of IDs and select node u as the leader.

Since all nodes' IDs are less than or equal to $n\, g(n)$, i^* is at most $\lceil \frac{n}{d} \rceil$. Hence, within $\lceil \frac{n}{d} \rceil$ rounds, the algorithm terminates. Moreover, there are at most $d\, g(n)$ nodes with IDs in $[(i^*-1)\, d\, g(n)+1, i^*\, d\, g(n)]$, hence, there are at most $d\, g(n)$ nodes that send their IDs to all other nodes at round i^*, giving the message complexity of $n\, d\, g(n)$. In particular, when $g(n)$ is a constant and d is $o(\log(n))$, then the message complexity is $o(n \log(n))$. □

3.5 Las Vegas Algorithms

As elaborated in more detail in Section 1, Kutten et al. [16] give a randomized Monte Carlo algorithm that successfully solves leader election with high probability using only $O(\sqrt{n} \log^{3/2} n)$ messages while terminating in just 2 rounds. Their algorithm is for implicit leader election (that is, not every node needs to know the name of the leader eventually [18]), and is a Monte Carlo one, i.e., fails with small probability of error. One may ask whether those two assumptions were necessary. First, we comment that it is trivial to turn their Monte Carlo algorithm into a Las Vegas explicit one. (Their algorithm elects a leader in the second round; one can add a step where the leader announces its identity to everybody at the third round: a node who does not receive an announcement in the third round restarts the algorithm.) Unfortunately, this obvious transformation of their algorithm increases the message complexity to $O(n)$. Below, we show that this is necessary, even for implicit leader election, i.e., if the nodes do not need to learn the identity of the leader.

THEOREM 3.16. *Any randomized Las Vegas leader election algorithm requires $\Omega(n)$ messages in expectation. Moreover, there exists an algorithm that with high probability, achieves $O(n)$ messages and terminates in 3 rounds.*

PROOF. The upper bound follows immediately from the previous discussion.

For the lower bound, suppose that there exists a Las Vegas randomized algorithm that solves implicit leader election, i.e., exactly one node becomes leader whereas everyone else becomes non-leader, with an expected message complexity of $o(n)$. Let E be the event that the algorithm sends $o(n)$ messages, and notice that E is guaranteed to happen with constant probability, for every assignment. Now consider some ID assignment I on a network with an even number of nodes. Conditioned on E, there exists a set of $n/2$ nodes with IDs $S \subseteq I$ such that no node in this set receives or sends any message from any other node throughout the execution. By the correctness of the algorithm, all nodes in S must eventually terminate. Let L_S be the indicator random variable that is 1 if and only if some node in S becomes leader, i.e., $L_S = 0$ if none of them does. Clearly, $\mathbf{Pr}[L_S = 1 \mid E] + \mathbf{Pr}[L_S = 0 \mid E] = 1$ and thus, one of the two events must occur with probability at least $\frac{1}{2}$.

Now consider two additional ID assignments I' and I'' such that $I, I',$ and I'' are mutually disjoint. By a similar argument as before, there exists a set $n/2$ nodes with IDs $S' \subseteq I'$ such that none of these nodes receives or sends any messages if event $E_{I'}$ happens, and we can also identify such a subset $S'' \subseteq I'$, conditioned on $E_{I''}$. We define the random variables $L_{S'}$ and $L_{S''}$ in exactly the same way as we did for S.

Since we have 3 ID assignments, it follows that there must be two of them, say I and I', such that $\mathbf{Pr}[L_S = b \mid E] \geqslant \frac{1}{2}$ and $\mathbf{Pr}[L_{S'} = b \mid E] \geqslant \frac{1}{2}$, for some $b \in \{0, 1\}$. Consider first the case that that $b = 1$.

Now consider the execution of the algorithm on the ID assignment $S \cup S'$. Since this execution is indistinguishable for nodes with IDs in S and S' from the executions with IDs I and I', respectively, it follows that the state transitions of every node have the same probability distribution in both executions. Consequently, the event that neither the nodes with IDs in S nor the ones with IDs in S' send or receive any messages has nonzero probability to occur. Moreover, the events $L_S = 1$ and $L_{S'} = 1$ are independent, and thus the algorithm elects 2 leaders with nonzero probability, which provides a contradiction.

A similar argument yields a contradiction when $b = 0$, here, this argument shows that the event where there is no leader in the execution $S \cup S'$ has nonzero probability. □

4   THE SYNCHRONOUS CLIQUE UNDER ADVERSARIAL WAKE-UP

We now consider the synchronous clique under *adversarial wake-up*, which means that the adversary chooses a nonempty subset of nodes that start the execution in round 1. At any later round, the adversary may choose additional nodes that are not awake (if such exist) and wake them up too. Any node not woken up by the adversary is asleep initially, and only wakes up at the end of a round if it received a message in that round. To make the proofs easier to read, we actually make the additional unnecessary assumption that the adversary only wakes up nodes in round 1. For the specific algorithm we present in this section, it is easy to get rid of this assumption. For the lower bound, since it holds under the unnecessary assumption, it certainly holds without it.

We first give a simple randomized 2-round leader election algorithm that succeeds with probability $1 - \epsilon - \frac{1}{n}$, has $O\big(n^{3/2} \log\big(\frac{1}{\epsilon}\big)\big)$ message complexity in expectation and never sends more than $O\big(n^{3/2} \log n\big)$ messages with high probability. Then, we show that this bound is tight, by proving that $\Omega\big(n^{3/2}\big)$ is a lower bound on the message complexity for randomized 2-round algorithms that succeed with sufficiently large probability. In fact, this lower bound holds not only for leader election, but even just for the problem of waking-up every node in the network.

4.1 An Optimal 2-Round Algorithm with $O(n^{3/2} \log n)$ Message Complexity

Fix some parameter $\epsilon \geqslant \frac{1}{\text{poly}(n)}$; the resulting algorithm will succeed with probability at least $1 - \epsilon - \frac{1}{n}$. A node that is awoken by the adversary sends $\lceil \sqrt{n} \rceil$ messages over uniformly at random sampled ports (without replacement). In round 2, any node that was awoken by the receipt of a round-1 message (i.e., *not* by the adversary) becomes a candidate with probability $\frac{\log(1/\epsilon)}{\lceil \sqrt{n} \rceil}$, whereas non-candidate nodes immediately become non-leaders. Every candidate samples a random rank from $[n^4]$ and sends its rank to all other nodes in round 2. At the end of round 2, a candidate becomes leader if and only if all rank messages that it has received carried a lower rank than its own.

> THEOREM 4.1. *Consider the synchronous clique under adversarial wake-up and fix any small $\epsilon \geqslant \frac{1}{\text{poly}(n)}$. There is a 2-round randomized leader election algorithm that succeeds with probability at least $1 - \epsilon - \frac{1}{n}$ and has an expected message complexity of $O(n^{3/2} \log(\frac{1}{\epsilon}))$. Moreover, it sends at most $O(n^{3/2} \log n)$ messages (whp).*

PROOF. Since the adversary wakes up at least one node at the start of round 1, at least $\lceil \sqrt{n} \rceil$ nodes will be awoken by a message and hence try to become candidate at the start of round 2. The probability that none of the nodes becomes a candidate is at most

$$\left(1 - \frac{\log(1/\epsilon)}{\lceil \sqrt{n} \rceil}\right)^{\lceil \sqrt{n} \rceil} \leqslant \exp\left(-\log\left(\frac{1}{\epsilon}\right)\right) = \epsilon.$$

Consider some candidate u. The probability that all of the at most $n - 1$ other candidates sample a rank different than u's is at least $\left(1 - \frac{1}{n^4}\right)^n \geqslant e^{(-2/n^3)} \geqslant 1 - \frac{2}{n^3}$. Taking a union bound implies that all candidates have distinct ranks with probability at least $1 - \frac{2}{n^2} \geqslant 1 - \frac{1}{n}$. Thus, the probability that there is at least one candidate and that all candidates have unique ranks is at least $1 - \epsilon - \frac{1}{n}$. Conditioned on these two events, it is clear that each candidate learns the ranks of all candidates by the end of round 2, and hence there is exactly one leader.

For the message complexity bound, we observe that, in the worst case, every node is awake in round 1 and sends $O(\sqrt{n})$ messages. In round 2, the expected number of candidates is $\sqrt{n} \log(\frac{1}{\epsilon})$, and each of these candidates sends $n - 1$ messages, which shows that the expected message complexity is $O(n^{3/2} + n^{3/2} \log(\frac{1}{\epsilon}))$.

Since $\epsilon \geqslant \frac{1}{\text{poly}(n)}$ and hence $\log(\frac{1}{\epsilon}) = O(\log n)$, a standard Chernoff bound shows that the number of candidates is $O(\log n)$, and hence we obtain the claimed high-probability bound of $O(n^{3/2} \log n)$ on the message complexity. □

4.2 A Tight Lower Bound for Randomized Algorithms

We now prove a lower bound that holds not only for leader election, but even just for the problem of waking-up every node in the network. In the proof, we assume that nodes are anonymous and do not have IDs. However, recall that it is straightforward for a randomized algorithm to generate IDs that are unique with high probability, by instructing each node to independently sample a random ID from the range $[n^4]$ (see the proof of Theorem 4.1 for a detailed argument).

THEOREM 4.2. *Consider the synchronous clique, where an arbitrary nonempty subset of nodes is awoken by the adversary at the start of round* 1. *Any randomized* 2-*round algorithm that wakes up every node with probability at least* $1 - \epsilon$, *has an expected message complexity of* $\Omega(n^{3/2})$, *for any positive constant* $\epsilon < 1$.

We point out that Theorem 4.2 extends to algorithms in the *asynchronous* clique that terminate within 2 units of time. Moreover, it improves significantly even over the previously-best lower bound of $\Omega(n^{5/4})$ messages [1] for *deterministic* 2-round algorithms under adversarial wake-up; see Section 1.2 for more details about their result.

4.2.1 Proof of Theorem 4.2. Since the proof is somewhat involved, let us first explain its highlights informally. The proof exploits the fact that limiting the time to 2 rounds, makes it difficult for a node to learn how many other nodes are awake, and hence it may need to send as many messages (on average) as it would in an execution where it is the only one who was woken up. Intuitively, this is why we are able to demonstrate a gap between the message complexity for a small constant time and the case of a large constant time. That is, for say, 9 rounds algorithms, $O(n)$ messages are possible [14]. This gap stands in contrast to the results shown by Afek and Gafni [1] for deterministic algorithms. There, for *any constant* time complexity of t rounds, the messages complexity was $\Omega(n^{1+t/2})$ messages. (Recall also that we have shown in Subsection 4.1 above an optimal $O(n^{3/2})$ messages randomized algorithm even for the 2-rounds case.) The source of the high complexity in the randomized case of 2 rounds algorithms seems different than the one exploited in the lower bound proof of [1]. (Moreover, formalizing the argument for randomized algorithms that may fail with some small probability turns out to be more involved.) Our high-level strategy is as follows: We call the nodes that are awake initially *roots* and any node that receives a message from a root becomes a *child* of that root (and also of any other root that sent it a message). We first show that root nodes cannot afford to send more than $o(\sqrt{n})$ messages, and then use this fact to show that many root nodes and child nodes will operate "undisturbed", in the sense that they do not receive any additional message on top of what they would have received in an execution where only a single root was awake. We show that, in expectation, this requires the child nodes to send at least $\Omega(n^{3/2})$ messages to ensure every other node wakes up by the end of round 2.

We now present the formal argument. Assume towards a contradiction that there exists a 2-round algorithm with a message complexity of $\frac{n^{3/2}}{f(n)}$, for some function $f(n) = \omega(1)$.

Events and Notation: Our proof relies on the occurrence of certain probabilistic events for which we introduce some notation.

- For any node u, let random variable $M(u, r)$ denote the set of round r messages sent by u annotated by the respective port numbers, and let $m(u, r) = |M(u, r)|$. We define $m(r) = \sum_u m(u, r)$ to be the total number of ports over which a message is sent in round r.
- Suppose that the adversary wakes up a set of nodes $R = \{u_1, \ldots, u_\ell\}$. We say that each u_i is a *root*, and define u_i's *children* to be the set of nodes $\{v_{i,1}, \ldots, v_{i,M(u_i,1)}\}$ that were asleep until receiving a message from u_i in round 1. Note that it is possible that $v_{i,j}$ is the child of multiple roots.

- Few_u captures the event that node u sends few messages in round 1. Formally, Few_u happens iff node u is a root and sends at most $\frac{\sqrt{n}}{g(n)}$ messages, for a function $g(n) = \omega(1)$, where $g(n) \geqslant f(n)$. We also define $\mathsf{Few}_R = \bigwedge_{u \in R} \mathsf{Few}_u$, for the set of nodes R.
- Event Undist_v ("v is *undisturbed*") occurs iff one of the following cases is true:
  (1) v is a root and does not receive any messages in round 1.
  (2) v is a child of some root u_i and only receives a message in round 1 from u_i.

Let us start the proof of Theorem 4.2 by showing several bounds on the probability of the events introduced above. Throughout this section, we define Λ to be an execution in which the adversary awakens a set of $\lceil \sqrt{n} \rceil$ root nodes R, and $R' \subseteq R$ the subset of roots that remain undisturbed in round 1 of Λ, i.e., $\bigwedge_{u_i \in R'} \mathsf{Undist}_{u_i}$.

LEMMA 4.3. *Let R be the set of roots and suppose that $|R| = \Theta(\sqrt{n})$. There exists a function $g(n) = \omega(1)$ such that $\mathbf{Pr}[\mathsf{Few}_R] \geqslant 1 - O\left(\frac{1}{g(n)}\right)$.*

PROOF. Let Γ be the execution where every node is a root, i.e., awoken in round 1. Since $M(u_i, 1)$ only depends on u_i's private coin flip and, in particular, is independent of the round-1 computation of other nodes, it follows that $\mathbf{Pr}[\mathsf{Few}_{u_i}] = \mathbf{Pr}[\mathsf{Few}_{u_j}]$ for all nodes u_i and u_j, and for both executions Λ and Γ. Assume towards a contradiction that $\mathbf{Pr}[\neg\mathsf{Few}_{u_i}] \geqslant \frac{c}{\sqrt{n}}$, for some constant $c > 0$. Since n nodes are awake, it follows that the expected value of $m(1)$ is $\Omega(n^{3/2})$ in both executions Γ and Λ, which is a contradiction.

So far, we have shown that $\mathbf{Pr}[\neg\mathsf{Few}_R] \leqslant \frac{1}{\sqrt{n}\,g(n)}$, for some function $g(n) = \omega(1)$ and, as argued above, this also holds in the execution Λ where only the nodes in R are roots. It follows that, in execution Λ,

$$\mathbf{Pr}[\mathsf{Few}_R] = 1 - \mathbf{Pr}[\exists u \in R \colon \neg\mathsf{Few}_u] \geqslant 1 - \frac{|R|}{\sqrt{n}\,g(n)} \geqslant 1 - O\left(\frac{1}{g(n)}\right).$$

$\square$

LEMMA 4.4. *Let $R' \subseteq R$ be the set of undisturbed roots, i.e., it holds that $\bigwedge_{u_i \in R'} \mathsf{Undist}_{u_i}$. Then,*

(A) $\mathbf{Pr}[\mathsf{Undist}_{u_i} \mid \mathsf{Few}_R] \geqslant 1 - O\left(\frac{1}{g(n)}\right)$, and

(B) $\mathbf{Pr}[|R'| \geqslant \frac{1}{2}|R| \mid \mathsf{Few}_R] \geqslant 1 - O\left(\frac{1}{g(n)}\right)$.

PROOF. Since we condition on event Few_R, we know that every root sends at most $\sqrt{n}/g(n)$ messages in round 1, and thus the total number of messages sent in round 1 is at most $\frac{|R|\sqrt{n}}{g(n)}$. Consider any root $u_i \in R$. Since the port connections are chosen uniformly at random, the probability that any message of another root u_j reaches u_i is at most $\frac{1}{\sqrt{n}\,g(n)}$. Thus,

$$\mathbf{Pr}[\mathsf{Undist}_{u_i} \mid \mathsf{Few}_R] \geqslant \left(1 - \frac{1}{\sqrt{n}\,g(n)}\right)^{|R|} \geqslant \exp\left(-\frac{2|R|}{\sqrt{n}\,g(n)}\right) \geqslant 1 - O\left(\frac{1}{g(n)}\right),$$

which proves (A). For (B), note that $\mathbf{E}[|R| - |R'| \mid \mathsf{Few}_R] \leqslant O\left(\frac{|R|}{g(n)}\right)$, and, by applying Markov's Inequality, we derive that $\mathbf{Pr}[|R| - |R'| \geqslant \frac{1}{2}|R|] \leqslant O\left(\frac{1}{g(n)}\right)$.

$\square$

We are now ready to derive a lower bound on the message complexity of the assumed algorithm. To this end, we lower-bound the total expected message complexity by only counting the number of messages sent by the children of each undisturbed root $u_i \in R'$ in round 2. We have

$$\mathbf{E}[m(2)] \geqslant \mathbf{E}\left[\sum_{u_i \in R'} \sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2)\right]$$

$$\geqslant \mathbf{Pr}[\mathsf{Few}_R]\, \mathbf{E}\left[\sum_{u_i \in R'} \sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2)\,\Bigg|\, \mathsf{Few}_R\right].$$

$$\text{(by Lemma 4.3)} \quad \geqslant (1 - o(1))\, \mathbf{E}\left[\sum_{u_i \in R'} \sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2)\,\Bigg|\, \mathsf{Few}_R\right].$$

Next, we condition on the event that $|R'| \geqslant \frac{1}{2}|R|$ and, without loss of generality, assume that $R' = \{u_1, \ldots, u_\ell\}$, where $\ell = \lfloor |R|/2 \rfloor$. We get

$$\mathbf{E}[m(2)] \geqslant (1 - o(1))\, \mathbf{Pr}\left[|R'| \geqslant \tfrac{1}{2}|R| \,\big|\, \mathsf{Few}_R\right]$$

$$\cdot \sum_{i=1}^{\ell} \mathbf{E}\left[\sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2)\,\Bigg|\, |R'| \geqslant \tfrac{1}{2}|R|, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right].$$

$$\text{(by Lem. 4.4)} \quad \geqslant (1 - o(1))$$

$$\cdot \sum_{i=1}^{\ell} \mathbf{E}\left[\sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2)\,\Bigg|\, |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right]. \tag{5}$$

Note that the conditioning on Undist_{u_i} is implied by the fact that the outer sum ranges only over undisturbed roots. Let C_i' be the set of undisturbed child nodes of root u_i. We make use of the following lemma, whose proof we postpone to Section 4.2.2:

LEMMA 4.5. *Consider execution Λ. For every root $u_i \in R$, we have*

$$\mathbf{Pr}\left[|C_i'| \geqslant \tfrac{1}{2}m(u_i, 1), m(u_i, 2) = o(n)\,\Big|\, |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right] \geqslant 1 - o(1).$$

Let $\mathcal{E}_i$ denote the event

$$\left(|C_i'| \geqslant \tfrac{1}{2}m(u_i, 1)\right) \wedge \left(m(u_i, 2) = o(n)\right) \wedge \left(|R'| \geqslant \tfrac{\sqrt{n}}{2}\right) \wedge \mathsf{Few}_R \wedge \mathsf{Undist}_{u_i} \tag{6}$$

where we assume (w.l.o.g.) that $C_i' = \{v_{i,1}, \ldots, v_{i,k}\}$, for some $k \geqslant \frac{1}{2}m(u_i, 1)$. By restricting the inner-most summation to range only over the undisturbed children in (5), we get

$$\mathbf{E}[m(2)] \geqslant (1 - o(1))$$

$$\cdot \mathbf{Pr}\left[|C_i'| \geqslant \tfrac{m(u_i,1)}{2}, m(u_i, 2) = o(n) \;\middle|\; |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R\right]$$

$$\cdot \sum_{i=1}^{\ell} \mathbf{E}\left[\sum_{j=1}^{k} m(v_{i,j}, 2) \;\middle|\; \mathcal{E}_i\right]$$

$$\text{(by Lem 4.5)} \quad \geqslant (1 - o(1)) \sum_{i=1}^{\ell} \mathbf{E}\left[\sum_{j=1}^{k} m(v_{i,j}, 2) \;\middle|\; \mathcal{E}_i\right]. \tag{7}$$

Let Corr be the event that the algorithm succeeds in waking up all nodes. Notice that

$$\mathbf{Pr}[\mathsf{Corr} \mid \mathcal{E}_i] \geqslant \mathbf{Pr}[\mathsf{Corr}] - \mathbf{Pr}[\neg \mathcal{E}_i]$$

$$\geqslant (1 - \epsilon) - o(1). \tag{8}$$

Continuing from the right-hand side of (7), we get

$$\mathbf{E}[m(2)] \geqslant (1 - o(1))$$

$$\cdot \mathbf{Pr}[\mathsf{Corr} \mid \mathcal{E}_i] \sum_{i=1}^{\ell} \mathbf{E}\left[\sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2) \;\middle|\; \mathsf{Corr}, \mathcal{E}_i\right],$$

$$\text{(by (8))} \quad \geqslant (1 - \epsilon - o(1)) \sum_{i=1}^{\ell} \mathbf{E}\left[\sum_{j=1}^{m(u_i,1)} m(v_{i,j}, 2) \;\middle|\; \mathsf{Corr}, \mathcal{E}_i\right]. \tag{9}$$

Let Γ be the execution where only u_i is awake in round 1. Due to conditioning on $\mathcal{E}_i$ (defined in (6)), it holds that

$$m(u_i, 1) + m(u_i, 2) \leqslant \frac{\sqrt{n}}{g(n)} + o(n) = o(n),$$

and hence, conditioned on Corr, we know that

$$\sum_{j=1}^{k} m(v_{i,j}, 2) \geqslant n - o(n) \geqslant \frac{n}{2} \tag{10}$$

in Γ, i.e., the children of u_i become responsible for waking up $\Omega(n)$ nodes in round 2. Since $u_i \in R'$, we know that u_i is undisturbed, and thus the distribution of $m(u_i, 2)$ is the same in both executions Γ and Λ (in which there are $\Theta(\sqrt{n})$ roots). Furthermore, the summation in (10) ranges only over the undisturbed child nodes, which means that $m(v_{i,j}, 2)$ has the same distribution in Γ and Λ. Therefore, the bound (10) must also hold in Λ, for the children of each $u_i \in R'$. Plugging (10) into (9) and recalling that $\ell = \Theta(\sqrt{n})$, we conclude that

$$\mathbf{E}[m(2)] \geqslant (1 - \epsilon - o(1))\frac{\ell n}{2} = \Omega\left(n^{3/2}\right).$$

4.2.2 Proof of Lemma 4.5.

LEMMA 4.5 (RESTATED). *Consider execution Λ. For every root $u_i \in R$, we have*

$$\mathbf{Pr}\left[|C_i'| \geqslant \tfrac{1}{2}m(u_i, 1), m(u_i, 2) = o(n) \,\middle|\, |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right] \geqslant 1 - o(1).$$

The following Lemmas 4.6 and 4.7 together with a simple application of the chain rule will complete the proof of Lemma 4.5.

LEMMA 4.6. *Let $u_i \in R$ be a root in execution Λ. Then,*

$$\mathbf{Pr}\left[m(u_i, 2) = o(n) \,\middle|\, |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right] \geqslant 1 - o(1).$$

PROOF. Let event $\mathcal{F}_i = (|R'| \geqslant \tfrac{\sqrt{n}}{2}) \wedge \mathsf{Few}_R \wedge \mathsf{Undist}_{u_i}$. Note that

$$\mathbf{Pr}\left[(|R'| \geqslant \tfrac{\sqrt{n}}{2}) \wedge \mathsf{Few}_R \wedge \mathsf{Undist}_{u_i}\right] = \mathbf{Pr}\left[|R'| \geqslant \tfrac{\sqrt{n}}{2} \,\middle|\, \mathsf{Undist}_{u_i}, \mathsf{Few}_R\right] \mathbf{Pr}\left[\mathsf{Undist}_{u_i} \,\middle|\, \mathsf{Few}_R\right] \mathbf{Pr}[\mathsf{Few}_R]$$

$$\geqslant \mathbf{Pr}\left[|R'| \geqslant \tfrac{\sqrt{n}}{2} \,\middle|\, \mathsf{Few}_R\right] \mathbf{Pr}\left[\mathsf{Undist}_{u_i} \,\middle|\, \mathsf{Few}_R\right] \mathbf{Pr}[\mathsf{Few}_R],$$

where the last inequality follows because removing the conditioning on Undist_{u_i} can only decrease the probability that we have at least $|R'| \geqslant \tfrac{\sqrt{n}}{2}$ many undisturbed roots. By applying Lemmas 4.4 and 4.3, it follows that

$$\mathbf{Pr}\left[(|R'| \geqslant \tfrac{\sqrt{n}}{2}) \wedge \mathsf{Few}_R \wedge \mathsf{Undist}_{u_i}\right] \geqslant 1 - O\left(\frac{1}{g(n)}\right). \tag{11}$$

To complete the proof, we argue that u_i must send many messages in round 2, if $\mathcal{F}_i$ happens. Let

$$p = \mathbf{Pr}[m(u_i, 2) = \Omega(n) \mid \mathcal{F}_i],$$

and note that our goal is to show that $p = o(1)$. By assumption, all nodes in R' are undisturbed in round 1 of Λ, and hence the distribution $m(u_i, 2)$ is the same for all $u_i \in R'$. It follows that the expected message complexity is at least

$$\mathbf{E}[m(2) \mid \mathcal{F}_i] \geqslant \mathbf{E}\left[\sum_{u_i \in R'} m(u_i, 2) \,\middle|\, \mathcal{F}_i\right]$$

$$\geqslant \sum_{u_i \in R'} \mathbf{E}[m(u_i, 2) \mid \mathcal{F}_i]$$

$$\geqslant p \cdot \sum_{u_i \in R'} \mathbf{E}[m(u_i, 2) \mid m(u_i, 2) = \Omega(n), \mathcal{F}_i]$$

$$= \Omega(p\,n\,|R'|)$$

$$= \Omega\left(p\,n^{3/2}\right). \tag{12}$$

It follows that

$$\mathbf{E}[m(2)] \geqslant \mathbf{Pr}[\mathcal{F}_i]\,\mathbf{E}[m(2) \mid \mathcal{F}_i]$$

$$\text{(by (11) and (12))} \quad \geqslant \left(1 - \frac{1}{g(n)}\right)\Omega\left(p\,n^{3/2}\right)$$

$$= \Omega\left(p\,n^{3/2}\right),$$

and since the algorithm has an assumed message complexity of $o\left(n^{3/2}\right)$, this implies $p = o(1)$, as required. □

LEMMA 4.7. *Let C_i' be the set of undisturbed child nodes of root u_i. It holds that*

$$\mathbf{Pr}\left[|C_i'| \geqslant \tfrac{1}{2}m(u_i, 1) \;\middle|\; m(u_i, 2) = o(n), |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right] \geqslant 1 - O\left(\tfrac{1}{g(n)}\right).$$

PROOF. Let $v_{i,j}$ be a child of root $u_i \in R'$ in execution Λ. We first show that

$$\mathbf{Pr}\left[\neg\mathsf{Undist}_{v_{i,j}} \;\middle|\; m(u_i, 2) = o(n), |R'| \geqslant \tfrac{\sqrt{n}}{2}, \mathsf{Few}_R, \mathsf{Undist}_{u_i}\right] \leqslant O\left(\tfrac{1}{g(n)}\right). \tag{13}$$

Recall that any root node in R' does not receive messages in round 1. To account for the conditioning on the event $|R'| \geqslant \tfrac{1}{2}|R|$, we pessimistically assume that $|R'| = |R|$, which means that all messages sent in round 1 must go to $n - |R| = n - \Theta(\sqrt{n}) \geqslant \tfrac{1}{2}n$ non-root nodes, one of which is $v_{i,j}$, as this assumption can only decrease the probability of $\mathsf{Undist}_{v_{i,j}}$. It follows that the probability that some root sends a round-1 message to $v_{i,j}$ is at most $\frac{2\sqrt{n}}{n\,g(n)}$. Thus the probability of $\neg\mathsf{Undist}_{v_{i,j}}$ is at most

$$1 - \left(1 - \frac{2}{\sqrt{n}\,g(n)}\right)^{|R|}$$

$$\text{(since } 1 - x \geqslant e^{-2x}, \text{ for } x < \tfrac{1}{2}) \quad \leqslant 1 - \exp\left(-\frac{4|R|}{\sqrt{n}\,g(n)}\right)$$

$$\text{(since } 1 - x \leqslant e^{-x}) \quad \leqslant 1 - \left(1 - \frac{4|R|}{\sqrt{n}\,g(n)}\right)$$

$$= O\left(\tfrac{1}{g(n)}\right),$$

which proves (13). It follows that the expected number of disturbed children of u_i is at most $m(u_i, 1)\left(1 - O\left(\tfrac{1}{g(n)}\right)\right)$, and Markov's Inequality ensures that the probability of

$$m(u_i, 1) - |C'| \geqslant \tfrac{1}{2}m(u_i, 1)$$

is at most $O\left(\tfrac{1}{g(n)}\right)$. Conversely, we have $|C'| \geqslant \tfrac{1}{2}m(u_i, 1)$ with probability at least $1 - O\left(\tfrac{1}{g(n)}\right)$. □

5 A COMMUNICATION-TIME TRADEOFF IN THE ASYNCHRONOUS CLIQUE

In this section, we present a randomized leader election algorithm that achieves the first trade off between messages and time in the asynchronous model.

Indeed, in addition to the adversarial delay of messages, we also allow an adversarial wake up.[3] (Note that the lower bound of Section 4 applies here too, since this section deals with adversarial wake up too). We assume that the adversary must choose the port mapping *obliviously*, i.e., before the first node is awoken and hence the port connections are independent of the random bits used by the nodes. However, the adversary may adaptively schedule the steps taken by the nodes: in fact, after fixing the port mappings, we allow the adversary to inspect all random bit strings before deciding which node takes the next step and at what time. Equivalently to the synchronous model, the *message complexity* of an asynchronous algorithm is the worst-case total number of messages sent during its execution. Note that we assume that each communication link guarantees FIFO message delivery. We follow the standard of defining the *asynchronous time complexity* as the worst-case total number of time units since the first node was awoken until the last message was received, whereas a *unit of time* refers to an upper bound on the transmission time of a message.

When a node u wakes up, either by the adversary or due to receiving a message, it samples uniformly at random $\Theta(n^{1/k})$ ports for sending its wake-up message, where k is a parameter that controls the tradeoff between messages and time. Then, u becomes a *leader candidate* with probability $\Theta(\log n/n)$, and, if it is a candidate, chooses a number called *rank* from a range of a polynomial size that is, a rank that is unique with high probability. If u is a candidate, it samples $\Theta(\sqrt{n \log n})$ ports (without replacement) uniformly at random and sends a compete message that carries its rank; the receiving nodes will serve as u's *referees*. Each node v keeps track of the "winning" candidate's rank ρ_{winner} that v has seen so far: if v is a candidate, it will store its own rank in its variable ρ_{winner} (which is empty initially) upon becoming candidate. Had this been a synchronous network, the referee would have known when all the compete messages arrived, and would have been able to reply "you win!" to the highest ranked node (possibly itself). Due to the asynchronous arrival of the compete messages, each referee responds "you win!" already to the very first compete message it receives from some node u, if v is not a candidate; in that case, v also stores the rank of u in ρ_{winner}. However, the above "winnings" may be revoked later. That is, whenever v receives a compete message with rank ρ from u when ρ_{winner} is not empty, node v immediately replies with a you lose!-message if $\rho \leqslant \rho_{\text{winner}}$. Otherwise, v consults with the node w whose current rank v stores in ρ_{winner} to see whether w has already decided to become leader. (Node w may be v itself.) Assuming that w has not become elected (for instance, because w is still waiting to hear back from its other referees), then w itself drops out of the competition upon receiving v's message. Node v, in turn, will consider u to be the current winning candidate, update the value in ρ_{winner} accordingly, and send a you win! message to u. Eventually, the candidate that receives you win! messages from all its referees and has not yet dropped out of the competition becomes leader. We provide the full pseudo code in Algorithm 2.

THEOREM 5.1. *For any integer k where $2 \leqslant k \leqslant O\left(\frac{\log n}{\log \log n}\right)$, there exists an asynchronous leader election algorithm that, with high probability, elects a unique leader in at most $k + 8$ units of time and sends at most $O\left(n^{1+1/k}\right)$ messages.*

---

[3]Recall that in the synchronous case we addressed the two kinds of wake up separately—we assumed simultaneous wake-up in Section 3 since this made our lower bounds harder to show, and addressed the synchronous clique with adversarial wake-up in Section 4 since this allowed for an even higher lower bound.

Algorithm 2 An asynchronous leader election algorithm that takes a tradeoff parameter k, where $2 \leqslant k \leqslant O(\log n/\log \log n)$.

1: Initially, every node is *undecided* and eventually may *decide* to either become leader or non-leader.
2: Each node u has a variable called *winner-rank* ρ_{winner}, which is empty initially, and is used to store the rank of received messages (explained below). In addition, it keeps track of the corresponding node that sent the rank currently stored in winner-rank.
3: **if** an asleep node u receives a message or is awoken by the adversary **then**
4:     u sends a message $\langle$wake up!$\rangle$ over $\Theta(n^{1/k})$ ports chosen uniformly at random.
5: u becomes a *candidate* with probability $\frac{4 \log n}{n}$.
6: **if** node u is a candidate **then**
7:     Node u chooses a *rank* ρ_u uniformly at random from $[n^4]$ and stores it in ρ_{winner}.
8:     u samples a set R_u of $\lceil 4\sqrt{n \log n} \rceil$ ports uniformly at random (without replacement).
9:     u sends $\langle \rho_u, \text{compete} \rangle$ over the ports in R_u, and considers the nodes in R_u its *referees*.
10:     **if** u has received $\langle$you win!$\rangle$ from all its referees and is still undecided **then**
11:         u decides to be leader and informs all nodes, who change their statuses to non-leader.
12:             $\triangleright$ A node responds to received $\langle \rho, \text{compete} \rangle$-messages even if it has already decided.
13:     **if** u receives a $\langle$you lose!$\rangle$ message **then**
14:         u decides to become non-leader.
15: Every (decided or undecided) node v processes a received $\langle \rho_u, \text{compete} \rangle$ message as follows:
16: **if** v's winner-rank ρ_{winner} is empty **then**
17:     v stores ρ_u in ρ_{winner}, sends $\langle$you win!$\rangle$ to u, and v becomes non-leader.
18: **else if** ρ_u is not larger than v's winner-rank ρ_{winner} **then**
19:     v sends $\langle$you lose!$\rangle$ to u, causing u to become non-leader.
20: **else** $\triangleright$ ρ_u is larger than v's current value of ρ_{winner}
21:     Let w be the node associated with the value in ρ_{winner} of v.
22:     v asks w whether it is already a leader.
23:     **if** w has not yet become leader at the time it receives v's request **then**
24:         w becomes non-leader and informs v.
25:         Upon receiving w's answer, v responds to u by sending $\langle$you win!$\rangle$.
26:         v stores u's rank in ρ_{winner}.
27:     **else**
28:         w informs v that it has already become leader.
29:         Upon receiving w's answer, v sends $\langle$you lose!$\rangle$ to u, causing u to become non-leader.

Let us illustrate the tradeoff achieved by Theorem 5.1 by considering the two extremes: For $k = 2$, the algorithm terminates in at most 10 units of time and sends $O(n^{3/2})$ messages, thus matching the lower bound of Theorem 4.2, whereas for $k = \Theta(\log n/\log \log n)$, we obtain a time complexity of $O(\log n)$ and a message complexity of $O(n \log n)$.

In the remainder of this section, we prove Theorem 5.1. For the analysis, we consider the execution as being split into two phases: a *wake-up phase*, where the goal is to wake up every node in the network eventually and an *election phase*, in which the nodes determine the leader. Since the network

is asynchronous, it can happen that some nodes enter (or even complete) their election phase, while other nodes are still asleep.

5.1 Analysis of the Wake-Up Phase

LEMMA 5.2. *Assume that $2 \leqslant k = O\left(\frac{\log n}{\log \log n}\right)$ and consider the asynchronous model. If each node sends $\gamma n^{1/k}$ messages initially over uniformly at random sampled ports, where γ is a sufficiently large constant, then each node wakes up within $k + 4$ units of time with high probability.*

In the remainder of this section, we prove Lemma 5.2. We call a point in time t_i a *wake-up step of node u_i*, if u is awoken by the adversary at time t_i, or, because u received a message from another node at time t_i. Without loss of generality, we assume that exactly one node wakes up in each wake-up step and that the adversary wakes up only one node initially. It is straightforward to verify that our analysis also holds for the case where the adversary wakes up multiple nodes simultaneously, since every node behaves the same way upon wake-up.

Definition 5.3 (Covered Nodes). For any given point in time t, we define C_t to be the set of *covered nodes*, which contains every node u that is either awake at time t or, if u is still asleep, then u must be the destination of some in-transit message that was sent at some time $t' < t$. We use $\bar{C}_{t_i}$ to denote the complement of C_{t_i}, which contains every node that is still asleep and does not have any in-transit message addressed to it at time t_i.

LEMMA 5.4. *Suppose that the i-th wake-up step occurs at time t_i. There exists a constant $c \geqslant 1$ such that, with high probability, for all $i \geqslant 1$, if $|C_{t_i}| \leqslant \frac{n}{16}$, then,*

$$|C_{t_i+1}| \geqslant |C_{t_i}| + c\, n^{1/k}. \tag{14}$$

PROOF. Let u_i be the node that woke up at time t_i, which means that u_i sends $\gamma n^{1/k}$ messages over uniformly at random chosen ports, where γ is a sufficiently large constant. Fix some enumeration $m_1, \ldots, m_{\gamma n^{1/k}}$ of u_i's messages, and, for each m_i, define the indicator random variable Z_i such that $Z_i = 1$ if and only if m_i is sent to C_{t_i}. Since u_i sends m_i over a port that is distinct from the ports used for $m_1, \ldots, m_{i-1}$ and there are $n - 1$ ports in total, it follows that

$$\mathbf{Pr}\left[\bigwedge_{i=1}^{\ell} Z_i = 1\right] = \prod_{i=1}^{\ell} \mathbf{Pr}\left[Z_i = 1 \mid \bigwedge_{j=1}^{i-1} Z_j = 1\right]$$

$$\leqslant \prod_{i=1}^{\ell} \frac{|C_{t_i}| - (i-1)}{n - 1 - (i-1)}$$

$$(\text{since } i - 1 \leqslant \gamma n^{1/k}) \quad \leqslant \prod_{i=1}^{\ell} \frac{|C_{t_i}|}{n - 1 - \gamma n^{1/k}}$$

$$(\text{since } |C_{t_i}| \leqslant n/16 \text{ and } \gamma n^{1/k} + 1 \leqslant n/16) \quad \leqslant \left(\frac{1}{15}\right)^{\ell} \tag{15}$$

We make use of the following generalized Chernoff bound:

LEMMA 5.5 (IMPLICIT IN THEOREM 1.1 IN [10]). *Let $X_1, \ldots, X_N$ be (not necessarily independent) Boolean random variables and suppose that, for some $\delta \in [0, 1]$, it holds that, for every index set $S \subseteq [N]$, $\mathbf{Pr}[\bigwedge_{i \in S} X_i] \leq \delta^{|S|}$. Then, for any $\beta \in [\delta, 1]$, we have $\mathbf{Pr}\big[\sum_{i=1}^{N} X_i \geq \beta N\big] \leq e^{-2N(\beta - \delta)^2}$.*

According to (15), we can instantiate Lemma 5.5 for the random variables $Z_1, \ldots, Z_{\gamma n^{1/k}}$ by setting $\delta = \frac{1}{15}$ and $\beta = \frac{1}{2}$, which ensures that $\beta - \delta \geq \frac{1}{4}$. It follows that

$$\mathbf{Pr}\left[\sum_{i=1}^{\gamma n^{1/k}} Z_i \geq \frac{\gamma}{2} n^{1/k}\right] \leq e^{-2\gamma(\beta - \delta)^2 n^{1/k}} \leq e^{-\frac{\gamma n^{1/k}}{8}}.$$

Since $n^{1/k} = \Omega(\log n)$, it holds with high probability that at least $\frac{\gamma}{2} n^{1/k}$ nodes in $\bar{C}_{t_i}$ will receive a message from u_i and hence are added to $C_{t_{i+1}}$. Finally, we take a union bound over the at most $O(n)$ wake-up steps for which the premise $|C_{t_i}| \leq n/16$ holds, which shows that (14) holds for all of them, thereby completing the proof of Lemma 5.4. □

So far, we have shown that the number of nodes that eventually wake up grows by an additive $\Theta(n^{1/k})$ factor at each wake-up step. We now leverage this result to show that the growth is indeed geometric as expected. To this end, consider the time t_* of the latest wake-up step such that $|C_{t_*}| \leq \frac{n}{16}$ holds. We define the *cover tree* $\mathcal{T}_{t_*}$ on the nodes in C_{t_*}, which has as root the node that was awoken by the adversary and a node u is the parent of node v in $\mathcal{T}_{t_*}$ if and only if v is woken up by a message sent by u at some time $t' < t_*$.

LEMMA 5.6. *Each node in the network appears at most once in $\mathcal{T}_{t_*}$, and $V(\mathcal{T}_{t_*}) = C_{t_*}$. The edges of $\mathcal{T}_{t_*}$ form a directed tree where each non-leaf node has at least $c_1 n^{1/k}$ and at most $\gamma n^{1/k}$ children. Moreover, any node that has not yet awoken by time t_*, but to which a message was sent before time t_*, is a leaf in $\mathcal{T}_{t_*}$.*

PROOF. By definition, each node in $\mathcal{T}_{t_*}$ (except the root) is awoken due to a message sent at some time before t_*, and linearity of time ensures that the tree is acyclic. Thus, every node in the network shows up at most once in $\mathcal{T}_{t_*}$. Moreover, if a node v is a leaf, then it must be woken up by a message that was sent but not yet received before time t_*, as otherwise v itself would have added (at least) $c_1 n^{1/k}$ children to $\mathcal{T}_{t_*}$ according to Lemma 5.4. From Definition 5.3 is straightforward to verify that a node is in $\mathcal{T}_{t_*}$ if and only if it is in C_{t_*}. The upper bound on the degree is immediate from the fact that a node sends $\gamma n^{1/k}$ messages upon wake-up, whereas the lower bound follows by applying Lemma 5.4 to each non-leaf node in $\mathcal{T}$. □

LEMMA 5.7. *With high probability, every node in the tree $\mathcal{T}_{t_*}$ is woken up by a messages that was sent before time $k + 2$.*

PROOF. Let u_1 be the root of the tree. Assume towards a contradiction that there exists a path $p = u_1, u_2, \ldots, u_\ell$ in $\mathcal{T}_{t_*}$ such that the message that wakes up the leaf u_ℓ is sent at time $t_\ell \geq k + 2$, and suppose that path p is chosen such that t_ℓ is the latest time for all paths satisfying the property. Since each message takes at most 1 unit of time, it follows that $\ell \geq k + 3$. (Note that we assume that the root u_1 is awoken at time 0.) Now consider any path q starting from the root u_1 and continuing over a distinct child $u_2' \neq u_2$. Let u_m' be the last node on path q, i.e., u_m' is a leaf.

We first argue that the message that wakes up u'_m must be sent at some time

$$t' \geqslant t_\ell - 1 \geqslant k + 1.$$

Recall that $\mathcal{T}_{t_*}$ contains all nodes that are either awake or to which a message is in-transit at any time before t_*, which, in particular, includes time t_ℓ, and assume towards a contradiction that $t' < t_\ell - 1$. By assumption, u'_m must wake up by time

$$t' + 1 < t_\ell \leqslant t_*.$$

Since $t_\ell \leqslant t_*$, it follows that $|C_{t_\ell}| \leqslant |C_{t_*}| \leqslant \frac{n}{16}$, and hence Lemma 5.4 ensures that u'_m sends messages to previously asleep nodes by time $t' + 1 < t_*$, contradicting the assumption that u'_m is a leaf.

Thus, we can assume that $t' \geqslant t_\ell - 1 \geqslant k + 1$, which implies that q contains at least $k + 2$ nodes, and $k + 1$ of them must be non-leaf nodes. Therefore, any path from the root consists of at least $k + 1$ non-leaf nodes that have a minimum degree of $c_1 n^{1/k}$, according to Lemma 5.6. Overall, this means that there are at least $c_1^{k+1} n^{1+1/k} > n$ nodes in $\mathcal{T}_{t_*}$, since $c_1 \geqslant 1$, thus providing a contradiction. □

To complete the proof of Lemma 5.2, we show that, once we have reached a "critical mass" of $\frac{n}{16}$ nodes that are either awake (or about to wake up), the remaining nodes wake up within 2 additional time steps, resulting in a time complexity of $t_* + 2 \leqslant k + 4$, as claimed.

LEMMA 5.8. *With high probability, all nodes wake up by time $t_* + 2$.*

PROOF. Recall the cover tree $\mathcal{T}_{t_*}$ for time t_* defined above, which was the latest wake-up step at which $|C_{t_*}| \leqslant \frac{n}{16}$. Since $|C_{t_*}| \geqslant \frac{n}{16} - 1 \geqslant \frac{n}{32}$, it follows that $\mathcal{T}_{t_*}$ contains at least $|\mathcal{T}_{t_*}| - \frac{|\mathcal{T}_{t_*}|}{c_1 n^{1/k}} \geqslant \frac{n}{32} - \frac{n^{1-1/k}}{c_1} \geqslant \frac{n}{64}$ leafs. According to Lemma 5.6, a leaf is a node that has not yet woken up by time t_*, which means that there are at least $\frac{n}{64}$ nodes that will wake up due to messages sent before time t_*, all of which must arrive at some point in the time interval $[t_*, t_* + 1]$.

Let M be the set of the messages sent by these nodes during their wake-up steps and note that $|M| \geqslant \frac{\gamma n^{1+1/k}}{64}$. The probability that a node $u \in \bar{C}_{t_*}$ is *not* the destination of at least one of the messages in M is at most

$$\left(1 - \frac{1}{n}\right)^{\gamma n^{1+1/k}/64} \leqslant e^{\left(-\frac{\gamma n^{1+1/k}}{64}\right)} \leqslant n^{-4},$$

where the last inequality follows from the assumption that $n^{1/k} = \Omega(\log n)$ and that γ is a sufficiently large constant. Observing that all messages in M must arrive by time $t_* + 2$, and taking a union bound over the at most $\frac{31}{32} n$ nodes that are not in C_{t_*}, ensures that every node will be awoken by time $t_* + 2$ with high probability. □

5.2 Analysis of the Election Phase

LEMMA 5.9. *Suppose that every node wakes up eventually. Then there is exactly one leader with high probability.*

PROOF. Since each node becomes candidate with probability $4 \log n / n$, it follows that the probability there is a candidate is at least $1 - \left(1 - \frac{4 \log n}{n}\right)^n \geqslant 1 - e^{-4 \log n} = 1 - \frac{1}{n^4}$. By a simple calculation that is analogous to the one in the proof of Theorem 4.1 in Section 4, it follows that every candidate will have a

unique ID with high probability. For the remainder of the proof, we condition on these high-probability events.

Now, assume towards a contradiction that there exist two candidates u and v that both become leaders. Let R_u and R_v be the set of referees chosen by u respectively v. Since it holds that

$$\mathbf{Pr}[R_u \cap R_v = \emptyset] \leqslant \left(1 - \frac{4\sqrt{\log n}}{\sqrt{n}}\right)^{4\sqrt{n \log n}} \leqslant \frac{1}{n^2},$$

we know that u and v will share some common referee w. By the code of the algorithm, we know that w sends a winner-message at most once to each node that contacts it. By the code of the algorithm, any undecided node becomes non-leader upon receiving a ⟨you lose!⟩ message, which tells us that w sent a message ⟨you win!⟩ to both u and v. Without loss of generality, assume that w sent the ⟨you win!⟩ first to u and later on to v. It follows that w must have processed the message ⟨ρ_u⟩ before ⟨ρ_v⟩. However, by the code of the algorithm, w stores u's rank in its winner-rank variable upon sending ⟨you win!⟩ to u at some time t_1. Moreover, since w sent a ⟨you win!⟩ also to v at some later time $t_2 > t_1$, it follows that u cannot have elected itself as a leader before t_2, as otherwise w would have sent ⟨you lose!⟩ to v upon receiving u's reply. This implies that u must have already decided to become non-leader upon being contacted by v. We have arrived at a contradiction. □

Lemma 5.10. *Each node decides in at most* 4 *units of time after its wake-up step.*

Proof. Suppose node u wakes up at time t. If u does not become a candidate, it decides instantly; thus, assume that u does become a candidate and contacts a set of referees. Upon receiving u's message, a referee may need to confirm the status of the current winner, which takes one additional round-trip before it sends its response to u. Thus, it takes at most 4 units of time until u receives a response from each of its referees and decides. Note that u may also serve as referee for other nodes, however, responding to these messages does not slow down its own decision process. □

5.3 Putting Everything Together

We have now assembled all the ingredients for completing the proof of Theorem 5.1. We first argue correctness: With high probability, the algorithm will elect exactly one node as the leader since Lemma 5.2 shows that every node wakes up and, in that case, Lemma 5.9 guarantees that exactly one of them will become leader.

For the claimed time complexity of $k + 8$, we observe that Lemma 5.2 ensures that every node wakes up within $k + 4$ units of time, and Lemma 5.10 tells us that everyone decides to become leader or non-leader within 4 additional time units.

Finally, note that each node sends $\Theta(n^{1+1/k})$ messages upon wake-up. We have $\Theta(\log n)$ candidates (whp), and each of them communicates with $\Theta(\sqrt{n \log n})$ referees. Upon being contacted by a candidate, a referee may need to communicate with its currently stored winner node, which requires an additional 2 messages. Thus, we have an overall message complexity of $O(n^{1+1/k}) + O(\sqrt{n} \log^{3/2} n) = O(n^{1+1/k})$ with high probability.

5.4 Asynchronizing the Trade-off of Afek and Gafni under Simultaneous Wake-up

The following algorithm can be viewed as an asynchronous version of the synchronous algorithm of Afek and Gafni [1]. However, Afek and Gafni argue that their algorithm would not work in asynchronous networks, and thus suggested three asynchronous algorithms where the $O(n \log n)$ messages were obtained at the cost of $\Omega(n)$ time. We managed to make their original algorithm asynchronous without increasing their original time, assuming that we count only from the last spontaneous (adversarial) wake up of a node. Alternatively, for the sake of this algorithm only, we can assume that all the nodes wake up simultaneously. The algorithm works as follows: Each surviving candidate v (initially everybody) is at a level in $[0, 1, 2, \dots \log n]$ (initially zero). It sends 2^i request messages to its first 2^i neighbors (v is its own neighbor number 1; we also assume that the ports are numbered $2, 3, \dots n$). If v gets acks for all the messages it sends (and has not meanwhile been killed) then it climbs to level $i + 1$. If it sent messages to all the nodes and received acks, then it terminates as a leader.

A node v that gets a request message for level i and has not sent an ack yet, now sends an ack. If v did send an ack to some u and now receives a request in (level i) from $w > u$ (w.r.t. to their IDs) then v sends a conditional cancel message to u. If u is already in a level higher than i, then u refuses, and v kills w. Otherwise, u is killed and v sends an ack to w (unless v meanwhile received a request from some $z > w$).

Analysis.

LEMMA 5.11. *If $2^i < n$ and there was a (live) candidate at level i, then there is a candidate at level $i + 1$.*

PROOF. Consider the highest candidate w to reach level i. Suppose that w never reaches level $i + 1$. Since w is the highest, the only way it can get killed is if it sent a request to some node v, and v sent as a result a cancel message to w because some node u (with a smaller ID than w) already reached level $i + 1$. This proves the lemma. □

LEMMA 5.12. *No more than $\frac{n}{2^i}$ candidates reach level i.*

PROOF. Consider a candidate v who reached level $i+1$. Then v received an ack from 2^i nodes. Among above mentioned 2^i nodes, consider some node u. If u had sent an ack to some other node w before sending an ack to v, then u had also sent a cancel message to w and has verified that w stopped being a candidate, before u sent an ack to v. Moreover, if u has not sent an ack to any node after sending to v, since otherwise, this would have meant that v accepted a cancel request and ceased being a candidate before reaching level $i + 1$. Hence, for every candidate v who reached level $i + 1$ there exist a set of 2^i nodes who supported v at the time it increased its level to $i + 1$, and none of them supported any other node w when w increased its own level to $i + 1$. The lemma follows. □

COROLLARY 5.13. *The number of levels is at most $O(\log n)$, the total number of messages sent by nodes on each level is $O(n)$, and the time duration of each level is constant.*

The above lemmas and corollary establish the following theorem:

THEOREM 5.14. *There exist a leader election algorithm for asynchronous networks under simultaneous wake-up that uses $O(\log n)$ time and $O(n \log n)$ messages, if the time is counted from the last spontaneous (adversarial) wake up of a node.*

6 CONCLUSION

The problem of leader election has been introduced a long time ago, and has been heavily investigated ever since. Yet, quite a few significant new results have been discovered very recently. At the time some of the classic results were obtained, it may have seemed that these issues were very well understood. This suggests that it does makes sense to look again even at classic results such as the those of Afek and Gafni (for tradeoffs), Afek and Matias (for randomized algorithms), and Frederikson and Lynch (for methods of generalizing lower bounds from comparison to general algorithms), which are some of the papers that we revisited in this work (and improved some of their results, even though they may have appeared to be tight at the time). It will be interesting to know whether some additional classic results can do with a refresh.

An interesting open question raised by our work is the precise impact of the size of the node ID space on the message complexity of deterministic algorithms for leader election and other problems. As elaborated in Section 1.1, showing lower bounds for an ID space of polynomial size is necessary in order to make the lower bound applicable to the standard CONGEST model. For instance, the work of Awerbuch, Goldreich, Peleg, and Vainish [4] proved that solving single-source broadcast in general networks has a message complexity of $\Omega(m)$, (where m is the number of edges) when assuming an exponentially large ID space, whereas the subsequent work of King, Kutten, and Thorup [11] showed that a message complexity of $\tilde{O}(n)$ is indeed possible for smaller ID spaces. Another open problem is whether we can show a tradeoff lower bound in the asynchronous model, similar in spirit to our result in Section 3.2.

ACKNOWLEDGMENTS

REFERENCES

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

[2] Yehuda Afek and Yossi Matias. 1994. Elections in anonymous networks. *Information and Computation* 113, 2 (1994), 312–330.

[3] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. 2004. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*. 290–299.

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

[5] Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W Richa, and Christian Scheideler. 2015. Leader election and shape formation with self-organizing programmable matter. In *International Workshop on DNA-Based Computers*. Springer, 117–132.

[6] Greg N Frederickson. 1983. Tradeoffs for selection in distributed networks (Preliminary Version). In *Proceedings of the second annual ACM symposium on Principles of distributed computing*. 154–160.

[7] Greg N Frederickson and Nancy A Lynch. 1987. Electing a leader in a synchronous ring. *Journal of the ACM (JACM)* 34, 1 (1987), 98–115.

[8] Saurabh Ganeriwal, Ram Kumar, and Mani B Srivastava. 2003. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*. 138–149.

[9] Pierre A Humblet. 1984. *Selecting a leader in a clique in 0 (N log N) messages*. Laboratory for Information and Decision Systems, Massachusetts Institute of . . . .

[10] Russell Impagliazzo and Valentine Kabanets. 2010. Constructive proofs of concentration bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 617–631.

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

[12] Ephraim Korach, Shay Kutten, and Shlomo Moran. 1990. A modular technique for the design of efficient distributed leader finding algorithms. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 12, 1 (1990), 84–101.

[13] Ephraim Korach, Shlomo Moran, and Shmuel Zaks. 1984. Tight lower and upper bounds for some distributed algorithms for a complete network of processors. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*. 199–207.

[14] Shay Kutten, William K Moses Jr, Gopal Pandurangan, and David Peleg. 2020. Singularly Optimal Randomized Leader Election. In *34th International Symposium on Distributed Computing*.

[15] Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. 2015. On the complexity of universal leader election. *Journal of the ACM (JACM)* 62, 1 (2015), 1–27.

[16] Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. 2015. Sublinear bounds for randomized leader election. *Theoretical Computer Science* 561 (2015), 134–143.

[17] Gérard Le Lann. 1977. Distributed Systems-Towards a Formal Approach.. In *IFIP congress*, Vol. 7. Toronto, 155–160.

[18] Nancy A Lynch. 1996. *Distributed algorithms*. Elsevier.

[19] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. 2010. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review* 44, 3 (2010), 2–19.

[20] David Peleg. 2000. *Distributed computing: a locality-sensitive approach*. SIAM.

[21] Murali Krishna Ramanathan, Ronaldo A Ferreira, Suresh Jagannathan, Ananth Grama, and Wojciech Szpankowski. 2007. Randomized leader election. *Distributed Computing* 19, 5 (2007), 403–418.

[22] Gurdip Singh. 1992. Leader election in complete networks. In *Proceedings of the eleventh annual ACM symposium on Principles of distributed computing*. 179–190.

[23] Yong Yao and Johannes Gehrke. 2002. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod record* 31, 3 (2002), 9–18.