Scalable and Secure Computation Among Strangers: Message-Competitive Byzantine Protocols

John Augustine
Dept. of Computer Science & Engg, Indian Institute of Technology Madras, Chennai 600036, India
augustine@iitm.ac.in

Valerie King
Department of Computer Science, University of Victoria, Vancouver BC V8P 5C2, Canada
val@uvic.edu

Anisur Rahaman Molla
Computer and Communication Sciences, Indian Statistical Institute, Kolkata 700108, India
molla@isical.ac.in

Gopal Pandurangan
Department of Computer Science, University of Houston, Houston, TX 77204, USA
gopalpandurangan@gmail.com

Jared Saia
Department of Computer Science, University of New Mexico, NM 87131, USA
saia@cs.unm.edu

Abstract

The last decade has seen substantial progress on designing Byzantine agreement algorithms which do not require all-to-all communication. However, these protocols do require each node to play a particular role determined by its ID. Motivated by the rise of permissionless systems such as Bitcoin, where nodes can join and leave at will, we extend this research to a more practical model where initially, each node does not know the identity of its neighbors. In particular, a node can send to new destinations only by sending to random (or arbitrary) nodes, or responding to messages received from those destinations. We assume a synchronous and fully-connected network, with a full-information, but static Byzantine adversary. A major drawback of existing Byzantine protocols in this setting is that they have at least $\Omega(n^2)$ message complexity, where n is the total number of nodes. In particular, the communication cost incurred by the honest nodes is $\Omega(n^2)$, even when Byzantine node send no messages. In this paper, we design protocols for fundamental problems which are *message-competitive*, i.e., the total number of bits sent by honest nodes is not significantly more than the total sent by Byzantine nodes.

We describe a message-competitive algorithm to solve Byzantine agreement, leader election, and committee election. Our algorithm sends an expected $O((T+n)\log n)$ bits and has latency $O(\text{polylog}(n))$ (even in the CONGEST model), where $T = O(n^2)$ is the number of bits sent by Byzantine nodes.[1] The algorithm is resilient to $(\frac{1}{4} - \epsilon)n$ Byzantine nodes for any fixed $\epsilon > 0$, and succeeds with high probability.[2] Our message bounds are essentially optimal up to polylagarithmic factors, for algorithms that run in polylogarithmic rounds in the CONGEST model.

We also show lower bounds for message-competitive Byzantine agreement regardless of rounds. We prove that, in general, one cannot hope to design Byzantine protocols that have communication cost that is significantly smaller than the cost of the Byzantine adversary.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Probabilistic algorithms; Mathematics of computing → Discrete mathematics

[1] Our algorithm will never send more $O(n^2 \log n)$ bits if $T \geq n^2$.
[2] Throughout, "with high probability" means with probability at least $1 - n^{-c}$ for some constant $c > 0$.

34th International Symposium on Distributed Computing (DISC 2020).
Editor: Hagit Attiya; Article No. 31; pp. 31:1–31:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases Byzantine protocols, Byzantine agreement, Leader election, Committee election, Message-competitive protocol, Randomized protocol

Digital Object Identifier 10.4230/LIPIcs.DISC.2020.31

Related Version A full version of the paper is available at `https://arxiv.org/abs/1907.10308`.

Funding *John Augustine*: Research supported in part by an Extra-Mural Research Grant (file number EMR/2016/003016) and a MATRICS grant (file number MTR/2018/001198), both funded by the Science and Engineering Research Board, Department of Science and Technology, Government of India and by the VAJRA faculty program of the Government of India.
Valerie King: This work is supported by an NSERC Discovery Grant.
Anisur Rahaman Molla: Research supported by DST Inspire Faculty research grant DST/INSPIRE/04/2015/002801 and ISI DCSW/TAC Project (file number E5412).
Gopal Pandurangan: Research supported, in part, by NSF grants CCF-1527867, CCF-1540512, IIS-1633720, CCF-BSF-1717075, BSF award 2016419, and by the VAJRA faculty program of the Government of India.
Jared Saia: This work is supported by the National Science Foundation grants CNS-1318880 and CCF-1320994.

1 Introduction

What happens when you don't know your neighbors? Permissionless systems, such as cryptocurrency [12, 23], anonymous communication [27, 63], and wireless [44, 48, 64, 61], allow nodes to join and leave with little or no admission control. In such systems, nodes are generally known only by self-generated identifiers[3]; and communication primitives may be limited to: sending a message to all nodes, sending a message to a random (or arbitrary) node, and responding to a message sent directly. Unfortunately, all algorithms to coordinate such networks in the presence of malicious faults seem either to require all-to-all communication, or make cryptographic assumptions.

A major challenge in permissionless systems is dealing with malicious, or *Byzantine* nodes, which can try to foil the protocols executed by honest, or *good* nodes. Byzantine-resistant protocols are at the heart of secure systems. Consider the example of Bitcoin – a decentralized, digital currency [12]. A crucial problem faced by Bitcoin is fault-tolerant agreement on a set of ordered transactions.

The problem of achieving agreement under Byzantine faults, *Byzantine agreement*, is a fundamental and long-studied problem in distributed computing [56, 7, 50]. In this problem, all good nodes start with an input bit, and we must ensure: (1) *All* good nodes output the same input bit (*consensus condition*); and (2) this common bit is the input bit of some good node (*validity condition*). This must be done despite the presence of a constant fraction of Byzantine nodes that can deviate arbitrarily from the protocol executed by the good nodes. Byzantine agreement is a "keystone" problem in distributed computing, in that it provides a critical building block for creating attack-resistant distributed systems. It has been used in many domains including: sensor networks [60], grid computing [6], peer-to-peer networks [59] and cloud computing [65]. However, despite intensive research, there is still no practical solution to Byzantine agreement for large networks. A main reason for this is the *large message complexity* of currently known protocols, as has been suggested by

[3] Such as the public key for a digital signature.

many systems papers [3, 5, 17, 51, 66]. The best known Byzantine protocols have (at least) quadratic message complexity, i.e., $\Theta(n^2)$, where n is the number of nodes in the network (e.g., [32, 9, 26, 46]). This is especially true of protocols that run fast, i.e., in $O(\text{polylog}\, n)$ rounds (e.g.,[32, 9]). As noted in many papers [1, 38, 34] the message complexity plays an important role in performance.

King and Saia [40] described the first Byzantine agreement algorithm for *synchronous, complete networks* that *breaks* the quadratic message barrier *under the assumption that nodes a priori know the identities of all their neighbors*. This assumption is called the KT_1 model [57]. A more challenging model is KT_0 (also called the *clean network* model [57]), where nodes do not know the identity of their neighbors a priori, but do learn a node's identity upon receiving a message from it. In the KT_1 model, [40] presented an algorithm where *each* good node sends only $\tilde{O}(\sqrt{n})$ messages, and thus total message complexity is $\tilde{O}(n^{1.5})$. Braud-Santoni et al. [14] improved this to $O(n \text{ polylog}(n))$ total message complexity, however, their protocol might require some node to send $O(n)$ messages.

The KT_0 model is more applicable to permissionless networks, where nodes enter and leave at will, and hence it is not reasonable to assume that nodes a priori know the identities of all other nodes in the system. We can convert algorithms for KT_1 to KT_0 by including an initial step where each node communicates with all its neighbors to obtain their identities, but this incurs a $\Theta(n^2)$ message cost. It is better to avoid such costly, potentially all-to-all, communication cost. Hence a fundamental question is:

Can we design Byzantine protocols that require sub-quadratic messages in KT_0?

In this paper, we address the above question. Our focus is on the fundamental problems of Byzantine agreement, leader election and committee election. Our main result (Theorem 1) is an algorithm to solve these problems while sending a number of bits that is $O((T + n) \log n)$, where T is *the number of bits sent by Byzantine nodes*, and n is the network size.[4] We show (Theorem 2) that this is essentially the best possible bound if one desires fast (i.e., polylogarithmic rounds) algorithms.

To the best of our knowledge, our result introduces *message-competitive analysis* to the study of Byzantine protocols. In particular, our algorithm is *message-competitive* in the sense that the number of messages sent by good nodes competes well with the number sent by Byzantine nodes; if Byzantine nodes send fewer messages then our algorithm also sends fewer. An alternate way to interpret our result is that Byzantine nodes have to incur significant message complexity (up to quadratic in n) in order to make the good nodes to have large message complexity. This kind of result where algorithmic cost is measured with respect to adversarial cost, is an example of *resource-competitive analysis* [11, 30].

Our work can be considered as an improvement in a long line of work that focuses on designing message-optimal Byzantine protocols, see e.g., [38] and the references therein and the recent work of [1]. We note that prior work on Byzantine protocols all incurred at least quadratic message complexity (in KT_0), regardless of the behavior of the Byzantine nodes. One exception is the result of Hadzilacos and Halpern [38] (see also [22]) that gives a message bound of $O(nt)$ (deterministically) when there are t Byzantine nodes; however this protocol takes $O(n)$ rounds. This protocol's message complexity is proportional to the number of Byzantine nodes, but not to the total number of messages sent by them. Our protocol is more general and (essentially) subsumes[5] the protocol of [38], while being significantly faster. It is significantly more message-efficient when t is large and T is small.

[4] Our algorithm sends at most $O(n^2 \log n)$ bits if $T \geq n^2$.

[5] Note that if there are t Byzantine nodes our protocol has a message complexity of $O(nt \log n)$ with high probability, which is a logarithmic factor larger than the $O(nt)$ bound of [38].

The lower bound results of Hadzilacos and Halpern [38] imply that our protocol is the best possible if one wants fast algorithms (i.e., finishing in polylogarithmic rounds) in the CONGEST model, where good nodes send only small-sized messages, i.e., $O(\log n)$ bits per edge per round. Our protocol is also lightweight and fast (has low latency) and can be used as a building block for designing secure and scalable (where communication and latency scales efficiently with network size) systems.

1.1 Model

We consider a network of n nodes. There are $t \leq (\frac{1}{4} - \epsilon_0)n$ *bad* nodes which are controlled by the adversary, for fixed $\epsilon_0 > 0$. The remaining nodes are *good* and follow our algorithm.

We consider a synchronous, fully-connected network in the KT_0 model [55, 57]. In particular, each node has ports to every other node in the network, but learns the identity of each node reachable through a port only by receiving a communication from that node. Thus a node sends to a new destination only by selecting a port, or by responding to messages received. We note that our algorithm technically only requires two primitives to send to unknown nodes: the ability to write to (1) a random unknown ID; and (2) all unknown IDs. Thus, it may be useful for models beyond KT_0, such as a gossiping-based communication model [47].

The n nodes are assumed to have distinct ID's which lie in $[1, n^k]$ for k is a (large) constant.[6] Our adversary is full-information in that it knows the states of all nodes at any time, is assumed to be computationally unbounded, and is also *rushing* in the sense that it can read messages sent by good nodes before sending out its own messages. However, the adversary is *static*, so that it must decide which nodes are bad prior to the start of the algorithm. A Byzantine node can choose its identity initially, but once chosen, that identity is presented to all nodes which receive messages from the Byzantine node. We expect this last requirement to useful in conjunction with algorithms in [35, 36, 37] that require some effort, such as solving a computational puzzle, in order to create a new identity.

We seek to design algorithms with low latency, i.e., the number of rounds until termination, and low message complexity, i.e., the total number of messages sent by good nodes.

1.2 Our Contributions

We solve three classic problems in this model. In *Byzantine agreement*, all good nodes must output the same bit, which is the input bit of some good node. In *leader election*, all good nodes must agree on a leader, and this leader must be good with constant probability. In *committee election*, all nodes must agree on a subset of $O(\log n)$ nodes where the fraction of bad nodes in the subset is within a small ϵ fraction of the overall fraction of bad nodes.

Our main result is as follows.

▶ **Theorem 1.** *There exists a randomized algorithm that solves Byzantine agreement, leader election and committee election in the above model. This algorithm sends an expected $O((T + n) \log n)$ messages, and has latency $O(\text{polylog}(n))$, where T is the minimum of n^2 and the total number of bits sent by the bad nodes to good nodes. (If $T \geq n^2$, then the algorithm sends $O(n^2 \log n)$ bits.) It is resilient to $t \leq (\frac{1}{4} - \epsilon_0)n$ Byzantine faults for any fixed $\epsilon_0 > 0$, and succeeds with probability $1 - 1/n^c$ for any constant c.*

[6] This means that an ID can be represented using $O(\log n)$ bits, which can be sent in a message.

We note that our $O(\text{polylog}(n))$ latency bound holds even in the CONGEST model, where each message is $O(\log n)$ bits. The algorithm *KT0-ByzantineAgreement* described in Section 3 achieves the result in Theorem 1, and the proof of this theorem is in Section 4.

The interesting regime for T is subquadratic, where our algorithm sends only subquadratic messages (actually proportional to T), unlike prior works (cf. Section 1) that incurred quadratic message complexity in general. Our algorithm is resilient up to a *constant* fraction of nodes – up to essentially $n/4$ – being bad. It is a open question whether one can improve this tolerance further up to $n/3$ bad nodes, which is the best possible[56].

As mentioned in Section 1, the work of Hadzilacos and Halpern [38] shows a tight bound of $\Theta(nt)$ for message complexity[7] of Byzantine agreement with $t \leq n$ Byzantine nodes (t is unknown). It gives a deterministic algorithm and a lower bound proof that holds for *Monte Carlo randomized algorithms that succeed with high probability* as well. If one desires fast, i.e., $\text{polylog}(n)$ rounds algorithms (as is the case with our randomized algorithm), then the above $\Theta(nt)$ bound shows that our message complexity is essentially the best possible in general. This is because, since in each round at most $\tilde{O}(n)$ bits can be sent by a Byzantine node to good nodes in the CONGEST model[8], and since the number of rounds is bounded by $O(\text{polylog}\, n)$, the number of Byzantine nodes t is $\tilde{\Omega}(T/n)$, where T is the total number of bits sent by Byzantine nodes to good nodes.[9] By the $\Omega(nt)$ lower bound of [38], we have the following lower bound theorem.

▶ **Theorem 2** (follows from [38]). *Let $T \in [\Theta(n), \Theta(n^2)]$ be the total number of bits sent by the Byzantine nodes to good nodes. Then any Byzantine agreement algorithm (including randomized Monte Carlo algorithms that succeed with high probability) that finishes in* $\text{polylog}(n)$ *rounds in the CONGEST model needs, in general, at least $\tilde{\Omega}(T)$ messages in expectation.*

The above theorem implies that our randomized algorithm with message complexity of $O((T + n)\log n)$ and $O(\text{polylog}\, n)$ latency is essentially optimal in CONGEST.

To the best of our knowledge, our protocol is the first that is message-competitive. As discussed above, all prior protocols (excepting [38] and [22]) require at least quadratic messages, regardless of the behavior of Byzantine nodes. This is especially true for protocols that take small number of rounds (e.g., [9, 32]). Algorithms sending $O(nt)$ messages [22, 38] have linear latency. One can view these upper bounds that depend on t as a special case of our result. Our message-competitive bound is more general, in the sense, it is proportional to the total number of messages sent by Byzantine nodes. The case when a large number of Byzantine nodes send a small number of messages is not captured in the prior bounds.

We also show lower bounds for message-competitive Byzantine agreement that hold regardless of rounds (see Section 6). We prove that, in general, one cannot hope to design Byzantine protocols that have communication cost that is significantly smaller than the communication cost of the Byzantine adversary. We first show a lower bound for *deterministic* BA protocols which is essentially tight with respect to our randomized algorithm (see Section 6.1). We show that if $T = O(n^2)$ is the budget on the message bits of the Byzantine nodes, then for any deterministic protocol, the total number of messages sent by the good nodes is $\Omega(T)$ (see Theorem 12). The deterministic lower bound holds even in the KT_1 model. We

[7] This bound holds even for messages of size one bit.
[8] We only consider algorithms where Byzantine nodes follow the CONGEST bound. Otherwise, if a Byzantine node sends $\omega(\log n)$ bits to a good node, it will be ignored.
[9] The $\tilde{O}$ notation hides a $\text{polylog}(n)$ factor and $\tilde{\Omega}$ notation hides a $1/\text{polylog}(n)$ factor.

then show a somewhat weaker lower bound on the message competitiveness of randomized Las Vegas (that always succeed) BA protocols (see Section 6.2) where we assume Byzantine nodes can fake their IDs. The argument for the randomized case is more involved compared to the deterministic case, as the algorithm's (future) random choices are unknown to the Byzantine adversary. We show that if $T = n^{1+\alpha}$ for some $\alpha \in (0, 1]$ is the budget of the Byzantine nodes, then for any (randomized) BA algorithm in the KT_0 setting, the expected number of messages sent by good nodes, is at least $\Omega(n^{1+\frac{\alpha}{2}})$ (see Theorem 14).

All omitted proofs and additional details will be given in the full paper [8].

1.3 Related Work

Message-Competitive Analysis. This paper introduces *message-competitive analysis* which can be considered as a special case of the more general *resource-competitive* analysis [11, 30] to the study of Byzantine agreement. In resource-competitive analysis, the computational cost of the attacker, T, is incorporated as a parameter in performance analysis. That is, the cost of executing an algorithm over a network of n nodes is measured not only as a function of n, but also as a function of T. Messages are an important resources and as mentioned in [38], "the number of messages used by a protocol is important, possibly the most important, factor that determines its performance."

Resource competitive analysis has been applied to designing algorithms for: jamming-resistant wireless communication [29, 31, 43]; attack-resistance on multiple access channels [10], tolerating adversarial channel noise [4, 20, 21], and efficiently distributing bridges for anonymity networks such as TOR [67]. See [11, 30] for detailed surveys.

Communication Efficient Byzantine Agreement and Leader Election. Byzantine agreement enables participants in a distributed network to reach agreement on a decision, even in the presence of a malicious minority. Thus, it is a fundamental building block for many applications including: cryptocurrencies [13, 24, 28, 33]; trustworthy computing [15, 16, 17, 18, 19, 45, 62]; peer-to-peer networks [2, 54]; and databases [53, 58, 68].

In 2006, King, Saia, Sanwalani, and Vee [41] gave a (randomized) algorithm to solve Byzantine agreement, leader election and committee election problems in a model differing from the one in this paper only in the assumption of KT_1 communication. This was the first algorithm to use only $\tilde{O}(1)$ bits of communication per node, and $\tilde{O}(1)$ time to bring almost all processors to agreement. This result can also be achieved in a particular sparse network [42]. This initial work produced agreement among all but $o(n)$ nodes. Further work extended this result to achieve everywhere agreement, while using a number of bits that is $\tilde{O}(n^{3/2})$ (load-balanced) [39]; and $\tilde{O}(n)$ (not load-balanced) [14]. All of these algorithms required each node to play a particular role as determined by its unique ID in $[1, n]$, and to send to specific neighbors. In other words, these algorithms critically rely on the KT_1 model. These bounds hold even if the bad nodes send any number of bits. Establishing Byzantine agreement via the use of committees is a common approach; for examples, see [28, 41, 49]. Recent work by Abraham et al. [1] revisits the problem of communication efficient Byzantine agreement in the KT_1 model. They show that achieving sub-quadratic message cost with an adaptive adversary in this model requires that the adversary not have the ability to erase messages already sent by the nodes it adaptively takes over. More related work is given in full paper.

2 High-level Overview of Algorithms and Techniques

We focus first on Byzantine agreement, our solutions to leader and committee election use similar techniques. Our algorithm depends on solutions to two new problems: *Implicit Agreement* and *Promise Agreement*. In the *Implicit Agreement* problem, success means that strictly greater than a t/n fraction of good nodes decide on the same (correct) bit and the remaining good nodes do not decide; and failure means that no good nodes decide. Next, the *Promise Agreement* problem assumes there has first been either success or failure in *Implicit Agreement*. In the case of success, *Promise Agreement* ensures all nodes decide on the same value and terminate; in the case of failure, no nodes decide.

KT0-ByzantineAgreement runs in epochs. In each epoch, we (1) run an algorithm for *Implicit Agreement*; (2) run an algorithm for *Promise Agreement*; and (3) terminate in the case of success, or increase the number of messages sent in the case of failure.

The number of messages sent for *Implicit Agreement* is tuned by increasing the number of *active* nodes. In particular, during a run of *Implicit Agreement*, the active nodes first attempt to solve Byzantine agreement among themselves, and then to communicate the output to all other nodes in the network. Our *Implicit Agreement* algorithm ensures that, unless the bad nodes send a number of messages that is n times the number of active nodes, then *Implicit Agreement* will succeed. Next, we solve *Promise Agreement*. This ensures that if *Implicit Agreement* succeeded, then all nodes will decide on the same value and terminate; and if *Implicit Agreement* failed, then no nodes decide. In the latter case, all nodes proceed to the next epoch, where the number of active nodes doubles in expectation.

When there is partial knowledge of participants. We say that a node x has a view of node y if x knows y's ID and the port to y. With a fair amount of technical work, we show that it is possible to modify an algorithm by King et al. [41] to ensure agreement even among nodes whose views only "mostly" overlap, provided that the range of all IDs is only polynomially large. We call this modified algorithm *LargeCoreBA*, and summarize its properties in Lemma 3 below; we believe the result may be of independent interest.

▶ **Lemma 3.** *Let G be a set of good nodes which wish to come to agreement. For each $x \in G$, let S_x be the set of nodes in the view of x. Let B be the set of bad nodes in $\bigcup_{x\in G} S_x$. Assume $G \subseteq \bigcap_x S_x$; $|B| \leq (1-\epsilon)|G|/2$ for some fixed constant $\epsilon > 0$; and all nodes have distinct ID's in $[1, n^k]$. Then there is an algorithm* LargeCoreBA *which computes almost everywhere agreement (i.e., computes agreement among $(1 - 1/\log n)$ fraction of nodes in G) with high probability in time and communication per node which is polylogarithmic in $|G| + |B|$. In one more round, if each good node broadcasts to all other nodes, and then each node takes the majority, all nodes will come to agreement using $|G|(|G| + |B|)$ total messages, and latency polylogarithmic in $|G| + |B|$.*

Implicit Agreement. Our solution to *Implicit Agreement* is given in Steps 1 to 6 of our main algorithm in Section 3.1. There are two key technical problems that must be addressed.

First, how do we ensure that each active node x maintains a set S_x so that the conditions of Lemma 3 are matched? Also, in order to achieve a good competitive ratio, we need the conditions of Lemma 3 to hold unless the adversary sends $\Omega(nA)$ messages, where A is the number of active nodes. If each active node x naively adds to S_x all nodes y that it receives an initial message from, then the adversary can add A Byzantine nodes to each S_x while sending only A^2 messages. Thus, we must enlist the aid of non-active nodes to establish the S_x sets. Initially, each active node sends its ID to *all* nodes. Call a good node *light* if it has

received a number of IDs approximately equal to A. Then the light nodes convey information about their S_x sets to the nodes in S_x. They cannot send out *all* the IDs in S_x, since that would be too many bits. Instead, they just send out a single random ID, and a node y adds an ID to S_y if it was received from at least enough (β) nodes that claim to be light.

Unfortunately, an adversary can still cause problems by making the size of the union of the bad nodes in each S_x large, so that $|B|$ is large in Lemma 3, even when the adversary does not send out too many messages. To solve this problem, we use a "validation" step, whereby each active node, for each ID in S_x, queries $\Theta(\log n)$ random nodes about whether they have the ID in their S_x sets, and filters out the ID unless enough of these queries are answered affirmatively. Based on information obtained during this process (Step 1 through Step 3c in Section 3.1), the active nodes determine if the number of light nodes is sufficient for favorable success in this epoch.

This brings us to the second problem. How can the active nodes agree on one of two options for this epoch: (1) conditions are favorable for agreement; or (2) conditions are not favorable? We can make use of *LargeCoreBA* in coming to agreement on an option. However, this is still challenging given that, under certain conditions, some active nodes may run *LargeCoreBA*, while other active nodes may not even have a small enough S_x set to run it. To address this issue requires careful decisions about whether a node will run *LargeCoreBA*, what its input will be, and whether or not it will trust the output, all based on the node's estimate of the number of light nodes (See Step 4, Section 3.1 for details). In particular, nodes will sometimes run *LargeCoreBA*, because other nodes are relying on them to do so, even when they plan to ignore the output. If active nodes decide conditions are favorable via the first call to *LargeCoreBA* (Step 4), they will all run it again (Step 5) to decide on a bit. Lemma 7 in Section 4 shows that no matter what the number of light nodes, these two steps ensure all active nodes come to agreement on the same decision.

Finally, in Step 6, active nodes send their decision to all other nodes. Nodes that have small S_x sets take the majority of the messages received in this step, whereas other nodes default to a decision to wait for the next epoch. We can thereby guarantee the post-condition for *Implicit Agreement*: either (1) a strictly greater than t/n fraction of good nodes decide, or (2) no good nodes decide. We obtain this result even when the adversary floods some good nodes but not others.

Promise Agreement. A final technical challenge is to determine whether or not we need to run another epoch. After solving *Implicit Agreement*, either (1) strictly greater than a t/n fraction of the good nodes have decided on the same correct bit; or (2) no good nodes have decided. We must then ensure that *all* good nodes decide either to terminate or to run another epoch. To do this, we run an algorithm, *PromiseAgreement* that solves the *Promise Agreement* problem (see Section 5.2). The solution simply has each node sample a logarithmic number of other nodes, and take a majority vote. It does not increase the overall asymptotic number of messages sent, but some non-active nodes can be forced by the adversary to respond to $O(n)$ requests. If the outcome of *PromiseAgreement* is not agreement then all nodes proceed to the next epoch, where the number of active nodes doubles in expectation. In this way, we can guarantee that that *KT0-ByzantineAgreement* succeeds within $\log(n)$ expected epochs.

3 KT0-ByzantineAgreement

The overview and intuition for the steps of the main algorithm *KT0-ByzantineAgreement* are described in Section 2. Here, we give its pseudocode and define the problem *Promise Agreement*. *KT0-ByzantineAgreement* calls *LargeCoreBA* and an algorithm *PromiseAgreement* that solves *Promise Agreement*. A node x calls *LargeCoreBA* with a set of possible participants S_x, which may include nodes which do not themselves participate.

The algorithm below runs correctly with probability $1 - 1/n^c$ for any constant c, when constant C below is chosen to be sufficiently large, depending on c. We let ϵ be a small constant such that $0 < \epsilon < \epsilon_0^2$. We set $max_a = (1+\epsilon)p(n-t)$ and $min_a = (1-\epsilon)p(n-t)$ so that w.h.p. the number of *active* nodes lies in this range.

We call a good node *active* if it sets its state to active in Step 2. We call a good node *light* if the number of IDs received by it from alleged *active* nodes in Step 2 is less than $max_a + \epsilon pn$. We use bounds $Low = n - 2t - \epsilon n$ and $High = Low + t$ to describe the number of light and purported light nodes. For $p > 1/(C \log n)$, if there are at least $Low - t$ light nodes and each sends a random ID from their list of nodes that reported being active in Step 2, then w.h.p., at least $\beta = \frac{(1-\epsilon)(Low-t)}{max_a+\epsilon pn}$ copies of all their common IDs, in particular, the IDs of all *active* nodes, will be received by every *active* node. Finally, an element in an *active* node x's set S_x is validated when x queries a random set of $C \log n$ nodes and $\delta C \log n$ nodes respond yes. $\delta = \frac{(1-\epsilon)(Low-t)}{n}$ is chosen so that w.h.p., every ID in *active* will be validated but not many ID's of nodes which are bad.

3.1 Pseudocode for KT0-ByzantineAgreement

1. **Initialize:** Every node x sets $p \leftarrow (C \log n)/n$. Each node x sets $\textit{ready-out}_x \leftarrow 0$, $\textit{ready-in}_x \leftarrow 0$, and sets its state to $\neg$*active* and $\neg$*light*.
2. **Nodes become *active* and notify others:** With probability p, x sets its state to *active* and sends its ID to all nodes. Every node x sets S_x to the set of IDs received. A node sets its state to *light* if $|S_x| \leq max_a + \epsilon pn$.
3. ***Active* nodes learn of other *active* nodes:**[10]
   a. Every *light* node x randomly selects an ID in S_x and sends it to the nodes in S_x.
   b. Every active node x sets n_x to be the number of nodes which sent to x in Step 3a. If $n_x \geq Low - t$ then x resets S_x to be the set of IDs which were received from at least β nodes in step 3a. For each ID in S_x, x sends the query $< ID? >$ to a random set of $C \log n$ nodes.
   c. Every light node x answers a query $< ID? >$ if ID is in S_x and the query is sent by a node in S_x. An ID in S_x is considered *validated* if x received at least $\delta C \log n$ responses to the query for ID. Each *active* node x that sent queries removes from S_x all IDs which are not validated.
4. **Can we proceed?** Each *active* node x with $n_x \geq Low - t$ runs *LargeCoreBA* with the other nodes in S_x. The input bit to *LargeCoreBA*, $\textit{ready-in}_x \leftarrow 1$ iff $n_x \geq High$. If $n_x \geq Low$ then $\textit{ready-out}_x \leftarrow$ output of *LargeCoreBA*.
5. **Compute Byzantine Agreement** Each *active* node x with $\textit{ready-out}_x = 1$ runs *LargeCoreBA* with nodes in S_x, with input bit, $value_x$, set to the node's initial input bit.
   Node x then sets $value_x$ to the output of this *LargeCoreBA*.

[10] Note: The S_x for inactive nodes x are unaffected by this step

6. **Take Majority:** Each active node, x, sends $(ready\text{-}out_x, value_x)$ to all nodes. Then, each node x with $n_x \geq Low - t$ sets $ready\text{-}out_x$ to the majority *ready-out* bit received from nodes in S_x. If this bit is 1, then $value_x$ is set to the majority *value* bit received from nodes in S_x.
7. **Promise Agreement:** Each node x runs PROMISEAGREEMENT with the tuple $(ready\text{-}out_x, value_x)$, and resets the tuple based on the outcome.
   a. If $ready\text{-}out_x = 1$, then node x terminates and outputs value $value_x$;
   b. Else if $p < 1/(C \log n)$, then p doubles and x repeats from Step 2.
   c. Else (i.e., when $pn \geq n/(C \log n)$), every node sends to all other nodes to determine their IDs, and then the protocol resorts to running LARGECOREBA.

3.2 Promise Agreement

Here we define a variant of the almost-everywhere to everywhere Byzantine agreement problem, which we call *Promise Agreement*. In Section 5.2, we describe an algorithm, PROMISEAGREEMENT, to solve this problem.

▶ **Definition 4.** *An algorithm is said to solve the* Promise Agreement *problem if it has the following properties.*

1. *If (i) there is at least a* $t/n+2\epsilon$ *fraction of good nodes with tuple* $(ready\text{-}out, value) = (1, v)$*, for the same bit* v*; and (ii) all remaining good nodes have ready-out value of* 0*, then all nodes terminate with tuple* $(ready\text{-}out, value) = (1, v)$*.*
2. *If all good nodes have* $ready\text{-}out = 0$*, then all nodes terminate with* $ready\text{-}out = 0$*.*

4 Analysis of KT0-ByzantineAgreement

4.1 Correctness

We call one run of all the steps in the KT0-BYZANTINEAGREEMENT algorithm an epoch. We assume $t \leq (\frac{1}{4} - \epsilon_0)n$ for a fixed $\epsilon_0 > 0$. We note that $p < 1/(C \log n)$ except in Step 7c.

▶ **Lemma 5.** *The following events occur w.h.p. in* n*.*

1. *The number of* active *nodes is between* min_a *and* max_a*.*
2. *If there are at least* $Low - t$ *light nodes, then all active nodes receive at least* β *copies of the ID of every* active *node in Step 3a.*
3. *If there are at least* $Low - t$ *light nodes, then all* active *nodes will consider all IDs of* active *nodes validated after Step 3c.*
4. *If an ID is contained in the* S_x *sets of at most* $(1 - \epsilon)(\delta - t/n)n$ *light nodes in Step 3b, then that ID will not be validated.*

Proof. For each of these items there is a random variable X which is the number of successful independent trials. In each case, we will show that $E[X] \geq C' \log n$ for some constant C'. Then, Chernoff bounds imply that $Pr(|X - E[X]| \geq \lambda E[X]) \leq n^{-c}$, for any fixed $\lambda < 1$, and any fixed c, for C' sufficiently large [52]. The details are deferred to the full paper [8]. ◀

For a fixed epoch, let *CORE* be the set of active nodes that run LARGECOREBA in Step 4. We show that the nodes participating in LARGECOREBA have the desired properties to successfully complete it when there are at least $Low - t$ light nodes. (See Lemma 3.)

From Lemma 5, we can observe the following.

▶ **Lemma 6.** *If there are at least $Low - t$ light nodes then w.h.p., we have the following*

1. *Every* active *node is in the CORE, and therefore $|CORE| \geq min_a$.*
2. *For all $x \in CORE$, $CORE \subseteq S_x$ after Step 3c.*
3. *Let B be the bad nodes in $\bigcup_{x \in CORE} S_x$. At the conclusion of Step 3, if there are at least $Low - t$ light nodes, $|B| \leq \epsilon' pn$ for any $\epsilon' > 0$, and $\frac{|B|}{|CORE|} \leq 1/2 - \epsilon''$ for any $\epsilon'' > 0$.*

The proof of this lemma is deferred to the full paper [8]. Lemma 6 and Lemma 3 imply that LargeCoreBA can be successfully run when there are at least $Low - t$ light nodes.

▶ **Lemma 7.** *Let L be the number of light nodes in an epoch of* KT0-ByzantineAgreement. *Then w.h.p.,*

1. *If $High \leq L$,*
   1) All active *nodes have ready-in $= 1$, they run* LargeCoreBA *and decide on ready-out $= 1$ when run in Step 4; and*
   2) All active *nodes y run* LargeCoreBA *in Step 5 and set their value bit to the input bit $value_x$ of some* active *node x.*
2. *If $Low \leq L < High$,*
   1) All active *nodes successfully run* LargeCoreBA *but they may start with differing values for ready-out in Step 4.*
   2) If the output is a 1, all active *nodes y set ready-out $= 1$ and they will successfully run* LargeCoreBA *in Step 5 and set $value_y$ to the input $value_x$ for some* active *node x.*
   3) If the output is a 0, all active *nodes set ready-out $= 0$.*
3. *If $Low - t \leq L < Low$, all* active *nodes will successfully run* LargeCoreBA *in Step 4, though some nodes will disregard the output. All* active *nodes will start with ready-in $= 0$ and all* active *nodes will have ready-out $= 0$.*
4. *If $L < Low - t$, some* active *nodes may run a possibly flawed* LargeCoreBA *in Step 4, though all* active *nodes will disregard the output. All* active *nodes will start with ready-in $= 0$ and end with ready-out $= 0$.*

Proof. If $L < Low - t$, then $n_x < Low$ for all nodes x, thus in Step 4, all *active* nodes have *ready-in* $= 0$, disregard the output of LargeCoreBA, and set *ready-out* $= 0$.

If $Low > L \geq Low - t$, by Lemmas 6 and 3, LargeCoreBA will run successfully. All *active* nodes x have $n_x < High = Low + t$, so in Step 4, all *active* nodes x have $ready\text{-}in_x = 0$. Thus, by the consistency property of LargeCoreBA, all *active* nodes x have $ready\text{-}out_x = 0$.

If $Low \leq L < High$, then all active nodes running LargeCoreBA in Step 4 may start with different *ready-in* values, but by the correctness of LargeCoreBA, they will all end with the same *ready-out* value. If the *ready-out* vale is 1, in Step 5, LargeCoreBA will run correctly and they will all set their *value* bit to the input bit, $value_x$ of some *active* node x.

If $L \geq High$, then any *active* node x has $n_x \geq High$, and so has $ready\text{-}in_x = 1$. Thus, after Step 4, by the validity of LargeCoreBA, all active nodes will have *ready-out* $= 1$. Thus, they will all run LargeCoreBA in Step 5 and will all set their value bit to the input value bit of some *active* node. ◀

▶ **Lemma 8.** *At the end of each epoch, w.h.p., all nodes either terminate and output the same value or they all go to the next epoch.*

Proof. By Lemma 7, if any *active* node x sets *ready-out* $= 1$ after Step 5, all *active* nodes will set their tuple $(ready\text{-}out, value)$ to the value $(1, v)$, and v will be the input bit of some node in *CORE*. Moreover, there must be at least *Low* light nodes. Since every light node y has at least min_a IDs of *active* nodes in S_y, and $|S_y| \leq max_a + \epsilon n$, in Step 6, the majority of the messages received from nodes with IDs in S_y will be $(1, v)$ and y will set *ready-out* $= 1$

and $value_y = v$. Since $Low = n - 2t + \epsilon \geq t + 2\epsilon$, all good nodes will come to agreement on $(1, v)$ in Step 7, when the Promise Agreement problem is solved correctly (by Lemma 11 in Section 5.2).

On the other hand, if any *active* node x sets their value $ready\text{-}out_x$ to 0, then we must be in Case 2, 3 or 4 of Lemma 7. In these cases, all *active* nodes have $ready\text{-}out = 0$, at the end of Step 5. Thus, all light nodes set $ready\text{-}out = 0$ since it is the majority value received in Step 6, and all nodes which are not light do not change their initial *ready-out* value from 0. Therefore, all nodes agree on $ready\text{-}out = 0$. With $ready\text{-}out = 0$, all nodes execute Steps 7b or 7c, depending on the value of p. ◀

4.2 Message Costs

▶ **Lemma 9.** *In any epoch, w.h.p., the algorithm sends $O((pn)^2 \log n + pn^2 + n \log n + T_e)$ messages, where T_e is the minimum of n^2 and the number of messages sent by bad nodes in that epoch. Moreover, in any epoch, the algorithm takes time polylogarithmic in n.*

Proof. There are $O(pn)$ *active* nodes which send to all nodes and each light node sends one message to $O(pn)$ nodes, for a total of $O(pn^2)$ messages. When S_x is reset, it is reset to be no larger than $n/\beta = O(np)$. To validate its S_x, each *active* node sends $O(\log n)$ messages for each element in S_x. There are $O(pn)$ *active* nodes, each with $|S_x| = O(n/\beta) = O(pn)$. Hence, issuing queries requires $O((pn)^2 \log n)$ messages by good nodes. There are at most T_e queries sent by bad nodes, so responding to queries requires $O(T_e + (pn)^2 \log n)$ messages.

Computing LargeCoreBA in Steps 4 and 5, requires $O((pn)^2)$ messages by Lemma 3. Then in Step 6, all *active* nodes send to all nodes for $O(pn^2)$ messages. Finally, in Step 7, all nodes send $O(n \log n)$ messages to solve PromiseAgreement, as shown in Lemma 11. Thus, the total number of messages sent in the epoch is $O((pn)^2 \log n + pn^2 + n \log n + T_e)$.

The time to perform all steps in an epoch is dominated by the cost of performing LargeCoreBA which is polylogarithmic. ◀

▶ **Lemma 10.** *The algorithm terminates in a decision in a given epoch, unless the adversary sends $\Omega(pn^2)$ messages.*

Proof. There are at least $High$ light nodes unless the adversary causes bad nodes to send more than $pn\epsilon$ messages to $n\epsilon$ nodes, for a total of $\Omega(pn^2)$ messages. If there are at least $High$ light nodes in an epoch, then by Lemma 7 the algorithm terminates with a decision. ◀

Note that $O((pn)^2 \log n + pn^2) = O(pn^2)$ except when $p > 1/\log n$, in which case our algorithm runs LargeCoreBA on all the nodes, by messaging all n of their neighbors, for a total cost of $O(n^2)$. This is the bottleneck in the algorithm which causes it to be $O(\log n)$-competitive instead of $O(1)$-competitive.

Let T be the minimum of n^2 and the total number of messages sent by the adversary, and n be the number of nodes in the network. We can now prove Theorem 1.

4.3 Proof of Theorem 1

Proof. By Lemma 10, the algorithm will terminate in an epoch, unless the adversary sends cpn^2 messages in that epoch, for some constant c. In epoch i, $p = (2^{i-1} \log n)/n$. If we do not terminate in epoch i, then $T \geq c2^{i-1} n \log n$. In epoch i, by Lemma 9, the total number of messages sent is $O((pn)^2 \log n + pn^2 + n \log n + T_e)$.

We first consider the case where it's always true that $p \leq 1/\log n$, and note that $O((pn)^2 \log n + pn^2) = O(pn^2)$. Thus, the message cost in epoch i is $O(n2^i \log n + T_i)$, where T_i is the number of messages sent by the adversary in epoch i. The T_i terms clearly sum to $O(T)$. If ℓ is the last epoch, then $O(\sum_{i=1}^{\ell} 2^i n \log n) = O(2^\ell n \log n) = O(T + n \log n)$. Thus the total number of messages sent in this case is $O(T + n \log n)$.

We next consider the case where $p > 1/\log n$. In this case, our algorithm runs *LargeCoreBA* on all the nodes, by messaging all n of their neighbors, for a total cost of $O(n^2)$. The value of T in this case is $\Omega(n^2/\log n)$, so our total message cost is $O(T \log n)$.

Since epoch i has latency polylogarithmic in n (by Lemma 3), and there are at most $\log n$ epochs, the total latency is $O(\text{polylog}(n))$. Additionally, we note that when the algorithm terminates, by Lemma 10, all good nodes come to agreement on an input bit of some node in *CORE*.

Finally we note that we can also solve the leader election and committee election problems. To do this, the active nodes use Feige's leader election algorithm [25] to elect a committee in one step, or a leader in $\log^* n$ steps among the $CORE_x$ sets for every active node x. This is done instead of selecting an agreement value as in the KSSV algorithm [41]. ◀

5 Additional Algorithms

5.1 LargeCoreBA

Here we prove Lemma 3. We do this by adapting the algorithm from [41]. In that paper, all nodes have a view of all of other nodes and nodes are numbered $[1, n]$.

The main idea of our adaptation is to show that for any s, $\log^{10} n \leq s \leq n$, there exists a deterministic assignment of IDs in $[1, n^k]$ to a set of $s/\ln n$ committees, so that for every subset of size s IDs, a $1 - 1/\ln^2 n$ fraction of committees are (1) "sufficiently large"; and (2) contain a nearly representative fraction of both good and bad nodes.

The algorithm in [41] is built upon a family of bipartite graphs with expansion-like properties. The existence of such graphs are proved using the probabilistic method (see Section 3 of [41]). We need the same properties here, but for a possibly much smaller subset of $s \leq n$ identities, which come from a much larger name space ($[1, n^k]$). We show that we can start with identities in the range $[1, n^k]$, of which s are active and generate a set of committees which have the required properties with respect to the active nodes, as is needed in each layer of the "election graph" in [41], Corollary 3.2. The details are deferred to the full paper [8].

5.2 PromiseAgreement

We now present a simple algorithm to solve the *Promise Agreement* problem, defined in Section 3.2.

PromiseAgreement

1. Each node y sends a request to a random set of $c \log n$ nodes.
2. Each node x, upon receiving a request from a node y, responds to the request by reporting $(\textit{ready-out}_x, \textit{value}_x)$.
3. If greater than a $t/n + \epsilon$ fraction of nodes sampled by x respond with $\textit{ready-out} = 1$, then x sets $\textit{ready-out} \leftarrow 1$ and sets $\textit{value}_x$ to the majority of the *value* bits sent by sampled nodes. Else $\textit{ready-out}_x \leftarrow 0$.

▶ **Lemma 11.** PROMISEAGREEMENT *solves the* Promise Agreement *problem (Definition 4), with* $O(1)$ *latency, and sending* $O(T' + n \log n)$ *bits, where* T' *is the minimum of* n^2 *and the number of messages sent by the adversary during this algorithm.*

Proof. Assume there are at least a $t/n + 2\epsilon$ fraction of good nodes with $(\textit{ready-out}, \textit{value}) = (1, v)$ for the same bit v, and all remaining good nodes have *ready-out* values of 0. By Chernoff and union bounds, every good node then has greater than a $t/n + \epsilon$ fraction of good nodes with *ready-out* values of 1, and less than a $t/n + \epsilon$ fraction of bad nodes in their sample. Hence, all good nodes will terminate with tuple values of $(\textit{ready-out}, \textit{value}) = (1, v)$.

Assume that all good nodes have *ready-out* values of 0. Then by Chernoff and union bounds, each sample has less than a $t/n + \epsilon$ fraction of bad nodes. Hence, all good nodes will terminate with *ready-out* values of 0.

The number of bits sent is just the number of queries sent which is $O(T' + n \log n)$. ◀

6 Lower Bounds for Message-Competitive Byzantine Agreement

We now study lower bounds for message-competitive Byzantine agreement (BA). We first show a tight lower bound on the message competitiveness of deterministic BA protocols. Then we show a lower bound on the message competitiveness of randomized BA protocols.

6.1 Deterministic Lower Bound

As per our model in Section 1.1 we assume a complete n-node network with $\hat{\epsilon}n$ Byzantine nodes and $(1 - \hat{\epsilon})n$ good nodes (i.e., non-Byzantine) for some small constant $\hat{\epsilon}$. We assume the KT_0 model. The Byzantine nodes are controlled by a non-adaptive rushing adversary. It is assumed that Byzantine nodes cannot fake their own identities.

In the above setting, the goal is to show a lower bound on the message bits spent by the good nodes in any deterministic algorithm solving Byzantine everywhere agreement. The lower bound also holds in the KT_1 model, in which a node knows the ID of its neighbors.

Suppose there is a deterministic algorithm solving BA. The output of the algorithm, i.e., the agreed value depends on the ID, input distribution of the nodes and the information exchanged among the nodes during the execution of the algorithm. More precisely, the output of a node u (with id ID_u) is a function $f_u(ID_u, b_u, X_u) \to \{0, 1\}$, where the argument b_u is the input bit of u and X_u is the set of received message bits during the execution of the algorithm. Let us call this information (ID_u, b_u, X_u) as the "transcript" of u. The algorithm is deterministic and known to the adversary which controls the Byzantine nodes. Further, the algorithm should work for any input distribution (i.e., the $0-1$ value distribution). Given an input distribution over the nodes, the complete execution of the algorithm is known to the adversary. Based on the execution, the adversary selects Byzantine nodes (in the beginning) in such a way that the algorithm fails to achieve agreement everywhere unless it spends enough messages. In fact, we prove the following result.

▶ **Theorem 12.** *Suppose the budget of messages of the Byzantine nodes is* $T \leq cn^2$ *bits, for some constant* c*. Then any deterministic algorithm, which solves Byzantine everywhere agreement, incurs an expected* $\Omega(\min\{T, n^2\})$ *bits of messages.*

Proof. We give a proof-sketch here. The detailed proof can be found in the full version [8]. Let there be a deterministic algorithm $\mathcal{A}$ that solves the Byzantine agreement everywhere and incurs only $o(T)$ messages. We show a contradiction, that the agreement is wrong in the sense that there exists two nodes with two different output values for some input distribution.

Consider an arbitrary input distribution $\mathcal{I}$ over n nodes. Since the total messages sent by the good nodes is $o(T)$, there must exist a good node, say, u that exchanges (sends and receives) less than $\delta T/((1-\hat{\epsilon})n)$ message bits in total for some small constant $\delta < 1$ (the actual value of δ to be fixed later); otherwise the sum of the messages of all the good nodes would be $\Omega(T)$.Let S_u be the set of nodes which exchange messages with u throughout the execution of $\mathcal{A}$ on the given input $\mathcal{I}$. Note that, given $\mathcal{A}$ and $\mathcal{I}$, u and S_u are fixed and known to the adversary in the beginning. (Further, for different input $\mathcal{I}$, the pair (u, S_u) might be different.) The adversary then selects all the nodes in S_u as Byzantine nodes before the execution starts. Thus the *transcript* of u is fully controlled by the Byzantine nodes as X_u is determined by the nodes in S_u. The transcript of u is the total history of messages between u and the rest of the nodes. Clearly, the decision of u depends on the choice of u's input value (0 or 1), u's ID and its transcript (which might also include the IDs of the nodes that it communicated with). Also, in a valid protocol, every node (with every distinct ID and input value) will have a distinct transcript for deciding 0 or 1, respectively. Essentially, the adversary can decide a transcript for u (depending on its input value and ID) such that the output value of u would be different than the output value of all other good nodes (assuming all other good nodes execute the algorithm without any influence from the Byzantine nodes). This will give a contradiction to the everywhere agreement. ◀

6.2 Randomized Lower Bound

Let us first consider the anonymous KT_0 setting, i.e., nodes do not have any identifiers. Later, we extend this to the non-anonymous KT_0 setting, where good nodes have unique identities and Byzantine nodes can fake their identities (in full version). Each node u has $n-1$ ports through which it connects to the $n-1$ other nodes. Thus, if a node u sends a message through a port $p \in [n-1]$ to another node v, then any message u receives through p is guaranteed to be from v. For our lower bound purpose, we assume that the ports for each node u are assigned uniformly at random and independent of port assignments for other nodes. As before, among the n nodes, a small fraction $\hat{\epsilon}n$ (assumed to be integral) for a fixed $\hat{\epsilon} > 0$ are Byzantine and denoted V^b; let $V^g = V \setminus V^b$. Nodes can individually generate uniform and independent random bits, but availability of common coins is not assumed.

We assume that our Byzantine adversary is full-information (i.e., knows the states of all nodes at all times), computationally unbounded, and is also *rushing* (i.e., it can read messages sent by good nodes before sending out its own messages). The adversary is limited to being *static*, so that it must decide which nodes are bad prior to the start of the algorithm. We formally define message complexity as follows.

▶ **Definition 13.** *For a given BA algorithm $\mathcal{A}$, the message complexity $M_{\mathcal{A}}$ (or just M when clear from context) is defined as the maximum expected number of the sum of the bits sent by good nodes. The maximum is taken over all possible adversarial strategies (i.e., choice of IDs, port assignments, input bits, and the behaviour of the Byzantine nodes) and the expectation is over the random bits used by the nodes.*

Overview of Our Approach. We show that if bad nodes can send $\Omega(n^{1+\alpha})$ messages, then the good nodes must send at least $\Omega(n^{1+\alpha/2})$ messages, for any $\alpha \in (0, 1]$. If we assume not (for the sake of contradiction), then, good nodes can reach agreement while only sending $o(n^{\alpha/2})$ messages on average. Under this situation, when any good node u sends a message to any other good node v, the bad nodes can bombard v with $n^{\alpha/2}$ messages intended for denial of service (DoS). Node v will be unable to distinguish between the legitimate message

from u and these DoS messages from bad nodes. As a result, v will have to respond, on average, to $\Omega(n^{\alpha/2})$ messages from bad nodes first. This is a greater number of messages than what good nodes can afford on average. Thus, several good nodes will not be able to establish two-way contact with any other good node, which we then exploit to show the impossibility via an indistinguishability argument. The proof has been deferred to the full version [8].

▶ **Theorem 14.** *Consider any BA algorithm $\mathcal{A}$ that guarantees that good nodes reach a valid agreement in the anonymous KT_0 setting as long as the number of messages sent by Byzantine nodes is at most $B = n^{1+\alpha}$ for some $\alpha \in (0, 1]$. Then, the message complexity $M_{\mathcal{A}}$ is at least $\Omega(n^{1+\frac{\alpha}{2}})$.*

7 Conclusion

We have described an efficient randomized message-competitive algorithm to solve Byzantine agreement, Leader election and Committee election, in the synchronous communication model, with a static and full-information adversary, where nodes don't know the IDs of other nodes a priori. Our algorithm is efficient in the sense that message cost and latency grow slowly with the number of messages sent by the adversary. We also show lower bounds on message-competitive Byzantine agreement algorithms. Our lower bounds show that in general, it is not possible to do significantly better than our algorithm with respect to the number of bits sent by Byzantine nodes. A key open problem is to close the gap between upper and lower bounds for randomized protocols across all budget values as well improve the fault-tolerance to 1/3-fraction of all nodes.

References

1 Ittai Abraham, TH Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *PODC*, pages 317–326, 2019.

2 Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, and Roger P. Wattenhofer. FARSITE: Federated, Available, and Reliable Storage for Incompletely Trusted Environment. In 5^{th} *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 1–14, 2002.

3 A. Agbaria and R. Friedman. Overcoming byzantine failures using checkpointing. *University of Illinois at Urbana-Champaign Coordinated Science Laboratory technical report no. UILU-ENG-03-2228 (CRHC-03-14)*, 2003.

4 Abhinav Aggarwal, Varsha Dani, Thomas P. Hayes, and Jared Saia. Sending a message with unknown noise. In *Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN)*, pages 8:1–8:10, 2018.

5 Yair Amir, Claudiu Danilov, Jonathan Kirsch, John Lane, Danny Dolev, Cristina Nita-Rotaru, Josh Olsen, and David John Zage. Scaling byzantine fault-tolerant replication towide area networks. In *Proceedings of International Conference on Dependable Systems and Networks (DSN)*, pages 105–114, 2006.

6 D. P. Anderson and J. Kubiatowicz. The worldwide computer. *Scientific American*, 286(3):28–35, 2002.

7 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*. John Wiley Interscience, 2004.

8 John Augustine, Valerie King, Anisur Rahaman Molla, Gopal Pandurangan, and Jared Saia. Scalable and secure computation among strangers: Message-competitive byzantine protocols. *CoRR*, abs/1907.10308, 2019.

9 Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in o(log n) rounds. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 179–186. ACM, 2006.

10 Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, and Maxwell Young. How to Scale Exponential Backoff: Constant Throughput, Polylog Access Attempts, and Robustness. In *Proceedings of the* 27^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 636–654, 2016.

11 Michael A. Bender, Jeremy T. Fineman, Mahnush Movahedi, Jared Saia, Varsha Dani, Seth Gilbert, Seth Pettie, and Maxwell Young. Resource-Competitive Algorithms. *SIGACT News*, 46(3):57–71, September 2015.

12 Bitcoin. Bitcoin website https://bitcoin.org/.

13 Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 104–121, 2015.

14 Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast byzantine agreement. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 57–64. ACM, 2013.

15 Christian Cachin and Jonathan A. Poritz. Secure Intrusion-Tolerant Replication on the Internet. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 167–176, 2002.

16 Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. *Operating Systems Review*, 33:173–186, 1998.

17 Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

18 Allen Clement, Mirco Marchetti, Edmund Wong, Lorenzo Alvisi, and Mike Dahlin. Byzantine Fault Tolerance: The Time is Now. In *Proceedings of the Second Workshop on Large-Scale Distributed Systems and Middleware (LADIS)*, pages 1–4, 2008.

19 Allen Clement, Edmund Wong, Lorenzo Alvisi, Mike Dahlin, and Mirco Marchetti. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In *Proceedings of the Sixth USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 153–168, 2009.

20 Varsha Dani, Tom Hayes, Mahnush Movahedi, Jared Saia, and Maxwell Young. Interactive Communication with Unknown Noise Rate. *Information and computation*, 261(Part):464–486, 2018.

21 Varsha Dani, Mahnush Movahedi, Jared Saia, and Maxwell Young. Interactive Communication with Unknown Noise Rate. In *Proceedings of the Colloquium on Automata, Languages, and Programming (ICALP)*, pages 575–587, 2015.

22 Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985.

23 Ethereum. Ethereum website https://ethereum.org/.

24 Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *Proceedings of the* 13^{th} *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 45–59, 2016.

25 Uriel Feige. Noncryptographic selection protocols. In *Proc. of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 142–153, 1999.

26 Paul Feldman and Silvio Micali. Byzantine agreement in constant expected time (and trusting no one). In *FOCS*, pages 267–276, 1985.

27 Freenet. Freenet website https://freenetproject.org/author/freenet-project-inc.html.

28 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, pages 51–68, 2017.

29 Seth Gilbert, Valerie King, Seth Pettie, Ely Porat, Jared Saia, and Maxwell Young. (Near) Optimal Resource-Competitive Broadcast with Jamming. In *Proceedings of the* 26^{th} *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 257–266, 2014.

30 Seth Gilbert, Valerie King, Jared Saia, and Maxwell Young. Resource-Competitive Analysis: A New Perspective on Attack-Resistant Distributed Computing. In *Proceedings of the* 8^{th} *ACM International Workshop on Foundations of Mobile Computing*, page 1, 2012.

31 Seth Gilbert and Maxwell Young. Making Evildoers Pay: Resource-Competitive Broadcast in Sensor Networks. In *Proceedings of the* 31^{th} *Symposium on Principles of Distributed Computing (PODC)*, pages 145–154, 2012.

32 Shafi Goldwasser, Elan Pavlov, and Vinod Vaikuntanathan. Fault-tolerant distributed computing in full-information networks. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 15–26. IEEE Computer Society, 2006.

33 Sergey Gorbunov and Silvio Micali. Democoin: A Publicly Verifiable and Jointly Serviced Cryptocurrency. Cryptology ePrint Archive, Report 2015/521, 2015. http://eprint.iacr.org/2015/521.

34 Jim Gray. The cost of messages. In *PODC*, pages 1–7. ACM, 1988.

35 Diksha Gupta, Jared Saia, and Maxwell Young. Proof of work without all the work. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, pages 1–10, 2018.

36 Diksha Gupta, Jared Saia, and Maxwell Young. Peace through superior puzzling: An asymmetric sybil defense. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1083–1094. IEEE, 2019.

37 Diksha Gupta, Jared Saia, and Maxwell Young. Resource burning for permissionless systems. *arXiv preprint arXiv:2006.04865*, 2020.

38 Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement. *Mathematical Systems Theory*, 26(1):41–102, 1993. Conference version in PODC 1991.

39 Valerie King, Steven Lonargan, Jared Saia, and Amitabh Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. In *International Conference on Distributed Computing and Networking*, pages 203–214. Springer, 2011.

40 Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. *J. ACM*, 58(4):18:1–18:24, 2011.

41 Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the Seventeenth annual ACM-SIAM Symposium on Discrete Algorithm*, pages 990–999. Society for Industrial and Applied Mathematics, 2006.

42 Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 87–98. IEEE, 2006.

43 Valerie King, Jared Saia, and Maxwell Young. Conflict on a Communication Channel. In *Proceedings of the* 30^{th} *Symposium on Principles of Distributed Computing (PODC)*, pages 277–286, 2011.

44 Jiejun Kong. *Anonymous and untraceable communications in mobile wireless networks*. Citeseer, 2004.

45 Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: Speculative Byzantine Fault Tolerance. In *Proceedings of* 21^{st} *ACM SIGOPS Symposium on Operating Systems Principles*, pages 45–58, 2007.

46 Dariusz R. Kowalski and Achour Mostéfaoui. Synchronous byzantine agreement with nearly a cubic number of communication bits: synchronous byzantine agreement with nearly a cubic

number of communication bits. In Panagiota Fatourou and Gadi Taubenfeld, editors, *PODC*, pages 84–91. ACM, 2013.

47 Harry C Li, Allen Clement, Edmund L Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. Bar gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 191–204, 2006.

48 Na Li, Nan Zhang, Sajal K Das, and Bhavani Thuraisingham. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, 7(8):1501–1514, 2009.

49 Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 17–30, 2016.

50 Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

51 Dahlia Malkhi and Michael K. Reiter. Unreliable intrusion detection in distributed computations. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW)*, pages 116–125, 1997.

52 Michael Mitzenmacher and Eli Upfal. *Probability and computing: randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

53 Hector Garcia Molina, Frank Pittelli, and Susan Davidson. Applications of Byzantine Agreement in Database Systems. *ACM Transactions on Database Systems (TODS)*, 11:27–47, 1986.

54 Oceanstore. The Oceanstore Project. http://oceanstore.cs.berkeley.edu.

55 Gopal Pandurangan. *Distributed Network Algorithms*. https://sites.google.com/site/gopalpandurangan/dna, 2018.

56 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

57 D. Peleg. *Distributed Computing: A Locality Sensitive Approach*. SIAM, 2000.

58 Nuno Preguiça, Rodrigo Rodrigues, Christóvão Honorato, and João Lourenço. Byzantium: Byzantine-Fault-Tolerant Database Replication Providing Snapshot Isolation. In *Proceedings of the Fourth Conference on Hot Topics in System Dependability*, page 9. USENIX Association, 2008.

59 Sean C. Rhea, Patrick R. Eaton, Dennis Geels, Hakim Weatherspoon, Ben Y. Zhao, and John Kubiatowicz. Pond: The oceanstore prototype. In *Proceedings of the FAST '03 Conference on File and Storage Technologies*, pages 1–14, 2003.

60 Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Commun.*, 11(6):38–43, 2004.

61 Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer networks*, 76:146–164, 2015.

62 SINTRA. SINTRA - Distributed Trust on the Internet. http://www.zurich.ibm.com/security/dti/.

63 Tor. Tor website https://www.torproject.org/.

64 Rolf H Weber. Internet of things–new security and privacy challenges. *Computer law & security review*, 26(1):23–30, 2010.

65 Alex Wright. Contemporary approaches to fault tolerance. *Commun. ACM*, 52(7):13–15, 2009.

66 Hiroyuki Yoshino, Naohiro Hayashibara, Tomoya Enokido, and Makoto Takizawa. Byzantine agreement protocol using hierarchical groups. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 64–70, 2005.

67 Mahdi Zamani, Jared Saia, and Jedidiah Crandall. TorBricks: Blocking-Resistant Tor Bridge Distribution. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 426–440. Springer, 2017.

68 Wenbing Zhao. A Byzantine Fault Tolerant Distributed Commit Protocol. In *Proceedings of the 3rd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 37–46, 2007.