Distributed Fast Crash-Tolerant Consensus with Nearly-Linear Quantum Communication

Mohammad T. HajiAghayi ✉
University of Maryland, College Park, MD, USA

Dariusz R. Kowalski ✉
School of Computer and Cyber Sciences, Augusta University, GA, USA

Jan Olkowski ✉
University of Maryland, College Park, MD, USA

Abstract

Fault-tolerant *Consensus* is about reaching agreement on some of the input values in a limited time by non-faulty autonomous processes, despite of failures of processes or communication medium. This problem is particularly challenging and costly against an adaptive adversary with full information. Bar-Joseph and Ben-Or (PODC'98) were the first who proved an absolute lower bound $\Omega(\sqrt{n/\log n})$ on expected time complexity of Consensus in any classical (i.e., randomized or deterministic) message-passing network with n processes succeeding with probability 1 against such a strong adaptive adversary crashing processes.

Seminal work of Ben-Or and Hassidim (STOC'05) broke the $\Omega(\sqrt{n/\log n})$ barrier for consensus in the classical (deterministic and randomized) networks by enhancing the model with quantum channels. In such networks, quantum communication between every pair of processes participating in the protocol is also allowed. They showed an (expected) constant-time quantum algorithm for a linear number of crashes $t < n/3$.

In this paper, we improve upon that seminal work by reducing the number of quantum and communication bits to an arbitrarily small polynomial, and even more, to a polylogarithmic number – though, the latter in the cost of a slightly larger polylogarithmic time (still exponentially smaller than the time lower bound $\Omega(\sqrt{n/\log n})$ for the classical computation models).

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Distributed algorithms

Keywords and phrases distributed algorithms, quantum algorithms, adaptive adversary, crash failures, Consensus, quantum common coin, approximate counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.80

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2305.10618`

Funding *Mohammad T. HajiAghayi*: Partially supported by DARPA QuICC, ONR MURI 2024 award on Algorithms, Learning, and Game Theory, Army-Research Laboratory (ARL) grant W911NF2410052, NSF AF:Small grants 2218678, 2114269, 2347322.
Jan Olkowski: Partially supported by DARPA QuICC, ONR MURI 2024 award on Algorithms, Learning, and Game Theory, Army-Research Laboratory (ARL) grant W911NF2410052, NSF AF:Small grants 2218678, 2114269, 2347322.

1 Introduction

Consensus is about making a common decision among the processes' initial input values in a limited time by every non-faulty process, despite the faulty behaviour of some of the players. Since its introduction by Pease, Shostak and Lamport [31] (JACM'80), who ruled out trivial solutions (such as always deciding on the same bit), fault-tolerant Consensus has constantly been among foundation problems in distributed computing. This problem has been studied in synchronous/asynchronous and deterministic/randomized computation models and under various fault-tolerant or adversarial models: Fail-Stop (crashes) and Byzantine, Static and Adaptive, Computationally Bounded and Unbounded Adversaries - just to name a few (see Section 2.1 for related work).

While the landscape of Consensus problem under the classic model of computation is well-developed, much less is known if we allow for quantum computation. The seminal work of Ben-Or and Hassidim [9] (albeit a short 5-pager in STOC'05) broke the $\Omega(\sqrt{n/\log n})$ rounds time barrier for classic computing by employing quantum computing to further use the power of randomization in distributed computing. They showed an (expected) constant-time quantum algorithm for a linear number of crashes $t < n/3$, however, the algorithm is inefficient in terms of communication bits and, even more importantly, in terms of the number of quantum bits (qubits), as it uses $\Omega(n)$ of them per process. Since then no algorithm has managed to reduce the quantum resources needed to solve Consensus. Because generating, maintaining and sending quantum bits is extremely costly (today's quantum devices use less than 500 qubits), thus the main question of our paper emerges naturally:

Could the number of qubits be substantially reduced without harming the time complexity?

1.1 Distributed setting

We consider a quantum synchronous message-passing model (c.f., [6]), consisting of n synchronous processes (also called players), each with common clock (a clock tick is called a round or a step) and unique id from the known set $\mathcal{P} = [n] = \{1, \ldots, n\}$.

Between any pair of processes we assume the existence of a quantum channel being able to transmit reliable[1] messages caring quantum bits, qubits. For the sake of completeness, we also augment the model by classic point-to-point channels between any pair of processes. In each round, a process can send (personalized) quantum and classic messages to any selected subset of other processes. After *multicasting messages*, in the same round a process *receives messages* that were just sent to it by other processes, and performs *local computation*, which involves both quantum and classic bits.[2]

Processes are prone to *crash failures*, also called *fail-stops.* A crashed process permanently stops any activity, including sending and receiving messages.

We model crashes as incurred by a full-information *adversary* (the same as in [7, 9]) that knows the algorithm, the exact pure quantum state (see Section 3) and the classic state of the system at any point of an execution, and has an unbounded computational power. The adversary decides which processes to fail and when. The adversary is also *adaptive* – it can make a decision on-line based on its current full-knowledge of the system. However, the adversary does not know the future computation, which means that it does not know future random bits drawn by processes.

As for the quantum part, the adversary can apply no quantum operation to the system, but it is aware of all quantum and classic changes of state that the network undergoes. If a process is crashed by the adversary, we assume that its quantum bits are not destroyed (in particular, entangled qubits in other processes do not collapse but maintain their entanglement), however they cannot be used in further computation.

[1] Messages are not lost nor corrupted while in transit.
[2] Local computation also decides what messages to send in the next round and to whom.

Failures are *not clean* – when a process crashes when attempting to multicast a message, then some of the recipients may receive the message and some may not; this aspect is controlled by the adversary. A t*-adversary* is additionally restricted by the the number of crashed processes being *smaller than* t; if $t = n$ then the n-adversary is also called an unbounded adversary (note that even in such case, at least one process must survive for Consensus to make sense). Throughout the paper, we will be calling the adversary described above "adaptive", for short.

Consensus problem. each process p has its own initial value $input_p$ and has to output a (common) decision value, so that the following conditions hold: *validity* – decision must be on an input value of some process; *agreement* – no two processes decide on different values; and *termination* – each process eventually decides on some value, unless it is faulty. All those three requirements must hold with probability 1. We focus on *binary Consensus*, in which initial values are in $\{0, 1\}$.

Correctness and complexity – in terms of time (the number of rounds needed for all processes to terminate) and the number of quantum bits (qubits) and communication bits – are analyzed and maximized (worst-case) against an adaptive adversary.

We say that a random event occurs *with high probability* (*whp* for short), if its probability could be made $1 - O(n^{-c})$ for any sufficiently large positive constant c by linear scaling of parameters.

2 Our Results

In this work, we focus on improving quantum bits' and communication complexities (without harming time complexity) of quantum algorithms solving Consensus problem with probability 1 against an adaptive full-information adversary capable of causing processes' crashes. We observe that the maximum, per process, number of communication bits in Consensus problem is $\Omega(n)$, therefore one can only hope to improve amortized communication complexity (per process), see the full version of the paper.[3]

Our first main result is a quantum algorithm that solves Consensus in expected constant number of rounds and amortized number of qubits and classical communication bits per process being an arbitrarily low polynomial. This directly improves, by a polynomial factor, on the result of Ben-Or and Hassidim [9], which required $\Theta(n)$ qubits and communication bits, amortized per process. The detailed description of the algorithm and the proof of its correctness is presented in Section 4.

▶ **Theorem 1.** *For any $\epsilon > 0$, there is an algorithm solving consensus against an adaptive $n/3$-adversary in expected $O(1)$ rounds while using $O(n^{\epsilon})$ qubits and communication bits (amortized) per process, w.h.p.*

To achieve this result, we give improved protocols for several existing tools that have been historically used in consensus algorithms. Combining all our advancements together provides a tighter approach to the general problem of consensus.

Our first technique is a new quantum implementation of a weak global coin.

▶ **Definition 2** ([9]). *Let $\mathcal{C}$ be a protocol for n players (with no input), where each player i outputs a (classical) bit $v_i \in \{0, 1\}$. We say that the protocol $\mathcal{C}$ is a t-resilient weak global coin protocol (or computes a weak global coin, for short) with fairness $\rho > 0$, if for any adaptive t-adversary and any value $b \in \{0, 1\}$, with probability at least ρ, $v_i = b$ for all good players i.*

[3] All omitted proofs and materials can be found in the full version of the paper, under the link given in "Related Version" on page 1.

We show a $\frac{1}{3}$-resilient weak global coin that works in constant, $O\left(\frac{1}{\epsilon}\right)$, time, uses an arbitrarily small polynomial number of bits $O(n^{\epsilon})$, amortized per process (for any chosen constant $\epsilon > 0$), and results in all non-faulty processes returning the same value with a non-zero constant probability, c.f., Theorem 12. The main idea behind our improvement is to couple a quantum protocol for selecting a leader in the system, proposed in [9], with generalized sparse patterns of fault-tolerant communication induced on random graphs, which in a classical version (i.e., non-quantum) was firstly used in [23].

Our second technical result is a deterministic algorithm for counting, which returns, in constant time, a count of the number of input 1 (or 0, resp.) among the processes. Specifically, our algorithm solves the following problem.

▶ **Definition 3** (Fuzzy Counting [23]). *A Fuzzy Counting is a distributed problem in which every active process is required a number between the initial and the final number of active processes. The notion of "being active" usually depends on the goal of the counting, e.g., all non-faulty processes, non-faulty processes with initial value* 1*, etc.*

The authors of [23] provided an algorithm to this problem that works in $O(\log^3 n)$ rounds and uses $O(\log^7 n)$ communication bits per process. By employing random graphs of asymptotically larger degree than those in [23], we generalize this result by showing a solution that works in constant time. As consequence of this improvement a tradeoff between the number of rounds and the communication complexity arises. Our protocol is faster than the one in [23], but on the other hand it uses a polynomial, yet of an arbitrarily small degree, number of communication bits, amortized per process. On the way to achieve this result, we also propose a new solution to Gossip – a distributed problem in which every non-faulty process, despite the presences of the adversary, have to collect inputs of every other non-faulty processes.[4] The novelty in this approach is that our solution, by exploiting similar properties of random graphs to the approach used for generating a weak global coin, we can solve Gossip in constant time $O\left(\frac{1}{\epsilon}\right)$ while using only $O(n^{\epsilon})$ bits per processes, for any constant $1 > \epsilon > 0$. The formal statement of the Gossip algorithm is provided in Theorem 13 and the analysis of our implementation of a weak global coin can be found in the Section 5 and the analysis of our implementation of Fuzzy Counting in Section 6.

Although constant-time algorithms cannot low (sub-polynomial) amortized communication complexity, we show that our main algorithm could be re-instantiated in such a way that it uses only a *polylogarithmic* number of qubits and classical communication bits to solve consensus in just a *polylogarithmically* larger number of rounds (see Section 4, and technical counterpart Theorem 7 for more details).

▶ **Theorem 4.** *There is an algorithm solving consensus against an unbounded adaptive adversary in polylogarithmic number of rounds, in expectation, while using a polylogarithmic number of qubits and communication bits (amortized) per process, whp.*

We believe that the newly developed techniques could be also applied to other types of failures, after failure-specific modifications. For example, although message omission failures require linear amortized communication per process (c.f., [22]), one could still use a small polynomial or even a polylogarithmic number of qubits (together with a linear number of classic communication bits) per process, if qubits are handled according to our techniques while some additional classic communication bits are introduced to handle message omissions. We leave details to follow-up work.

---

4 The main difference between Gossip and Fuzzy Counting is that the latter allows for aggregating information, i.e. counting the number of 1's in the system vs collecting a set of identifiers of these active processes that start with 1.

2.1 Previous and Related Work

Consensus in the classic (non-quantum) model. Bar-Joseph and Ben-Or [7] (see also their extended version [8]) proved a lower bound $O(\sqrt{\frac{n}{\log n}})$ on expected time complexity of consensus against an adaptive adversary. They also complemented it by time-optimal randomized algorithm. Their algorithm uses expected $O(\frac{n^{3/2}}{\log n})$ number of communications bits, amortized per process, which has been recently improved by Hajiaghayi et al [23] to $O(\sqrt{n})$ (while maintaining the almost-optimal round complexity $O(\sqrt{n}\log n)$).

Fisher and Lynch [19] proved a lower bound $f+1$ on *deterministic* consensus with f crashes (that actually occurred, i.e., $f < t$), thus separating deterministic solutions from randomized. Regarding communication complexity, Amdur, Weber and Hadzilacos [3] showed that the amortized number of messages per process is at least constant, even in some failure-free execution. Dwork, Halpern and Waarts [18] found a solution with $O(\log n)$ messages per process, but requiring an exponential time, and later Galil, Mayer and Yung [20] developed a message-optimal algorithm working in super-linear $O(n^{1+\varepsilon})$ time, for any $0 < \varepsilon < 1$ and any $f < n$. They also improved the communication further to a constant number of communication bits per process, but the resulting algorithm was exponential in the number of rounds. Chlebus and Kowalski [11] showed that consensus can be solved in $O(f+1)$ time and with $O(\log^2 f)$ messages if only the number $n-f$ of non-faulty processors satisfies $n-f=\Omega(n)$. It was later improved in [12] to $O(f+1)$ time and $O(\text{polylog } n)$ number of communication bits. All the abovementioned communication complexities are amortized per process.

Quantum consensus. To the best of our knowledge, almost all previous papers on quantum consensus concentrated on assuring feasibility of the problem against strong Byzantine adversaries, c.f., [14, 24, 26], or on optimizing time complexity, including the work of Ben-Or and Hassidim [9] achieving constant time against an adaptive adversary.

In recent years, the primary application of (classical) consensus has been in the synchronization of distributed blockchains. A blockchain algorithm based on asymmetric quantum encryption was proposed in [34]. Blockchain algorithms often use leader election as an important subroutine. Quantum algorithms for leader election in anonymous networks, a problem intrinsically unsolved in the classical setting, has been considered in [32, 29]. [36] implemented the first algorithm for the anonymous leader election problem on a quantum computer. A more thorough study of challenges arising from practical implementations of quantum distributed networks is a pending, but unresolved issue [17]. The dynamics of quantum consensus and gossip over a contiguous space was studied in [28].

Sparse quantum communication has been considered by Chlebus, Kowalski and Strojnowski in [13], in the context of solving some version of consensus, but their protocols work correctly only with some probability smaller than 1 and for a specific number of failures corresponding to the probability of success. Another difference is that they used quantum operations to encode the classical inputs in quantum registers and used it to directly solve consensus. In this paper, we show another, more-efficient approach, in which we first create a quantum, weak global coin and later employ this tool to the state-of-the-art framework of solving consensus based on the common coin. Other distributed computing problems, not necessarily fault-prone, were also analyzed in quantum models, c.f., [10, 33, 35]. Finally, the readers interested in more broad treatment of quantum consensus are referenced to a recent survey [27].

More efficient classic randomized solutions against *weak adversaries*. Whenever weaker oblivious adversaries are considered, randomness itself proved to be enough in reducing time complexity to a constant. Chor, Merritt and Shmoys [15] developed constant-time algorithms for consensus against an *oblivious adversary* – that is, the adversary who knows the algorithm but has to decide which process fails and when before the execution starts. Their solution, however, requires a linear number of communication bits per process. Gilbert and Kowalski [21] gave a randomized consensus algorithm that achieves optimal communication complexity of $O(1)$ amortized communication bits per process while terminating in $O(\log n)$ time w.h.p., as long as the number of crashes $f < n/2$.

Classic consensus in more demanding models. Dolev and Reischuk [16] and Hadzilacos and Halpern [22] proved the $\Omega(f + 1)$ lower bound on the amortized message complexity per process of deterministic consensus for *omission or (authenticated) Byzantine failures*. However, under some limitation on the adversary and requiring termination only whp, the sublinear expected communication complexity $O(\sqrt{n}\ \mathrm{polylog}\ n)$ per process can be achieved even in case of Byzantine failures, as proved by King and Saia [25]. Such limitations are apparently necessary to achieve subquadratic time complexity for Byzantine failures, c.f., Abraham et al. [1]. *Asynchrony* also implies large communication – Aspnes [4] proved a lower bound $\Omega(n/\log^2 n)$ on communication complexity per process. The complexity bounds in this setting have been later improved, see e.g., [5, 2].

3 Technical Preliminaries

Quantum model of computation. We provide a short review of those parts of the quantum model of computing that are relevant to our results. The reader can find a comprehensive introduction to quantum computing in e.g. [30]. We use only the pure state of qubits. A pure state of a single qubit is a vector in a 2-dimensional Hilbert space $\mathcal{H}$. A pure quantum state of d qubits, denoted $|x\rangle$, is a vector of 2^d-dimensional Hilbert space $\mathcal{H}^{\otimes d} = \mathcal{H} \otimes \ldots \otimes \mathcal{H}$. In our paper, we use only the standard computational basis of the Hilbert space, which consists of vectors $\{|b_1 \ldots b_d\rangle : b_1 \ldots, b_d \in \{0,1\}^d\}$, to describe the system. Therefore, any state $|x\rangle$ can be expressed as $|x\rangle = \sum_{i=0}^{2^d-1} \alpha_i |i\rangle$, with the condition that $\sum_i |\alpha_i|^2 = 1$, since quantum states can be only normalized vectors.

Transitions, or equivalently – changes of states of a quantum system, are given by unitary transformations on the Hilbert space of d qubits. These unitary transformations are called *quantum gates*. These operations are exhaustive in the sense that any quantum computation can be expressed as a unitary operator on some Hilbert space. There are small-size sets of quantum gates working on two-dimensional space that are universal – any unitary transformation on a 2^d-dimensional quantum space can be approximated by a finite collection of these universal gates. In our applications, any quantum algorithm computation run by a process requires a polynomial (in n) number of universal gates.

Finally, an important part of quantum computation is also a quantum measurement. Measurements are performed with respect to a basis of the Hilbert space – in our case, this is always the computational basis. A complete measurement in the computational basis executed on a state $|x\rangle = \sum_{i=0}^{2^d-1} \alpha_i |i\rangle$ leaves the state in one of the basis vectors, $|i\rangle$, for $i \in \{0,1\}^d$, with probability α_i^2. The outcome of the measurement is a classic register of d bits, informing to which vector the state has been transformed. It is also possible to measure only some qubits of the system, which is called a partial measurement. If A describes the subset of qubits that we want to measure and B is the remaining part of the system, then the partial measurement is defined by the set of projectors $\{\Pi_i = |i\rangle_A \langle i|_A \otimes I_B \mid \text{for } i \in \{0,1\}^d\}$.[5] In the former, a subscript refers to the part of the system on which the object exists, I denotes the identity function, while $\langle i|$ is a functional of the dual space to the original Hilbert space (its matrix representation is the conjugate transpose of the matrix representation of $|i\rangle$). If before the measurement the system was in a state $|x\rangle_{AB}$ then, after the measurement, it is in one of the states $\{\Pi_i |x\rangle_{AB} \mid \text{for } i \in \{0,1\}^d\}$, where state $\Pi_i |x\rangle_{AB}$ is achieved with probability $\langle x|_{AB} \Pi_i |x\rangle_{AB}$.[6] We would like to note that, similarly to all other quantum operations, measurements are local in our model. The result of a measurement is visible only to the process that performed this measurement; however, any quantum operation in some process may affect quantum bits stored in some other process.

Graph notations. Let $G = (V, E)$ denote an undirected graph. Let $W \subseteq V$ be a set of nodes of G. We say that an edge (v, w) of G is *internal for* W if v and w are both in W. We say that an edge (v, w) of G *connects the sets* W_1 *and* W_2, or *is between* W_1 *and* W_2, for any disjoint subsets W_1 and W_2 of V, if one of its ends is in W_1 and the other in W_2. The *subgraph of* G *induced by* W, denoted $G|_W$, is the subgraph of G containing the nodes in W and all the edges internal for W in G. A node adjacent to a node v is a *neighbor of* v and the set of all the neighbors of a node v is the *neighborhood of* v. $N^i_G(W)$ denotes the set of all the nodes in V that are of distance at most i from some node in W in graph G. In particular, the (direct) neighborhood of v is denoted $N_G(v) = N^1_G(v)$.

The following combinatorial properties are of utter importance in the analysis of our algorithms. Graph G is said to be ℓ*-expanding*, or to be an ℓ*-expander*, if any two subsets of ℓ nodes each are connected by an edge. Graph G is said to be (ℓ, α, β)*-edge-dense* if, for any set $X \subseteq V$ of *at least* ℓ nodes, there are at least $\alpha|X|$ edges internal for X, and for any set $Y \subseteq V$ of *at most* ℓ nodes, there are at most $\beta|Y|$ edges internal for Y. Graph G is said to be $(\ell, \varepsilon, \delta)$*-compact* if, for any set $B \subseteq V$ of at least ℓ nodes, there is a subset $C \subseteq B$ of at least $\varepsilon\ell$ nodes such that each node's degree in $G|_C$ is at least δ. We call any such set C a *survival set for* B.

4 Consensus Algorithm

The very high-level description of our consensus algorithm CheapQuantumConsensus is as follows. Each process starts by setting its preferred value (the value which it would like to decide for now) to the input bit. Then, the processes repeatedly use the counting procedure FastCounting, specified in Section 6, to compute the number of preferred 0's and 1's stored by processes that have not crashed yet, see line 3. Due to the definition of Fuzzy Counting, the outcomes of FastCounting can be slightly different across processes, but by no more than the number of crashes. Depending on the outcome, each process may change its preferred value to the dominating one (among the received preferred values), decide if the domination is substantial, or run the quantum common coin procedure, if the number of preferred 0's and 1's are very close to each other – see lines 14-17 in the pseudocode of CheapQuantumConsensus in Figure 1; all lines involving communication are underlined. Repeating the entire process (i.e., counting) for a constant number of rounds and then calculating the weak global coin and updating the decision either deterministically or based on the global coin, guarantees to produce one preferred value in all correct processes with probability 1, provided strong enough protocol for computing the global coin.

[5] We follow the standard notation in quantum computing and skip writing normalizing factors.
[6] $\Pi_i |x\rangle_{AB}$ and $\langle x|_{AB} \Pi_i |x\rangle_{AB}$ are simply linear operations on matrices and vectors.

In general, this type of framework is well-known for solving consensus and was proposed first by Bar-Joseph and Ben-Or [8] in the context of classical randomized computation against an adaptive adversary. However, for classical randomized computation the limitation of this approach is in the fact that any weak global coin obtained so far can tolerate at most $O(\sqrt{n})$ crashes to guarantee constant fairness of the coin. In contrast, by employing quantum communication we propose a new quantum protocol that computes a weak global coin with constant fairness even when a linear number of processes crash.

Besides this improvement, we also propose two new techniques for sparse classical and quantum communication, employed in lines 3 and 18 of the pseudocode in Figure 1: fast and communication-efficient counting and, as mentioned before, fast and quantum-communication-efficient weak global coin, respectively. Both these techniques use parameters x, d, α, which, roughly speaking, correspond to the density of random communication graphs used in these algorithms. The detailed performance formulas of these algorithms, with respect to those parameters, are stated in Theorems 12 and 14.

In the heart of these two techniques lies a crucial observation: consensus (as well as common coin and counting) could be achieved quickly even if many processes do not directly exchange messages, but use some carefully selected sparse set of communication links instead. This way, instead of creating qubits for each pair of processes, we could do it only per some pairs corresponding to some communication links to be used. This set of links, modeled as an evolving communication graph, needs to be maintained adaptively and locally by processes throughout the execution – otherwise, an adaptive adversary would learn it and design crashes to separate processes and prevent consensus.

Algorithm's description. Each process p stores its current choice in b_p (which is initialized to p's input). The value b_p at the end of the algorithm indicates p's decision. Now, processes use $O(1)$ (in expectation) *phases* to update their values b_p such that eventually every process keeps the same decision. To do so, in a round r every process p calculates the number of processes whose current choice is 1 and the number of processes whose current choice is 0, denoted O_p^r and Z_p^r respectively. Based on these numbers, process p: either sets b_p to 1, if the number O_p^r is large enough; or it sets b_p to 0 if the number Z_p^r is large; or it replaces b_p with a random bit if the numbers of zeros and ones are close to each other. If for generating the random bit, in line 18, processes use a quantum implementation of a weak global coin (implemented with CheapQuantumCoin algorithm, specified in Section 5), they will all have the same value b_p with constant probability unless more than third of alive processes crash. Assuming the presence of the adaptive adversary, this could not be achieved quickly if using classical communication only. Once it happens with the help of the quantum weak global coin, the conditional statements in lines 14-17, run in the next iteration of the "while" loop, guarantee that once the majority of processes have the same value b_p, the system converges to this value in at most 2 phases. Since the probability of this event is constant (guaranteed by the quantum weak global coin combined), the expected number of phases before the consensus algorithm terminates is constant. That reasoning holds, assuming that at most 1/3 fraction of processes crashed (we will generalize it to any $t \leq n$ at the end of this section).

As mentioned earlier, the major improvement in the above protocol comes from using novel techniques for counting and weak global coin. For the former, we use the FastCounting algorithm (Theorem 14), which, with the choice of parameters given in line 3, works in $O\Big(\big(\frac{1}{\epsilon}\big)^4\Big)$ rounds and uses $O\big(n^{1+3\epsilon}\log^2 n\big)$ (classic) communication bits in total. Similarly, the CheapQuantumCoin algorithm (Theorem 12), executed in line 18, terminates in $O\Big(\big(\frac{1}{\epsilon}\big)^3\Big)$ rounds and uses $O\Big(n^{1+2\epsilon}\log^2 n\Big)$ both quantum and classical bits; we need to divide the communication formulas by n to obtain the complexity amortized per process.

Algorithm 1 CheapQuantumConsensus for process p.

input: $\mathcal{P}$, p, $input_p$

1 $b_p \leftarrow input_p$; $r \leftarrow 1$; `decided` $\leftarrow FALSE$;
2 **while** $TRUE$ **do**
3 participate in FastCounting$(\mathcal{P}, p, b_p)$ (run with parameters $x = n^\epsilon, d = \log n, \alpha = n^\epsilon$) that counts the processes who have $b_p = 1$ and the processes who have $b_p = 0$; let O_p^r, Z_p^r be the numbers of ones and zeros (resp.) returned by FastCounting;
4 $N_p^r \leftarrow Z_p^r + O_p^r$;
5 **if** $(N_p^r < \sqrt{n/\log n})$ **then**
6 1) send b_p to all processes; 2) receive all messages sent to p in round $r+1$;
7 3) implement a deterministic protocol for $\sqrt{n/\log n}$ rounds;
8 **end**
9 **if** $\mathtt{decided} = TRUE$ **then**
10 `diff` $\leftarrow N_p^{r-3}$ - N_p^r;
11 **if** $(\mathtt{diff} \leq N_p^{r-2}/10)$ **then** STOP;
12 **else** `decided` $\leftarrow FALSE$;
13 **end**
14 **if** $O_p^r > (7N_p^r - 1)/10$ **then** $b_p \leftarrow 1$, `decided` $\leftarrow TRUE$;
15 **else if** $O_p^r > (6N_p^r - 1)/10$ **then** $b_p \leftarrow 1$;
16 **else if** $O_p^r < (4N_p^r - 1)/10$ **then** $b_p \leftarrow 0$, `decided` $\leftarrow TRUE$;
17 **else if** $O_p^r < (5N_p^r - 1)/10$ **then** $b_p \leftarrow 0$;
18 **else** set b_p to the output of CheapQuantumCoin$(\mathcal{P}, p)$ executed with parameters $d = \log n$, $\alpha = n^\epsilon$;
19 $r \leftarrow r + 1$;
20 **end**
21 **return** b_p ; /* p outputs final decision */

Algorithm's analysis. To analyze the CheapQuantumConsensus algorithm we first recall a combinations of lemmas from [8].

▶ **Lemma 5** (Lemmas 4.1, 4.2, 4.3 in [8]). *If all processes have the same value at the beginning of an iteration of the main while loop, then the algorithm returns the decision after at most two iterations.*

▶ **Theorem 6.** *For any $\epsilon > 0$, the CheapQuantumConsensus algorithm solves Consensus against $n/3$-adversary in $O\left(\left(\frac{1}{\epsilon}\right)^4\right)$ rounds in expectation while using $O(n^{3\epsilon})$ qubits and communication bits per process (amortized), whp.*

Proof. First, we argue for correctness. Compared to the protocol of Bar-Joseph and Ben-Or [8], which works for an arbitrary number of failures $t \leq n$, we changed the method of deriving random coin, c.f., line 18. Observe that even if CheapQuantumCoin fails to meet conditions of t-resilient coin flip, it always outputs some bit in every non-faulty process. Thus, regardless of the number of crashes the output could be an output of local coin flips with a non-zero probability. Since Bar-Josephs's and Ben-Or's algorithm works with probability 1 (see Theorem 3 in [8]), thus CheapQuantumConsensus also achieves correctness with probability 1.

Next, we estimate the expected number of phases (i.e. the number of iterations of the main while loop). We consider only *good* phases, i.e. phases in which the adversary crashed at most $\frac{1}{10}$ fraction of processes that were correct at the beginning of this iteration. Note, that there can be at most $\frac{1}{3}/\frac{1}{10} < 4$ "bad" phases. Let x be the number of non-faulty processes at the beginning of some good phase. We consider the following cases:
Case *a)* There exists a process that in this iteration executes line 15 or line 14. In this case, all other processes have to execute line 14, 15 or line 18, since the number of ones counted in different processes may differ by $\frac{x}{10}$ at most. All processes that in this iteration execute CheapQuantumCoin will set b_p to 1 with probability $\frac{1}{4}$ at least. What follows, in the next iteration all processes will start with b_p set to 1, and by Lemma 5 the algorithms will decide within two next phases.
Case *b)* There exists a process that in this phase executes line 16 or line 17. Similarly to the previous case, we observe that all other processes have to execute line 17 or line 18, since, again, the number of ones counted in different processes may differ by $\frac{x}{10}$ at most. Therefore the same arguments apply but know the final decision will be 0.
Case *c)* None of processes executes one of lines 14, 15, 16, or 17. Thus, all processes participated in CheapQuantumCoin in line 18. By Theorem 12, with probability at least $\frac{1}{4}$, all processes will set value b_p to the same value. Thus, again by applying Lemma 5, we get that the algorithms will decide within the two next phases.

We showed that if a good phase happens, then the algorithm terminates within 2 next iterations with probability at least $\frac{1}{4}$. Since there can be at most 4 bad iterations, thus we can calculate the expected number of iterations as follows: $\mathbb{E}(\#\text{iterations}) = \sum_{i=4}^{\infty} i\left(\frac{1}{4}\right)^i = O(1)$.

Executing a single phase takes $O\left(\left(\frac{1}{\epsilon}\right)^4\right)$ rounds, which is the round complexity of the FastCounting algorithm and an upper bound of the time complexity of the CheapQuantumCoin algorithm, therefore the algorithm terminates in $O\left(\left(\frac{1}{\epsilon}\right)^4\right)$ rounds in expectation. Similarly, by taking the upper bounds on the communication complexity of the algorithms FastCounting and CheapQuantumCoin we get that the expected number of amortized communication bits used by the algorithm is $O(n^{3\epsilon})$. ◀

Handling arbitrary number of crashes. Consider $O(\log n)$ repetitions of the main loop (phases) of the CheapQuantumConsensus algorithm. If during these phases, the processes with value $b_p = 1$ become a large majority (at least $\frac{6}{10}$ fraction of alive processes), then, as discussed before, every process will decide within the next two rounds. The same holds if processes with value $b_p = 0$ start to overpopulate by a ratio of $\frac{6}{10}$ all non-faulty processes. On the other hand, if the cardinalities of the two groups with different values b_p are close to each other, then the processes execute the CheapQuantumCoin algorithm. It outputs a random bit (the same in every participating process), under the assumption that at least a $\frac{2}{3}$ fraction of processes that started this phase as non-faulty have not crashed during this phase. However, in these $O(\log n)$ phases there must be at least one phase in which the property of a $\frac{2}{3}$ fraction of processes survive holds. In what follows, we argue that if the adversary can crash arbitrarily many processes, but smaller than n, then the expected number of phases should still be $O(\log n)$. Now, to obtain the algorithm stated in Theorem 4, we make two more adjustments to the original CheapQuantumConsensus algorithm. In lines 3 and 18, processes execute the algorithms FastCounting and CheapQuantumCoin, respectively, with parameters x, d, α set as follows: $x = 2, d = \log n, \alpha = \log n$. This corresponds to the use of a sparse graph for communication (of degree roughly $O(\log n)$). In consequence, the time complexity of the FastCounting algorithm increases to $O(\log^4 n)$, but the communication complexity decreases to $O(\log^8 n)$ amortized per process. The details of the implementation of the FastCounting algorithm are presented in Section 6, and performance follows from putting the aforementioned parameters to the formulas in Theorem 14. Similarly, the time complexity of the CheapQuantumCoin algorithm increases to $O(\log^3 n)$, but the communication complexity (both quantum and classical) decreases to $O(\log^7 n)$ amortized per process. Formally:

▶ **Theorem 7.** *The modified version of the CheapQuantumConsensus algorithm solves Consensus against any adversary in $O(\log^5 n)$ rounds in expectation while using $O(\log^8 n)$ qubits and communication bits per process (amortized), whp.*

Proof. For correctness, we argue exactly the same as in the proof of the previous Theorem. We also define good and bad phases, with only this difference that now the number of bad phases is at most $O(\log n)$ since the adversary has the full power of crashing an arbitrary number of processes. This being said we get that, by the very same reasoning, the expected number of phases is

$$\mathbb{E}(ITE) = \sum_{i=\log n}^{\infty} i \left(\frac{1}{4}\right)^i = \Theta(\log n) .$$

By examining the time and bits complexity of the algorithms FastCounting and CheapQuantumCoin (c.f. Theorems 14, 12) with parameters $x = 2, d = \log n, \alpha = \log n$, we get a single phase lasts $O(\log^4 n)$ rounds and contributes $O(n \log^7 n)$ bits to the total communication complexity. The latter, after dividing by n, gives the sought complexity amortized per process. Thus, the theorem follows. ◀

5 Qubit-and-Communication Efficient Quantum Common Coin

In this section, we design a new t-resilient weak global coin, for $t < n$, with the help of quantum communication and computation.

On a high level, our protocol CheapQuantumCoin chooses a leader process uniformly at random and all other processes agree on the random bit proposed by the leader. Quantum phenomena are used to hide the random choices of the leader and its output from the adaptive adversary when processes communicate with each other. The idea was first proposed in [9], yet there are key differences between that work and our algorithm. Instead of all-to-all communication, which required large number of qubits, we use a sequence of random graphs of node degrees $d, d\alpha^1, \ldots, d\alpha^k$, respectively, where $d, \alpha \in \Omega(\log n)$ and $k = \lceil \log n / \log \alpha \rceil$ are some integer parameters. The vertices of these graphs correspond to processes and edges correspond to communication links – each process communicates with neighbors in one of the graphs at a time. If the graph is chosen properly (i.e., so that there is no situation in which too many processes use denser graphs), it reduces the communication complexity but simultaneously imposes a new challenge. Mainly, the communication procedure has to now assure the delivery of quantum bits between every two non-faulty processes regardless of the pattern of crashes. For instance, if only one random graph of degree d was used then the adversary could easily isolate any vertex using only $O(d)$ crashes (i.e., by crashing all its neighbors). Hence, strictly speaking, assuring such delivery is not possible while using a sparse communication graph as relays, but we show that a certain majority could still come up with a common coin value based only on their exchanges with neighbors in the communication graphs; they could later propagate their common value to other processes by adaptively controlling their (increasing) set of neighbors, taken from subsequent communication graphs of increasing density. A thorough analysis shows that in this way it is possible to achieve the same quantum properties that are guaranteed by Ben-Or's and Hassidim's global coin [9], and at the same time reducing the quantum communication by a polynomial factor.

Algorithm's description. We now describe the CheapQuantumCoin algorithm. Its pseudocode is presented in Figure 2. It takes as input: a process name p and two integers, d and α. The two latter parameters are used to determine communication patterns between non-faulty processes, and their choice determines the complexity of the algorithm.

As mentioned before, processes use the quantum equivalent of a procedure in which processes draw random names from the range $[1, \ldots, n^3]$ and decide on a random bit proposed by the process with the largest name.[7] We view the quantum part of the algorithm as a quantum circuit on the joint space of all qubits ever used by different processes. Due to the distributed nature of the system, not all quantum operations are allowed by the quantum circuit. That is, (i) any process can perform arbitrary unitary gates on its part of the system, (ii) unitary gates on qubits of different processes might be performed but must be preceded by quantum communication that sends qubits involved in the operation to a single process. The communication can be either direct or via relays. We use the following unitary gates to simulate the classical algorithm of the leader election in the quantum distributed setting: `Hadamard` gate, `CNOT` and `Pairwise_CNOT` gates, and `F_CNOT` and `Controlled_Swap` gates. (Although they are all standard quantum gates, a thorough description of these gates can be found in the full version.)

Let us now explain how all the gates listed above work together. As mentioned in the description of the Hadamard gate, after line 2 ends, the main registers of the system are in the state of being a uniform superposition of all vectors of the computational basis. Starting from this point, the composition of all gates applied to different registers along the execution can be viewed as a single unitary gate on the entire system, consisting of the qubits that any process ever created. Note that the unitary transformation might be different depending on the failure in communication, i.e., a failure in the delivery of some block of qubits between two processes may result in abandoning gates involving these qubits, but for a moment let us assume that the links are reliable. Since the unitary transformation is linear, it is enough to consider how it affects the vectors of the computational basis. However, all the gates described above behave on the computational basis as their classical equivalents. More precisely, let $|x\rangle$ be a vector from the computational basis spanning the whole circuit. Let p be the process whose main register has the largest[8] value in the state $|x\rangle$. From the point of view of this register, the following happens in the algorithm. In each round, p creates an entangled state on $6 \log n + 2$ qubits (see point (2)) that has the same qubits on its new block of $3 \log n + 1$ qubits as it has on the main register. Then, it propagates the new block to its neighbors (lines 9- 12). The neighbors compare the content of received qubits and exchange them with their main register if their content is smaller (gates `F_CNOT` and `Controlled_Swap` in lines 16- 17). This operation is then repeated $(k+2)^2(\gamma_\alpha + 1)$ times on the set of links defined by some random evolving graphs, see the later paragraph about adaptive communication pattern. In the end, the processes who, either directly or via relays, received the content of the largest main register, have the same value in their main register. Therefore, the result of the measurement in line 27 must be the same in all these processes.

Assume now that we are able to achieve bidirectional quantum communication between any pair of processes of an τ fraction of the entire system, regardless of the (dynamic) actions of the adversary, for some constant $0 \leq \tau \leq 1$. From the above, the algorithm transforms any vector whose largest main register is one of the registers of the τ fraction of the processes to a

[7] Note that the latter procedure cannot be used against an adaptive adversary, as it could crash such a leader.

[8] The probability of a tie is polynomially small.

vector such that the processes from the τ fraction have the same values in the main registers. The initial state of the system is a uniform superposition which is a linear combination of all vectors from the computational basis. From this perspective, the normalized sum of coefficients of these vectors of the superposition that have the property that the largest register is one of the registers of the τ fraction is $\tau - o(1)$.[9] Thus, we get that the probability of measuring the same values in the processes of the τ fraction is at least $\tau - o(1)$ and we can claim the following lemma.

▶ **Lemma 8.** *Let A be a set of correct processes such that any pair of them was connected by quantum communication either directly or by relays. Then the probability that all processes from A output the same bit from the algorithm* CheapQuantumConsensus *is at least* $\frac{|A|}{n} - o(1)$.

Adaptive communication pattern. As explained, we not only need that communication should be efficient in terms of the number of qubits and classical bits, but also it should be such that any two correct processes of a large fraction of the entire system are connected by a short path of the correct process so that quantum registers can be relayed. Let d, α be two integer parameters. We define $k = \left\lceil \frac{\log(n/d)}{\log \alpha} \right\rceil$, $\gamma_\alpha = \frac{\log n}{\log \alpha}$, and $\delta_\alpha = \frac{2}{3}\alpha$. Initially, each process p draws independently $k+1$ sets $\mathcal{N}_p(d), \mathcal{N}_p(d\alpha^1), \ldots, \mathcal{N}_p(d\alpha^k)$, where a set $\mathcal{N}_p(d\alpha^i)$, for $0 \le i \le k$, includes each process from $\mathcal{P}$ with probability $\frac{d\alpha^i}{n}$.

Communication is structured into $(k+2)^2$ *epochs*, see line 4. Each epoch consists of $2(\gamma_\alpha + 1)$ communication rounds, also called *testing* rounds. They are scheduled in $\gamma_\alpha + 1$ iterations within the loop "for" in line 7, each iteration containing two communication rounds (underlined in the pseudocode): sending/receiving inquiries in line 9 and sending/receiving responses in line 12. In the testing rounds of the first epoch, a process p sends inquiries to processes in set $\mathcal{N}_p(d)$. The inquired processes respond by sending in the next round (line 12) their current classical state and specially prepared, in line 6, quantum register. However, if in a result of crashes p starts receiving less than δ_α responses per round, it switches its communication neighborhood from $\mathcal{N}_p(d)$ to the next, larger set, $\mathcal{N}_p(d \cdot \alpha)$. A similar adaptation to a crash pattern is continued in the remaining epochs.

Process p stores the cardinally of the set being inquired in an epoch in the variable $\texttt{degree}_p$ (initialized to d in line 2). For the purpose of testing rounds, p copies the value $\texttt{degree}_p$ to a variable $\texttt{adaptive\_degree}_p$ (line 6). In every testing round, p adapts its variable $\texttt{adaptive\_degree}_p$ to the largest value $x \le \texttt{adaptive\_degree}_p$ such that it received at least δ_a responses from processes that have their variable `adaptive_degree` at least x (loop "while" in line 19). If p had to decrease the value $\texttt{adaptive\_degree}_p$ in testing rounds of an epoch, it then *increases* the main variable $\texttt{degree}_p$ by the factor α before the next epoch, see line 24. The latter operation formally encodes the intuition that the process p expected to have δ_α non-faulty neighbors with their values of `degree` at least as big as its own, but due to crashes it did not happen; Therefore, p increases the number of inquired processes, by adopting the next, larger neighborhood set $\mathcal{N}_p(\cdot)$, randomly selected, in order to increase the chance of communication with the majority of non-faulty processes in the next epoch. On the other hand, the adaptive procedure of reducing `adaptive_degree` in testing rounds of a single epoch helps neighbors of p to estimate correctly the size of the

[9] The $o(1)$ part contributes to the normalized sum of these vectors that correspond to having more than one largest register.

Algorithm 2 CheapQuantumCoin for process p.

input: p, two parameters: d, α

For $0 \le i \le \left\lceil \frac{\log(n/d)}{\log\alpha} \right\rceil$: $\mathcal{N}_p(d\alpha^i) \leftarrow$ a set of processes such that each process is chosen independently with probability $\frac{d\alpha^i}{n}$;

$\texttt{degree}_p \leftarrow d$, $\quad \gamma_\alpha \leftarrow \frac{\log n}{\log\alpha}$, $\quad \delta_\alpha \leftarrow \frac{2}{3}\log n$;

$|Leader\rangle_p |Coin\rangle_p \leftarrow H^{\otimes 3\log n+1} |00\ldots0\rangle$ (a gate on $3\log n + 1$ qubits) ;

for $i = 1$ *to* $(k+2)^2$; /* iter. of epochs */

do

    $\texttt{adaptive\_degree} \leftarrow \texttt{degree}_p$;

    for $j = 1$ *to* $\gamma_\alpha + 1$; /* iter. of testing rounds */

    do

        send to each process in $\mathcal{N}_p(\texttt{degree}_p)$: an inquire bit 1 ;

        $\mathcal{I} \leftarrow$ the set of processes who sent an inquire bit to p ;

        $\forall_{q\in\mathcal{I}} : |B\rangle_q \leftarrow \texttt{Pairwise\_CNOT}\left(|LeaderCoin\rangle_p, |0\ldots0\rangle\right)$;

        send to each process $q \in \mathcal{I}$: a quantum message containing first $3\log n+1$ bits of $|B_q\rangle$, and a classical message containing $\texttt{adaptive\_degree}_p$;

        $\mathcal{R} \leftarrow$ the set of processes who responded to p's inquires ;

        foreach $q \in \mathcal{R}$ **do**

            $|CLeader\rangle_q |CCoin\rangle_q \leftarrow$ received quantum bits from q, $\quad |S\rangle \leftarrow |0\rangle$;

            $\texttt{F\_CNOT}_{|Leader\rangle_p > |CLeader\rangle_q}\left(|Leader\rangle_p, |CLeader\rangle_q, |S\rangle\right)$;

            $\texttt{Controlled\_Swap}\left(|LeaderCoin\rangle_p, |CLeaderCCoin\rangle_q, |S\rangle\right)$;

        end

        while $|\{q \in \mathcal{R} : \textit{adaptive\_degree}_q \ge \textit{adaptive\_degree}_p\}| < \delta_\alpha$ *and* $\textit{adaptive\_degree}_p \ge d$; /* adapting #neighbors during testing */

        do

            $\texttt{adaptive\_degree}_p \leftarrow \frac{1}{\alpha}\texttt{adaptive\_degree}_p$;

        end

    end

    if $\textit{adaptive\_degree}_p < \textit{degree}_p$ **then**

        $\texttt{degree}_p \leftarrow \min\{\texttt{degree}_p \cdot \alpha, d\alpha^k\}$; /* neighborhood for next epoch grows */

end

$b_p \leftarrow$ be the last bit of the result of measuring $|Leader\rangle_p |Coin\rangle_p$ in the computational basis;

return b_p ; /* p outputs random bit */

neighborhood that process p is using in the current testing round, which might be much smaller than the value $\texttt{degree}_p$ from the beginning of the epoch. This, in turn, calibrate the value of $\texttt{adaptive\_degree}$ of the neighbors of p, and this calibration can propagate to other processes of distance up to γ_α from p in the next iterations of testing rounds.

Analysis. Let us define graphs $\mathcal{G}(d\alpha^i)$, for $0 \le i \le k$, as the union of random sets $\cup_{p\in\mathcal{P}}\mathcal{N}_p(d\alpha^i)$. The probability distribution of the graph $\mathcal{G}(d\alpha^i)$ is the same as the random graph $G(n, y)$ for $y = \frac{d\alpha^i}{n}$. Chlebus, Kowalski and Strojnowski [12] showed in their Theorem 2, applied for $k = \frac{64n}{d\alpha^{i-1}}$, that the graph $\mathcal{G}(d\alpha^i)$ has the following properties, whp:

(i) it is $(\frac{n}{d\alpha^{i-1}})$-expanding, which follows from $(\frac{n}{d\alpha^{i-1}}, \frac{2}{3}\frac{n}{d\alpha^{i-2}}, \frac{4}{3}\frac{n}{d\alpha^{i-2}})$-edge-expanding property,

(ii) it is $(\frac{n}{d\alpha^{i-1}}, \frac{1}{3}\alpha, \frac{2}{3}\alpha)$-edge-dense,

(iii) it is $(16\frac{n}{d\alpha^{i-1}}, 3/4, \frac{2}{3}\alpha)$-compact,

(iv) the degree of each node is at most $\frac{21}{20}d\alpha^i$.

Since the variable $\texttt{degree}_p$ takes values in the set $\{d, d\alpha^1, \ldots, d\alpha^k\}$, the pigeonhole principle assures that eventually p will participate in an epoch in which $\texttt{degree}_p$ has not been increased. In the most severe scenario, p will use the set $\mathcal{N}_p(d\alpha^k)$, which consists of all other processes – because it contains each process, as a neighbor of p, with probability 1. The $(\gamma_\alpha, \delta_\alpha)$-dense-neighborhood property of random graphs composed from neighborhoods $\mathcal{N}(\texttt{degree}_p)$ implies that p will then contact a majority of other non-faulty processes via at most $\gamma_\alpha + 1$ intermediate processes (during testing rounds). Formally, the following holds:

▶ **Lemma 9.** *If a process p does not change its variable $\texttt{degree}_p$ at the end of an epoch i, then at the beginning of epoch i there exists a $(\gamma_\alpha, \delta_\alpha)$-dense-neighborhood of p in the graph $\mathcal{G}(\texttt{degree}_p)$ consisting of non-faulty processes with variable $\texttt{degree}$ being at least $\texttt{degree}_p$ in the epoch i, whp.*

On the other hand, $(16n/d\alpha^{i-1}, 3/4, 2\alpha/3)$-compactness of the (random) graph composed of processes that have the variable $\texttt{degree}$ at least $d\alpha^i$, guarantees that the total number of processes that use sets $\mathcal{N}(d\alpha^i)$ during the epoch i is at most $\frac{n}{\alpha^{i-2}}$, which amortizes communication complexity.

▶ **Lemma 10.** *For any integer i, such that $0 \le i \le k$, at the beginning of each epoch there is at most $\frac{16n}{d\alpha^{i-2}}$ processes with the variable $\texttt{degree}$ greater than $d\alpha^i$, whp.*

In the above proof of Lemma 10, we use the fact that a suitable set C of processes that have been correct throughout the whole epoch exists. We may choose this set and argue about it after the epoch, as the communication pattern in the algorithm is deterministic. Hence, in any round of the epoch, processes in C have at least as many non-faulty neighbors in the communication graph as they have neighbors from set C. We use this number of neighbors in C as a *lower bound* to argue about the behavior of variables $\texttt{degree}$; therefore, our arguments do not depend on the behavior of processes outside of C and whether/when some of them fail during the epoch.

▶ **Lemma 11.** *Any two non-faulty processes p and q were connected by a quantum path of communication during the run of the algorithm.*

The above lemmas yield the following result.

▶ **Theorem 12.** *For two integer parameters $d, \alpha \in \Omega(\log n)$, the algorithm* QuantumCoin-Flip *generates a weak global coin, provided that at most $\frac{1}{3}$-fraction of initially non-faulty processes have crashed. It terminates in $O\Big(\big(\frac{\log n}{\log \alpha}\big)^3\Big)$ rounds and with high probability uses $O\Big(\big(\frac{\log n}{\log \alpha}\big)^4 d\alpha^2 \log n\Big)$ both quantum and classical communication bits (amortized) per process.*

Proof of Theorem 12. Let $\mathcal{H} \subseteq \mathcal{P}$ be the set of initially non-faulty processes. Assume that at least $\frac{2}{3}|\mathcal{H}|$ of them remains non-faulty during the execution of the algorithm. By Lemma 11, any pair of processes from $\mathcal{H}$ is connected by quantum communication, therefore applying Lemma 8 to this set, give that there is at least $\frac{2}{3} - o(1)$ (which is greater than $\frac{1}{2}$ for sufficiently large n) probability that all non-faulty processes return the same output bit. Since 0 and 1 are equally likely, thus the probabilistic guarantee on the weak global coin follows.

The number of rounds is deterministic and upper bounded by $O(k^2 \cdot \gamma_\alpha) = O\Big(\big(\frac{\log(n/d)}{\log \alpha}\big)^2 \frac{\log n}{\log \alpha}\Big) = O\Big(\big(\frac{\log n}{\log \alpha}\big)^3\Big)$. To bound the communication complexity, assume that each graph $\mathcal{G}(d\alpha^i)$, for $0 \le i \le k$, satisfies the desired expanding properties listed in the description of the algorithm. This, by the union bound argument, holds whp. By Lemma 10 at the beginning of each epoch there are at most $\frac{16n}{d\alpha^{i-2}}$ processes that inquire more than $d\alpha^i$ other processes in testing rounds of this epoch, for each $0 \le i \le k$. Since each message uses at most $O(\log n)$ bits and qubits, thus a single testing round in an epoch adds at most $\sum_{i=0}^{k} \frac{16n}{d\alpha^{i-2}} \cdot d\alpha^i \le 16kd\alpha^2 \cdot n\log n$ qubits and bits to communication complexity. Since there is exactly $O\Big(\big(\frac{\log n}{\log\alpha}\big)^3\Big)$ testing rounds, thus the $O\Big(\big(\frac{\log n}{\log\alpha}\big)^4 d\alpha^2 \cdot n\log n\Big)$ upper bound on the total communication complexity of the algorithm follows. Dividing the latter by n we receive amortized formula per process. ◀

6 Constant-Time Communication-Efficient Fuzzy Counting

Although generating a weak global coin in a constant number of rounds against an adaptive adversary requires quantum communication (due to the lower bound by Ben-Or and Bar-Joseph [7]), the CheapQuantumCoin algorithm, even without quantum communication, achieves few other goals. As discussed in the previous section, its random communication pattern guarantees, whp, that any additional classical message, also called a rumor, of a non-faulty process can be conveyed to any other non-faulty process if added to every classical message sent/received in line 12. Even more, assume that there is a set of x messages/rumors $M = \{m_1, \ldots, m_x\}$, distributed as inputs among some subset of processes (one message from M per process). If processes always convey all the known rumors from set M when using classical communication (avoiding repetition), then they solve a Gossip problem, whp, i.e., every rumor m_i given to a non-faulty process, for $1 \le i \le x$, is known at the end of the protocol to every other non-faulty process. Observe that processes resign from the quantum content of communication for this purpose, and instead of $\log n$ bits (or qubits) per message, they use $|M|$ bits, where $|M|$ denotes the number of bits needed to encode all rumors from M. Finally, if processes work in a model where the names of other processes are commonly known, they can withdraw from using random communication. Instead, they can use a deterministic family of graphs $\mathcal{G}(d\alpha^i)$, for the same choice of parameters d and α. The proof of existence of such graphs, using the probabilistic argument, was described in [12] (Theorem 2). In such case, the set $\mathcal{N}_p(d\alpha^i)$ is the neighborhood of process p in the deterministic graph $\mathcal{G}(d\alpha^i)$. (Processes compute the same copies of graphs $\mathcal{G}$ locally in the beginning of the algorithm.) The above augmentation of the algorithm, together with the proof of Theorem 12, from which we take the upper bound on the number of messages send and the upper bound on the number of rounds, gives:

▶ **Theorem 13.** *For integer parameters $d, \alpha \in \Omega(\log n)$, there is a deterministic algorithm that solves the gossip problem in $O\left(\left(\frac{\log n}{\log\alpha}\right)^3\right)$ rounds using $O\left(\left(\frac{\log n}{\log\alpha}\right)^4 d\alpha^2 \cdot (|M| + \log n)\right)$ communication bits per process (amortized), where $|M|$ is the number of bits needed to encode all rumors in M.*

Generalized Fuzzy Counting. In this part, we refine the state-of-art method of solving the Fuzzy Counting problem (c.f. Definition 3), even deterministically, and propose a new recursive algorithm with any branching factor x, called FastCounting. We prove the following:

▶ **Theorem 14.** *For any $2 \le x \le n$ and $\delta, \alpha \in \Omega(n)$, the FastCounting deterministic algorithm solves the Fuzzy Counting problem in $O\Big(\frac{\log n}{\log x}\big(\frac{\log n}{\log\alpha}\big)^3\Big)$ rounds, using $O\Big(\frac{\log n}{\log x}\big(\frac{\log n}{\log\alpha}\big)^4 d\alpha^2 \cdot x\log n\Big)$ communication bits (amortized) per process.*

Obviously, the constant-time is achieved in Theorem 14 when $x, \alpha = n^{\epsilon}$, for a constant $\epsilon \in (0,1)$. In this case, the number of rounds is $O\left(\left(\frac{1}{\epsilon}\right)^4\right)$, while the communication complexity is $O(n^{3\epsilon} \log^2 n)$ (amortized) per process. In our approach, we generalize the method of Hajiaghayi et al. [23] to denser communication graphs of certain properties, which allows us to achieve constant running time. The constant running time is the key feature of the algorithm, since it is used in the implementation of (expected) constant-round quantum consensus protocol CheapQuantumConsensus. The main difference between ours and the state-of-art approach is a different Gossip protocol used in the divide-and-conquer method.

The FastCounting algorithm is recurrent. It takes as an input the following values: $\mathcal{P}$ is the set of processes on which the algorithm is executed; p is the name of a process which executes the algorithm; $a_p \in \{0,1\}$ denotes if p is active ($a_p = 1$) or not; and parameters x, d, α, where x is the branching factor and d, α steer the density of the communication graph in the execution. Knowing the set $\mathcal{P}$ of n processes, FastCounting splits the set into x disjoint groups of processes, each of size between $\lfloor \frac{n}{x} \rfloor$ and $\lceil \frac{n}{x} \rceil$. Name these groups $\mathcal{P}_1, \ldots, \mathcal{P}_x$. The groups are known to each participating process. The algorithm then makes x parallel recursive calls on each of these groups. As a result, a process p from a group $\mathcal{P}_i$, for $1 \leq i \leq x$, gets the number of the active processes in its group $\mathcal{P}_i$. In the merging step, all processes execute Gossip algorithm, proposed in Theorem 13, with parameters d, α, where the input rumors are the numbers calculated in the recursive calls. To keep the communication small, when processes learn new rumors they always store at most one rumor corresponding to each of the x groups. This way, the number of bits needed to encode all rumors is $O(x \log n)$. Let $r_1, \ldots, r_\ell$ be the rumors learned by process p from the execution of the Gossip algorithm. The final output of p is the sum $\sum_{i=1}^{\ell} r_i$. The pseudocode of the algorithm can be found in the full version.

7 Future Work

We believe that the outcome of our work effectively establishes the foundation for the efficient integration of classical fault-tolerant distributed and quantum computing. We show that in the basic model of failures, crash failures, a fundamental problem of reaching consensus fast is feasible with only polylog n qubit communication. An immediate open question arises whether such performance improvement is possible in the case of more severe failures, such as omissions, authenticated Byzantine, or even full Byzantine. Another interesting idea is to approach the communication bound from the lower bound side. We believe that investigating even conditional lower bounds on the minimal quantum communication needed to solve various distributed problems, foremost the Consensus problem, could lead to many breakthroughs in both distributed and quantum realms.

References

1. I. Abraham, T. -H. Hubert Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the acm Symposium on Principles of Distributed Computing (podc)* and 2019, editors, *Distributed Computing*, pages 317–326, 2018.
2. Dan Alistarh, James Aspnes, Valerie King, and Jared Saia. Communication-efficient randomized consensus. *Distributed Computing*, 31:489–501, 2018.
3. S. Amdur, S. Weber, and V. Hadzilacos. On the message complexity of binary agreement under crash failures. *Distributed Computing*, 5:175–186, 1992.

4 J. Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. *J. ACM*, 45(3):415–450, 1998.

5 H. Attiya and K. Censor. Tight bounds for asynchronous randomized consensus. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 11–13, California, USA, June, 2007. San Diego.

6 H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. Wiley, 2nd edition, 2004.

7 Z. Bar-Joseph and M. Ben-Or. A tight lower bound for randomized synchronous consensus. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 193–199, 1998.

8 Z. Bar-Joseph and M. Ben-Or. A tight lower bound for randomized synchronous consensus, 2002. `doi:10.1145/277697.277733`.

9 M. Ben-Or and A. Hassidim. Fast quantum byzantine agreement. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

10 A. Broadbent and A. Tapp. Can quantum mechanics help distributed computing? *SIGACT News*, 39(3):67–76, 2008.

11 B. S. Chlebus and D. R. Kowalski. Robust gossiping with an application to consensus. *Journal of Computer and System Sciences*, 72(2006):1262, 2009.

12 B. S. Chlebus, D. R. Kowalski, and M. Strojnowski. Fast scalable deterministic consensus for crash failures. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 111–120, 2009.

13 B. S. Chlebus, D. R. Kowalski, and M. Strojnowski. Scalable quantum consensus for crash failures. In *Proceedings of the 24th International Symposium on Distributed Computing (DISC)*, 2010.

14 V. Cholvi. Quantum byzantine agreement for any number of dishonest parties. *Quantum Inf. Process.*, 21:1–11, 2022.

15 B. Chor, M. Merritt, and D. B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *J. ACM*, 36(3):591–614, 1989.

16 D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM*, 32:191–204, 1985.

17 D. Dong and I. R. Petersen. Quantum estimation, control and learning: Opportunities and challenges. *Annual Reviews in Control*, 54:243–251, 2022.

18 C. Dwork, J. Halpern, and O. Waarts. Performing work efficiently in the presence of faults. *SIAM Journal on Computing*, 27:1457–1491, 1998.

19 M. Fisher and N. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14:183–186, 1982.

20 Z. Galil, A. Mayer, and M. Yung. Resolving message complexity of byzantine agreement and beyond. In *Proceedings of the* 36*th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1995.

21 S. Gilbert and D. R. Kowalski. Distributed agreement with optimal communication complexity. In *Proceedings of the* 21*st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 965–977, 2010.

22 V. Hadzilacos and J. Y. Halpern. Message-optimal protocols for byzantine agreement. *Mathematical Systems Theory*, 26:41–102, 1993.

23 M. Hajiaghayi, D. R. Kowalski, and J. Olkowski. Improved communication complexity of fault-tolerant consensus. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

24 L. K. Helm, quantum distributed consensus. In *Proceedings of the 27th ACM symposium on Principles of distributed computing (PODC)*, 2008.

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

26 Q. Luo, K. Feng, and M. Zheng. Quantum multi-valued byzantine agreement based on d-dimensional entangled states. *International Journal of Theoretical Physics*, 58:4025–4032, 2019.
27 M. Marcozzi and L. Mostarda. Quantum consensus: an overview. *arXiv preprint*, 2021. arXiv:2101.04192.
28 L. Mazzarella, A. Sarlette, and F. Ticozzi. Consensus for quantum networks: Symmetry from gossip interactions. *IEEE Transactions on Automatic Control*, 60(1):158–172, 2014.
29 C. Mochon. Quantum weak coin flipping with arbitrarily small bias. *arXiv preprint*, 2007. arXiv:0711.4114.
30 M. A. Nielsen and I. Chuang. Quantum computation and quantum information. *Phys. Today*, 54(2):60, 2001.
31 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27:228–234, 1980.
32 S. Tani, H. Kobayashi, and K. Matsumoto. Exact quantum algorithms for the leader election problem. *ACM Transactions on Computation Theory*, 4:1–24, 2012.
33 J. van Apeldoorn and T. de Vos. A framework for distributed quantum queries in the congest model. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2022.
34 W. Wang, Y. Yu, and L. Du. Quantum blockchain based on asymmetric quantum encryption and a stake vote consensus algorithm. *Scientific Reports*, 12(1):1–12, 2022.
35 X. Wu and P. Yao. Quantum complexity of weighted diameter and radius in congest networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2022.
36 L. Zhang and S. Fulton. Discussion of quantum consensus algorithms. *arXiv preprint*, 2022. arXiv:2206.04710.