

Byzantine-Resilient Counting in Networks

Soumyottam Chatterjee

Gopal Pandurangan

Peter Robinson

Abstract

We present two distributed algorithms for the *Byzantine counting problem*, which is concerned with estimating the size of a network in the presence of a large number of Byzantine nodes.

In an n -node network (n is unknown), our first algorithm, which is *deterministic*, finishes in $O(\log n)$ rounds and is time-optimal. This algorithm can tolerate up to $O(n^{1-\gamma})$ arbitrarily (adversarially) placed Byzantine nodes for any arbitrarily small (but fixed) positive constant γ . It outputs a (fixed) constant factor estimate of $\log n$ that would be known to all but $o(1)$ fraction of the good nodes. This algorithm works for *any* bounded degree expander network. However, this algorithm assumes that good nodes can send arbitrarily large-sized messages in a round.

Our second algorithm is *randomized* and most good nodes send only small-sized messages.¹ This algorithm works in *almost all* d -regular graphs. It tolerates up to $B(n) = n^{\frac{1}{2}-\xi}$ (note that n and $B(n)$ are unknown to the algorithm) arbitrarily (adversarially) placed Byzantine nodes, where ξ is any arbitrarily small (but fixed) positive constant. This algorithm takes $O(B(n) \log^2 n)$ rounds and outputs a (fixed) constant factor estimate of $\log n$ with probability at least $1 - o(1)$. The said estimate is known to most nodes, i.e., $\geq (1 - \beta)n$ nodes for any arbitrarily small (but fixed) positive constant β .

To complement our algorithms, we also present an impossibility result that shows that it is impossible to estimate the network size with any reasonable approximation with any non-trivial probability of success if the network does not have sufficient vertex expansion.

Both algorithms are the first such algorithms that solve Byzantine counting in sparse, bounded degree networks under very general assumptions. Both algorithms are fully local and need no global knowledge. Our algorithms can be used for the design of efficient distributed algorithms resilient against Byzantine failures, where the knowledge of the network size — a global parameter — may not be known a priori.

Keywords: Byzantine counting, expander graphs, Byzantine faults, randomization, network size estimation.

¹Throughout this paper, a small-sized message is defined to be one that contains $O(\log n)$ bits in addition to at most a constant number of node IDs.

1 Introduction

The recent surge in the popularity of decentralized peer-to-peer protocols has renewed the interest in achieving Byzantine fault-tolerance in sparse networks of untrusted participants. In this work, we study the fundamental problem of *Byzantine counting* where the goal is to estimate the number of nodes in a network in the presence of a large number of Byzantine nodes. We say that a node u is *good* or *honest* if u is not a Byzantine node. We assume that the Byzantine nodes are *arbitrarily* distributed in the network and that when a Byzantine node sends a message over an edge, it cannot fake its ID. We note that both of these assumptions are quite typical in the literature [18, 46, 32, 7, 3, 4]. Please refer to Section 2 for more details on the distributed computing model.

We focus on the Byzantine counting problem in the context of *sparse* networks because of the following reasons.

1. Peer-to-peer networks and most other large-scale, real-world networks happen to be sparse.
2. In a d -regular network, if the degree d is a non-constant function of n , e.g., if $d = \Theta(\log n)$, then it might become trivial for a node to estimate n from its knowledge of its own degree d .

Essentially all known algorithms studied in the literature for solving problems like *Byzantine consensus* and *Byzantine leader election* in sparse networks require an underlying *expander graph*: the expansion property is *needed* in tolerating a large number of Byzantine nodes. The vertex expansion of a graph $G = (V, E)$ on n nodes is defined as

$$h(G) = \min_{0 < |S| \leq \frac{n}{2}} \frac{|Out(S)|}{|S|},$$

where S is any subset of V of size at most $\frac{n}{2}$ and $Out(S)$ is the set of neighbors of S in $V \setminus S$. In particular, the seminal paper of Dwork et al. [18], which introduced and studied the problem of almost-everywhere Byzantine agreement in bounded degree graphs showed that such an agreement is achievable in *almost all* d -regular graphs (i.e., all but a vanishingly small fraction of such graphs). We exploit the following fact in our current work: almost all d -regular graphs possess good expansion properties.

However, these algorithms assume knowledge of at least an estimate of the *size of the network* (in many cases, an estimate of the logarithm of the network size suffices) and related parameters such as the network diameter or the mixing time. In fact, the result of Dwork et al. assumes that all nodes know the global network topology. This suggests that it is non-trivial to design algorithms that work *without* knowledge of these global network parameters in bounded-degree (or d -regular) expander networks. In such networks, nodes have a limited local view that is highly symmetric, and this enables Byzantine nodes to fake the presence (or absence) of parts of the network.

The goal of our algorithms is to guarantee that most of the honest (i.e., non-Byzantine) nodes obtain a good estimate of the network size. We note that obtaining “almost-everywhere” knowledge is the best one can hope for in such networks [18]. Byzantine counting is related to, yet different from, other fundamental problems in distributed computing, namely, *Byzantine agreement* and *Byzantine leader election*. Similar to the latter two problems, it involves solving a global problem under the presence of Byzantine nodes. However, it is a different problem, since protocols for Byzantine agreement or leader election do not necessarily yield a protocol for Byzantine counting. In fact, many existing algorithms for these two problems (discussed below and in Section 1.1) assume knowledge of n , the number of nodes in the network. In sparse networks, they require at least a reasonably good estimate of n , typically a constant factor estimate of $\log n$ is needed and usually sufficient (as explained in Section 1.1). Indeed, one of the main motivations for this paper is

to design distributed protocols in sparse networks that can work with little or no global knowledge, including the network size. An efficient protocol for the Byzantine counting problem can serve as a preprocessing step for protocols for Byzantine agreement, leader election, and other problems that either require or assume knowledge of an estimate of $\log n$ [5] (cf. Section 1.1).

Byzantine agreement and leader election have been studied extensively for several decades. Dwork et al. [18], Upfal [46], and King et al. [32] studied the Byzantine agreement problem in *sparse (bounded-degree) expander networks* under the condition of *almost-everywhere* agreement, where *almost* all (honest) processors need to reach agreement as opposed to *all* nodes agreeing as required in the standard Byzantine agreement problem. Dwork et al. [18] showed how one can achieve almost-everywhere agreement under up to $\Theta(\frac{n}{\log n})$ of Byzantine nodes in a bounded-degree *expander* network (n is the network size). Subsequently, Upfal [46] gave an improved protocol that can tolerate up to a linear number of faults in a bounded degree *expander* of *sufficiently large spectral gap*. These algorithms required polynomial number of rounds in the CONGEST model (where honest nodes send only small-sized messages) and required $O(\log n)$ rounds in the LOCAL model (where there is no restriction on the message sizes) and polynomial (in n) number of messages. (For a comparison, similarly, our Local algorithm takes $O(\log n)$ rounds and our Congest algorithm takes polynomial number of rounds.) Moreover, for Upfal’s algorithm the local computation required by each processor is exponential. The work of King et al. [32] was the first to study scalable (polylogarithmic communication and number of rounds, and polylogarithmic computation per processor) algorithms for Byzantine leader election and agreement. All of the above algorithms require knowledge of the network topology (including the knowledge of n) — nodes need to have this information hardcoded from the very start.

The works of [7], [3], and [4] studied stable agreement, Byzantine agreement, and Byzantine leader election (respectively) in dynamic networks (see also [5]), where in addition to Byzantine nodes there is also adversarial churn. All these works assume that there is an underlying bounded-degree regular expander graph (in fact, Dwork et al. among others assume d -regular random graphs which are expanders with high probability) and *all nodes are assumed to have knowledge of n* . It was not clear how to estimate n without additional information under presence of Byzantine nodes in such (essentially, regular and constant degree expander) networks. In fact, the works of [5, 4] raised the question of designing protocols in expander networks that work when the network size is not known and may even change over time, with the goal of obtaining a protocol that works when nodes have strictly local knowledge. This requires devising a distributed protocol that can measure global network parameters such as size, diameter, average degree, etc. in the presence of Byzantine nodes in sparse networks, especially in sparse *expander* networks.

Motivated by the above considerations, the work of Chatterjee et al. [14] studied the Byzantine counting problem in a “*small-world*” expander network under the assumption that the Byzantine nodes are *randomly* distributed (cf. Section 1.2 for more details). They present a distributed algorithm running in polylogarithmic (in n) rounds in the CONGEST model that can output a constant factor estimate of $\log n$, where n is the (unknown) network size under the presence of $O(n^{1-\gamma})$ Byzantine nodes, where $\gamma > 0$ can be any arbitrarily small (but fixed) constant. While this presents the first known Byzantine counting algorithm under this setting, it has two major drawbacks.

First, it does not work when Byzantine nodes are *arbitrarily distributed* — it crucially needs that they be *randomly distributed*.

Second, it does not work for (just) expander networks; it needs additional structure, namely a *small-world* network, i.e., a network that has a large clustering coefficient.² The work of Chatterjee

²i.e., a Watts-Strogatz type network similar to [48, 9].

et al. crucially relies on the small-world property in its estimation of the network size. Hence the algorithm and techniques used in that paper [14] are *not* directly applicable to the present paper. Indeed, this paper uses a different approach compared to that of [14] (cf. Section 1.2). While prior works on Byzantine agreement and leader election required only (sparse) expander networks [18, 46, 32] under an arbitrary distribution of Byzantine nodes, Chatterjee et al. remark that:

“... for the Byzantine counting problem, which seems harder, however, expansion by itself does not seem to be sufficient.”

In this paper, we show that Byzantine counting can indeed be solved in expander networks and almost all d -regular graphs under arbitrarily (adversarially) placed Byzantine nodes. This is the setting that is typically assumed in prior works on Byzantine agreement and leader election problems (e.g., [18, 46, 32, 7, 3, 4]).

Throughout this paper, we use the following terminology.

1. We use the terms *sparse network* and *bounded-degree network* synonymously — each describing a network where the maximum degree of a node is bounded by a constant, and hence the number of edges is linear in the number of vertices.
2. A *small-sized message* is defined to be one that contains $O(\log n)$ bits in addition to at most a constant number of node IDs.
3. We use the term *most nodes* or *most good nodes* to indicate $\geq (1 - \beta)n$ nodes, where n is the total number of nodes in the network (and is an unknown quantity in the context of this paper) and β is any arbitrarily small (but fixed) positive constant.
4. By *efficient algorithms* we mean algorithms that use small-sized messages and run in $\text{polylog}(n)$ time.

1.1 Our Contributions

We present two distributed algorithms for the *Byzantine counting problem*, which is concerned with estimating the size (more specifically, the logarithm of the size, as considered here) of a sparse network in the presence of a large number of Byzantine nodes.

Let the network be denoted by $G = (V, E)$; let $n = |V|$ denote the (unknown) network size. Our first algorithm is *deterministic* and finishes in $O(\log n)$ rounds in the LOCAL model and is time-optimal. This algorithm can tolerate up to $O(n^{1-\gamma})$ adversarially placed Byzantine nodes for any arbitrarily small (but fixed) positive constant γ . It outputs a constant factor estimate of $\log n$ that is known to all but $o(1)$ fraction of the good nodes. This algorithm works for *any* bounded degree expander network.

Our second algorithm is *randomized*. This algorithm works in *almost all* d -regular graphs (i.e., all but a vanishingly small fraction of such graphs). We note that this is the same model used in the seminal work of Dwork et al.[18]. Our algorithm works in the CONGEST model, where honest nodes use only *small-sized* messages (unlike the first algorithm). See Section 2 for more details about the network model. It tolerates up to $B(n) = n^{\frac{1}{2}-\xi}$ adversarially placed Byzantine nodes, where ξ is any arbitrarily small (but) fixed positive constant. This algorithm takes $O(B(n) \log^2 n)$ rounds (hence $o(\sqrt{n})$ rounds for $B(n) = n^{\frac{1}{2}-\xi}$) and outputs a constant factor estimate of $\log n$ with probability at least $1 - o(1)$. The said estimate is known to at least $(1 - \beta)n$ nodes for any arbitrarily small positive constant β .

We note that similar to our result, many prior Byzantine protocols [18, 10, 11, 46] in bounded-degree networks take a polynomial number of rounds in the Congest model (where honest nodes are limited to small-sized messages). However, all these protocols assume knowledge of n , and in certain cases, even the entire network topology. A notable exception is the protocol of King et al. [32] that takes a polylogarithmic number of rounds in the Congest model, but this protocol also requires knowledge of the entire network topology (including the value of n). On the other hand, these protocols tolerate a substantially larger number of Byzantine nodes, even up to $\Theta(n)$ Byzantine nodes. It is unknown whether one can achieve the same level of fault-tolerance for Byzantine counting.

To complement our algorithms, we also present an impossibility result that shows that it is impossible to estimate the network size (or the logarithm of it) with any reasonable approximation and with any non-trivial probability of success if the network does not have sufficient vertex expansion. This shows that the assumption of the expansion property of the network is *necessary* for solving Byzantine counting.

Both our algorithms are the first such algorithms that solve Byzantine counting in sparse, bounded degree networks under very general assumptions: they are fully local and need no global knowledge. Our algorithms can serve as a building block for implementing other non-trivial distributed computational tasks in Byzantine networks such as agreement and leader election where the network size (or its estimate) is not known a priori.

Applying our counting protocols. To illustrate, we consider the *Byzantine agreement* protocol of [3] that applies to sparse bounded-degree expander networks. It applies even when the network is dynamic with adversarial churn, but the network size is assumed to be stable. This protocol uses two main ideas to solve binary agreement, where the requirement is that most good nodes should decide on a common value (0 or 1) which should be an input value of a good node: (1) *random walks* to sample nodes uniformly at random from the network and (2) *a majority protocol* to converge to the correct value. Both ideas require knowledge of $\log n$, in particular, a constant factor upper bound of $\log n$. For random walks, $O(\log n)$ is the *mixing time*, which is needed for walks to converge to the stationary distribution in a bounded degree expander; nodes need to know an upper bound on the mixing time to ensure that only sufficiently “mixed” random walks are used for sampling. The majority protocol uses the following simple idea: In one iteration, each node samples two random nodes and updates its value to the majority value among the three values: its own value and the two other values. It is shown that $O(\log n)$ iterations are needed to converge to the almost-everywhere agreement with high probability, provided the number of Byzantine nodes is bounded by $O(\sqrt{n})$.

It is important to note that the above protocol assumes knowledge of $c \log n$, for some constant $c > 1$. However, using the Byzantine counting protocol of this paper as a preprocessing step, the above assumption can be removed. The counting protocol ensures that most honest nodes have a constant factor estimate of $\log n$ (this constant is fixed in the analysis). Although the counting protocol does not guarantee that all (or most) honest nodes have the *same* estimate of $\log n$, it is easy to ensure that most honest nodes have an estimate that is some constant factor larger than $\log n$. This estimate suffices to run the Byzantine agreement protocol of [3].

1.2 Technical Challenges and Drawbacks of Previous Approaches

The main challenge is designing and analyzing distributed algorithms in the presence of Byzantine nodes in networks where the honest nodes have only local knowledge, i.e., knowledge of their immediate neighborhood. For example, in a constant degree regular network, a node’s local view

does not yield any information on the network size. It is possible to solve the counting problem exactly in networks *without* Byzantine nodes by simply building a spanning tree and converging the nodes' counts to the root, which in turn can compute the total number of nodes in the network. A more robust and alternate way that works also in the case of *anonymous* networks is the technique of *support estimation* [7, 5] which uses *exponential* distribution. Alternatively, one can use a geometric distribution (see e.g., [33, 40, 39]) to accurately estimate the network size.

Consider the following simple protocol for estimating the network size that uses the geometric distribution. Each node u flips an unbiased coin until the outcome is heads; let X_u denote the random variable that denotes the number of times that u needs to flip its coin. Then, nodes exchange their respective values of X_u whereas each node only forwards the highest value of X_u (once) that it has seen so far. We observe that X_u is geometrically distributed and denote its global maximum by \bar{X} ; it can be shown that $\bar{X} = \Theta(\log n)$ with high probability and hence can be used to estimate $\log n$.

The geometric distribution protocol fails when even just one Byzantine node is present. Byzantine nodes can fake the maximum value or can stop the correct maximum value from spreading and hence can violate any desired approximation guarantee. The work of [14] successfully adapts the geometric distribution to work for their purpose. However, their work [14] assumes additional structural properties of the network — they assume “small-world” networks, i.e., networks with constant expansion *and* large clustering coefficient. The latter property implies that for every node, many of its neighbors are well-connected among themselves. The protocol of [14] exploits this fact to detect fake values sent by Byzantine nodes. This protocol does not work for graphs that *only* have the expander property (which as we show in the impossibility result is needed to estimate the network size within a non-trivial factor). Hence a new approach is needed as shown in this paper.

The work of [14] also assumes that the Byzantine nodes are *randomly* distributed in the network. This assumption coupled with the fact that their number is only $O(n^{1-\gamma})$ (where γ is any arbitrarily small, but fixed, positive constant), results in (with high probability) every honest node having a significant number of honest neighbors (the number of neighbors depends on γ). The algorithm of [14] *fails* to work for expander networks with arbitrary or adversarial Byzantine node distribution, which is typically assumed in previous works on Byzantine protocols [18, 46, 32, 7, 3, 4].

Prior localized techniques that have been used successfully for solving other problems such as Byzantine agreement and leader election such as random walks and majority agreement (e.g., [3, 4]) do not imply efficient algorithms for Byzantine counting. For instance, random walk-based techniques crucially exploit a uniform sampling of tokens (generated by nodes) after Θ (mixing time) number of steps. However, the main difficulty in this approach is that the mixing time is unknown (since the network size is unknown) — and hence it is unclear a priori how many random walk steps the tokens should take. Similar approaches based on the return time of random walks fail due to long random walks having a high chance of encountering a Byzantine node.

One can also use “birthday paradox” ideas to try to estimate n , e.g., as in the work of [21] in a non-Byzantine setting. However it fails too in the Byzantine case.

We note that one can possibly solve Byzantine counting if one can solve Byzantine leader election, as observed in [14], however, all known algorithms for Byzantine leader election (or agreement) *assume a priori knowledge (or at least a good estimate) of the network size*. Hence we require a new protocol that solves Byzantine counting from “scratch”. In our network model, where most nodes, with high probability, see (essentially) the same local topological structure (and constant degree) even for a reasonably large neighborhood radius (see Lemma 2), it is difficult for nodes to break symmetry or gain a priori knowledge of n .

We point out that with constant probability, in our network model, due to the property of the d -regular random graph, an expected constant number of nodes might have multi-edges — this can

potentially be used to break ties; however, this approach *fails* with constant probability.

Another approach is to try to estimate the diameter of the network, which, being $\Theta(\log n)$ for sparse expanders, can be used to deduce an approximation of the network size. Assuming that there exists a leader in the network, one way to do this is for the leader to initiate the flooding of a message and it can be shown that a large fraction of nodes (say a $(1 - \epsilon)$ -fraction, for some small $\epsilon > 0$) can estimate the diameter by recording the time when they see the first token, since we assume a synchronous network. However, this method fails since it is not clear, how to break symmetry initially by choosing a leader — this by itself appears to be a hard problem in the Byzantine setting without knowledge of n (or an estimate of $\log n$).

1.3 A High-level Description of Our Protocols

We now give a high-level intuition behind our protocols. Our first protocol works for *any* expander network as long as the nodes have knowledge of some lower bound on the expansion. The main idea is to show that honest nodes that have a sufficiently large distance from any of the Byzantine nodes will be able to detect any deviations in the network structure caused by Byzantine nodes. The honest nodes can accomplish that by checking the expansion of their i -hop neighborhood, for some $i = \Omega(\log n)$. This algorithm is *time-optimal* and runs in time proportional to the network diameter. However, it is designed for the **Local** model, as the expansion check requires nodes to send messages of polynomial size.

The second algorithm achieves Byzantine counting by ensuring that most good nodes will send only small-sized messages. The main idea here is the following. The algorithm proceeds in phases. In phase i , i is the current estimate of $\log(n)$. In each i -hop neighborhood of some node consisting only of good nodes, there are likely to be $\Theta(i)$ nodes that are generating *beacon messages*, which are propagated for at least i rounds through the network.

Upon receiving a beacon message, a node assumes that the value of i is not yet too large and hence proceeds without *deciding*. On the other hand, the probability of any good node generating a beacon message becomes $\frac{1}{\text{poly}(n)}$ once $i = \Omega(\log n)$, and hence good nodes that do not observe a beacon message within $O(i)$ rounds of phase i , decide on i as their estimate.

To avoid the scenario where Byzantine nodes simply keep generating new beacon messages (to falsely induce a larger network size), the algorithm implements a *blacklisting mechanism* that uses properties of random regular graphs to prevent nodes from generating multiple beacon messages within the same phase. This ensures that the Byzantine nodes will be blacklisted if they attempt to generate fake beacon messages.

1.4 Other Related Work

There have been several works on estimating the size of the network, see e.g., the works of [21, 27, 36, 45, 44], but all these works do not work under the presence of Byzantine adversaries. There have been some work on using network coding for designing byzantine protocols (see e.g., [28]); but these protocols have polynomial message sizes and are highly inefficient for problems such as counting, where the output size is small. There are also some works on topology discovery problems under Byzantine setting (e.g., [38]), but these do not solve the counting problem.

Several recent works deal with Byzantine agreement, Byzantine leader election, and fault-tolerant protocols in dynamic networks. We refer to [24, 7, 3, 2, 4] and the references therein for details on these works. These works crucially assume the knowledge of the network size (or at least an estimate of it) and don't work if the network size is not known.

There has been significant work in designing peer-to-peer networks that are provably robust to a large number of Byzantine faults [19, 26, 37, 43]. These focus only on (robustly) enabling storing and retrieving data items. The works of [32, 30, 31] address the Byzantine agreement problem, and the work of [24] presents a solution for maintaining a clustering of the network. In particular, [30] use a spectral technique to “blacklist” malicious nodes leading to faster and more efficient Byzantine agreement. All these works assume a sufficiently good estimate of the network size; in particular, none of them solves the Byzantine counting problem in sparse networks.

The work of [13] shows how to implement uniform sampling in a peer-to-peer system under the presence of Byzantine nodes where each node maintains a local “view” of the active nodes. We point out that the choice of the view size and the sample list size of $\Theta(n^{\frac{1}{3}})$ necessary for withstanding adversarial attacks requires the nodes to have a priori knowledge of a polynomial estimate of the network size. [27] considers a dynamically changing network *without* Byzantine nodes where nodes can join and leave over time and provides a local distributed protocol that achieves a polynomial estimate of the network size.

In [47], the authors present a gossip-based algorithm for computing aggregate values in large dynamic networks (but without the presence of Byzantine failures), which can be used to obtain an estimate of the network size. The work of [16] focuses on the consensus problem under crash failures and assumes knowledge of $\log n$, where n is the network size. Lenzen et al. [35] study the synchronous counting problem under Byzantine nodes which is a different problem: the goal here is to synchronize pulses among correct nodes. They study the problem in a complete network, and hence the network size is trivially known.

Byzantine fault detection in the context of asynchronous distributed systems. There have also been several works on Byzantine fault detection — see, e.g., [1], [29], [25], and [23]. Alvisi et al. [1] consider the problem of fault detection in Byzantine *quorum systems* and design statistical methods to compute the current number of failures at any point of time. Their model assumes the knowledge of n , the total number of servers, whereas their goal is also clearly different. There have also been works on Byzantine fault detectors [29, 25], but these assume the complete graph where the knowledge of n becomes trivial.

Kihlstrom et al. [29] propose and analyze new classes of Byzantine fault detectors to solve the consensus problem in an asynchronous distributed system of n processes, in which the number of (Byzantine-) faulty processors is strictly less than $\frac{n}{3}$. Haeberlen et al. [25] proposes a new idea for Byzantine fault detection by achieving *eventual strong completeness* where every faulty node is eventually blacklisted by every correct node. The underlying communication graph is a *complete graph* in both the network models of [29] and [25], thus the knowledge of n becomes immediate and trivial.

Greve et al. [23] design and analyze a powerful Byzantine failure detector that works in dynamic distributed systems, where both the number of processors and the topology of the communication graph can change from round to round. Their work does not assume any knowledge of n ; however, their work does *not* solve the Byzantine counting problem either — *no* estimate about the *global* network size can be made during the execution of their algorithm.

2 Computing Model and Problem Definition

The distributed computing model. We consider a synchronous network represented by a graph G whose nodes execute a distributed algorithm and whose edges represent connectivity in the network. The computation proceeds in synchronous rounds, i.e., we assume that nodes run

at the same processing speed (and have access to a synchronized clock) and any message that is sent by some node u to its neighbors in some round $r \geq 1$ will be received by the end of round r . We consider the **Local** model, where there is no restriction on the size of the messages that can be transmitted per edge per round, [40, 42]; but we point out that our second algorithm ensures that most good nodes send only small-sized messages.

As is usual [40, 42], we assume local computation (within a node) is free and instantaneous.

Byzantine nodes. Among the n nodes (n or its estimate is not known to the nodes initially), up to $B(n)$ can be *Byzantine*. The Byzantine nodes have unbounded computational power and can deviate arbitrarily from the protocol. This setting is commonly referred to as the *full information model*.

We say that a node u is *good* or *honest* if u is not a Byzantine node. Byzantine nodes are *adaptive* — they have complete knowledge of the entire states of all nodes at the beginning of every round (including random choices made by all the nodes), and thus can take the current state of the computation into account when determining their next action. The Byzantine nodes also know the future random choices of the honest nodes, i.e., the Byzantine nodes are *omniscient*. We assume that the Byzantine nodes are *arbitrarily* distributed in the network and that when a Byzantine node sends a message over an edge, it cannot fake its id. We note that both of these assumptions are quite typical in the literature [18, 46, 32, 7, 3, 4].

Distinct IDs. We assume that all nodes (including the Byzantine nodes) have *distinct IDs*, chosen from an arbitrarily large set whose size is unknown a priori. In other words, the node IDs can be viewed as comparable black boxes that do not leak any information about the network size. We point out that this precludes most nodes from estimating $\log n$ by looking at the length of their IDs.

Network topology for the first (deterministic) algorithm. Let $G = (V, E)$ be the graph representing the network. We assume G to be a bounded-degree expander network. For the sake of a self-contained exposition, we recall the definition of *vertex expansion* below.

Definition 1 (Vertex expansion of a graph). *The vertex expansion of a graph $G = (V, E)$ on n nodes is defined as*

$$h(G) = \min_{0 < |S| \leq \frac{n}{2}} \frac{|Out(S)|}{|S|},$$

where S is any subset of V of size at most $\frac{n}{2}$ and $Out(S)$ is the set of neighbors of S in $V \setminus S$.

We assume that the network graph G has a constant *vertex expansion* $\alpha > 0$, where α is a fixed positive constant.

Network topology for the second (randomized) algorithm. Here we assume G to be a random d -regular graph model (d is a constant) that is constructed by the union of $\frac{d}{2}$ (assume $d \geq 8$ is an even constant) random Hamiltonian cycles of n nodes. We call this random graph model the $H(n, d)$ *random graph model*, also called the *permutation model* (please refer to [15] for a detailed exposition of the $H(n, d)$ random graph model and its various properties). It is known that such a random graph is an expander (in fact a Ramanujan expander [20, 34]) with high probability. The $H(n, d)$ model is a well-studied and popular random graph model (see e.g., [49]), and has been used as a model for peer-to-peer networks and self-healing networks [34, 41].

We note that the usual d -regular random graph model is the model where a graph is selected with uniform probability among all (simple) d -regular graphs [49]. Thus if one can show a result that holds with high probability in a d -regular random graph, then it holds for *almost all* d -regular graphs (as in Dwork et al [18]). Since it is hard to work directly with the above model, one usually works with the so-called *configuration (or pairing) model* [12] that can be used to generate a d -regular random graph. The advantage of the configuration model is that if one can show a high probability bound on the configuration model, then this implies a similar bound for d -regular (d is a constant) random graphs [18]. The configuration model is closely related to the $H(n, d)$ (i.e., permutation) model, which is sometimes easier to work with compared to the configuration model. It was shown by Greenhill et al. [22] that an event that holds with probability at least $1 - o(1)$ in the configuration model also holds with probability $1 - o(1)$ in the $H(n, d)$ model and vice versa. Thus, the results that we show for the $H(n, d)$ model also hold for the configuration model with probability at least $1 - o(1)$. Therefore they also hold for d -regular random graphs with the same probability. Hence they hold for *almost all* d -regular graphs.

Problem definition. Since we assume a *sparse* (constant bounded degree) network and a large number of Byzantine nodes, it is difficult to ensure that every honest node eventually knows an exact estimate of n . This motivates us to consider the following “approximate, almost everywhere” variant of counting:

Definition 2 (Byzantine counting). *Suppose that there are $B(n)$ Byzantine nodes in the network and let ϵ be an arbitrarily small (but fixed) positive constant. We say that an algorithm solves Byzantine Counting in T rounds if the following properties hold in all runs:*

1. *Every honest node u (irrevocably) decides on an estimate of $\log n$, denoted by \mathcal{L}_u , within T rounds.*
2. *There is a set S of at least $(1 - \epsilon)n - B(n)$ honest nodes such that each $u \in S$ has a constant factor estimate of $\log n$; i.e., there are fixed constants $c_1, c_2 > 0$, such that*

$$c_1 \log n \leq \mathcal{L}_u \leq c_2 \log n.$$

Some terminology. We recall the following terminology that are used throughout this paper.

1. We use the terms *sparse network* and *bounded-degree network* synonymously — each describing a network where the maximum degree of a node is bounded by a constant, and hence the number of edges is linear in the number of vertices.
2. A *small-sized message* is defined to be one that contains $O(\log n)$ bits in addition to at most a constant number of node IDs.
3. We use the term *most nodes* or *most good nodes* to indicate $\geq (1 - \beta)n$ nodes, where n is the total number of nodes in the network (and is an unknown quantity in the context of this paper) and β is any arbitrarily small (but fixed) positive constant.
4. By *efficient algorithms* we mean algorithms that use small-sized messages and run in $\text{polylog}(n)$ time.

3 Preliminaries

We use the notation $\mathcal{B}_G(u, i)$ to refer to the inclusive i -hop neighborhood of node u in graph G and we omit G when it is clear from the context. For a set of nodes S , we define $\mathcal{B}_G(S, i) = \bigcup_{u \in S} \mathcal{B}_G(u, i)$.

Both of our algorithms make use of a structural result that shows that Byzantine nodes have a somewhat limited impact on *most* good nodes in expander graphs.

Lemma 1. *Consider an n -node graph $G = (V, E)$ with maximum degree $\Delta = O(1)$ and vertex expansion $\alpha > 0$. Let Byz , an arbitrary subset of V , denote the set of Byzantine nodes with the restriction that $|\text{Byz}| \leq n^{1-\gamma}$, where γ is any arbitrarily small (but fixed) positive constant. Then, for any $o(n)$ -sized $F \subset V$, there exists a set $\text{Good} \subseteq V \setminus F$ of good nodes such that*

$$|\text{Good}| \geq n - 2|F| - o(n). \quad (1)$$

Moreover, for each $u \in \text{Good}$, the following hold:

1. $\mathcal{B}(u, \lfloor \frac{\gamma}{2} \log_{\Delta} n \rfloor)$ does not contain any Byzantine nodes.
2. Let H be the subgraph induced by nodes in Good . Then, for any constant $c > 0$ such that $|\mathcal{B}_H(u, c \log n)| \leq \frac{|\text{Good}|}{2}$, it holds that every vertex subset $S \subseteq \mathcal{B}_H(u, c \log n)$ has a vertex expansion of $\geq \alpha'$ in graph H , for any fixed constant $\alpha' < \alpha$.

Proof. Consider the set Byz of Byzantine nodes. We first instantiate Lemma 13 by removing the set $V(G) \setminus (\text{Byz} \cup F)$ from G , and thus obtain a connected subgraph H of size $\geq n - o(n)$. By Lemma 13, H contains at least

$$n - (|F| + |\text{Byz}|) \left(1 + \frac{1}{\phi(1-c')}\right) = n - |F| \left(1 + \frac{1}{\phi(1-c')}\right) - o(n)$$

good nodes. Also, every one of its subsets of size $\geq \frac{|H|}{2}$ has a vertex expansion of at least α' , for any constants $\alpha' < \alpha$ and $c' < 1$. Choosing $c' = 1 - \frac{1}{\phi}$, implies that H contains $n - 2|F| - o(n)$ nodes, as required. Assuming a maximum degree of Δ , we get $|\mathcal{B}(\text{Byz}, j)| \leq |\text{Byz}| \Delta^j$, for any $j \geq 0$.

We observe that

$$|\mathcal{B}(\text{Byz}, \lfloor \frac{\gamma}{2} \log_{\Delta} n \rfloor)| \leq |\text{Byz}| \cdot \Delta^{(\gamma/2) \log_{\Delta} n} = n^{1-\gamma/2}.$$

It follows that the set $\text{Good} = V(H) \setminus \mathcal{B}(\text{Byz}, \frac{\gamma}{2} \log_{\Delta} n)$ satisfies (1) and (2). \square

3.1 The “locally tree-like” property of an $H(n, d)$ random graph

We refer to [15] for a detailed exposition of the $H(n, d)$ random graph model and its various properties. For the sake of completeness, we merely state the main definitions and lemmas needed here. The “locally tree-like” property of an $H(n, d)$ random graph says that for most nodes w , the subgraph induced by $B(w, r)$ up to a certain radius r looks like a tree. More specifically, let G be an $H(n, d)$ random graph and w be any node in G . Consider the subgraph induced by $B(w, r)$ for $r = \frac{\log n}{10 \log d}$. Let u be any node in $Bd(w, j)$, $1 \leq j < r$. u is said to be *typical* if u has only one neighbor in $Bd(w, j-1)$ and $(d-1)$ -neighbors in $Bd(w, j+1)$; otherwise it is called *atypical*.

Definition 3 (Locally Tree-Like Property). *We call a node w locally tree-like if no node in $B(w, r)$ is atypical. In other words, w is locally tree-like if the subgraph induced by $B(w, r)$ is a $(d-1)$ -ary tree.*

Using the properties of the $H(n, d)$ random graph model and standard concentration bounds, it can be shown that most nodes in G are locally tree-like:

Lemma 2. *In an $H(n, d)$ random graph, with high probability, at least $n - O(n^{0.8})$ nodes are locally tree-like.*

The proof of this lemma as well as a more detailed discussion of the $H(n, d)$ random graph model can be found in the appendix of [15].

4 A Time-Optimal Deterministic Algorithm

In this section, we present and analyze a simple algorithm that solves the Byzantine counting problem in the Local model — see Algorithm 1 for the pseudocode.

Our goal here is to show that a set called **Good** consisting of $\geq n - o(n)$ good nodes achieve a constant factor approximation of $\log n$ when executing our algorithm. Lemma 1 formalizes the criteria for a good node to be in **Good**: in particular, a good node needs to have a distance of $\Omega(\log n)$ from all Byzantine nodes and the graph induced by **Good** must have nearly the same vertex expansion as the original network.

4.1 Description of the algorithm

Throughout the algorithm, each node u locally builds an approximation of its i -hop neighborhood for rounds $i = 1, 2, 3, \dots$, which we denote by $\hat{B}(u, i)$. To this end, we instruct nodes to simply forward the content of their current $\hat{B}(u, i)$ at the start of round i . Considering that we assume (at most) $n^{1-\gamma}$ Byzantine nodes, node u needs to be careful when integrating any newly received knowledge.

There are two possibilities for triggering a decision of node u . Firstly, u immediately decides if it notices some structural inconsistencies in the received topology information, such as a degree larger than Δ , or the addition of spurious edges to vertices that it had already learned about previously (cf. Line 6).

Furthermore, after obtaining $\hat{B}(u, i + 1)$ by adding the received topology information in round r into $\hat{B}(u, i)$, node u also decides if any of the subsets of $\hat{B}(u, i)$ do not have sufficient vertex expansion with respect to $\hat{B}(u, i + 1)$. Intuitively speaking, this second condition ensures that Byzantine nodes cannot trick u into continuing forever. The algorithm's correctness crucially rests on the original network having constant expansion — a point that is further emphasized by our impossibility result in Theorem 3.

Remark 1. We observe that, for $o(n)$ nodes in $G \setminus \text{Good}$, the adversary essentially controls the termination time. This is *not* simply a drawback of our algorithm, but, instead, *unavoidable* when assuming a worst-case placement of Byzantine nodes in the network: For instance, consider a d -regular expander and a set of $\lfloor n^{1-\gamma}/d \rfloor$ good nodes U that are surrounded by roughly $n^{1-\gamma}$ Byzantine nodes, i.e., none of the edges emanating from U to $G \setminus U$ are connected to good nodes. Then, the Byzantine nodes could simply send information corresponding to a large fake network of some arbitrary size n' with sufficiently high expansion to the nodes in U . It is easy to see that no algorithm can distinguish this case from U being indeed part of a network of size n' .

We state below the main result of this section. The rest of this section is devoted to proving it.

Theorem 1. *Let $\gamma \in (0, 1)$ and $\Delta > 0$ be arbitrary fixed constants. Consider an n -node network with a maximum node degree bounded by Δ and a constant vertex expansion α . There exists a*

deterministic *LOCAL* algorithm such that $n - o(n)$ good nodes decide on a $\left(\frac{\gamma}{2 \log \Delta}\right)$ -approximation of $\log n$ in $O(\log n)$ rounds in the presence of up to $n^{1-\gamma}$ arbitrarily (adversarially) placed Byzantine nodes.

Algorithm 1 An $O(\log n)$ time algorithm in the *Local* model. Code for node u .

```

1: Let  $\hat{B}(u, 1)$  be the set of nodes in the inclusive neighborhood of  $u$ .
2: for round  $i = 1, 2, \dots$  do
3:   Broadcast  $\hat{B}(u, i)$  to all neighbors.
4:   Let  $I$  be the information received in the current round.
5:   if inconsistent( $\hat{B}(u, i), I$ ) or (some neighbor is mute) then
6:     Decide on  $i$  and terminate.
7:   end if
8:   Create  $\hat{B}(u, i + 1)$  by incorporating  $I$  into  $\hat{B}(u, i)$ .
9:   for each vertex subset  $S \subseteq V(\hat{B}(u, i))$  do
10:    Let  $\alpha' > 0$  be the constant given by Lemma 1.
11:    if  $S$  does not have vertex expansion  $\geq \alpha'$  in graph  $\hat{B}(u, i + 1)$  then
12:      Decide on  $i$ ; terminate.
13:    end if
14:  end for
15: end for

16: predicate inconsistent( $\hat{B}(u, i), I$ ) returns true iff
17:    $I$  contains a node with degree  $> \Delta$ , or
18:    $I$  contains a set of incident edges for some node  $v$ , but already  $v \in \hat{B}(u, j)$  for some  $j \leq i - 1$ .

```

4.2 Analysis of the algorithm

Lemma 3. *All nodes in Good decide on a value of at least $\lfloor \frac{\gamma}{2} \log_{\Delta} n \rfloor$.*

Proof. We will proceed by induction over the number of rounds. Consider the graph H given by Lemma 1 and recall that $u \in V(H)$ by definition. Since H has a vertex expansion $\geq \alpha'$, it follows that u 's neighborhood (in H) has size at least $1 + \alpha'$, which guarantees that u passes the expansion-check in Line 11 in Round 1. Moreover, u has a distance of at least $\lfloor \frac{\gamma}{2} \log_{\Delta} n \rfloor$ from any Byzantine node, and hence it does not receive any inconsistent information during the first $\lfloor \frac{\gamma}{2} \log_{\Delta} n \rfloor$ rounds. This ensures u will not decide in Line 6 Round 1, which completes the inductive base.

Now, consider the inductive step $1 < i < \lfloor \frac{\gamma}{2} \log_{\Delta} n \rfloor$, and suppose that u has not decided at the end of round $i - 1$. Similarly as in the case $i = 1$, it holds that u does not decide due to receiving inconsistent information. Moreover, note that

$$|\mathcal{B}(u, i)| \leq \Delta^{i+1} \leq n^{\gamma/2} < \frac{|H|}{2},$$

since $|H| \geq n - o(n)$ by Lemma 1. Hence every subset of $\hat{B}(u, i)$ is guaranteed to have a vertex expansion of at least α' , which ensures that u continues to round $i + 1$ without deciding. \square

The next lemma tells us that, if a good node u that has not yet decided, then its local i -neighborhood approximation $\hat{B}(u, i)$ does not contain inconsistent information concerning the nodes in *Good*. We will make use of this property in Lemma 5 below.

Lemma 4. *Suppose that $u \in \text{Good}$ has not decided by the end of round i , and consider graph H given by Lemma 1. Then, for each $v \in \mathcal{B}_H(u, i)$ and any node w , it holds that $e = \{v, w\} \in \hat{B}(u, i)$ if and only if $e \in E(G)$.*

Proof. By the definition of H , it follows that, for each $v \in \mathcal{B}_H(u, i)$, there exists a shortest path $p = (v = p_1, p_2, \dots, p_j = u)$ consisting of $j \leq i$ good nodes connecting u and v . Moreover, it is easy to see that node p_k ($1 \leq k < j$) on path p must have broadcast its topology information in round $i - j + k$, since otherwise its neighbor p_{k+1} would have terminated at the end of round $i - j + k$, because of having noticed that p_k was mute. This in turn would have caused p_{k+1} to terminate at the end of round $i - j + k + 1$ and hence would have propagated to u by round i , contradicting the premise of the lemma.

Since each of the good nodes in p forwards the received topology information towards u , it follows that node u receives a message from some good neighbor, which contains the exact set E_v of edges incident to v in G . Suppose that, in some round during round i , node u also receives a message containing a set of edges $E'_v \neq E_v$ of v , possibly injected by Byzantine nodes. However, Line 6 ensures that u decides instantly in round i , since it has added inconsistent information to $\hat{B}(u, i)$. This results in a contradiction. \square

Lemma 5. *Every node in Good decides on a value of at most $\text{diam}(G) + 1$.*

Proof. Assume toward a contradiction that there is a node $u \in \text{Good}$ that decides on a value strictly greater than $\text{diam}(G) + 1$. By the description, of the algorithm, this means that u did not decide when executing round i , where $i = \text{diam}(G) + 1$. Consider the content of $\hat{B}(u, i)$ after receiving all messages for round i . Note that it is possible that $\hat{B}(u, i)$ also contains information that was injected by Byzantine nodes.

Let F denote the *Byzantine part* of $\hat{B}(u, i)$, i.e.,

$$F \stackrel{\text{def}}{=} \hat{B}(u, i) \setminus \text{Good}$$

and call

$$R \stackrel{\text{def}}{=} \hat{B}(u, i) \cap \text{Good}$$

its *honest part*.

We can assume that Byzantine nodes do not send any inconsistent information regarding the graph induced by R , as otherwise u will decide in round i and we are done. Similarly, we can rule out that any node in R has already decided: For if some w decided and remained mute, this would cause its good neighbors to decide in the next round (cf. Line 6), which in turn would propagate (through good nodes) to u , causing it to decide. Consequently, Lemma 4 tells us that all edges emanating from nodes in $R \setminus \text{Byz}$ in graph $\hat{B}(u, i)$ also exist in G . In particular, there are no edges between $R \setminus \text{Byz}$ and F . Since every node in G has distance at most $\text{diam}(G) = i - 1$ to u , it follows that $R \subseteq \hat{B}(u, i - 1)$ and thus u will check R 's vertex expansion with respect to graph $\hat{B}(u, i)$ at the end of round $i + 1$.

To complete the proof, we analyze the expansion-check in Line 11 for the set R . Observe that R contains all nodes within distance $\text{diam}(G)$ from u in graph H (see Lemma 1). Given that $\text{diam}(H) \leq \text{diam}(G)$ and the fact that nodes in Good are connected in H , we know that

$$|R| \geq |\text{Good}| = n - o(n).$$

Recall that there are at most $n^{1-\gamma}$ Byzantine nodes in $R \cap \hat{B}(u, i)$. Since we assumed that Byzantine nodes did not send inconsistent information, each Byzantine node has at most Δ neighbors in F (cf. Line 17). It follows that there is a set S' of at most $\Delta n^{1-\gamma} = o(n)$ fake vertices in

the set $(\hat{B}(u, i) \setminus R) \subseteq F$ that have an edge to R . To satisfy the expansion-check (see Line 11), the number of neighbors of vertices in R would need to be $|R|(1 + \alpha') = \Omega(n)$, far exceeding the $o(n)$ fake vertices in S' . Hence the expansion-check fails for set R , causing u to decide on $i = \text{diam}(G) + 1$. \square

Combining Lemmas 3 and 5 shows the claimed bound on the approximation achieved by the $n - o(n)$ nodes in set **Good**. From Lemma 5, it follows that the round complexity until all but $o(n)$ nodes have decided is $O(\text{diam}(G)) = O(\log n)$. This completes the proof of Theorem 1.

5 Byzantine Counting with Small Messages

We now describe an algorithm that guarantees most good nodes will achieve a constant factor approximation of $\log(n)$ while sending only messages of small size (proportional to the number of bits of any node's ID). We give the detailed correctness proof in Section 5.1. As mentioned in Section 2, our algorithm works in the $H(n, d)$ d -regular random graph model with high probability, i.e., with probability at least $1 - n^{-c}$, for some constant $c \geq 1$. As discussed in Section 2, this implies that the algorithm works in almost all d -regular graphs with probability at least $1 - o(1)$.

In Algorithm 2, each node keeps track of its current estimate in a variable i that is initialized to a fixed constant. A node increases i whenever it enters a new *phase*, where the goal of a phase is to determine whether i is already a sufficiently good approximation of $\log(n)$. On the other hand, once a node concludes that its current value of i is sufficiently large, it *decides on i* and stops participating in future phases. Each phase i consists of roughly $e^{(1-\gamma)i} + 1$ iterations, and each iteration of phase i takes $2i + 5$ rounds: During the first $i + 2$ rounds, nodes disseminate so called “beacon messages” (described next) whereas, during the following $(i + 3)$ -rounds, all yet-undecided nodes ensure that everyone in their $(i + 3)$ -neighborhood knows that they have not yet decided by sending a “continue” message.

Beacon Messages and Path Fields. At the start of an iteration, a good node u chooses to become *active* with probability $\Theta(\frac{i}{d^i})$, where d is u 's degree. The intuition behind this probability is that this ensures that on the average there are approximately $O(i)$ nodes that are active in a ball of radius i — note that the tree-like property of expander graphs (see Section 3.1) ensures that the number of nodes in a ball is $\Theta(d^i)$.

If u becomes active, it broadcasts a *beacon message* to its neighbors, which is then forwarded for $i + 2$ rounds. Intuitively speaking, these beacon messages signal to other nodes that they should not yet decide on their current estimate.

In more detail, a beacon message $\langle \text{beacon}, u, P \rangle$ has an *origin id* u , and a *path field* P , which is the path of nodes that the message has visited so far. That is, whenever the message is sent from a node w to a node v , we append v to the path field before forwarding the message. Of course, it is entirely possible that these fields contain bogus information if the message passed through a Byzantine node.

Blacklisting. Whenever a node v receives a beacon message, it inspects the attached path field $P = (u_1, u_2, \dots, u_k)$ by performing a series of checks.

First, v checks whether the neighbor from which it received the message does indeed have id u_k . If v finds that the sender has an id different from u_k , it simply discards that message. Node v also maintains a *blacklist* set BL , which is reset at the start of each phase and is gradually filled throughout a phase's iterations.

In more detail, let us suppose that the above mentioned beacon message was the *first* one received by v in iteration 1 of phase i , from some neighbor w . Then, v adds all nodes except the ones in the $\lfloor(1 - \epsilon)\rfloor i$ -suffix of P to its blacklist BL , where this suffix consists of the last $\lfloor(1 - \epsilon)\rfloor i$ nodes on the path to the destination v . The intuition behind this rule is that u blindly trusts all nodes that are close to it, but won't accept another beacon message in the future if it has traversed the same (far away) nodes twice in this iteration.

In addition, v sets its variable $\text{shortestPath} \leftarrow P$, which indicates the (supposedly) shortest path over which v received a beacon message in this iteration. (If v receives two or more beacon messages simultaneously, it discards all but one.) Note that v resets shortestPath at the end of each iteration. Then, assuming that we are still in the first $i + 1$ rounds of the iteration upon the reception of this beacon message, v broadcasts the message (beacon, P') with the modified path field P' to its neighbors where P' is obtained by appending v to P .

As mentioned, blacklisting ensures that v does not accept a beacon message if the message took a path leading through the same nodes from which it has already seen a beacon message in this phase. Blacklisting is implemented as follows: If v receives a beacon message m' in some iteration $\ell > 1$ of phase i and the node IDs contained in the path field that are at least $\lfloor(1 - \epsilon)\rfloor i$ away from v intersect with the nodes already added to BL during the previous iterations, then v will not use m' to update its shortestPath variable. However, it is important to keep in mind that, even in this case, v still broadcasts the message with the updated path field to its neighbors, assuming we are still in the first $(i + 1)$ -rounds of the iteration.

Consequently, if $(i + 2)$ rounds have passed and node v did not set shortestPath in this iteration either because it did not receive any beacon messages or all received beacons carried already blacklisted node IDs, then v decides on its current value of i .

Introducing blacklisting avoids the scenario where Byzantine nodes keep generating new beacon messages that trigger good nodes to continue progressing to the next phase, possibly significantly overshooting the actual value of $\log(n)$ before deciding. The blacklisting mechanism kicks in once $i = \Omega(\log n)$ since the algorithm ensures that (see Lemma 11):

1. there is no iteration in which a good node still generates a new beacon message (whp);
2. the number of iterations performed in phase i exceeds the number of Byzantine nodes.

For instance, suppose that a Byzantine node b generates a beacon message with a fake path field in iteration 1. Even though b can trick all good nodes into accepting this beacon message in this iteration, it will fail to convince a set U of good nodes that have a distance of at least $\lfloor(1 - \epsilon)\rfloor i$ from b into accepting such a message in any future iteration of this phase.

To see why this is the case, observe that when a node $u \in U$ receives a message where b was involved in faking the path field, b will be added to u 's blacklist because its ID will not be in the $\lfloor(1 - \epsilon)\rfloor i$ suffix of the path field by the time the message reaches u (cf. Line 32), assuming that the message did not pass through other Byzantine nodes that are closer to u . (Recall that i is large enough such that good nodes have ceased from generating beacon messages and hence every beacon message that is still in transit must have been injected by Byzantine nodes.) The upshot is that a good node u that has all the Byzantine nodes at a distance of at least $\lfloor(1 - \epsilon)\rfloor i$ will blacklist at least 1 Byzantine node b in each iteration if b generates a beacon message. Hence, u will encounter an iteration in which its shortestPath variable is not set, thus causing it to decide on i .

Technical challenges. There are several technical difficulties that we need to handle in our correctness proof. For instance, we need to choose the probability of generating beacon messages in a way such that i does not become too large before most nodes have reached a decision, as we

might end up with a value of i where almost all good nodes are *within* distance $\lfloor (1 - \epsilon)i \rfloor$ of some Byzantine node, thus disarming the blacklisting mechanism.

On the other hand, the blacklisting process itself reduces the number of nodes that a good node considers for beacon messages, which may cause too many nodes to decide early due to not seeing a beacon message in each iteration. We use two techniques to avoid this second problem:

1. We use the *tree-like* property of the regular expander graphs (see Section 3.1). This shows that the remaining nodes provide sufficient expansion even if a large number of paths have been invalidated due to at least one of their nodes being blacklisted.
2. We instruct undecided nodes to send out **continue** messages that are forwarded for $(i + 3)$ rounds in phase i . Upon reception of such a message, a node that has possibly already decided and stopped increasing its phase counter, will become active again and generate beacon messages with the appropriate probability.

We state below the main result of this section. The rest of this section is devoted to proving it.

Theorem 2. *Let ξ and β be any arbitrarily small (but fixed) positive constants. Let $B(n) = n^{\frac{1}{2} - \xi}$ denote the number of Byzantine nodes in the network. Consider the $H(n, d)$ random regular graph G of n nodes with constant vertex expansion, where d is a sufficiently large constant. Then there exists an algorithm such that, with high probability, at least $(1 - \beta)n$ nodes send messages of at most $O(\log n)$ bits and decide on a constant factor approximation of $\log n$ in time $O(B(n) \cdot \log^2 n)$ in the presence of up to $B(n)$ arbitrarily (adversarially) placed Byzantine nodes.*

5.1 Analysis of Algorithm 2

For the analysis, we assume at most $n^{1-\gamma}$ Byzantine nodes, where γ needs to satisfy

$$\gamma \geq \frac{1}{2 - \delta} + \eta, \quad (2)$$

for any fixed constants $0 < \delta \leq \frac{1}{2}$ and $\eta > 0$. Note that the smaller δ is, the smaller γ is. Therefore maximum Byzantine tolerance is achieved when δ is very close to (but slightly greater than) zero and γ is very close to (but slightly greater than) $\frac{1}{2}$. In that case, the maximum Byzantine tolerance, i.e., the maximum number of Byzantine nodes that our algorithm can tolerate, boils down to $n^{\frac{1}{2} - \xi}$, as stated in Theorem 2.

The parameter ϵ that we use to determine the distance outside of which the blacklisting becomes effective in our algorithm, is fixed as

$$\epsilon = 1 - \frac{(1 - \delta)}{\log d} \gamma. \quad (3)$$

Let $\text{GoodTL} = \text{Good} \cap \text{TreeLike}$ be the set of nodes that have a sufficiently large distance to all Byzantine nodes due to being in set **Good**, and that also have the property of d -ary trees up to some radius of length $\frac{\log_d n}{10}$ (see Section 3.1).

We will first study the progress of the algorithm at nodes in **GoodTL**, for the phases up to radius ρ , where

$$\rho = \left\lfloor \min \left((1 - \delta) \gamma \log_d n, \frac{1}{10} \log_d n \right) \right\rfloor - 2, \quad (4)$$

since, in phase i , we require the tree-like property to hold up until radius $(i + 2)$. We also recall that c_1 is any large constant, as defined in Line 5 of the pseudocode of Algorithm 2.

Algorithm 2 A Byzantine-Resilient Counting algorithm using messages of size $O(\log n)$ (at most nodes), assuming $O(n^{1-\gamma})$ Byzantine nodes. Nodes do not have any other global knowledge apart from γ .

```

1: for phase  $i = c, c + 1, \dots$  do //  $c \geq \frac{2 \log 2}{(2-\delta)\eta}$  is a sufficiently large constant, see (2)
2:   Each node initializes its phase  $i$  blacklist  $BL = \emptyset$ .
3:   for iteration  $j = 1, \dots, \lfloor e^{(1-\gamma)i} \rfloor + 1$  of phase  $i$  do // each iteration takes  $(2i + 5)$  rounds.
4:     Each node  $u$  initializes  $\text{shortestPath} \leftarrow \text{none}$ .
5:      $u$  becomes active with probability  $\frac{c_1 \cdot i}{d^i}$ , for a sufficiently large constant  $c_1$ .
6:     if  $u$  is active then
7:        $u$  updates  $\text{shortestPath} \leftarrow (u)$ .
8:        $u$  broadcasts message  $m = \langle \text{beacon}, u, P \rangle$ ,  $u$ 's own id denotes the origin,
9:       and  $P$  is the path of nodes visited previously by  $m$ ; i.e.,  $P = \text{none}$ .
10:      Message  $m$  is forwarded by correct nodes for exactly  $(i + 2)$  rounds (see below).
11:    end if

12:    // During the first  $i + 2$  rounds of the iteration:
13:    if node  $u$  receives a set of beacon messages from its neighbors then
14:       $u$  discards all but one arbitrarily chosen message.
15:      Let  $m = \langle \text{beacon}, v, Q \rangle$  be this message; assume that it was received from neighbor  $w$ .
16:       $u$  appends  $w$ 's id to path  $Q$  yielding  $Q'$ .
17:      if we are still within the first  $i$  rounds of the iteration then
18:         $u$  broadcasts  $\langle \text{beacon}, v, Q' \rangle$ .
19:      end if
20:      Let  $S$  be the set of all except the last  $\lfloor (1 - \epsilon) i \rfloor$  nodes in  $Q'$ , where  $\epsilon$  is defined in (3).
21:      if  $S \cap BL = \emptyset$  then
22:        if  $\text{shortestPath} = \text{none}$  then
23:           $u$  updates  $\text{shortestPath} \leftarrow Q'$ .
24:        end if
25:      end if
26:    end if

27:    // After the first  $i + 2$  rounds of iteration  $j$ :
28:    if  $\text{shortestPath} = \text{none}$  at node  $u$  and  $u$  has not yet decided then
29:       $u$  decides on  $i$ .
30:    end if
31:    Let  $S$  be the set of all except the last  $\lfloor (1 - \epsilon) i \rfloor$  nodes in  $\text{shortestPath}$ .
32:     $u$  adds all nodes in  $S$  to blacklist  $BL$ .
33:  end for // End of the  $j^{\text{th}}$  iteration of phase  $i$ 

34:  if  $u$  has not decided then
35:     $u$  broadcasts a  $\langle \text{continue} \rangle$  message, which is forwarded for  $i + 3$  rounds by other nodes.
36:  end if
37:  When receiving multiple  $\langle \text{continue} \rangle$  messages simultaneously, all but one are discarded.
38:  if  $v$  has decided and  $v$  did not receive  $\langle \text{continue} \rangle$  in  $i + 3$  rounds then
39:     $v$  exits the for-loop.
40:  end if
41:  // Iteration  $j$  ends after the  $\text{continue}$  messages have been in transit for  $i + 3$  rounds.
42: end for // End of phase  $i$ 

43: if  $v$  has decided (possibly in some earlier phase and  $v$  receives a  $\langle \text{continue} \rangle$  message then
44:    $v$  reenters the for-loop and updates its value of  $i$  to the current phase value. (It can do so by keeping track of the number of rounds since starting.)
45: end if

```

5.1.1 Analysis of the early phases of the algorithm: when $i < \rho$

We first show that, during the early phases of the algorithm, nodes in **GoodTL** do not add corrupted information to their **shortestPath** variable (see Lemma 6).

Lemma 6. *Consider any phase $i < \rho$, some iteration j , and some $u \in \text{GoodTL}$. At the end of iteration j , it holds that either **shortestPath** = **none** or **shortestPath** corresponds to a shortest path in G starting at some node v that generated a beacon message and ending at u .*

Proof. Since $u \in \text{GoodTL}$ and $i < \rho$, all Byzantine nodes are at a distance of at least $i + 2$ from u , and hence no information injected by a Byzantine node can reach u until it stops waiting for beacon messages in iteration j . It follows that any information that was added to **shortestPath** corresponds to a path in G . \square

In Lemma 7 we show an upper bound on the number of nodes that are all located at the closest possible distance to $u \in \text{GoodTL}$ such that u will blacklist them if they generate beacon messages. We note that some of or even all of such nodes may be good nodes, but that does not cause any conflicts with our argument here. We will use this lemma together with the tree-like property in to argue that the remaining non-blacklisted nodes (and their expanded neighbors) provide a sufficiently large set (see Lemma 8) for making it likely that some node generates a beacon message (see Lemma 9 and Lemma 10).

Lemma 7. *Consider any phase $i < \rho$ and some good node $u \in \text{GoodTL}$ that has not yet decided at the start of i . For each iteration j , node u blacklists at most one node in its $\lceil (1 - \epsilon)i \rceil$ -boundary $\mathcal{D}(u, \lceil (1 - \epsilon)i \rceil)$ (and none of the nodes that are at a lesser distance).*

Proof. Assume towards a contradiction that, in some iteration j , node u adds at least two nodes $w_1, w_2 \in \mathcal{D}(u, \lceil (1 - \epsilon)i \rceil)$ to its blacklist. By the code of the algorithm, it follows that the ids of w_1 and w_2 must both be in **shortestPath** at the end of iteration j . Without loss of generality, suppose that **shortestPath** = $(v, \dots, w_1, \dots, w_2, \dots, u)$, i.e., v is the origin of the beacon message that caused the update to **shortestPath** in iteration j . Since $w_1, w_2 \in \mathcal{D}(u, \lceil (1 - \epsilon)i \rceil)$ it follows that there exists a path $P_1 = (w_1, \dots, u)$ of length $\lceil (1 - \epsilon)i \rceil$ between w_1 and u that does not contain w_2 .

However, this means that u must have received a beacon message containing a path field that contains the concatenation of paths $Q' = (v, \dots, w_1)$ and P_1 , where $|Q'| < |\text{shortestPath}|$. This contradicts the assumption that both w_1 and w_2 are in **shortestPath**, and completes the proof. \square

Lemma 8. *Let BL_u^* denote the set of nodes added to u 's blacklist during phase $i < \rho$ and let A_u^* be the set of nodes in $\mathcal{B}(u, i + 2) \setminus BL_u^*$ having a shortest path to u that does not traverse nodes in BL_u^* . Then, it holds that $|A_u^*| \geq d^i$.*

Proof. Lemma 7 tells us that during phase i , the number of nodes in $\mathcal{D}(u, \lceil (1 - \epsilon)i \rceil)$ that are added to BL_u is at most

$$\begin{aligned} e^{(1-\gamma)i} + 1 &\leq 2e^{(1-\gamma)i} \\ &\leq e^{(1-\gamma)i + \log 2}. \end{aligned} \tag{5}$$

On the other hand,

$$\begin{aligned} |\mathcal{D}(u, \lceil (1 - \epsilon)i \rceil)| &\geq d^{(1-\epsilon)i} \\ &= e^{(1-\epsilon)i \log d} \\ &= e^{(1-\delta)\gamma i}. \end{aligned} \tag{by (3)}$$

This implies that

$$\frac{|\mathcal{D}(u, \lceil (1-\epsilon)i \rceil)|}{2} \geq e^{(1-\delta)\gamma i - \log 2}, \quad (6)$$

To show that at most half of the nodes in the $\lceil (1-\epsilon)i \rceil$ -boundary of u are blacklisted, it suffices if the right-hand side of (5) is upper bounded by (6). This is true if

$$(1-\gamma)i + \log 2 \leq (1-\delta)\gamma i - \log 2,$$

which holds if

$$\gamma \geq \frac{1}{2-\delta} + \frac{2 \log 2}{(2-\delta)i}. \quad (7)$$

By the code of the algorithm, we know that $i \geq \frac{2 \log 2}{(2-\delta)\eta}$ (see Line 1) and hence (7) is guaranteed by the assumed bound on γ stated in (2).

So far, we have shown that there is a set S' of at least half of the nodes in the $\lceil (1-\epsilon)i \rceil$ -boundary of u that are *not* in the phase i blacklist BL_u^* of u , i.e.,

$$|S'| \geq \frac{|\mathcal{D}(u, \lceil (1-\epsilon)i \rceil)|}{2} \geq d^{(1-\epsilon)i - \log 2}. \quad (8)$$

Since $i < \rho$ and $u \in \text{GoodTL}$, it follows that each node in S' is the root of a d -ary subtree of depth at least $\lfloor \epsilon i \rfloor + 2$. By the tree-like property of u , we know that the sets of nodes in these trees are pairwise disjoint. Let T be the set of nodes that are in these trees. By the above,

$$\begin{aligned} |A_u^*| &\geq |T| \geq \frac{|\mathcal{D}(u, \lceil (1-\epsilon)i \rceil)|}{2} \cdot d^{\lfloor \epsilon i \rfloor + 2} \\ &\geq d^{\lceil (1-\epsilon)i \rceil - \log_d(2) + \lfloor \epsilon i \rfloor + 2} \end{aligned} \quad (\text{by (6)})$$

and, assuming $\log_d(2) \leq 1$, we get

$$\begin{aligned} |A_u^*| &\geq d^{\lceil (1-\epsilon)i \rceil + \lfloor \epsilon i \rfloor + 1} \\ &\geq d^i. \end{aligned}$$

In the remainder of the proof, we argue that none of the nodes in T is blacklisted by u .

Consider any node $w \in T$. The only way that w can be added to BL_u^* is that $w \in \text{shortestPath}$ during some iteration j . However, by the tree-like property, we know that any shortest path from w to u must pass through some node $w' \in S'$ and hence, by Lemma 7, shortestPath must contain w' . This contradicts the assumption that the nodes in S' are never blacklisted, and thus it follows that none of the nodes in T end up in u 's phase i blacklist BL_u^* . \square

We now show that a large fraction of the nodes in GoodTL do not decide in the first $o(\log n)$ phases.

Lemma 9. *For any $u \in \text{GoodTL}$, $\Pr[u \text{ decides in phase } i] \leq \exp(-\frac{c_1 i}{2})$.*

Proof. Consider some $u \in \text{GoodTL}$ that has not yet decided at the start of phase i . By Lemma 8, we know there is a set A_u^* of at least d^i nodes such that a beacon message generated by a node in A_u^* is guaranteed to reach u within $(i+2)$ rounds. By assumption, u has not yet decided at the start of phase i , and hence it must have sent out a $\langle \text{continue} \rangle$ message at the end of the previous phase $(i-1)$, which was forwarded for $(i-1) + 3 = i+2$ rounds. Given that $A_u^* \subseteq \mathcal{B}(u, i+2)$, it follows that this beacon message must have reached all nodes in A_u^* . Hence, any node in A_u^* that

has already decided (possibly during a much earlier phase), will nevertheless execute the for-loop and try to generate beacon messages in the current phase i . It follows that u does not decide if at least one node in A_u^* generates a beacon message.

The probability that none of the nodes in A_u^* becomes active during an iteration of phase i is at most $(1 - \frac{c_1}{d^i})^{d^i} \leq e^{-c_1 i}$. By taking a union bound over the $e^{(1-\gamma)i} + 1$ iterations of phase i , it follows that

$$\begin{aligned} \Pr[u \text{ decides in phase } i] &\leq \left(e^{(1-\gamma)i} + 1\right) e^{-c_1 i} \\ &= e^{(1-\gamma-c_1)i} + e^{-c_1 i} \\ &\leq 2e^{-(c_1-1)i} && (\text{since } 1 - \gamma \leq 1) \\ &\leq e^{-(c_1/2)i}. \end{aligned}$$

□

Lemma 9 promises us that any individual node has a small probability of error when $i < \rho$. So the expected number of nodes to make an error is also small. We, however, want to show a high probability bound on the number of nodes that make a mistake. In order to show that, we proceed along the usual way of formulating an indicator random variable and then computing the expectation of the sum of the individual indicator random variables by using the principle of linearity of expectation. We show the high probability bound by using the method of *bounded differences* (Azuma's Inequality, more specifically).

Now to the formal description. Let Y_i^v be an indicator random variable which is 1 if and only if v decides i to be a correct estimate of $\log n$. Lemma 9 shows that $\Pr[Y_i^v = 1] < \exp(-\frac{c_1 i}{2})$. Now let

$$Y_i = \sum_{v \in V} Y_i^v.$$

That is, Y_i denotes the number of nodes that decide *wrongly* in the i^{th} phase. We recall once again that here we are interested only in the case where $i < \rho$. Then Y_i cannot be too large, i.e., not too many nodes can decide wrongly in one phase.

Lemma 10. $\Pr[Y_i > 2n \cdot \exp(-\frac{c_1 i}{2})] < \exp(-\frac{\sqrt{n}}{2})$ if $i < \rho$.

Proof.

$$\begin{aligned} \mathbb{E}[Y_i] &= \mathbb{E}\left[\sum_{v \in V} Y_i^v\right] = \sum_{v \in V} \mathbb{E}[Y_i^v] && (\text{by linearity of expectation}) \\ &= \sum_{v \in V} \Pr[Y_i^v = 1] && (\text{since } Y_i^v \text{ is an indicator random variable}) \\ &< \sum_{v \in V} \exp(-\frac{c_1 i}{2}) && (\text{from Lemma 9}) \\ &= n \cdot \exp(-\frac{c_1 i}{2}). \end{aligned} \tag{9}$$

Two vertices v and w are *independent* if their $(i+3)$ -distance neighborhoods do not intersect, i.e., if the distance between them is $\geq 2i+7$. In other words, v going defective can affect only those vertices that are within a distance of $(2i+6)$ to v . In a $(d+1)$ -regular graph, the number of vertices

that are within a $(2i + 6)$ -distance of v is at most $d^{(2i+7)}$. By the Azuma-Hoeffding Inequality [17, Theorem 5.1],

$$\begin{aligned}
& \Pr[Y_i - \mathbb{E}[Y_i] \geq n \cdot \exp(-\frac{c_1 i}{2})] \\
& \leq \exp(-\frac{(n \cdot \exp(-\frac{c_1 i}{2}))^2}{2n \cdot d^{2(2i+7)}}) \\
& = \exp(-\frac{n \cdot \exp(-c_1 i)}{2d^{4i+14}}) \\
& = \exp(-\frac{n}{2 \cdot \exp(c_1 i) \cdot d^{4i+14}}) \\
& = \exp(-\frac{n}{2e^k}),
\end{aligned} \tag{10}$$

say, where

$$k \stackrel{\text{def}}{=} (4i + 14) \ln d + c_1 i. \tag{11}$$

But we have from Equation 4 that

$$\begin{aligned}
i < \rho & < \frac{\frac{1}{2} \ln n - 14 \ln d - 1}{4 \ln d + c_1} \\
& \implies 4i \ln d + c_1 i < \frac{1}{2} \ln n - 14 \ln d \\
& \text{or, } (4i + 14) \ln d + c_1 i < \frac{1}{2} \ln n \\
& \text{or, } k < \frac{1}{2} \ln n \quad (\text{follows from the definition of } k \text{ in Equation 11})
\end{aligned}$$

Substituting the value of k in Equation 10, we get that

$$\begin{aligned}
& \Pr[Y_i - \mathbb{E}[Y_i] \geq n \cdot \exp(-\frac{c_1 i}{2})] \\
& < \exp(-\frac{n}{2 \exp(\frac{1}{2} \ln n)}) \\
& = \exp(-\frac{n}{2\sqrt{n}}) = \exp(-\frac{\sqrt{n}}{2}).
\end{aligned} \tag{12}$$

Equation 12 combined with Equation 9 yields the result. \square

As an immediate consequence of Lemma 10, we can take a union bound over the phases up to $i = c, c + 1, c + 2, \dots, \rho$ to obtain that, whp, the total number of nodes in **GoodTL** that decide early is at most

$$\sum_{i=1}^{\rho} \frac{2n}{e^{(c_1/2)i}} \leq \frac{2n}{e^{c_1/2} - 1}. \tag{13}$$

5.1.2 Analysis of the later phases of the algorithm: when $i = \lceil \log n \rceil$

Once a node $u \in \text{GoodTL}$ proceeds beyond phase ρ , it has obtained a sufficiently good estimate of $\log n$, and hence our goal is to show that it is likely to decide. For this part of the analysis, we need to deal with the possibility that conflicting information originating at Byzantine nodes reaches u during an iteration. However, in the following analysis, we show that u is unlikely to increase its phase counter above $\lceil \log n \rceil$.

Lemma 11. Consider phase $i = \lceil \log n \rceil$. The following hold with probability at least $1 - O(\frac{1}{n})$:

1. No good node becomes active.
2. Every node in **GoodTL** decides at the end of phase $i = \lceil \log n \rceil$.

Proof. Let $\text{Active}(u, j)$ be the event that a good node u becomes active in iteration j of phase i . For any $u \in \text{GoodTL}$ and any iteration j , it holds that

$$\Pr[\text{Active}(u, j)] = \frac{c_1 \cdot i}{d^i} \leq \frac{c_1 \log n}{n^{\log d}}.$$

By taking a union bound over all good nodes and over all $e^{(1-\gamma)i} + 1$ iterations, we get

$$\begin{aligned} \Pr[\exists u: \text{Active}(u, j)] &\leq \frac{c_1 \log n}{n^{\log(d)-1}}; \\ \Pr[\exists j \exists u: \text{Active}(u, j)] &\leq \left(e^{\log(n)(1-\gamma)} + 1 \right) \frac{c_1 \log n}{n^{\log(d)-1}} \\ &\leq 2n^{1-\gamma} \frac{c_1 \log n}{n^{\log(d)-1}} \\ &\leq \frac{2c_1 \log n}{n^{\log d - 2}}. \end{aligned}$$

This completes the proof of Part (1), assuming $\log d \geq 4$.

To show Part (2), we *condition* on Part (1) being true, and assume towards a contradiction that there is an undecided node $u \in \text{GoodTL}$ that does not decide in phase i . We will argue that u blacklists at least one Byzantine node in each iteration j of phase i .

By the code of the algorithm, u has set $\text{shortestPath} \leftarrow (v_1, v_2, \dots, v_k)$, for some $k \leq i + 1$, which is the path information of the first beacon message that it received in iteration j .

Note that we cannot be sure that v_1 is the id of a Byzantine node, as it could have happened that some other Byzantine node v_ℓ ($\ell \in [2, k]$) tampered with the prefix $(v_1, \dots, v_{\ell-1})$ before that message reached u . However, by Lemma 1, we know any Byzantine node is at least $\lfloor (1 - \delta)\gamma \log_d n \rfloor$ hops away from u . In particular, this guarantees that the path suffix P' , which consists of the last $\lfloor (1 - \delta)\gamma \log_d n \rfloor$ nodes in path P , contains only ids of good nodes. Hence at least one Byzantine node's id must be in the path prefix Q (where we obtain Q by removing P' from P), as we have assumed that only Byzantine nodes generate beacon messages at this point.

We will now argue that all nodes in Q are blacklisted by u . By the description of the algorithm, u blacklists only nodes that have a distance of at least $\lfloor (1 - \epsilon)i \rfloor$ from u . We observe that

$$\lfloor (1 - \epsilon)i \rfloor \leq (1 - \epsilon)i = (1 - \delta)\gamma \log_d n. \quad (\text{by (3)})$$

It follows that the entire prefix Q will be blacklisted. Thus, we have shown that u does not accept a beacon message that visits any of the nodes in Q in a future iteration of this phase.

By the above reasoning, we know that u blacklists at least 1 Byzantine node in each iteration. Recall that u executes $e^{(1-\gamma)i} + 1 \geq n^{1-\gamma} + 1$ iterations in phase i . Given that there are only $n^{1-\gamma}$ Byzantine nodes in the network, it follows that there exists an iteration in which u does not set its variable shortestPath to a value different from **none** since all Byzantine nodes are already blacklisted at that point. We conclude that u decides in Line 29, yielding a contradiction. \square

Lemma 12. At least $(1 - \beta)n$ nodes decide within $O(n^{1-\gamma} \log^2 n)$ rounds of the algorithm.

Proof. Lemma 11(b) tells us that every node in **GoodTL** decides by phase $i = \lceil \log n \rceil$ with high probability. By the description of the algorithm, each phase i consists of $2i + 3$ rounds and hence the total number of rounds executed until that point is $O(\log^2 n)$. \square

We now combine the previous lemmas to complete the proof of Theorem 2.

Proof of Theorem 2. We focus on nodes in **GoodTL**. From Lemma 10 we know that $\Omega(n)$ nodes in **GoodTL** will proceed to at least phase $\rho = \Omega(\log n)$ before deciding and thus we can set the parameter β of the theorem statement accordingly. On the other hand, Lemma 11 guarantees that all of these nodes decide by the end of phase $\lceil \log n \rceil$ with high probability.

The claim on the running time follows immediately from Lemma 12. \square

Remark 2. The approximation factor mentioned in Theorem 2 is not universal. It may be different for different nodes, but in all cases it is bounded by the quantity $\lceil \frac{\log n}{\rho} \rceil$, where ρ is as defined in Equation (4). Also, while the estimates may vary by a constant factor, it holds with high probability that all the nodes in **GoodTL** have estimates that are upper-bounded by $\lceil \log n \rceil$, i.e., the estimates of $\log n$ are upper-bounded by an additive constant term (which is 1 basically).

Remark 3. We point out that a node in **GoodTL** may reenter the for-loop and participate in sending out beacons even after it has already decided (see Line 44). However, in Corollary 1 below, we show that in the benign case where there are no Byzantine nodes in the network, the algorithm computes $\log(n)$ exactly and all nodes terminate.

Corollary 1. *Suppose that all nodes are good. Then the algorithm terminates in $O(\log(n))$ rounds, and whp, $\Omega(n)$ nodes decide on $\lceil \log(n) \rceil$ and stop sending messages.*

Proof. Lemmas 10 and 11 tell us that $\Omega(n)$ nodes will proceed to phase $i = \lceil \log(n) \rceil$. Moreover, none of the good nodes will generate a beacon message with high probability at that point. Thus no node will send a continue message and all nodes will exit the for-loop. \square

6 Impossibility result

We have seen that both of our algorithms crucially rely on the expansion properties of the underlying network. In Theorem 3, we show that having sufficient expansion is necessary for obtaining *any* approximation of $\log(n)$. In the proof, we make use of the fact that a single Byzantine node can trick the honest nodes into believing that there may be some large number of nodes hidden “behind” the Byzantine node, and the honest nodes have no way of verifying whether this bottleneck actually exists.

Theorem 3. *There is no randomized algorithm that ensures that more than $\lceil \frac{n}{2} \rceil$ nodes output an approximation of $\log n$ in the presence of one Byzantine node with probability at least $(1 - \epsilon)$, if there are no restrictions on the network topology of the given n -node network, for any constant $0 < \epsilon < 1$.*

Proof. For the sake of a contradiction, suppose that there is an algorithm A that solves the counting problem and let C_n be an arbitrary graph of size n . Fix any approximation factor $c = c(n) > 0$, which includes c being a constant as a special case. Suppose that an execution γ of algorithm A results in a set S of more than $\lceil \frac{n}{2} \rceil$ nodes, where every $u_i \in S$ outputs an estimate $\hat{\ell}_i$ such that $\frac{\ell}{c} \leq \hat{\ell}_i \leq c\ell$, for some ℓ . Then we say that A decides on a common estimate of ℓ in execution γ .

For a given n , let $t \geq n$ be the smallest integer such that the probability of A deciding on a common estimate of $\log(nt)$ when executing on network C_n is at most $\frac{1-\epsilon}{t}$, where $\epsilon > 0$ is

the assumed constant error probability of the algorithm A . Since we assume that decisions are irrevocable (see Def. 2), we know that A can decide at most one common estimate in a single execution, the event of producing common estimate ℓ_1 and the event of producing common estimate ℓ_2 , where $c \ell_1 < \frac{\ell_2}{c}$, are mutually exclusive. If t does not exist, then the algorithm has probability $> \frac{1-\epsilon}{k}$ to output $\log(nk)$ as the common estimate, for all $k \geq n$. However, by summing up the probabilities of these (mutually exclusive) events we get $\sum_{k=n}^{\infty} \frac{1-\epsilon}{k} > 1$, i.e., the probabilities of outputting the common estimates do not form a valid probability distribution. It follows that t exists.

Now consider a graph H of t copies of C_n where the Byzantine node b is part of each copy, i.e., node b has degree $t \cdot \deg(b)$ where $\deg(b)$ is the degree of b in C_n . For each copy C_n , node b outputs the same set of messages and local state transitions, as are required by an execution of algorithm A in the network C_n for some given random coin flips when b is an honest node. For the algorithm to output a common estimate of $\log(nt)$ when executing on H , at least $> \frac{nt}{2}$ nodes need to output a common estimate of $\log(nt)$, which involves at least $\frac{n}{2}$ nodes in at least $\frac{t}{2}$ copies of C_n . Since the nodes in any given copy of C_n cannot distinguish the execution in C_n from the execution on H and nodes in each individual C_n have probability at most $\frac{1-\epsilon}{t}$ of outputting the required approximation of $\log(nt)$, we can take a union bound over the $\frac{t}{2}$ copies of C_n . Thus, for the probability of the algorithm to produce a common estimate in H , we obtain $\frac{1-\epsilon}{t} \frac{t}{2} \leq \frac{1-\epsilon}{2}$, a contradiction to the assumed probability of success being $\geq (1-\epsilon)$. \square

7 Conclusion and Open Problems

In this paper we take a step towards designing localized, secure, robust, and scalable distributed algorithms for large-scale networks. We presented two distributed protocols for the fundamental Byzantine counting problem. Our work leaves many questions open. While our deterministic algorithm runs in optimal $O(\log n)$ rounds, the randomized algorithm takes rounds that is essentially proportional to the number of Byzantine nodes in the network. Thus a main open problem would be to show a polylogarithmic round algorithm for Byzantine counting using small messages or to prove that this is not possible. Another open problem is to show a algorithm that can tolerate a significantly larger number of Byzantine nodes, e.g., $\Theta(n)$ Byzantine nodes.

References

- [1] Lorenzo Alvisi, Dahlia Malkhi, Evelyn Pierce, and Michael K. Reiter. Fault detection for Byzantine quorum systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(9):996–1007, September 2001. doi:10.1109/71.954640.
- [2] John Augustine, Anisur Rahaman Molla, Ehab Morsy, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Storage and search in dynamic peer-to-peer networks. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 53–62, New York, NY, USA, 2013. ACM. URL: <http://doi.acm.org/10.1145/2486159.2486170>, doi:10.1145/2486159.2486170.
- [3] John Augustine, Gopal Pandurangan, and Peter Robinson. Fast Byzantine agreement in dynamic networks. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 74–83, New York, NY, USA, 2013. ACM. URL: <http://doi.acm.org/10.1145/2484239.2484275>, doi:10.1145/2484239.2484275.

- [4] John Augustine, Gopal Pandurangan, and Peter Robinson. Fast Byzantine leader election in dynamic networks. In Yoram Moses, editor, *Distributed Computing: 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, pages 276–291, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. URL: http://dx.doi.org/10.1007/978-3-662-48653-5_19, doi:10.1007/978-3-662-48653-5_19.
- [5] John Augustine, Gopal Pandurangan, and Peter Robinson. Distributed algorithmic foundations of dynamic networks. *SIGACT News*, 47(1):69–98, March 2016. URL: <http://doi.acm.org/10.1145/2902945.2902959>, doi:10.1145/2902945.2902959.
- [6] John Augustine, Gopal Pandurangan, Peter Robinson, Scott Roche, and Eli Upfal. Enabling robust and efficient distributed computation in dynamic peer-to-peer networks. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 350–369, October 2015. doi:10.1109/FOCS.2015.29.
- [7] John Augustine, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12*, pages 551–569, Philadelphia, PA, USA, 2012. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=2095116.2095163>.
- [8] Amitabha Bagchi, Ankur Bhargava, Amitabh Chaudhary, David Eppstein, and Christian Scheideler. The effect of faults on network expansion. *Theory of Computing Systems*, 39(6):903–928, Nov 2006. doi:10.1007/s00224-006-1349-0.
- [9] Marc Barthélémy and Luís A. Nunes Amaral. Small-world networks: Evidence for a crossover picture. *Physical Review Letters*, 82:3180–3183, April 1999. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.82.3180>, doi:10.1103/PhysRevLett.82.3180.
- [10] Piotr Berman and Juan A. Garay. Cloture votes: $(\frac{n}{4})$ -Resilient distributed consensus in $(t+1)$ rounds. *Mathematical Systems Theory*, 26(1):3–19, 1993. doi:10.1007/BF01187072.
- [11] Piotr Berman and Juan A. Garay. Fast consensus in networks of bounded degree. *Distributed Computing*, 7(2):67–73, December 1993. doi:10.1007/BF02280836.
- [12] Bela Bollobas and Harold Neville Vazeille Temperley. *Random graphs*, pages 80–102. London Mathematical Society Lecture Note Series. Cambridge University Press, 1981. URL: <https://www.cambridge.org/core/books/combinatorics/random-graphs/560447624AEABC8C19A05BFF191453E1>, doi:10.1017/CB09780511662157.006.
- [13] Edward Bortnikov, Maxim Gurevich, Idit Keidar, Gabriel Kliot, and Alexander Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340 – 2359, 2009. Preliminary version in PODC 2008. URL: <http://www.sciencedirect.com/science/article/pii/S1389128609001182>, doi:<https://doi.org/10.1016/j.comnet.2009.03.008>.
- [14] Soumyottam Chatterjee, Gopal Pandurangan, and Peter Robinson. Network size estimation in small-world networks under Byzantine faults. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 855–865, May 2019. doi:10.1109/IPDPS.2019.00094.

- [15] Soumyottam Chatterjee, Gopal Pandurangan, and Peter Robinson. Network size estimation in small-world networks under Byzantine faults. *CoRR*, abs/2102.09197, 2021. URL: <https://arxiv.org/abs/2102.09197>, arXiv:2102.09197.
- [16] Bogdan S. Chlebus and Dariusz R. Kowalski. Locally scalable randomized consensus for synchronous crash failures. In *Proceedings of the Twenty-first Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '09, pages 290–299, New York, NY, USA, 2009. ACM. URL: <http://doi.acm.org/10.1145/1583991.1584063>, doi:10.1145/1583991.1584063.
- [17] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
- [18] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal on Computing*, 17(5):975–988, 1988. arXiv:<https://doi.org/10.1137/0217061>, doi:10.1137/0217061.
- [19] Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 94–103, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=545381.545392>.
- [20] Joel Friedman. On the second eigenvalue and random walks in random d-regular graphs. *Combinatorica*, 11(4):331–362, 1991. URL: <http://dx.doi.org/10.1007/BF01275669>, doi:10.1007/BF01275669.
- [21] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, Erwan Le Merrer, and Laurent Massoulié. Peer counting and sampling in overlay networks based on random walks. *Distributed Computing*, 20(4):267–278, 2007. URL: <http://dx.doi.org/10.1007/s00446-007-0027-z>, doi:10.1007/s00446-007-0027-z.
- [22] Catherine Greenhill, Svante Janson, Jeong Han Kim, and Nicholas C. Wormald. Permutation pseudographs and contiguity. *Combinatorics, Probability and Computing*, 11(3):273–298, 2002. doi:10.1017/S0963548301005065.
- [23] Fabiola Greve, Murilo Santos de Lima, Luciana Arantes, and Pierre Sens. A time-free Byzantine failure detector for dynamic networks. In *2012 Ninth European Dependable Computing Conference*, EDCC 2012, pages 191–202, May 2012. URL: <https://ieeexplore.ieee.org/document/6214774>, doi:10.1109/EDCC.2012.28.
- [24] Rachid Guerraoui, Florian Huc, and Anne-Marie Kermarrec. Highly dynamic distributed computing with Byzantine failures. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, PODC '13, pages 176–183, New York, NY, USA, 2013. ACM. URL: <http://doi.acm.org/10.1145/2484239.2484263>, doi:10.1145/2484239.2484263.
- [25] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. The case for Byzantine fault detection. In *Proceedings of the 2nd Conference on Hot Topics in System Dependability - Volume 2*, HOTDEP 2006, pages 5–5, Berkeley, CA, USA, 2006. USENIX Association. URL: <http://dl.acm.org/citation.cfm?id=1251014.1251019>.
- [26] Kirsten Hildrum and John Kubiawicz. *Asymptotically Efficient Approaches to Fault-Tolerance in Peer-to-Peer Networks*, pages 321–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. URL: http://dx.doi.org/10.1007/978-3-540-39989-6_23, doi:10.1007/978-3-540-39989-6_23.

- [27] Keren Horowitz and Dahlia Malkhi. Estimating network size from local information. *Information Processing Letters*, 88(5):237–243, 2003. URL: <https://bit.ly/2CUAcAh>.
- [28] Sidharth Jaggi, Michael Langberg, Sachin Katti, Tracey Ho, Dina Katabi, Muriel Médard, and Michelle Effros. Resilient network coding in the presence of Byzantine adversaries. *IEEE Trans. Information Theory*, 54(6):2596–2603, 2008. doi:10.1109/TIT.2008.921711.
- [29] Kim Potter Kihlstrom, Louise E. Moser, and P. M. Melliar-Smith. Byzantine fault detectors for solving consensus. *The Computer Journal*, 46(1):16–35, January 2003. URL: <https://ieeexplore.ieee.org/document/8129490>, doi:10.1093/comjnl/46.1.16.
- [30] Valerie King and Jared Saia. Faster agreement via a spectral method for detecting malicious behavior. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 785–800, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=2634074.2634132>.
- [31] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 990–999, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=1109557.1109667>.
- [32] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 87–98, Washington, DC, USA, 2006. IEEE Computer Society. URL: <http://dx.doi.org/10.1109/FOCS.2006.77>, doi:10.1109/FOCS.2006.77.
- [33] Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. On the complexity of universal leader election. *Journal of the ACM*, 62(1):7:1–7:27, March 2015. URL: <http://doi.acm.org/10.1145/2699440>, doi:10.1145/2699440.
- [34] Ching Law and Kai-Yeung Siu. Distributed construction of random expander networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2133–2143, Los Alamitos, CA, USA, April 2003. IEEE Computer Society. URL: <https://doi.ieeecomputersociety.org/10.1109/INFCOM.2003.1209234>, doi:10.1109/INFCOM.2003.1209234.
- [35] Christoph Lenzen, Joel Rybicki, and Jukka Suomela. Efficient counting with optimal resilience. *SIAM Journal on Computing*, 46(4):1473–1500, 2017. arXiv:<https://doi.org/10.1137/16M107877X>, doi:10.1137/16M107877X.
- [36] Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in anonymous dynamic networks under worst-case adversary. In *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*, ICDCS '14, pages 338–347, Washington, DC, USA, 2014. IEEE Computer Society. URL: <http://dx.doi.org/10.1109/ICDCS.2014.42>, doi:10.1109/ICDCS.2014.42.
- [37] Moni Naor and Udi Wieder. *A Simple Fault Tolerant Distributed Hash Table*, pages 88–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. URL: http://dx.doi.org/10.1007/978-3-540-45172-3_8, doi:10.1007/978-3-540-45172-3_8.

- [38] Mikhail Nesterenko and Sébastien Tixeuil. *Discovering Network Topology in the Presence of Byzantine Faults*, pages 212–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. URL: http://dx.doi.org/10.1007/11780823_17, doi:10.1007/11780823_17.
- [39] Calvin Newport and Peter Robinson. Fault-tolerant consensus with an abstract MAC layer. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 38:1–38:20, 2018. doi:10.4230/LIPIcs.DISC.2018.38.
- [40] Gopal Pandurangan. *Distributed Network Algorithms*. 2019. (*unpublished manuscript*).
- [41] Gopal Pandurangan and Amitabh Trehan. Xheal: a localized self-healing algorithm using expanders. *Distributed Computing*, 27(1):39–54, February 2014. doi:10.1007/s00446-013-0192-1.
- [42] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. URL: <https://epubs.siam.org/doi/abs/10.1137/1.9780898719772>, arXiv:<https://epubs.siam.org/doi/pdf/10.1137/1.9780898719772>, doi:10.1137/1.9780898719772.
- [43] Christian Scheideler. How to spread adversarial nodes?: Rotate! In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, STOC '05*, pages 704–713, New York, NY, USA, 2005. ACM. URL: <http://doi.acm.org/10.1145/1060590.1060694>, doi:10.1145/1060590.1060694.
- [44] Tallat M. Shafaat, Ali Ghodsi, and Seif Haridi. *A Practical Approach to Network Size Estimation for Structured Overlays*, pages 71–83. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. URL: http://dx.doi.org/10.1007/978-3-540-92157-8_7, doi:10.1007/978-3-540-92157-8_7.
- [45] Hakan Terelius, Damiano Varagnolo, and Karl Henrik Johansson. Distributed size estimation of dynamic anonymous networks. In *2012 IEEE 51st IEEE Conference on Decision and Control, CDC '12*, pages 5221–5227, December 2012. doi:10.1109/CDC.2012.6425912.
- [46] Eli Upfal. Tolerating a linear number of faults in networks of bounded degree. *Information and Computation*, 115(2):312 – 320, 1994. URL: <http://www.sciencedirect.com/science/article/pii/S0890540184710996>, doi:<http://dx.doi.org/10.1006/inco.1994.1099>.
- [47] Ruud van de Bovenkamp, Fernando Kuipers, and Piet Van Mieghem. Gossip-based counting in dynamic networks. In Robert Bestak, Lukas Kencl, Li Erran Li, Joerg Widmer, and Hao Yin, editors, *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part II*, IFIP '12, pages 404–417, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-30054-7_32.
- [48] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, June 1998. URL: <http://dx.doi.org/10.1038/30918>, doi:10.1038/30918.
- [49] Nicholas Wormald. *Models of random regular graphs*, pages 239–298. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

A Expander Subgraph Lemma

For completeness, we restate the following lemma; the original proof is in [6].

Lemma 13 (cf. Lemma 3 in [6]). *Let G be a n -node graph with expansion ϕ and constant node degree d and suppose that all nodes in a set F are removed from G , where $|F| = o(n)$. Then, for any positive constant $c < 1$, there exists a subgraph H of G such that*

1. H has expansion $c\phi$, and
2. $|H| \geq n - |F| \left(1 + \frac{1}{\phi(1-c)}\right)$.

Proof. We adapt the proof of Theorem 2.1 in [8]. Let G_F be the graph yielded by removing the set F from G . Perform the following iterative procedure (cf. Algorithm Prune in [8]):

1. Initialize $G_0 = G_F$.
2. In iteration $i \geq 0$, let S_i be any set of up to $|V(G_i)|/2$ nodes with expansion smaller than $c\phi$.
3. If S_i exists, prune S_i from G_i , i.e., $G_{i+1} = G_i \setminus S_i$.
4. Let H be graph that we get after the final iteration; note that H has expansion $\geq c\phi$.

We now lower bound the size of H . Suppose that the pruning procedure stops after m iterations. For the sake of a contradiction, suppose that

$$\frac{|F|}{\phi(1-c)} < \left| \bigcup_{i=0}^m S_i \right|.$$

Define the set $S = \bigcup_{i=0}^{\ell} S_i$ where ℓ is the smallest index such that exactly one of the following two cases holds:

First, assume that $|S| \leq n/2$. Let $N_{G_i}(S_j)$ denote the neighbors in $G_i \setminus S_j$ of nodes in set S_j . We make use of the following result whose proof follows analogously to Lemma 2.6 of [8]:

Lemma 14 (cf. Lemma 2.6 in [8]). *Suppose that we execute procedure Prune(c). For all j with $0 \leq j < m$, it holds that $|N_{G_F}(\bigcup_{i=0}^j S_i)| \leq c\phi |\bigcup_{i=0}^j S_i|$.*

Since G has expansion of ϕ , it holds that $|N_G(S)| \geq \phi|S|$. On the other hand, Lemma 14 implies that $|N_{G_F}(S)| \leq c\phi|S|$. Thus we get $|F| \geq \phi|S| - c\phi|S| = \phi(1-c)|S|$ and hence $|S| \leq \frac{|F|}{\phi(1-c)}$, yielding a contradiction.

Now, consider the case where $|S| > n/2$. Then, it follows that

$$\left| \bigcup_{i=0}^{\ell-1} S_i \right| \leq \frac{|F|}{\phi(1-c)}, \tag{14}$$

but $|S_\ell| > n/2 - \frac{|F|}{\phi(1-c)}$. (Note that if $|S_\ell| \leq \frac{n}{2} - \frac{|F|}{\phi(1-c)}$, then $|S| \leq \frac{n}{2}$ and the first case applies.) Recalling that $F = o(n)$, it follows that $|S_\ell| \in \Theta(n)$. Since S_ℓ was removed when executing Prune(c), we know that $|N_{G_\ell}(S_\ell)| \leq c\phi|S_\ell|$ and, by the expansion of G , we have $|N_G(S_\ell)| \geq \phi|S_\ell|$ as $|S_\ell| \leq n/2$. Thus

$$|N_G(S_\ell)| - |N_{G_\ell}(S_\ell)| \geq \phi(1-c)|S_\ell| = \Theta(n)$$

We observe that the size of the neighborhood of S_ℓ must have been reduced by $\Theta(n)$ either due to the removal of nodes in F or because of the pruning of the sets $S_0, \dots, S_{\ell-1}$. This, however, yields a contradiction, since

$$|F| + \bigcup_{i=0}^{\ell-1} S_i = O(|F|) = o(n).$$

Thus we have shown that

$$\left| \bigcup_{i=0}^m S_i \right| \leq \frac{|F|}{\phi(1-c)},$$

and hence

$$|H| \geq |G| - |F|(1 + \frac{1}{\phi(1-c)}),$$

which completes the proof. □