**Byzantine-Resilient Counting in Networks**

Soumyottam Chatterjee  
Department of Computer Science  
Ashoka University  
Sonipat, Haryana, India  
soumyottam.chatterjee@ashoka.edu.in

Gopal Pandurangan  
Department of Computer Science  
University of Houston  
Houston, Texas  
gopalpandurangan@gmail.com

Peter Robinson  
School of Computer & Cyber Sciences  
Augusta University  
Augusta, Georgia  
perobinson@augusta.edu

**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 \(O(n^{1-\epsilon})\) arbitrarily (adversarially) placed Byzantine nodes for any arbitrarily small (but fixed) positive constant \(\epsilon\). 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 algorithms 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 \(B(n) = n^{1/2-\xi}\) (note that \(n\) and \(B(n)\) are unknown to the algorithm) arbitrarily (adversarially) placed Byzantine nodes, where \(\xi\) is any arbitrarily small (but fixed) positive constant. This algorithm takes \(O(B(n) \log^2 n)\) rounds and outputs a 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 \(\beta\).

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.

**Index Terms**—Byzantine counting, expander graphs, Byzantine faults, randomization, network size estimation.

---

\(^1\)Throughout 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.

---

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** 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 \([1], [2], [3], [4], [5], [6]\). Please refer to Section II 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 **connected** 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. In particular, the seminal paper of Dwork et al. \([1]\), 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.

Byzantine agreement and leader election have been studied extensively for several decades. Dwork et al. [1], Upfal [2], and King et al. [3] 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. 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.

Motivated by the above considerations, the work of Chatterjee et al. [7] studied the Byzantine counting problem in a “small-world” expander network under the assumption that the Byzantine nodes are randomly distributed (cf. Section I-B 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 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 [7] are not directly applicable to the present paper. Indeed, this paper uses a different approach compared to that of [7] (cf. Section I-B). While prior works on Byzantine agreement and leader election required only (sparse) expander networks [1], [2], [3] under an arbitrary distribution of Byzantine nodes, Chatterjee et al. remark that:

\[^{2}\text{i.e., a Watts-Strogatz type network similar to [8].}\]

“... 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., [1], [2], [3], [4], [5], [6]).

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 \(\beta\) 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.

A. 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 \(\gamma\). 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.[1]. Our algorithm works in the CONGEST model, where honest nodes use only small-sized messages (unlike the first algorithm). See Section II for more details about the network model. It tolerates up to \(B(n) = n^{\frac{\gamma}{2}}\) adversarially placed Byzantine nodes,
where $\xi$ is any arbitrarily small (but) fixed positive constant. This algorithm takes $O(B(n)\log^2 n)$ rounds (hence $o(\sqrt{n})$ rounds for $B(n) = n^{2-\frac{\gamma}{2}}$) 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 $\beta$.

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.

**Omitted proofs, additional details and other related work:** Due to lack of space, these are deferred to the full version of this paper [9].

**B. 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 converge-casting 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 [4], [10] which uses exponential distribution. Alternatively, one can use a geometric distribution (see e.g., [11], [12], [13]) to accurately estimate the network size.

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 [7] successfully adapts the geometric distribution to work for their purpose. However, their work [7] 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 [7] 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 [7] 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 $\gamma$ 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 $\gamma$). The algorithm of [7] fails to work for expander networks with arbitrary or adversarial Byzantine node distribution, which is typically assumed in previous works on Byzantine protocols [1], [2], [3], [4], [5], [6].

We note that one can possibly solve Byzantine counting if one can solve Byzantine leader election, as observed in [7], 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.

**C. 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.

**Omitted proofs, additional details and other related work:** Due to lack of space, these are deferred to the full version of this paper [9].

II. COMPUTING MODEL AND PROBLEM DEFINITION

a) 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, [12], [14]; but we point out that our second algorithm ensures that most good nodes send only small-sized messages.

b) 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 [1], [2], [3], [4], [5], [6].

c) 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.

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

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

where $S$ is any subset of $V$ of size at most $\frac{n}{2}$ and $\text{Out}(S)$ 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 $\alpha$ is a fixed positive constant.

e) 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 $d$ (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. It is known that such a random graph is an expander (in fact a Ramanujan expander [15], [16]) with high probability. The $H(n, d)$ model is a well-studied and popular random graph model (see e.g., [17]), and has been used as a model for peer-to-peer networks and self-healing networks [16], [18].

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 [17]. 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[1]). Since it is hard to work directly with the above model, one usually works with the so-called configuration (or pairing) model [19] 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 [1]. 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. [20] 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.

f) 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 $\epsilon$ 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 $L_u$, 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 $c_1 \log n \leq L_u \leq c_2 \log n$.

### III. Preliminaries

We use the notation $B_G(u, i)$ 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 $B_G(S, i) = \bigcup_{u \in S} B_G(u, i)$.

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 $\text{Byz}$, an arbitrary subset of $V$, denote the set of Byzantine nodes with the restriction that $|\text{Byz}| \leq n^{1-\gamma}$, where $\gamma$ is any arbitrarily small (but fixed) positive constant. Then, for any $o(n)$-sized $F \subseteq V$, there exists a set $\text{Good} \subseteq V \setminus F$ of good nodes such that $|\text{Good}| \geq n - 2|F| - o(n)$. Moreover, for each $u \in \text{Good}$, the following hold:

1) $B(u, \lfloor \frac{2}{3} \log n \rfloor)$ 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 $|B_H(u, c \log n)| \leq \frac{|\text{Good}|}{2}$, it holds that every vertex subset $S \subseteq B_H(u, c \log n)$ has a vertex expansion of $\geq \alpha'$ in graph $H$, for any fixed constant $\alpha' < \alpha$.

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

We refer to [21] for various properties of an $H(n,d)$ random graph in some detail. We state the main definitions and lemmas needed here for the sake of completeness. 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. 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 $u$ locally tree-like if no node in $B(u, r)$ 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.

### IV. 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.

Our goal here is to show that a set called $\text{Good}$ 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 $\text{Good}$: in particular, a good node needs to have a distance of $\Omega(\log n)$ from all Byzantine nodes and the graph induced by $\text{Good}$ must have nearly the same vertex expansion as the original network.

A. Description of the algorithm

Throughout the algorithm, each node $u$ locally builds an approximation of its $i$-hop neighborhood for rounds $i = 1, 2, 3, \ldots$, which we denote by $\tilde{B}(u, i)$. To this end, we instruct nodes to simply forward the content of their current $\tilde{B}(u, i)$ 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 $\Delta$, or the addition of spurious edges to vertices that it had already learned about previously.

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 $B(u, i + 1)$. 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 $\Delta$ and a constant vertex expansion $\alpha$. There exists a deterministic LOCAL algorithm such that $n - o(n)$ good nodes decide on a $\left(\frac{1}{2} \log \Delta\right)$-approximation of $\log n$ in $O(\log n)$ rounds in the presence of up to $n^{1-\gamma}$ arbitrarily (adversarially) placed Byzantine nodes.

**B. Analysis of the algorithm**

**Lemma 3.** All nodes in $\text{Good}$ decide on a value of at least $\frac{1}{2} \log \Delta n$.

**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 Round 1. Moreover, $u$ has a distance of at least $\frac{1}{2} \log \Delta n$ from any Byzantine node, and hence it does not receive any inconsistent information during the first $\frac{1}{2} \log \Delta n$ rounds. This ensures $u$ will not decide in Round 1, which completes the inductive base.

Now, consider the inductive step $1 < i < \frac{1}{2} \log \Delta n$, 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

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

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 $\alpha'$, which ensures that $u$ continues to round $i + 1$ without deciding.

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

**B. Analysis of the algorithm**

**Lemma 3.** All nodes in $\text{Good}$ decide on a value of at least $\frac{1}{2} \log \Delta n$.

**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 Round 1. Moreover, $u$ has a distance of at least $\frac{1}{2} \log \Delta n$ from any Byzantine node, and hence it does not receive any inconsistent information during the first $\frac{1}{2} \log \Delta n$ rounds. This ensures $u$ will not decide in Round 1, which completes the inductive base.

Now, consider the inductive step $1 < i < \frac{1}{2} \log \Delta n$, 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

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

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 $\alpha'$, which ensures that $u$ continues to round $i + 1$ without deciding.

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 $\text{Good}$. 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 $v \in \hat{B}_H(u, i)$ and any node $w$, it holds that $e = \{v, w\} \in \hat{B}(u, i)$ if and only if $e \in E(G)$.

**Lemma 5.** Every node in $\text{Good}$ 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.,

$$F \triangleq \hat{B}(u, i) \setminus \text{Good}.$$

We call

$$R \triangleq \hat{B}(u, i) \cap \text{Good}$$

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, 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 $B(u, i)$ 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 $R \subseteq B(u, i - 1)$ and thus $u$ will check $R$’s vertex expansion with respect to graph $B(u, i)$ at the end of round $i + 1$.

To complete the proof, we analyze the expansion-check in our algorithm 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 $\text{Good}$ 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 $R \cap B(u, i)$. Since we assumed that Byzantine nodes did not send inconsistent information, each Byzantine node has at most $\Delta$ neighbors in $F$. It follows that there is a set $S'$ of at most $\Delta n^{1-\gamma} = o(n)$ fake vertices in the set $(B(u, i) \setminus R) \subseteq F$ that have an edge to $R$. To satisfy the expansion-check, 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 $i = \text{diam}(G) + 1$.

Combining Lemmas 3 and 5 shows the claimed bound on the approximation achieved by the $n - o(n)$ nodes in set $\text{Good}$. 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.

V. 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 V-A. As mentioned in Section II, 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 II, this implies that the algorithm works in almost all $d$-regular graphs with probability at least $1 - o(1)$.

In our algorithm, 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.

a) Beacon Messages and Path Fields: At the start of an iteration, a good node $u$ chooses to become active with probability $\Theta(\frac{1}{d'})$, 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 (cf. Definition 3 and Lemma 2) ensures that the number of nodes in a ball is $\Theta(d')$.

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 $(\text{beacon}, u, P)$ 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.

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

First, $v$ checks whether the neighbor from which it received the message does indeed have id $u_k$. To simplify the notation we use the same variable to denote a node and its id. 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 $[(1 - \epsilon)]i$-suffix of $P$ to its blacklist $BL$, where this suffix consists of the last $[(1 - \epsilon)]i$ 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 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 \( (\text{beacon}, P') \) 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 \( \lceil (1 - \epsilon)i \rceil \) away from \( v \) intersect with the nodes already added to \( BL \) during the previous iterations, then \( v \) will not use \( m' \) to update its \( \text{shortestPath} \) 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 \( \text{shortestPath} \) 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 \( \lceil (1 - \epsilon)i \rceil \) 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 \( \lceil (1 - \epsilon)i \rceil \) suffix of the path field by the time the message reaches \( u \), 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 \( \lceil (1 - \epsilon)i \rceil \) 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 \( \text{shortestPath} \) variable is not set, thus causing it to decide on \( i \).

c) 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 \( \delta \) distance \( \lceil (1 - \epsilon)i \rceil \) 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. 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 \( \xi \) and \( \beta \) 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 \( (1 - \beta) \) 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.

A. Analysis of the algorithm

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

\[ \gamma \geq \frac{1}{2 - \delta} + \eta, \]

for any fixed constants \( 0 < \delta \leq \frac{1}{2} \) and \( \eta > 0 \). Note that the smaller \( \delta \) is, the smaller \( \gamma \) is. Therefore maximum Byzantine tolerance is achieved when \( \delta \) is very close to (but slightly greater than) zero and \( \gamma \) is very close
to (but slightly greater than) $\frac{1}{3}$. In that case, the maximum Byzantine tolerance, i.e., the maximum number of Byzantine nodes that our algorithm can tolerate, boils down to $n^{\frac{1}{2} - \epsilon}$, as stated in Theorem 2.

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

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

(2)

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 $\text{Good}$, and that also have the property of $d$-ary trees up to some radius of length $\frac{\log d \cdot n}{\gamma}$.

We will first study the progress of the algorithm at nodes in $\text{GoodTL}$, for the phases up to radius $\rho$, where

$$
\rho = \left\lceil \min \left( (1 - \delta) \gamma \log_d n, \frac{1}{10} \log_d n \right) \right\rceil - 2,
$$

since, in phase $i$, we require the tree-like property to hold up until radius $(i + 2)$.

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 $\text{GoodTL}$ do not add corrupted information to their $\text{shortestPath}$ 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 $\text{shortestPath} = \text{none}$ or $\text{shortestPath}$ 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 node 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 $\text{shortestPath}$ corresponds to a path in $G$. \hfill $\square$

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 $[(1 - \epsilon)i]$-boundary $D(u, [(1 - \epsilon)i])$ (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 $w_1, w_2 \in D(u, [(1 - \epsilon)i])$ to its blacklist. By the code of the algorithm, it follows that the IDs of $w_1$ and $w_2$ must both be in $\text{shortestPath}$ at the end of iteration $j$.

Without loss of generality, suppose that $\text{shortestPath} = (v, \ldots, w_1, \ldots, w_2, \ldots, u)$, i.e., $v$ is the origin of the beacon message that caused the update to $\text{shortestPath}$ in iteration $j$. Since $w_1, w_2 \in D(u, [(1 - \epsilon)i])$ it follows that there exists a path $P_1 = (w_1, \ldots, w_2)$ of length $\lfloor (1 - \epsilon)i \rfloor$ 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 $Q' = (v, \ldots, w_1)$ and $P_1$, where $|Q'| < |\text{shortestPath}|$. This contradicts the assumption that both $w_1$ and $w_2$ are in $\text{shortestPath}$, and completes the proof. \hfill $\square$

**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 $B(u, i + 2) \setminus BL^*_u$ having a shortest path to $u$ that does not traverse nodes in $BL^*_u$. Then, it holds that $|A^*_u| \geq d^\delta$.

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

**Lemma 9.** For any $u \in \text{GoodTL}$,

$$
\Pr[u \text{ decides in phase } i] \leq \exp(-\frac{c_1 i}{2}).
$$

This lemma 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).

**Lemma 10.** For $i < \rho$, the following statement holds with probability greater than $1 - \exp(-\frac{c_2 i}{2})$: No more than $2n \cdot \exp(-\frac{c_1 i}{2})$ nodes in $\text{GoodTL}$ decide during phase $i$.

As an immediate consequence of Lemma 10, we can take a union bound over the phases up to $i = e(c+1, e+$
Lemma 11. Above we show that information originating at Byzantine nodes reaches 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 ⌈log n⌉.

Lemma 11. Consider phase i = ⌈log n⌉. The following hold with probability at least 1 − O(1/n):

1) No good node becomes active.
2) Every node in GoodTL decides at the end of phase i = ⌈log n⌉.

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

Pr[Active(u, j)] = \frac{c_1 \log n}{n^{d/2}} \leq \frac{c_1 \log n}{n^{d/2}}. By taking a union bound over all good nodes and over all \( e^{(1−\gamma)j} + 1 \) iterations, we get Pr[∃u: Active(u, j)] ≤ \frac{c_1 \log n}{n^{d/2}} + 1, and therefore

\[
Pr[\exists j \exists u: Active(u, j)] \leq \left( e^{(\log(n)(1−\gamma) + 1)} \right) \frac{c_1 \log n}{n^{d/2}} \leq 2n^{1−\gamma} \frac{c_1 \log n}{n^{d/2}} \leq \frac{2c_1 \log n}{n^{d/2}}.
\]

This completes the proof of Part (1), assuming log d ≥ 4.

To show Part (2), we condition on Part (1) being true, and assume towards a contradiction that there is an undecided node u ∈ 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 shortestPath ← (v_1, v_2, ..., v_k), for some k ≤ 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 (ℓ ∈ [2, k]) tampered with the prefix (v_1, ..., v_ℓ−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 prefix contains only IDs of good nodes. Hence at least one Byzantine node's ID must be in the path prefix prefix contains only IDs of good nodes. Hence at least one Byzantine node's ID must be in the path prefix.

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 \( \lfloor (1−\epsilon)j \rfloor \) from u. We observe that \( \lfloor (1−\epsilon)j \rfloor \leq (1−\epsilon)i = (1−\delta)\gamma \log_d n, by (2). \)

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 \( e^{(1−\gamma)j} + 1 \geq n^{1−\gamma} + 1 \) 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 at this point, yielding a contradiction.

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 = ⌈log n⌉ 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 \( O(\log^2 n) \).

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 \( \beta \) of the theorem statement accordingly. On the other hand, Lemma 11 guarantees that all of these nodes decide by the end of phase \( \lfloor \log n \rfloor \) with high probability.

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

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 \( \lfloor \log n \rfloor \), where \( \rho \) is as defined in Equation (3). 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 \( \lfloor \log n \rfloor \), i.e., the estimates of \( \log n \) are upper-bounded by an additive constant term (which is 1 basically).
VI. 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 $\left\lceil \frac{n}{2} \right\rceil$ 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 $0 < \epsilon < 1$.

VII. 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 present 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] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal, “Fault tolerance in networks of bounded degree,” *SIAM Journal on Computing*, vol. 17, no. 5, pp. 975–988, 1988. [Online]. Available: https://doi.org/10.1137/0217061

[2] E. Upfal, “Tolerating a linear number of faults in networks of bounded degree,” *Information and Computation*, vol. 115, no. 2, pp. 312 – 320, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0890540184710996

[3] V. King, J. Saia, V. Sanwaler, and E. Vee, “Towards secure and scalable computation in peer-to-peer networks,” in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 87–98. [Online]. Available: http://dx.doi.org/10.1109/FOCS.2006.77

[4] J. Augustine, G. Pandurangan, P. Robinson, and E. 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*, ser. SODA ’12. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2012, pp. 551–569. [Online]. Available: http://dl.acm.org/citation.cfm?id=2095116.2095163

[5] J. Augustine, G. Pandurangan, and P. Robinson, “Fast Byzantine agreement in dynamic networks,” in *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, ser. PODC ’13. New York, NY, USA: ACM, 2013, pp. 74–83. [Online]. Available: http://doi.acm.org/10.1145/2484429.2484425

[6] ——, “Fast Byzantine leader election in dynamic networks,” in *Distributed Computing: 29th International Symposium, DISC 2015, Tokyo, Japan, October 7–9, 2015, Proceedings*, Y. Moses, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 276–291. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-48653-5_19

[7] S. Chatterjee, G. Pandurangan, and P. Robinson, “Network size estimation in small-world networks under Byzantine faults,” in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2019, pp. 855–865.

[8] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, pp. 440–442, June 1998. [Online]. Available: http://dx.doi.org/10.1038/30918

[9] S. Chatterjee, G. Pandurangan, and P. Robinson, “Byzantine-resilient counting in networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 119–131, 2022. [Online]. Available: https://arxiv.org/abs/2204.11951

[10] J. Augustine, G. Pandurangan, and P. Robinson, “Distributed algorithmic foundations of dynamic networks,” *SIGACT News*, vol. 47, no. 1, pp. 69–98, March 2016. [Online]. Available: http://doi.acm.org/10.1145/2902945.2902959

[11] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan, “On the complexity of universal leader election,” *Journal of the ACM*, vol. 62, no. 1, pp. 7:1–7:27, March 2015. [Online]. Available: http://doi.acm.org/10.1145/2699440

[12] G. Pandurangan, *Distributed Network Algorithms, 2019, (unpublished manuscript).*

[13] C. Newport and P. 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*, pp. 38:1–38:20. [Online]. Available: https://doi.org/10.4230/LIPIcs.DISC.2018.38

[14] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719772

[15] J. Friedman, “On the second eigenvalue and random walks in random d-regular graphs,” *Combinatorica*, vol. 11, no. 4, pp. 331–362, 1991. [Online]. Available: http://doi.acm.org/10.1007/BF01275669

[16] C. Law and K.-Y. Siu, “Distributed construction of random expander networks,” in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. Los Alamitos, CA, USA: IEEE Computer Society, April 2003, pp. 2133–2143. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/INFCOM.2003.1209234

[17] N. Wormald, *Models of random regular graphs*, ser. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999, pp. 239–298.

[18] G. Pandurangan and A. Trehan, “Xheal: a localized self-healing algorithm using expanders,” *Distributed Computing*, vol. 27, no. 1, pp. 39–54, February 2014. [Online]. Available: https://doi.org/10.1007/s00446-013-0192-1

[19] B. Bollobas and H. N. V. Temperley, *Random graphs*, ser. London Mathematical Society Lecture Note Series. Cambridge University Press, 1981, pp. 80–102. [Online]. Available: https://www.cambridge.org/core/books/combinatorics/978-0-521-42140-6

[20] C. Greenhill, S. Janson, J. H. Kim, and N. C. Wormald, “Permutation pseudographs and contiguity,” *Combinatorics, Probability and Computing*, vol. 11, no. 3, pp. 273–298, 2002.

[21] S. Chatterjee, G. Pandurangan, and P. Robinson, “Network size estimation in small-world networks under Byzantine faults,” *CoRR*, vol. abs/2102.09197, 2021. [Online]. Available: https://arxiv.org/abs/2102.09197