How Efficient Can Gossip Be?
(On the Cost of Resilient Information Exchange)

Dan Alistarh[1], Seth Gilbert[1], Rachid Guerraoui[1], and Morteza Zadimoghaddam[1,2]

[1] EPFL
[2] MIT

Abstract. Gossip protocols, also known as epidemic dissemination schemes, are becoming increasingly popular in distributed systems. Yet, it has remained a partially open question to determine how robust such protocols can be. We consider a natural extension of the *random phone-call* model by Karp et al. [21], in which we analyze two different notions of robustness: the ability to tolerate *adaptive* failures, and the ability to tolerate *oblivious* failures.

For adaptive failures, we present a new gossip protocol, TrickleGossip, which achieves near-optimal $O(n \log^3 n)$ message complexity; to the best of our knowledge, this is the first epidemic-style protocol that can tolerate adaptive failures. We also show a direct relation between resilience and message complexity, demonstrating that gossip protocols which tolerate a large number of adaptive failures need to use a super-linear number of messages with high probability.

For oblivious failures, we present a new gossip protocol, CoordinatedGossip, that achieves optimal $O(n)$ message complexity. This protocol makes novel use of the *universe reduction* technique to limit the message complexity, resulting in a protocol that is both fast and message-efficient.

1 Introduction

Consider a distributed system consisting of n processes $p_1, \ldots, p_n$, each connected by a pairwise communication channel to every other process. Processes may have unique identifiers, but a process does not know the identifiers of the other processes until it has exchanged information with them.

Disseminating information efficiently and robustly in such a system is a fundamental problem, which can roughly be defined as follows. Each process p_i is initialized with a *rumor* r_i; eventually, each process should learn as many rumors as possible. Moreover, this exchange of information should be *fault-tolerant*: it should succeed even if some of the processes fail (i.e., crash). Any process that does not crash should succeed in disseminating its rumor to every other non-crashed process. This problem is a basic building block in many distributed settings, such as distributed databases [7], failure detection [32], group multicast [3, 14, 28], group membership [24], resource location [23], and, recently, update dissemination for mobile nodes [4].

In this paper, we investigate the relationship between *fault tolerance* and *efficiency* for such dissemination protocols. We consider a natural extension of the *random phone-call* model of Karp et al. [21][3]. In the case of *adaptive* failures, i.e. failures that are

[3] For a detailed discussion of the model distinctions, see Section 3.

history-dependent, we introduce the first protocol that achieves sub-quadratic message complexity. The protocol, which we call TrickleGossip, ensures dissemination while using $O(n \log^3 n)$ messages and $O(\log^2 n)$ communication rounds, with high probability. We also prove a super-linear lower bound for the message complexity of any gossip protocol that tolerates a large number of adaptive failures. In the case of *oblivious* failures, we introduce a new protocol, CoordinatedGossip, that achieves optimal $O(n)$ message complexity and terminates in $O(\log n)$ communication rounds. Surprisingly, this implies that one may achieve a similar level of efficiency in a network with oblivious failures as in a fault-free network. Together, our results highlight an inherent trade-off between efficiency and resilience: in order to tolerate more aggressive failures, protocols need to send more messages. On the other hand, we show that information exchange can be implemented efficiently even in the presence of adaptive failures.

We now describe our results in more detail.

1. Adaptive failures. Our first main result is a gossip protocol, TrickleGossip, that achieves $O(n \log^3 n)$ message complexity, with high probability, when failures are adaptive, i.e., when failures may depend on the ongoing execution. The protocol terminates in $O(\log^2 n)$ rounds, which is near-optimal–a simple modification of the result in [5] yields a lower bound of $\Omega(\log n / \log \log n)$ rounds for the problem.

The challenge, when failures are adaptive, is that the "adversary," which is dictating the failures, may select which processes to crash, and therefore can target specific rumors and prevent them from being disseminated. For example, the adversary may decide to "single out" a process p_i by failing every process that p_i chooses to send a message to. Unless process p_i sends a large number of messages, the adversary can always prevent the associated rumor r_i from spreading to any other process. Our algorithm introduces a scheme for each process to monitor the spread of its rumor, sending more and more messages if it realizes it is being "isolated" from other processes, until its rumor is successfully disseminated. At the same time, the protocol has to prevent too many processes from sending too many messages. Thus the protocol alternates *spreading* rounds that attempt to disseminate rumors, *collection* rounds that attempt to discover new rumors, and *sampling* rounds that attempt to gauge how far rumors have spread.

It is perhaps surprising that gossip can work efficiently when failures are adaptive. Typically, the robustness of epidemic gossip is explained by observing that *the protocol is making random choices*. That is, oblivious failures have a limited effect on the dissemination process. For adaptive failures, this is not the case: the failure pattern adapts to the actual execution of the protocol. One key observation we use is that randomization spreads messages roughly uniformly; our protocol ensures that, by failing a limited set of processes, the adversary can only prevent a proportional set of rumors from being spread. We also ensure that, once a rumor has been learned by enough processes, the adversary can do little to stop its spread to almost all correct processes.

We also show a trade-off between robustness and message complexity in the presence of *adaptive* faults. More precisely, any protocol that tolerates $n(1 - \epsilon)$ process crashes, where $1 \geq \epsilon > 0$ is a function of n, has message complexity $\Omega(n \log(1/\epsilon))$, with high probability. Of note, for small ϵ, e.g. $\epsilon < 1/\log \log n$, this results in a super-linear lower bound on message complexity, the first such bound for gossip protocols in the presence of adaptive faults.

2. Oblivious failures. Our second main result is a gossip protocol, CoordinatedGossip, that achieves optimal $O(n)$ message complexity, with high probability, when failures are oblivious, i.e., independent of the actual execution (and hence independent of random choices made by the processes). This implies that, asymptotically, we can achieve the same level of efficiency in a crash-prone network as in a fault-free network!

The key idea underlying this algorithm is *universe reduction*: we randomly select a small set of coordinators and use them to collect and then disseminate the rumors. The main challenge here is the need for coordinators to work efficiently together; specifically, they must avoid duplicating work (i.e., messages). However, the coordinators do not initially know the identities of the other coordinators, nor how to reach them. The protocol builds simple, yet robust, overlay structures that allow coordinators to communicate amongst themselves, and allow processes to communicate with coordinators.

This result is particularly surprising due to the existence of an $\Omega(n \log \log n)$ lower bound on the message complexity of gossip broadcast by Karp et al. [21]; in the problem of broadcast, a single process attempts to distribute its message to all others. In [21], they consider a model in which, in each round, each process can: (i) contact a random process and (ii) either *push* information to the random process or *pull* information from it. By contrast, in this paper, we allow a process to send more than one message per round, and to reply to processes that communicate with it at later rounds. The capacity to maintain a connection over multiple rounds allows us to circumvent the lower bound and obtain significant improvements.

2 Distributed System Model

We consider a synchronous distributed system consisting of n processes $\{p_1, \ldots, p_n\}$. An execution proceeds in rounds in which each process sends a set of messages, receives a set of messages, and updates its state. Every pair of processes is connected by a pairwise communication channel.

Faults. Up to $t < n/3$ processes may *fail* by crashing. When a process p_i crashes in some round r, any arbitrary subset of its round r messages may be delivered; process p_i then takes no further steps in any round $> r$. A process that does not fail is *correct*. We think of the failures as dictated by an *adversary*.

Unique Ids. Each process has a unique identifier. When the execution begins, a process does not know the identity of any other process in the system[4]. The first time that a process p_i receives a message from another process p_j, it learns the identity of process p_j, and hence it learns on which channel it can send messages to p_j. Formally, each process p_i has access to an unordered set of n channels S_i (including a channel connecting p_i to itself). In each round, each process can send a message on any subset of the available channels, and, in addition, it can reply to any message received in a previous round (using the channel on which that message was delivered).

[4] Often distributed algorithms assume that processes know the identity of their neighbors, since learning this information requires only a single exchange of messages on each channel. However, this neighbor-discovery process requires $\Theta(n^2)$ messages, and so we do not assume that a process knows the identities of its neighbors *a priori*.

This basic model is well suited for studying randomized gossip protocols, as it leads naturally to epidemic dissemination: each process can exchange information with randomly chosen partners by selecting random subsets of its channels[5].

Anonymous Systems. While we assume, for simplicity, that processes have unique identifiers, this assumption is not needed for any of the results in this paper. A process can simply choose an identifier at random from a suitably large namespace (e.g., $\{1, \ldots, \Theta(n^2)\}$), such that, with high probability, every process selects a unique identifier. All the results in this paper continue to hold.

Moreover, there are several reasons why anonymity may be desired. For example, in order to ensure privacy, processes may not want to reveal their identity. Along the same lines, processes may be communicating via an anonymous routing protocol (e.g., *onion routing*). All that is required for the protocols in this paper is the ability to establish a communication session with randomly chosen processes.

Oblivious and Adaptive Failures. We distinguish two types of failures: when failures are independent of the actual execution (and hence independent of the random choices made by the processes), we say that they are *oblivious* (or that the *adversary* is *oblivious*); by contrast, when failures may depend on the ongoing execution, we say that they are *adaptive* (or that the *adversary* is *adaptive*). Formally, when failures are *oblivious*, we assume that a *failure pattern* is fixed prior to the execution; the failure pattern specifies precisely in which round each faulty process fails, and which of its messages are delivered in that round. When failures are *adaptive*, we assume that failures in round r are determined by a *failure function* which takes as input the entire history of rounds $1, \ldots, r-1$.

When we say that a protocol has message complexity M with probability p, this means that for every failure pattern/function, the probability that more than M messages are sent in an execution is no greater than p. The term *with high probability* (w.h.p.) means with probability at least $1 - O(n^{-\gamma})$, for an arbitrary constant $\gamma > 0$.

3 Related Work

The idea of randomized rumor distribution dates back to Frieze and Grimmett [16], and later, Pittel [30] and Feige [15], who showed that one can distribute a single rumor in a complete graph with n vertices in time $\log n + \ln n + o(1)$, as well as in other types of random graphs. Demers et al. [7] showed that rumor spreading can be used for efficient distributed database maintenance, an idea extended in e.g., [27, 29].

In a landmark paper, Karp, Schindelhauer, Shenker and Vöcking [21] considered the problem in the *random phone-call model*. The paper shows how to distribute a rumor with high probability, in the presence of an oblivious adversary using $O(\log n)$ communication rounds and $O(n \log \log n)$ messages. This is shown to be optimal, in the absence of addresses. Moreover, it is shown that, even using addresses, no algorithm can achieve both time and communication optimality.

The results of [21] inspired a significant amount of research. Elsässer et al. [2, 11–13] study extensions of the random phone-call model over various types of connected

[5] By contrast, any deterministic protocol requires $\Omega(n^2)$ messages, since processes do not know the identities of the other processes prior to the execution; this precludes protocols based on constructing specially crafted overlays, such as expander graphs.

random graphs. In [9,10], Doerr et al. consider *quasirandom* rumor spreading, in which each process has a list of its neighbors through which it is allowed to cycle when sending the rumor. Surprisingly, the time bounds of [16] and [15] are still optimal in this augmented model. The robustness of the quasirandom model is analyzed in [8], under the assumption of probabilistic transmission failures. Later, Kempe, Kleinberg, and Demers [22] introduced a gossip-based algorithm for resource location. References listed so far consider an *oblivious* adversary.

Model Comparison. Our model may be seen as an enrichment of the random phone-call model of Karp et al. [21]. Of note, this allows us to achieve $O(n)$ message complexity for oblivious failures, and it allows us to develop protocols for a stronger, adaptive adversary. Compared to the original model of [21], and the later variations in [9,12,13], our model is stronger in that it allows processes to send several messages per round (for example, a process might choose to send messages on *all* n available links in a round), and it allows processes to reply to messages received. Also, note that we consider *gossip* (n rumors to be spread), rather than broadcast.

There has also been much work on gossip when addresses are available, i.e., when processes known in advance the identities of all the other processes in the system. Chlebus and Kowalski [5] present deterministic algorithms (based on an expander-graph construction) that ensure the dissemination of rumors within time $O(\log^2 t)$ using $O(n \text{ polylog } t)$ total messages, where t is the number of failures. Earlier research [6] determined time and cost trade-offs for gossip, in a model in which one neighbor is contacted per round. Gilbert et al. [17] provided upper and lower bounds for the complexity of gossip in an asynchronous environment. A detailed account of prior work on gossip in distributed networks can be found in [19].

A key aspect of the CoordinatedGossip algorithm present in Section 6 is the technique of *universe reduction*, which has been an important tool in developing distributed algorithms. Of particular note are recent algorithms by Ben-Or et al. [1], Kapron et al. [20], King et al. [26], and King and Saia [25] that use universe reduction to solve Byzantine agreement.

In a recent paper [18], Gilbert and Kowalski use the universe reduction technique to develop a crash-tolerant consensus protocol that has optimal communication complexity: it uses only $O(n)$ bits of communication. In the same paper, they discuss how these techniques can be used to solve gossip using $O(n)$ messages. The model in that paper, however, assumes that processes know the identities of their neighbors; as discussed in Section 2, this assumption adds a hidden implicit $\Omega(n^2)$ message cost which we avoid in this paper. Thus, we cannot take advantage of the *work sharing* techniques discussed in [18]; instead a more careful scheme is needed to avoid sending too many messages.

4 Gossip Against an Adaptive Adversary

In this section, we present a gossip algorithm, TrickleGossip, which uses $O(n \log^3 n)$ messages with high probability, and terminates in $O(\log^2 n)$ communication rounds. Previous work [5] shows that the algorithm is optimal in terms of round complexity, up to logarithmic factors.

4.1 Algorithm Description

The execution of the algorithm is divided into two epochs: the *dissemination* epoch and the *confirmation* epoch. We proceed to describe the structure of each epoch. Let β be a large integer constant (e.g., $\beta = 200$), and let $\alpha = 7/12$.

The Dissemination Epoch. The epoch is composed of $\log n$ individual phases, and each phase is composed of $3 \log n + 1$ communication rounds.

The first round of each phase is designed for distributing single rumors. Each process is initially *unmarked*. In the first round of phase number i, each unmarked process sends its rumor to a set of $2^i \log n$ randomly chosen processes (if $i \geq \log n - \log\log n$, then all n processes are chosen).

In the following $\log n$ rounds of phase i, each process, whether marked or unmarked, gathers all the distinct rumors it has received thus far in a *packet*, and sends this packet to $\log n$ randomly chosen neighbors, in every round.

Finally, the last $2 \log n$ rounds of a phase i are *testing* rounds for processes that are still unmarked. In each *odd* testing round, each unmarked process chooses at random $2^i \beta \log n$ processes, if $i < \log n - \log\log n - \log \beta$, or chooses all n processes, otherwise. The unmarked process sends a query regarding the rumors received to all selected processes. In the *even* testing rounds, each process replies to the queries with the list of rumors received so far. An *unmarked* process changes its state to *marked* if at least $\min(\alpha 2^i \beta \log n, n - t)$ of the processes that were contacted during the previous (even) communication round had received its rumor. Otherwise, the process remains unmarked. A marked process stops disseminating its message, but keeps replying to queries from other processes, and sends packets in the second part of every phase. Once a process is marked, it remains marked for the rest of the dissemination epoch.

The algorithm guarantees that, during the dissemination epoch, the number of unmarked processes is roughly halved in every phase–we say that information "trickles" down to the last uninformed processes.

The Confirmation Epoch. This epoch also consists of $\log n$ phases, each of which consists of $2 \log n$ communication rounds. At the beginning of this epoch, we set all processes to the *unmarked* state. In each odd round of phase i, each process in unmarked state queries $2^i \beta \log n$ processes chosen randomly (if $i \geq \log n - \log\log n - \log \beta$, then the process sends messages to all n processes), about the rumors they received so far. In even rounds, processes reply to the queries by sending the list of rumors they received in the current execution. If the phase number i is less than $\log n - \log\log n - \log \beta$, then the process needs to receive at least $\alpha 2^i \beta \log n$ replies to switch to the marked state. Otherwise, if $i \geq \log n - \log\log n - \log \beta$, then the process needs to receive at least $n - t$ replies in order to switch to the marked state. All processes terminate after the last phase of this epoch by delivering the rumors they received so far.

4.2 Correctness

Note that *termination* is ensured by the structure of the algorithm. We first show that every process delivers every rumor belonging to a correct process, with high probability. Second, we show that the algorithm sends $O(n \log^3 n)$ messages, w.h.p. Due to space restrictions, we defer the complete proofs to the appendix.

In terms of notation, let t be the maximum number of failures in an execution, and let f be the actual number of failures that occur in the current execution (not counting

processes that fail prior to the start of the execution; these latter processes have no effect on the outcome). The model section assumes that $f \leq t < n/3$. To simplify the analysis, in the following, we will consider a run of the algorithm in which $f < n/\kappa$, where κ is a large constant. Note that, because of the properties of the algorithm, we can "boost" its resiliency from $f < n/\kappa$ failures to $f \leq t < n/3$ failures, without affecting its asymptotic complexity, by re-running it for κ times (as during one of these κ executions, by the pigeonhole principle, there are no more than n/κ failures).

Delivery. We first show that, by the end of the first epoch, every rumor belonging to a correct (non-crashed) process has spread to at least $n/2$ processes w.h.p. The proof is based on the analysis of the sampling procedure used for marking processes.

Lemma 1 (Spreading). *For every correct process p, there is a majority of correct processes that have received its rumor during the first epoch, with high probability.*

During the second epoch, with high probability, every rumor belonging to a correct process is delivered to every other correct process through the majority that holds it at the end of the dissemination epoch.

Lemma 2 (Delivery). *By the end of the confirmation epoch, every correct process has received the rumors initiated by other correct processes, with high probability.*

Message Complexity. In this section, we prove that the algorithm sends $O(n \log^3 n)$ in every run, with high probability. The argument is based on the observation that, in every phase of an epoch, the number of processes that remain unmarked is roughly halved. This implies that the number of messages sent in a phase is always bounded by $O(n \log^2 n)$, w.h.p. Finally, since there are $O(\log n)$ phases in an execution, we obtain that the total number of messages sent is $O(n \log^3 n)$ with high probability.

In the following analysis, we consider, without loss of generality, that the adversary makes its choices of which process to fail at each round r, precisely at the beginning of round r, by inspecting the local state and the randomly chosen message receivers for each process that has not yet been crashed (by the beginning of the previous round $r-1$). This simplifies the exposition, without decreasing the power of the adversary.

The first Lemma proves that, during the dissemination epoch, the number of unmarked processes is halved in each phase, with high probability.

Lemma 3 (Dissemination Epoch). *At the beginning of phase number i of the first epoch, the number of unmarked processes is at most $n/2^{i-1}$, with high probability.*

Proof. (Sketch) We proceed by induction on the phase number i. The case $i = 1$ is trivial. For the induction step, we assume that, at the beginning of phase number i, there are at most $n/2^{i-1}$ unmarked processes.

Let U_i be the set of processes that are unmarked at the beginning of phase number i. We analyze the progressive spread of rumors in U_i during the current phase of the algorithm. Recall that, in the first round of the phase, each unmarked process sends its rumor to $2^i \log n$ randomly chosen processes. We say that a rumor from a process in U_i is *fast* in round r_1 if it is held by at least $2^{i-3} \log n$ processes that are not crashed at the second round of the phase. The first step proves that, by failing f_1 processes during the

first round of the phase, the adversary may prevent at most $cf_1/2^i$ processes being *fast*, where c is a constant. We denote the set of *fast* rumors for round r_1 by M_i.

The second step analyzes the following $\log n$ communication rounds, in which every process sends packets containing rumors it received so far. We prove that by failing f_2 processes in the second part of the phase, the adversary may stop at most $220f_2$ packets from spreading to at least $n(\alpha + 1/48)$ processes each, w.h.p. We say that a packet is *fast* in round r of this part, if it is sent by at least $\min(\log^r n/96^r, n(\alpha + 1/24))$ processes that do not crash before round $r + 1$. We notice that the rumors are spread uniformly in each of these rounds. By a counting argument, we obtain that, by crashing f_r processes in a round r, the adversary may stop at most cf_r rumors from being *fast* in round r, where c is a constant. Therefore, for each rumor that is *fast* throughout this part of the phase, we are analyzing an epidemic process with a growth rate of $\log n/96$. On the other hand, the adversary may stop the growth for individual rumors, by failing processes at every round. A careful analysis of this procedure, which can be found in the appendix, shows that, by failing f processes in the second part of the phase, the adversary may stop at most a total of $K \cdot f$ packets from being spread to at least $n(\alpha + 1/48)$ processes each, where K is a constant.

Since each rumor in M_i is distributed into at least $2^{i-3} \log n$ packets at the end of the first round in this phase, the adversary has to stop *all* packets containing the rumor from being *fast* in order to prevent the rumor from being spread to a large fraction of processes. On the other hand, by a Chernoff bound, a packet contains at most $10 \log n$ rumors from M_i, w.h.p. Therefore, by stopping x packets, the adversary may prevent at most $10x \log n/(2^{i-3} \log n) \leq 80f/2^i$ rumors from being spread to at least $n(1/2 + \epsilon/4)$ processes each, w.h.p. Finally, we obtain that, for suitable values of κ, there are at most $n/2^{i+1}$ rumors started by processes in U_i that are not spread to at least $n(\alpha+1/24)$ processes. So at least $|U_i| - n/2^{i+1}$ rumors are spread to at least $n(\alpha+1/24)$ processes.

The third and final step shows that there are at most $n/2^{i+1}$ processes associated to these $|U_i| - n/2^{i+1}$ rumors that are not marked during the testing part of the phase, with high probability. This concludes the proof of the induction step, and of the Lemma.

The final lemma in the proof of the message complexity upper bound proves that the number of non-marked processes is halved in each phase of the confirmation epoch as well. The proof is similar to that of Lemma 3, and can be found in the appendix.

Lemma 4 (Confirmation Epoch). *At the beginning of phase i of the second epoch, the number of unmarked processes is not more than $n/2^{i-1}$, with high probability.*

5 Relating Fault-Tolerance to Message Complexity

In this section, we prove a lower bound on the message complexity of any gossip algorithm that tolerates $t < n(1 - \epsilon)$ process crashes. We consider that ϵ is a function of n, with $1 > \epsilon(n) \geq 0$. For simplicity, in the rest of this section, we omit the argument of the $\epsilon(n)$ function, and simply write ϵ.

Theorem 1 considers the case in which every process starts out with a rumor, which has to be delivered by every correct process as long as its source is correct. The argument can be adapted to yield a similar result in the case of the *broadcast* problem, in which a rumor that is delivered by a correct process has to be delivered by every correct process. We present the full proof in the appendix.

Theorem 1 (Message Complexity Lower Bound). *Any algorithm that tolerates $t < n(1-\epsilon)$ process crashes, and guarantees gossip with constant positive probability, needs to send $\Omega(n \log(1/\epsilon))$ messages, with high probability.*

Proof. (Sketch) We consider an algorithm $\mathcal{A}$ that solves gossip with constant positive probability, tolerating $n(1-\epsilon)$ failures, and show that there exists an adversarial strategy which forces the algorithm to generate $\Omega(n \log(1/\epsilon))$ messages with high probability. The adversarial strategy is to crash any process that receives a new rumor, and to crash every process immediately after it generates $n/4$ messages. We split the execution into phases, with the property that, in each phase, the adversary crashes exactly half of the processes that were not crashed at the end of the previous phase. Our main claim is that, with high probability, by sending $n/64$ new messages, the algorithm makes the whole system progress for at most one phase. The main difficulty in proving the claim is that the random variables corresponding to messages sent are not independent; however, we are able to show that they are *negatively associated*, in the sense of [31], which allows the use of tail bounds. Finally, we show that, in order to achieve gossip with constant probability, the algorithm needs to make the system progress for $\Omega(\log(1/\epsilon))$ phases with high probability, which implies the lower bound on message complexity.

6 Gossip and Oblivious Failures

In this section, we present the CoordinatedGossip protocol, that disseminates every rumor initiated by a correct process, using only $O(n)$ messages, with high probability. The key idea behind the protocol is to elect a small set of $\Theta(\log n)$ *coordinators*, who then collect and disseminate the rumors. There are two main challenges:
1. Intra-coordinator communication: Since the coordinators are selected independently, they do not initially know how to contact the other coordinators. Therefore the coordinators select a set of $O(\sqrt{n} \log^2 n)$ *intermediaries* that form an overlay connecting all the coordinators.
2. Coordinator discovery: In order to collect and disseminate the rumors, the coordinators need to efficiently contact all the processes in the system. Each coordinator sending a message to each process would be too expensive, requiring $\Theta(n \log n)$ messages. Instead, each coordinator selects $\Theta(n/\log n)$ *relays*, leading to $\Theta(n)$ relays in total. To collect the rumors, each process forwards its rumor to a relay, which can then forward it to a coordinator. To disseminate rumors, the process is reversed.
We first present the protocol in Section 6.1. We then analyze the message complexity in Section 6.2. All omitted proofs can be found in the appendix.

6.1 CoordinatedGossip

We split the presentation of the CoordinatedGossip protocol in three parts. First, we describe a set of initial steps in which we select three special sets of processes: *coordinators*, *intermediaries*, and *relays*. We describe how the rumors are collected, and then how rumors are disseminated.

Selecting processes. In the first round, each process decides whether to be a coordinator, an intermediary, a relay, or none of the above. (Note that a process can play more than one role.)

***Coordinators**:* Each process decides to be a coordinator with probability $\Theta(\log n/n)$, specifically, $12c \log n/n$.
***Intermediaries**:* Each coordinator chooses $\Theta(\sqrt{n} \log n)$ processes at random, specifically, $6c\sqrt{n} \log n$ processes, and sends each an *intermediary-election* message; each process that receives such a message decides to be an intermediary. We say that a process is a *correct intermediary* if it receives an intermediary-election message and does not fail. For a given intermediary, we define the intermediary's *neighbors* to be the set of processes from which it received intermediary-election messages.
***Relays**:* Each coordinator participates in c relay elections. That is, for every $\ell \in \{1, \ldots, c\}$, each coordinator chooses $\Theta(n/\log n)$ processes at random, more precisely $n/(24c \log n)$ processes, and sends each a *relay-election-ℓ* message.

If, for any $\ell \in \{1, \ldots, c\}$, a process receives *exactly one* relay-election-ℓ message, then it decides to be an ℓ-relay. We define the *ℓ-parent* of a relay to be the coordinator that sent it a unique relay-election-ℓ message. We say that a process is a *good relay* for ℓ if: (i) it receives exactly one relay-election-ℓ message, for some ℓ; (ii) it is correct; and (iii) its ℓ-parent is correct.
We will argue that there are $\Theta(\log n)$ correct coordinators, that every pair of coordinators shares a correct intermediary, and that there are $\Theta(n)$ good relays.

Collecting rumors. After choosing coordinators, intermediaries, and relays, there is an collection phase. The goal of the collection phase is to ensure that every rumor from a correct process is known to every correct coordinator. Each process attempts to send its rumor to a relay; the relay forwards it to its parent, a coordinator; the coordinators then exchange rumors via the intermediaries. We proceed to describe this process in more detail. Specifically, the collection phase consists of $\Theta(\log n)$ iterations, specifically, $45(c+1) \log n$ iterations, of the following seven rounds:

a. Each process that has not *succeeded* in a prior iteration sends its rumor to a random process.
b. Each relay sends any messages received in Round (a) to its parents (for each $\ell \in \{1, \ldots, c\}$).
c. Each coordinator forwards all the rumors it has received to the set of intermediaries to which it previously sent intermediary-election messages.
d. Each intermediary forwards every rumor received to its neighbors.
e. Each coordinator sends a response to every relay from which it received a message in Round (b).
f. Each relay that received a response in Round (e) sends a response to every process from which it receives a message in step (a).
g. Each process that receives a response in Round (f) has *succeeded* and remains silent thereafter.

We will argue that the collection phase uses $O(n)$ messages, with high probability, and that, by the end, every rumor from a non-failed process has been received by every non-failed coordinator, with high probability.

Disseminating rumors. After the collection phase, there is a dissemination phase. In the first round of the dissemination phase, each coordinator sends all the rumors it has learned to the set of relays that it previously sent relay-election messages to. The

remainder of the dissemination phase proceeds much like the collection phase, except in reverse. As before, the dissemination phases consists of $\Theta(\log n)$ iterations, specifically $45(c+1)\log n$ iterations, of the following rounds:

a. Each process that has not *succeeded* in a prior iteration sends its message to a random process.
b. Each relay sends a response to every message received in Round (a); the response includes all the rumors previously received from the coordinators.
c. Each process that receives a response in Round (b) has *succeeded* and remains silent thereafter.

We will argue that the dissemination phase uses $O(n)$ messages, with high probability, and that every non-failed process learns every rumor that was previously collected by the coordinators.

6.2 Analysis

We begin by bounding the number of coordinators, which follows from a straightforward calculation:

Lemma 5 (Coordinator-Set Size). *With probability at least $(1-1/n^c)$: (a) There are at most $24c\log n$ coordinators; and (b) There are at least $c\log n$ correct coordinators.*

Next, we argue that for every pair of coordinators, there is some correct intermediary that has both coordinators as neighbors.

Lemma 6 (Good Intermediaries). *With probability at least $(1-1/n^c)$: for every pair of non-failed coordinators p_i, p_j, there exists some intermediary p_k such that both p_i and p_j are neighbors of p_k.*

Next, we argue that a constant fraction of the processes are good relays. This is perhaps the key fact that leads to good performance: the coordinators can efficiently contact a constant fraction of the world (i.e., the relays); and there are enough relays that the processes can easily find a relay, and hence a path to a coordinator. The lemma follows from a balls-and-bins style analysis.

Lemma 7 (Good Relays). *There exists some $\ell \in \{1,\ldots,c\}$ such that at least $n/18$ processes are good for ℓ, with probability at least $(1-2/n^c)$.*

Next, we argue that every process succeeds during the collection and dissemination phases, resulting in every correct process learning the entire set of rumors that were initiated at correct processes.

Lemma 8 (Gossip Success). *With probability at least $(1-4/n^c)$: (a) By the end of the collection phase, every process has succeeded. (b) By the end of the dissemination phase, every process has succeeded. (c) When the protocol terminates, every rumor from a correct process has been received by every correct process.*

Finally, we analyze the message complexity of the collection and dissemination phases. The key part of this analysis is showing that the process of finding relays is *efficient*. This fact follows from the following analogy: the attempts by processes to find relays is equivalent to flipping a (biased) coin until we get n heads; this requires only $O(n)$ flips, with high probability.

Lemma 9 (Message Complexity). *With probability at least* $(1 - 5/n^c)$*, the collection and dissemination phases have message complexity* $O(n)$*.*

We conclude with the main theorem:

Theorem 2. *The* CoordinatedGossip *protocol is correct and has message complexity* $O(n)$*, w.h.p.*

Proof. By Lemma 8, we see that every rumor is successful disseminated with probability at least $(1-4/n^c)$. In terms of message complexity, when Lemma 5 holds, i.e., with probability $(1 - 1/n^c)$: the selection of coordinators requires at most $O(\sqrt{n}\log^2 n)$ messages; the selection of relays requires at most $O(n)$ messages. Lemma 9 states that coordination and dissemination have message complexity $O(n)$ with probability $(1 - 5/n^c)$. Thus, overall, the message complexity is $O(n)$, with high probability.

References

1. Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in o(log n) rounds. In *STOC*, 2006.
2. Petra Berenbrink, Robert Elsaesser, and Tom Friedetzky. Efficient randomised broadcasting in random regular networks with applications in peer-to-peer systems. In *PODC*, 2008.
3. K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. on Computer Systems*, 17:41–86, 1999.
4. Augustin Chaintreau, Jean-Yves Le Boudec, and Nikodin Ristanovic. The age of gossip: spatial mean field regime. In *SIGMETRICS*, 2009.
5. Bogdan S. Chlebus and Dariusz R. Kowalski. Robust gossiping with an application to consensus. *J. Comput. Syst. Sci.*, 72(8):1262–1281, 2006.
6. A. Czumaj, L. Gasieniec, and A. Pelc. Time and cost trade-offs in gossiping. *SIAM Journal on Discrete Mathematics 11 (1998), 400-413.*
7. Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC*, 1987.
8. B. Doerr, A. Huber, and A. Levavi. Strong robustness of randomized rumor spreading protocols. *Proceedings of ISAAC'09.*
9. Benjamin Doerr, Tobias Friedrich, and Thomas Sauerwald. Quasirandom rumor spreading. In *SODA*, 2008.
10. Benjamin Doerr, Tobias Friedrich, and Thomas Sauerwald. Quasirandom rumor spreading: Expanders, push vs. pull, and robustness. In *ICALP (1)*, pages 366–377, 2009.
11. R. Elsässer and T. Sauerwald. On the runtime and robustness of randomized broadcasting. *Theor. Comput. Sci.*, 410(36):3414–3427, 2009.
12. Robert Elsässer. On the communication complexity of randomized broadcasting in random-like graphs. In *SPAA*, 2006.
13. Robert Elsässer and Thomas Sauerwald. The power of memory in randomized broadcasting. In *SODA*, 2008.
14. P. Eugster, R. Guerraoui, S. Handurukande, A-M Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4), 2003.
15. Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Randomized broadcast in networks. In *SIGAL '90*, pages 128–137, London, UK, 1990. Springer-Verlag.
16. Alan M. Frieze and G.R. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, (10):55–77, 1985.

17. Chryssis Georgiou, Seth Gilbert, Rachid Guerraoui, and Dariusz R. Kowalski. On the complexity of asynchronous gossip. In *PODC*, 2008.
18. Seth Gilbert and Dariusz Kowalski. Distributed agreement with optimal communication complexity. *SODA 2010*.
19. J. Hromkovic, R. Klasing, A. Pelc, P. Ruzika, and W. Unger. *Dissemination of information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance*. Springer-Verlag, 2005.
20. Bruce M. Kapron, David Kempe, Valerie King, Jared Saia, and Vishal Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. In *SODA*, pages 1038–1047, 2008.
21. R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. In *FOCS*, 2000.
22. D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms, 2002.
23. D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. *Journal of ACM*, pages 943–967, 2004.
24. A. Kermarrec, L. Massoulie, and A. Ganesh. Probabilistic reliable multicast in ad hoc networks. *IEEE Trans. on Parallel and Distr. Syst.*, 14, 2003.
25. Valerie King and Jared Saia. Fast, scalable byzantine agreement in the full information model with a nonadaptive adversary. In *DISC*, 2009.
26. Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *SODA*, pages 990–999, 2006.
27. Rivka Ladin, Barbara Liskov, Liuba Shrira, and Sanjay Ghemawat. Providing high availability using lazy replication. *ACM Trans. Comput. Syst.*, 10(4):360–391, 1992.
28. J. Luo, P. Eugster, and J.P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *INFOCOM*, 2003.
29. Dahlia Malkhi, Yishay Mansour, and Michael K. Reiter. On diffusing updates in a byzantine environment. In *SRDS*, pages 134–143. IEEE Computer Society, 1999.
30. Boris Pittel. On spreading a rumor. *SIAM J. Appl. Math.*, 47(1):213–223, 1987.
31. Devdatt Dubhashi Desh Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13:99–124, 1996.
32. R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Int-l Conference on Distributed Systems Platforms and Open Distributed Processing*, 1998.

Appendix

A Proofs omitted from Section 4

Lemma 10. *For every correct process p, there is a majority of correct processes that have received its rumor during the first epoch, with high probability.*

Proof. Let p be a correct process. It is easy to see that p is marked at the end of the dissemination epoch. This implies that process p queried $\min(2^i \cdot \beta \cdot \log n, n)$ processes during some round r in the dissemination phase, and at least a fraction of α of these had p's rumor at the end of round $r-1$. If process p is in a phase where it queries n processes in each round, the claim becomes trivial.

Otherwise, let x be the number of processes that had p's rumor at the end of round $r-1$. The expected number of processes that tell process p that they had its message before round r is $2^i\beta \log n \cdot x/n$. Since p did receive $\alpha \cdot 2^i \beta \log n$ positive responses, we can say that $2^i\beta \log n \cdot x/n$ is not less than $\frac{\alpha - 1/24}{\alpha} \cdot 2^i \alpha\beta \log n$, with high probability. This implies that $x \geq n(1/2 + 1/24)$ with high probability. This means that the number of processes that were not crashed and had process p's rumor at the end of round $r-1$ is at least $n(1/2 + 1/24)$. Since the number of failures in the current run of the algorithm is at most n/κ, it follows that, for sufficiently large κ, there are at least $n(1/2 + 1/48)$ *correct* processes that received p's rumor with high probability, which form a majority. $\square_{Lemma\ 1}$

Lemma 11. *By the end of the confirmation epoch, every correct process has received the rumors initiated by other correct processes, with high probability.*

Proof. Let p be an arbitrary correct process. It is easy to see that every such process gets marked by the end of the last phase of the confirmation epoch.

We want to prove that this process receives all rumors initiated by correct processes. Let i be the phase of the confirmation epoch in which we marked process p. If p queried all n processes in phase i, then the claim follows. Otherwise, process p queried $2^i\beta \log n$ random processes, and received responses from at least $2^i\alpha\beta \log(n)$ of these processes.

Lemma 1 ensures that, for every correct process q, there are at least $n(1 + 1/24)/2$ correct processes having its rumor by the end of the first epoch. On the other hand, the expected number of correct processes that have q's rumor and were queried by p in this round is at least $(1+1/24) \cdot 2^{i-1}\beta \log n$. By the Chernoff bound, with high probability, there are at least $\frac{1+1/48}{2} \cdot 2^{i-1}\beta \log n$ such processes.

On the other hand, p has received query responses from at least $\alpha 2^i \beta \log n$ processes. Since $\alpha > 1/2$, it follows that, with high probability, p receives a message from a correct process that had q's rumor. $\square_{Lemma\ 2}$

Lemma 12. *At the beginning of phase number i of the first epoch, the number of unmarked processes is at most $n/2^{i-1}$, with high probability.*

Proof. We proceed by induction on the phase number i. The case $i = 1$ is trivial. For the induction step, we assume that, at the beginning of phase number i, there are at most $n/2^{i-1}$ unmarked processes, and we prove that the number of unmarked processes at the end of the phase is at most $n/2^i$.

Let U_i be the set of processes that are unmarked at the beginning of phase number i. In round r_1, the first round of phase i, each unmarked process sends exactly $2^i \log n$ messages to other processes. This implies that each process receives on average $|U_i|/n \cdot 2^i \log n \leq 2 \log n$

messages in this round. Therefore, by the Chernoff bound, each process receives between $\log n$ and $10 \log n$ messages in this round, with high probability. The first step in the proof is to bound the number of rumors of unmarked processes that the adversary may stop from spreading at a large number of processes during round r_1.

Note that, since $t < n/3$, each process in U_i sends messages to at least $2^{i-2} \log n$ processes that were not crashed at the end of round r_1 - 1, with high probability. Given this observation, we say that a rumor initiated by a process p in U_i is *fast-spreading* in round r if at most half of the processes that 1) were not crashed at the end of round $r-1$ and 2) received the rumor in round r, are crashed by the adversary before they complete sending for round $r+1$. In particular, a rumor is fast-spreading in round r_1, with high probability, if the adversary crashed at most $2^{i-3} \log n$ of the processes that received the rumor in round r_1. Note that, if rumor m is fast-spreading in round r_1, then at least $2^{i-3} \log n$ processes successfully send the rumor in round $r_1 + 1$.

Next, we bound the number of rumors that the adversary may prevent from being fast spreading in round r_1, depending on the number of failures that the adversary is willing to expend in this round.

Since a process may receive at most $10 \log n$ messages in round r_1, with high probability, and the adversary has to fail at least $2^{i-3} \log n$ processes in this round to prevent a rumor from spreading fast in this round, if follows that by failing f processes during round r_1, the adversary may prevent at most $20f/2^{i-3}$ rumors from spreading fast in this round, whp. This implies that at least $|U_i| - 20f/2^{i-3} \geq |U_i| - \frac{20n}{2^{i-3}\kappa}$ rumors are fast spreading in round r_1. Let M_i denote the set of these fast spreading rumors.

Next, we analyze the following $\log n$ communication rounds, in which every process sends packets containing rumors it heard so far. The previous claim shows that each of the rumors in M_i is in at least $2^{i-3} \log n$ packets at the beginning of the second part of the phase. On the other hand, a packet contains at most $10 \log n$ rumors, whp. Next, we prove that by failing f processes in the second part of the phase, the adversary may stop at most $220f$ packets from spreading to at least $n(\alpha + 1/48)$ processes each, whp.

Recall that the second part of the phase contains $\log n$ rounds, in which each process sends its packet to $\log n$ randomly chosen neighbors. Let $p_1, p_2, \ldots, p_\ell$ be the processes that participate in the second part of the phase, and let $c_1, c_2, \ldots, c_\ell$ be their respective packets. Also, for each round $r \in \{1, 2, \ldots, \log n\}$ in the second part of this phase, let S_j^r be the set of processes that had packet c_j at the end of round r, and that successfully send the packet in round r, i.e. are not crashed during round r. For completeness, let $S_j^0 = \{p_j\}$. We say that a packet c_j is *fast* if for every round r in this phase, $|S_j^r| \geq \min(\log^r n/96^r, n(\alpha + 1/24))$. Otherwise, we say that the packet is *slow*.

We prove that there are at least $n - f_0 - 220 \sum_{k=1}^{\log n} f_k$ fast packets in this part of the phase, where f_k is the number of processes that are crashed by the adversary in round k of the phase. The statement is trivial for the first round. For the induction step, we assume that there are at least $n - f_0 - 220 \sum_{k=1}^{r} f_k$ fast packets up to round R, and f_0 is the number of processes that take no steps in this phase. If $\log^R n/96^R > n(\alpha + 1/24)$, then the claim follows. Otherwise, for each fast packet c_j, pick a set B_j such that $B_j \subseteq S_i^R$, and $|B_j| = \log^r n/96^r$. For simplicity, let $b = |B_j|$. We now analyze the spread of messages from processes in B_j in round r. We define H_j as the set of processes which received at least one message from a process in B_j in round r. Note that some of these processes may be crashed by the adversary at the beginning of round $r+1$.

There are $b \log n$ messages sent from each set B_j. For this part of the analysis, we assume without loss of generality that these messages are sent independently. We notice that, by the end of this round, $|H_j| \geq \min(n(\alpha + 1/24), b \log n/50)$. If $|H_j| < n(\alpha + 1/24)$, then there are at least $2n/3 - |H_j| \geq n/24$ correct processes that are not in B_j. Therefore, the size of B_j either surpasses $n(\alpha + 1/24)$, or it is increased to $b \log n/50$, with high probability.

We now analyze the number of rumors that remain fast in round r, given that the adversary fails f_r processes in this round. Note first that, if $|H_j| \geq n(\alpha + 1/24)$, then, since the adversary may fail a total of n/κ processes in the current run of the protocol, we obtain that there are at least $|H_j| \geq n(\alpha + 1/48)$ processes that have packet j and do not crash in this run, for large values of κ, thereby concluding our claim for packet j. Otherwise, $|H_j| \geq b \log n/50$ whp, therefore the adversary has to crash at least $b \log n/110$ processes in H_j at the beginning of round $r + 1$ in order to stop rumor i from being *fast*. On the other hand, every process receives on average $b \log n$ messages from processes in the sets B_j in this round. Therefore, by the Chernoff bound, every process receives at most $2b \log n$ messages from processes in the sets B_j in this round. It follows that, by failing x processes in round r, the adversary may stop at most $\frac{220bx \log n}{b \log n} = 220x$ rumors from being fast in this round.

Therefore, by failing f processes in the course of this part of the phase, the adversary may stop at most a total of $220f$ packets from being spread to at least $n(\alpha + 1/48)$ processes each. Since each rumor in M_i is distributed into at least $2^{i-3} \log n$ packets, the adversary has to stop *all* packets containing the rumor in order to prevent a rumor from being spread. On the other hand, a packet contains at most $10 \log n$ rumors from M_i. Therefore, by failing x packets, the adversary may prevent at most $10x \log n/(2^{i-3} \log n) \leq 80f/2^i$ rumors from being spread to at least $n(1/2 + \epsilon/4)$ processes each, with high probability. By failing $f < n/\kappa$ processes, the adversary may stop at most $220f$ packets from being spread. Finally, we obtain that, for suitable values of κ, there are at most $n/2^{i+1}$ rumors started by processes in U_i that are not spread to least $n(\alpha+1/24)$ processes. So at least $|U_i| - n/2^{i+1}$ rumors are spread to at least $n(\alpha+1/24)$ processes.

In the final step, we prove that among the rumors that are spread to at least $n(\alpha + 1/24)$ processes in the previous part, there are at most $n/2^{i+1}$ processes associated with these rumors that remain unmarked in the testing phases. Assume a process p among these processes. If it sends n messages, it will receive at least $n - t$ replies which makes it unmarked. Otherwise p sends $2^i \beta \log n$ messages. Since there are at most $n/2^{i-1}$ unmarked processes. Everybody receives at most $2\beta \log n$ messages in expectation. Using the Chernoff bound, this number is not more than $3\beta \log n$.

On the other hand, if p receives replies from at least a fraction of α of these processes, it will be marked. We expect at most $1/3$ of its recipients to be already failed. With high probability the number of already failed recipients of p is not more than a fraction of $1/3+\epsilon$. So the adversary can keep a process unmarked only by failing at least $(1-1/3-\epsilon\alpha) \times 2^i \beta \log n > 2^i \beta \log n/15$ of p's recipients for small enough ϵ. We also know that each process receives at most $3\beta \log n$ messages in each round. So the adversary can keep at most $x \cdot (3\beta \log n)/(2^i \beta \log n/15) = 45x/2^i$ processes unmarked by failing x processes in the testing phase. Again by choosing a suitable value for κ, the number of these rumors that remain unmarked is at most $n/2^{i+1}$. So there are at most $n/2^{i+1} + n/2^{i+1}$ rumors that are still unmarked. So at least $n/2^{i-1} - 2n/2^{i+1} = n/2^i$ are marked.

$\square_{Lemma\ 3}$

Lemma 13. *At the beginning of phase i of the second epoch, the number of unmarked processes is not more than $n/2^{i-1}$, with high probability.*

Proof. The induction base ($i = 1$) is trivial. Recall that the confirmation epoch is composed of $\log n$ phases, each of which has $2 \log n$ rounds. In each odd round, every unmarked processes sends $2^i \beta \log n$ queries, and it waits for the replies in the next round. If it receives at least $\alpha 2^i \beta \log n$ replies, the process moves to the "marked" state.

Let r be the current round. Let r be odd, and let Z_p be the set of processes that p sends a request message to in round r. So $|Z_p| = 2^i \beta \log n$. Some processes in Z_p are already failed.

Some of them might be failed by the adversary after they receive the message, i.e. before they complete sending their messages for round $r+1$. Note that $t < n/3$, so the expected number of processes in Z_p that are already failed is at most $(t/n) \times (2^i \beta \log n) \leq 2^i \beta \log n/3$. So, by the Chernoff bound, it is not more than $2^i \beta \log(n) \cdot (1/3 + 1/24)$, with high probability.

We also know that process p remains unmarked only if at least $2^i \log n(1-\alpha)$ processes in Z_p are either already failed or will be failed in this or next round. So the adversary has to crash at least $2^i \beta \log n(1-\alpha-1/3-1/24) \geq 2^i \beta \log n/8$ processes in Z_p in order to keep process p unmarked.

On the other hand, we know that each unmarked process sends $2^i \beta \log n$ messages, and there are at most $n/2^{i-1}$ unmarked processes in this phase. Therefore, the total number of messages sent in round r is at most $2\beta n \log n$. So the expected number of messages that a process receives is $2\beta \log n$, and no process receives more than $3\beta \log n$ messages whp.

Assume that the adversary fails f_r processes in this round. Each of these processes received at most $3\beta \log n$ messages in this round. So the total number of messages "lost" to failures is at most $3\beta \log n$. In order to keep process p unmarked in this round, the adversary has to fail at least $2^i \beta \log n/8$ processes in round r, whp. So the total number of processes that remain unmarked until the end of the execution is at most $24\beta f_r \log n/2^i \beta \log n = 24 f_r/2^i$. Note, however, that we only need to mark process p in one of the $\log r$ rounds of this phase, and, once a process is marked, it remains marked until the end of the execution. Since the number of failures in this run of the algorithm is at most n/κ, there exists a round R in phase i in which there are at most $n/(\kappa \log n)$ failures. The number of processes that remain unmarked after round R is at most $24 f_R/2^i \leq 24n/(2^i \kappa \log n)$. This is obviously less than $n/2^i$, which finishes the proof of the induction step and the proof of the lemma.

$\square_{Lemma\ 4}$

B Proof of the Lower Bound (Section 5)

Theorem 1. *Any algorithm that tolerates $t < n(1-\epsilon)$ process crashes, and guarantees gossip with constant positive probability, needs to send $\Omega(n \log(1/\epsilon))$ messages, with high probability.*

Proof. Let $\mathcal{A}$ be an algorithm that solves gossip with constant positive probability. In the following, we define an adversarial strategy which forces the algorithm to send $\Omega(n \log(1/\epsilon))$ messages in every execution, with high probability.

The adversarial strategy is the following. First, every time a message is sent in round $r \geq 1$, its receiver is failed as soon as it receives all its messages for round r. Second, if a process sends $n/4$ messages in total from the beginning of the execution, then the process is crashed. Let A be the set of non-failed processes at some stage in the execution, and let $x = |A|$. We split an execution into phases, with the property that, in phase $i \geq 1$, $n/2^{i-1} \geq x > n/2^i$. It is easy to see that this adversarial strategy prevents any communication between processes, as long as the adversary does not run out of failures. The strategy stops when $x = n/\epsilon$, i.e., after approximately $\log(1/\epsilon)$ phases.

We claim that, if less than $n/64$ messages are sent by the algorithm since the beginning of phase i, then, with high probability, the system does not progress for more than one phase. To prove the claim, fix phase i, and a process p that has not been failed at the beginning of phase i. The number of processes that p has not communicated with so far is at least $3n/4$ (otherwise p would be failed). Also, note that process p cannot find out about failures in the system from other processes, since there is no consistent addressing for processes.

Let $m_1, m_2, \ldots, m_{n/64}$ be the first $n/64$ messages sent during this phase, and let X_i be an indicator random variable that is 1 if message m_i was sent to a process that was not crashed at the beginning of the phase, and 0 otherwise. The probability that a message is sent to a process

that has not been failed is at most $x/(3n/4) \leq 1/2^{i-2}$. Then $E[\sum_i X_i]$, the expected number of non-failed processes that are hit by messages during phase i (and hence crashed by the adversary) is at most $(n/64) \cdot 1/2^{i-2} = n/2^{i+4}$.

Next, we notice that the dependence relation between the random variables X_i is negative. It is easy to see that $Pr(X_i = 1|X_j = 1) \leq Pr(X_i = 1)$, since there are less non-crashed processes to hit if one is already crashed. This implies that $E[X_i|X_j] \leq E[X_i]$ for all pairs i, j in $\{1, 2, \ldots, n/64\}$, which in turn implies that $E[X_i X_j] \leq E[X_i] \cdot E[X_j]$ for all pairs i, j. It follows that the random variables X_i are *negatively associated*, in the sense of [31]. Proposition 5 from [31] implies that, in this case, the Chernoff bound applies to the sum of the random variables X_i.

Therefore, with high probability, the number of messages that "hit" the set of processes that have not been crashed at the beginning of phase number i is less than $n/2^{i+2}$. The number of processes crashed during this phase because they have sent at least $n/4$ messages throughout the execution is $o(\log n)$, since otherwise the algorithm $\mathcal{A}$ sends $\Omega(n \log n)$ messages in this execution. These two facts imply that, with high probability, at most $n/2^{i+1}$ processes are crashed as long as the algorithm sends $n/64$ messages, which implies that, with high probability, the system will stay in phase i or move to phase $i + 1$.

The claim above means that by sending $n/64$ messages, the algorithm cannot make the system progress for more than one phase, with high probability. This implies that, after the algorithm sends $n/64 \cdot \log(1/\epsilon)$ messages, there are still, with high probability, n/ϵ processes that have not been failed, and did not receive any rumor except their own.

We analyze the case where the algorithm $\mathcal{A}$ ensures termination with constant positive probability, and delivery with constant positive probability. This means that, given a fixed adversarial failure pattern, in all but a constant fraction of the executions associated with that pattern, every correct process terminates, and every terminating process delivers the rumors of all correct processes. However, we presented an adversarial failure pattern which guarantees with high probability that at least $n/\epsilon > 2$ processes do not receive any rumor except their own, as long as at most $n \log(1/\epsilon)/128$ messages are sent. Let $\mathcal{E}$ be the set of executions associated with this failure pattern. The properties of algorithm $\mathcal{A}$ ensure that in all but a constant fraction of the executions in $\mathcal{E}$, every correct process terminates, and every terminating process delivers the rumors of all correct processes. This implies that, with high probability, the algorithm has to send $\Omega(n \log(1/\epsilon))$ messages in each one of these executions, which means that the algorithm $\mathcal{A}$ has to send $\Omega(n \log(1/\epsilon))$ messages with high probability.

$\square_{Theorem\ 1}$

C Proofs omitted from Section 6.2

Lemma 14. *With probability at least $(1 - 1/n^c)$: (a) There are at most $24c \log n$ coordinators; and (b) There are at least $c \log n$ correct coordinators.*

Proof. Let X_i be a $0/1$ variable indicating whether a process elects itself a coordinator, and recall that at least $n/3$ processes are correct. Since the adversary is oblivious, X_i is independent of whether process p_i is correct.

Since $\Pr X_i = 12c \log n/n$, we conclude that the expected number of coordinators is $12c \log n$, and the expected number of correct coordinators is at least $4c \log n$. The result follows by a straightforward Chernoff bound. $\square_{Lemma\ 5}$

Lemma 15. *With probability at least $(1 - 1/n^c)$: for every pair of non-failed coordinators p_i, p_j, there exists some intermediary p_k such that both p_i and p_j are neighbors of p_k.*

Proof. Fix some non-failed coordinators p_i and p_j. Both p_i and p_j send $\Theta(\sqrt{n}\log n)$ intermediary-election messages. Let I be the set of intermediaries chosen by p_i. The probability that p_j does not choose a *correct* intermediary in the set I is at most:

$$\begin{aligned}\left(1-\frac{|I|}{3n}\right)^{6c\sqrt{n}\log n} &\leq \left(1-\frac{2c\sqrt{n}\log n}{n}\right)^{6c\sqrt{n}\log n}\\ &\leq \left(\frac{1}{2}\right)^{12c^2\log^2 n}\\ &\leq \left(\frac{1}{n}\right)^{c+2}\end{aligned}$$

This implies that p_i and p_j have a shared intermediary with with probability at least $1-1/n^{c+2}$. Taking a union bound over all $\binom{n}{2}$ pairs of p_i and p_j, the probability that there exists some non-failed p_i and p_j without a shared intermediary is no greater than $1/n^c$. $\square_{Lemma\ 6}$

Lemma 16. *There exists some $\ell \in \{1,\ldots,c\}$ such that at least $n/18$ processes are good for ℓ, with probability at least $(1-2/n^c)$.*

Proof. We first argue that for each $\ell \in \{1,\ldots,c\}$, there is a probability at least $1/n$ that at least $n/24$ processes are good relays for ℓ. Recall that Lemma 5 states that with probability $(1-1/n^c)$, there are at least $c\log n$ correct coordinators, and at most $24c\log n$ coordinators in total. We assume for the remainder of the proof that this condition holds.

Fix some $\ell \in \{1,\ldots,c\}$. We first calculate the probability that a correct process is a good relay for ℓ. That is, we calculate the probability that a correct process is sent exactly one relay-election-ℓ message by a good coordinator, and no relay-election-ℓ messages by the at most $24c\log n - c\log n$ faulty coordinators. (Recall that each coordinator sends $n/(24c\log n)$ relay-election-ℓ messages.) That is, assuming Lemma 5 holds, the probability of being a good relay for some ℓ is at least:

$$\left(1-\frac{1}{n}\right)^{n/24-1}\left(1-\frac{1}{n}\right)^{23n/24} \geq \frac{1}{e}$$

Since there are $n/3$ correct processes, we conclude that, in expectation, there are $n/3e$ correct processes that are good relays for ℓ. By a straightforward Chernoff bound, there are at least $n/18$ processes that are good for ℓ, with probability $(1-1/n)$.

Thus, the probability that Lemma 5 holds and that for some ℓ there are at least $n/18$ relays that are good for ℓ is at least $(1-2/n^c)$. $\square_{Lemma\ 7}$

Lemma 17. *With probability at least $(1-4/n^c)$: (a) By the end of the collection phase, every process has succeeded. (b) By the end of the dissemination phase, every process has succeeded. (c) When the protocol terminates, every rumor from a correct process has been received by every correct process.*

Proof. Lemma 7 states that with probability at least $(1-2/n^c)$, there is some ℓ for which there are at least $n/18$ relays that are good for ℓ. Lemma 6 states that with probability at least $(1-1/n^c)$, every pair of non-failed coordinators shares an intermediary. Assume these events occur (with probability $(1-3/n^c)$), and fix ℓ for the remainder of this proof.

During the dissemination phase, it is easy to see that a process is successful as soon as it chooses a relay that is good for ℓ. The same is true of the collection phase: every message received

by a relay that is good for ℓ is forwarded to a correct coordinator, and via a correct intermediary, to every other non-failed coordinator, leading to a response. Thus, during both the collection and dissemination phases, when a process chooses a relay that is good for ℓ in some iteration, then it succeeds by the end of that iteration.

In each iteration, each process that has not succeeded has probability $1/18$ of choosing a relay that is good for ℓ. Over $45(c+1)\log n$ rounds, a process chooses a relay that is good for ℓ with probability at least $(1 - 1/n^{c+1})$. Thus, taking a union bound over n processes, and when combined with the probabilities from Lemmas 7 and 6, we conclude that every process succeeds with probability at least $(1 - 4/n^c)$.

For the final point, notice that since every non-failed process succeeds during the collection phase, we can conclude that every non-failed coordinator has received the rumor of every correct process. Thus in the first step of the dissemination phase, every non-failed rumor is sent to every relay, in particular, to every relay that is good for ℓ. From this it is easy to see that since every correct process succeeds during the dissemination phase, every correct process receives every "correct" rumor. $\square_{Lemma\ 8}$

Lemma 18. *With probability at least $(1 - 5/n^c)$, the collection and dissemination phases have message complexity $O(n)$.*

Proof. Lemma 5 states that with probability at least $(1-1/n^c)$ there are at least $c\log n$ coordinators and at most $12c\log n$ coordinators. Lemma 7 states that with probability at least $(1-2/n^c)$, there is some ℓ for which there are at least $n/18$ relays that are good for ℓ. Lemma 6 states that with probability at least $(1-1/n^c)$, every pair of non-failed coordinators shares an intermediary. Assume these events occur (with probability $(1-4/n^c)$), and fix ℓ for the remainder of this proof.

We first consider the message complexity of the collection phase. We divide the message complexity into two parts: the messages sent between coordinators and intermediaries, and the remaining messages.

Since there are $\Theta(\log n)$ coordinators (by Lemma 5), each of which selects $\Theta(\sqrt{n}\log n)$ intermediaries, there are at most $O(\sqrt{n}\log^2 n)$ intermediaries; we conclude that in each iteration of the collection phase, coordinators and intermediaries send $O(\sqrt{n}\log^2 n)$ messages.

We also observe that each process that has not yet succeeded induces at most $O(c)$ messages in each iteration: the message it sends to the relay, the (at most) c messages its relay sends to its parents, the (at most) c responses it receives, and the response the relay sends to the process. It thus remains to bound the number of times an unsuccessful process participates in an iteration.

Specifically, a process succeeds in an iteration as soon as it finds a relay that is good for ℓ. Each such attempt is independent. Thus the process is analogous to flipping a biased coin, with probability $1/18$ of getting a heads (as per Lemma 7), repeatedly until heads has appeared n times. In expectation, this requires $18n$ attempts. Thus, the probability that it takes more than $36n + c\log n \leq 45(c+1)\log n$ attempts is at most $1/n^c$. The same argument holds for dissemination.

That is, when including the probability that earlier lemmas do not hold, with probability at least $(1 - 5/n^c)$, the message complexity of collection and dissemination is $O(n)$. $\square_{Lemma\ 9}$