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] Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[2] Massachusetts Institute of Technology, Cambridge, USA

Abstract. Gossip, also known as epidemic dissemination, is becoming an increasingly popular technique in distributed systems. Yet, it has remained a partially open question: how robust are such protocols? We consider a natural extension of the *random phone-call* model (introduced by Karp et al. [1]), and 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.

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 have unique identifiers, but a process does not know the identifiers of the other processes until it has exchanged information with them.

The problem of disseminating information efficiently and robustly can, roughly, be described as follows. Each process p_i begins 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), and any non-failed process should succeed in disseminating its rumor to every other non-failed process. This problem is a basic building block in many distributed settings, such as distributed databases [2], group multicast [3,4], group membership [5], resource location [6], and, recently, update dissemination for mobile nodes [7].

In this paper, we investigate the relationship between *fault tolerance* and *efficiency*. We consider a natural extension of the *random phone-call* model, introduced by Karp et al. [1], and develop protocols for both *adaptive* and *oblivious* failures:

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 history of the ongoing execution. The protocol tolerates $t < n/3$ failures, and terminates in $O(\log^2 n)$ rounds. The time complexity is near-optimal–a simple modification of the result in [8] 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, failing every process that p_i sends a message to. Unless process p_i sends a large number of messages, the adversary can always prevent p_i's rumor from reaching any other process. Our algorithm introduces a scheme for processes to monitor the spread of their rumors, allowing isolated processes to send more and more messages, while preventing 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.

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 perhaps surprising due to the $\Omega(n \log \log n)$ lower bound in [1] on the message complexity of single-source gossip, in which a single process attempts to distribute its message to all others. In [1], 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.

Discussion. 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 may 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.

As a result, a process may not, *a priori*, know the identity of other processes in the system. In such a model, any deterministic protocol requires $\Omega(n^2)$ messages; the lack of knowledge about the world precludes protocols based on specially crafted overlays, such as expander graphs, as used previously in, e.g., [9]. Thus there is a significant difference in the efficiency of deterministic and randomized protocols when the adversary is adaptive.

Another interesting aspect is the gap between the TrickleGossip protocol, which requires $n - t \geq 2n/3$ correct processes, and the lower bound, which requires $n - t = o(n)$ to yield a super-linear bound on message complexity. We conjecture that this gap can be closed by making TrickleGossip more robust, i.e. requiring only $o(n)$ correct processes, while maintaining sub-quadratic message complexity.

Finally, the analysis in this paper focuses on the message complexity, while ignoring the size of messages. The protocols have very low communication overhead, as messages contain only a small number of "control bits." Thus, it seems likely that in the context of using gossip to aggregate data (e.g., calculate the maximum initial rumor), it would be possible, in addition, to achieve small message size.

2 Distributed System Model

We consider a synchronous distributed system consisting of n processes $\{p_1, \ldots, p_n\}$. An execution proceeds in synchronous 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*.

Communication. Each process has a unique identifier. When the execution begins, a process does not know the identity of any other process in the system[3]. The first time that a process p_i receives a message from another process p_j, it learns the identity of 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).

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* (i.e., the *adversary* is *oblivious*). By contrast, when failures may depend on the ongoing execution, we say that they are

[3] 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 any other process *a priori*.

adaptive (i.e., the *adversary* is *adaptive*). Formally, when failures are *oblivious*, we assume that the adversary fixes, prior to the execution, a *failure pattern* that specifies 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 $1, \ldots, r-1$, including the results of the random coin flips.

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 any constant $\gamma > 0$.

3 Related Work

The idea of randomized rumor distribution dates back to Frieze and Grimmett [10], and later, Pittel [11] and Feige [12], 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. [2] showed that rumor spreading can be used for efficient distributed database maintenance.

In a landmark paper, Karp, Schindelhauer, Shenker and Vöcking [1] considered the problem in the *random phone-call model*. They show how to distribute a rumor (in the presence of an oblivious adversary) using $O(\log n)$ rounds and $O(n \log \log n)$ messages, which is optimal in the absence of addresses. Moreover, they show that, even using addresses, no algorithm is simultaneously time and message optimal.

The results of [1] inspired a significant amount of research. Elsässer et al. [13–16] study extensions of the random phone-call model for various types of random graphs. In [17, 18], 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 [10] and [12] are still optimal in this augmented model. In [19], the robustness of the quasirandom model is analyzed when there are probabilistic transmission failures. All these papers consider an *oblivious* adversary.

The model in this paper can be seen as an enrichment of the random phone-call model of Karp et al. [1]. 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 [1], and the later variations in [13,15,17], our model is stronger in that it allows processes to send an arbitrary number of messages per round, and it allows processes to reply in later rounds to messages received in earlier rounds. Note that we consider *gossip* (i.e., n rumors to be spread), rather than broadcast.

There has also been much work on gossip when addresses are available, i.e., when processes know in advance the identities of all the other processes in the system. Kowalski et al. (e.g. [8, 9]) present deterministic algorithms (based on expander-graph constructions) that ensure the dissemination of rumors within time $O(\text{polylog } n)$ using $O(n \text{ polylog } n)$ total messages. These algorithms cannot be used when a process does not know the addresses of the other processes, or more generally, when the names are not derived from a fixed namespace. Earlier research [20] determined time and cost trade-offs for gossip, when only one neighbor is contacted per round. Georgiou et al. [21] provided upper and lower bounds for the complexity of gossip in an asynchronous environment. A survey of prior work on gossip can be found in [22].

A key aspect of the CoordinatedGossip algorithm, presented 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. [23], Kapron et al. [24], King et al. [25, 26] that use universe reduction to solve Byzantine agreement.

In a recent paper [27], 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 [27]; 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, that tolerates adaptive failures. The algorithm ensures, with high probability, that each correct process receives the rumors of other correct processes. It uses $O(n \log^3 n)$ messages, with high probability, and terminates in $O(\log^2 n)$ communication rounds. Previous work [8] implies that the round complexity is optimal, 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, and let $\alpha = 7/12$.

The Dissemination Epoch. The dissemination epoch consists of $\log n$ phases, where each phase is $3 \log n{+}1$ rounds, yielding a round complexity for the epoch of $O(\log^2 n)$.

In the first round of each phase, each process sends its own rumor to other processes. Each process is initially *unmarked*. In the first round of phase i, each unmarked process sends its rumor to a set of $2^i \log n$ randomly chosen processes. (If $2^i \log n \geq n$, then all n processes are chosen). Each process then gathers all the distinct rumors it has received thus far into a *packet*. In each of the following $\log n$ rounds of phase i, each process, whether marked or unmarked, sends its packet to $\log n$ randomly chosen neighbors.

Finally, during the remaining $2 \log n$ rounds of phase i, unmarked processes perform a *testing* process. In each *odd* testing round, each unmarked process sends a *query* to a random set of $2^i \beta \log n$ processes (or all n processes, if $2^i \beta \log n \geq n$). In the *even* testing rounds, each process that received a query sends a reply with the list of rumors it has received. 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) round had received its rumor. Once a process is marked, it stops sending queries, but continues replying to queries from other processes. (However, it continues to send a packet during rounds $2, \ldots, \log n + 1$ in the first part of each 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 $2 \log n$ rounds. At the beginning of the epoch, all processes are again designated *unmarked*. In each odd round of phase i, each unmarked process sends a query to a random set of $2^i \beta \log n$ processes (or all n processes, if $2^i \beta \log n \geq n$). In even rounds, each process that received a query sends a reply with the list of rumors it has received. An unmarked process changes its state to marked if it receives at least $\min(\alpha 2^i \beta \log n, n - t)$ responses. At the end of the epoch, each process delivers all the rumors it has received at any point.

4.2 Analysis

We first show that every process delivers every rumor belonging to a correct process, w.h.p. Second, we show that the algorithm sends $O(n \log^3 n)$ messages, w.h.p. The bound on round complexity follows trivially from the structure of the algorithm. Due to space restrictions, we defer the complete proofs to the the full version of the paper [28].

To simplify the analysis, we will consider executions of the algorithm where $t < n/\kappa$, where κ is a (large) constant. We can "boost" the resiliency from $t < n/\kappa$ failures to $t < n/3$ failures, without affecting its asymptotic complexity, by re-running the protocol κ times; during at least one of these κ executions, by the pigeonhole principle, there are no more than n/κ new failures. (Failures during a previous execution of the protocol have no affect on a later execution.)

We first show that by the end of the first epoch, every rumor belonging to a non-crashed process is spread to $> n/2$ processes w.h.p. This follows since a process p_j is marked only when a large fraction of the randomly sampled query respond that they have already received p_j's rumor; this implies that at least a majority of all processes have received p_j's rumor, w.h.p. Finally, every process is eventually marked, since in the last phase an unmarked process sends its rumor directly to all n other processes.

Lemma 1 (Spreading). *For every correct process, there is a majority of correct processes that have received its rumor during the dissemination epoch, w.h.p.*

During the second epoch, w.h.p., every rumor belonging to a correct process is delivered to every other correct process by the majority that holds it at the end of the first epoch.

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

We now examine the message complexity of TrickleGossip, showing that the algorithm sends $O(n \log^3 n)$ messages, w.h.p. 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 $O(n \log^3 n)$, w.h.p.

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 destinations 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 shows that during the dissemination epoch, the number of unmarked processes is halved in each phase, w.h.p. This lemma is the main technical contribution in this section. We refer the interested reader to the full version of the paper [28] for a complete proof.

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

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 i. We analyze the spread of rumors in U_i during the phase. 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 have not crashed before the beginning of the second round of the phase. We first observe that, by failing f_1 processes during the first round of the phase, the adversary may prevent at most $\Theta(f_1/2^i)$ processes from being *fast*. We denote the set of *fast* rumors for round r_1 by M_i.

Fix some sufficiently small $\delta < 1$ and $\epsilon < 1$. In the second step of the analysis, we argue that in the subsequent rounds $2, \ldots, \log n + 1$, if the adversary fails f_2 processes, then it can stop at most $\Theta(f_2)$ packets from each spreading to at least $n(\alpha + 2\epsilon)$ processes, w.h.p. We say that a packet is *fast* in round r of this part, if it is sent by at least $\min[(\delta \log n)^r, n(\alpha + 2\epsilon)]$ processes that do not crash before round $r + 1$. By a counting argument, we see that by crashing f_r processes in a round r, the adversary may stop at most $\Theta(f_r)$ rumors from being *fast* in round r. Therefore, for each rumor that is *fast* throughout this part of the phase, we are analyzing an epidemic process with a growth rate of $\Theta(\log n)$. Thus, by round $\log n + 1$, each fast rumor has reached at least $(\alpha + 2\epsilon)n$ processes. 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 shows that, by failing f_2 processes during rounds $2, \ldots, \log n + 1$, the adversary can stop at most a total of $\Theta(f)$ packets from being spread to at least $n(\alpha + \epsilon)$ processes.

We now combine the previous two steps: 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, since the destinations in the first round of the phase are chosen at random, a packet contains at most $\Theta(\log n)$ rumors from M_i, w.h.p. (by a Chernoff bound). Therefore, by failing f processes, and hence delaying $O(f)$ rumors in round 1 and $O(f)$ packets in rounds $2, \ldots, \log n + 1$, the adversary can prevent at most $\Theta(f \log n/(2^{i-3} \log n)) \leq \Theta(f/2^i)$ rumors from being spread to at least $n(\alpha + \epsilon)$ processes each, w.h.p. Finally, we conclude that, there are at most $n/2^{i+1}$ rumors started by processes in U_i that are not spread to at least $n(\alpha + \epsilon)$ processes. So at least $|U_i| - n/2^{i+1}$ rumors are spread to at least $n(\alpha + \epsilon)$ 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.

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 assume that ϵ is a function of n, with $1 > \epsilon(n) \geq 0$. For the rest of this section, we omit the argument of the $\epsilon(n)$ function, and simply write ϵ. Theorem 1 focuses on the problem of gossip; the argument can be adapted to yield a similar result in the case of *reliable broadcast* in which a rumor delivered by a correct process has to be delivered to every correct process.

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, w.h.p.*

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 w.h.p. The adversary constructs an execution as follows: it crashes any process that receives a new rumor, and it crashes 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, w.h.p., the algorithm sends $\Theta(n)$ messages in each 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 [29], 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, w.h.p.

6 Gossip and Oblivious Failures

In this section, we present the CoordinatedGossip protocol which sends only $O(n)$ messages, w.h.p. The key underlying idea 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 full version of this paper.

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 then describe how the rumors are collected, and finally 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.) *A. Coordinators:* Each process decides to be a coordinator with probability $\Theta(\log n/n)$. *B. Intermediaries:* Each coordinator chooses $\Theta(\sqrt{n}\log n)$ processes at random, 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. *C. Relays:* Each coordinator participates in c relay elections, for some constant c. That is, for every $\ell \in \{1, \ldots, c\}$, each coordinator chooses $\Theta(n/\log n)$ processes at random,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 of the following seven rounds:

a. Each process that has not *succeeded* in a previous iteration sends its rumor to one randomly selected 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 phase consists of $\Theta(\log n)$ 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 immediately from the fact that each process elects itself coordinator with probability $\Theta(\log n/n)$:

Lemma 5 (Coordinator-Set Size). *There are $\Theta(\log n)$ coordinators, w.h.p.*

Next, we argue that for every pair of coordinators, there is some correct intermediary that has both coordinators as neighbors. This follows from a "birthday-paradox" style argument that randomly chosen sets of size $\sqrt{n}$ intersect with constant probability:

Lemma 6 (Good Intermediaries). *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, w.h.p.*

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 relays for ℓ, w.h.p.*

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. This follows from showing that during one of the $\Theta(\log n)$ iterations of collection/dissemination, each process finds a good relay.

Lemma 8 (Gossip Success). *W.h.p.: (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 finding relays is *efficient*. This fact follows from the following analogy: the attempts by processes to find relays is equivalent to flipping a coin until we get n heads; this requires only $O(n)$ flips, w.h.p.

Lemma 9 (Message Complexity). *The collection and dissemination phases have message complexity $O(n)$, w.h.p.*

We conclude with the main theorem:

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

Proof. By Lemma 8, we see that every rumor is disseminated w.h.p., and by observation, it runs for $O(\log n)$ rounds. In terms of message complexity, when Lemma 5 holds, i.e., w.h.p.: 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 collection and dissemination have message complexity $O(n)$, w.h.p. Thus, overall, the message complexity is $O(n)$, w.h.p.

Acknowledgements. We would like to thank Prof. Hagit Attiya and the anonymous reviewers for their useful comments on earlier drafts of this paper.

References

1. R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking, "Randomized rumor spreading," in *FOCS*, 2000.
2. A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *PODC*, 1987.
3. K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Trans. on Comp. Sys.*, vol. 17, pp. 41–86, 1999.

4. P. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, and P. Kouznetsov, “Lightweight probabilistic broadcast,” *ACM Trans. on Comp. Sys.*, vol. 21, no. 4, 2003.
5. A. Kermarrec, L. Massoulie, and A. Ganesh, “Probabilistic reliable multicast in ad hoc networks,” *IEEE Trans. on Parallel and Distr. Syst.*, vol. 14, 2003.
6. D. Kempe, J. Kleinberg, and A. Demers, “Spatial gossip and resource location protocols,” *Journal of ACM*, pp. 943–967, 2004.
7. A. Chaintreau, J.-Y. Le Boudec, and N. Ristanovic, “The age of gossip: spatial mean field regime,” in *SIGMETRICS*, 2009.
8. B. S. Chlebus and D. R. Kowalski, “Robust gossiping with an application to consensus,” *J. Comput. Syst. Sci.*, vol. 72, no. 8, pp. 1262–1281, 2006.
9. B. S. Chlebus and D. R. Kowalski, “Time and communication efficient consensus for crash failures,” 2006.
10. A. M. Frieze and G. Grimmett, “The shortest-path problem for graphs with random arc-lengths,” *Discrete Applied Mathematics*, no. 10, pp. 55–77, 1985.
11. B. Pittel, “On spreading a rumor,” *SIAM J. Appl. Math.*, vol. 47, no. 1, pp. 213–223, 1987.
12. U. Feige, D. Peleg, P. Raghavan, and E. Upfal, “Randomized broadcast in networks,” in *SIGAL ’90*, (London, UK), pp. 128–137, Springer-Verlag, 1990.
13. R. Elsässer, “On the communication complexity of randomized broadcasting in random-like graphs,” in *SPAA*, 2006.
14. P. Berenbrink, R. Elsaesser, and T. Friedetzky, “Efficient randomised broadcasting in random regular networks with applications in peer-to-peer systems,” in *PODC*, 2008.
15. R. Elsässer and T. Sauerwald, “The power of memory in randomized broadcasting,” in *SODA*, 2008.
16. R. Elsässer and T. Sauerwald, “On the runtime and robustness of randomized broadcasting,” *Theor. Comput. Sci.*, vol. 410, no. 36, pp. 3414–3427, 2009.
17. B. Doerr, T. Friedrich, and T. Sauerwald, “Quasirandom rumor spreading,” in *SODA*, 2008.
18. B. Doerr, T. Friedrich, and T. Sauerwald, “Quasirandom rumor spreading: Expanders, push vs. pull, and robustness,” in *ICALP (1)*, 2009.
19. B. Doerr, A. Huber, and A. Levavi, “Strong robustness of randomized rumor spreading protocols,” *ISAAC*, 2009.
20. A. Czumaj, L. Gasieniec, and A. Pelc, “Time and cost trade-offs in gossiping,” *SIAM Journal on Discrete Mathematics 11, 400-413.*, 1998.
21. C. Georgiou, S. Gilbert, R. Guerraoui, and D. R. Kowalski, “On the complexity of asynchronous gossip,” in *PODC*, 2008.
22. 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.
23. M. Ben-Or, E. Pavlov, and V. Vaikuntanathan, “Byzantine agreement in the full-information model in O(log n) rounds,” in *STOC*, 2006.
24. B. M. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani, “Fast asynchronous byzantine agreement and leader election with full information,” in *SODA*, pp. 1038–1047, 2008.
25. V. King, J. Saia, V. Sanwalani, and E. Vee, “Scalable leader election,” in *SODA*, pp. 990–999, 2006.
26. V. King and J. Saia, “Fast, scalable byzantine agreement in the full information model with a nonadaptive adversary,” in *DISC*, 2009.
27. S. Gilbert and D. Kowalski, “Distributed agreement with optimal communication complexity,” *SODA*, 2010.
28. D. Alistarh, S. Gilbert, R. Guerraoui, and M. Zadimoghaddam, “How efficient can gossip be? (technical report).” https://infoscience.epfl.ch/record/148544.
29. D. Dubhashi and D. Ranjan, “Balls and bins: A study in negative dependence,” *Random Structures and Algorithms*, vol. 13, pp. 99–124, 1996.