Distrib. Comput. (2018) 31:503–513
https://doi.org/10.1007/s00446-017-0312-4

Breaking the $\log n$ barrier on rumor spreading

Chen Avin[1] · **Robert Elsässer**[2]

Received: 22 January 2015 / Accepted: 7 September 2017 / Published online: 15 September 2017

Abstract $O(\log n)$ rounds has been a well known upper bound for rumor spreading using `push&pull` in the *random phone call* model (i.e., uniform gossip in the complete graph). A matching lower bound of $\Omega(\log n)$ is also known for this special case. Under the assumption of this model and with a natural addition that nodes can call a partner once they learn its address (e.g., its IP address) we present a new distributed, address-oblivious and robust algorithm that uses `push&pull` with pointer jumping to spread a rumor to all nodes in only $O(\sqrt{\log n})$ rounds, w.h.p. This algorithm can also cope with $F = O(n/2^{\sqrt{\log n}})$ node failures, in which case all but $O(F)$ nodes become informed within $O(\sqrt{\log n})$ rounds, w.h.p.

Keywords Rumor spreading · Gossip algorithms · Random phone call · Push & pull

An extended abstract of this work appeared in [1]. The work of the second author was partially supported by the Austrian Science Fund (FWF) under contract P25214-N23 *"Analysis of Epidemic Processes and Algorithms in Large Networks"*. The main result of this paper solves an open problem presented at Dagstuhl Seminar 13042 *"Epidemic Algorithms and Processes: From Theory to Applications"*.

✉ Chen Avin
avin@cse.bgu.ac.il

Robert Elsässer
elsa@cosy.sbg.ac.at

1 Communication Systems Engineering, Ben Gurion University of the Negev, Beer-Sheva, Israel

2 Department of Computer Sciences, University of Salzburg, Salzburg, Austria

1 Introduction

Gossiping, or rumor-spreading, is a simple stochastic process for dissemination of information across a network. In a *round* of gossip, *each* node chooses a single, usually random, neighbor as its *communication partner* according to a *gossip algorithm* (e.g., selecting a random neighbor). Once a partner is chosen the node *calls* its partner and a limited amount of data is transferred between the partners, as defined by the gossip *protocol*. Three basic actions are considered in the literature: either the caller pushes information to its partner (`push`), pulls information from the partner (`pull`), or does both (`push&pull`). In the most basic information dissemination task, a token or a rumor in placed arbitrary in the network and we are interested in the number of rounds and message transmissions until all nodes in the network receive the rumor. The selection of the protocol can lead to significant differences in the performance. Take for example the star graph, let nodes call a neighbor selected uniformly at random and assume the rumor is placed at one of the leafs. It is easy to see that both `push` and `pull` will require $\omega(n)$ rounds to complete the spreading of a single rumor while `push&pull` will take only two rounds.

Somewhat surprisingly, but by now well understood, randomized rumor-spreading turned out to be very efficient in terms of time and message complexity while keeping robustness to failures [13,23]. In addition, this type of algorithms are very simple and distributed in nature so it is clear why gossip protocols have gained popularity in recent years and have found many applications both in communication networks and social networks. To name a few examples: updating a database replicated at many sites [9,23], resource discovery [22], computation of aggregate information [24], multicast via network coding [8], membership services [19], or the spread of influence and gossip in social networks [6,25].

In this paper we consider the most basic scenario, the *random phone call model* [23], where the underlying network is the complete graph and nodes can call a random neighbor according to some given distribution. In addition, the model requires the algorithm to be *distributed* and *address-oblivious*: it cannot use the address of the current communication partners to determine its state (for an exact definition see Sect. 2). For example this setting fits well to applications which require communication over the internet such as peer-to-peer protocols and database synchronization. A node can pick and call any (random or given) neighbor via its IP address, but it is desired to keep the algorithm address-oblivious otherwise it may have critical points of failure. For example agreeing beforehand on a leader to contact (by its IP address) is *not* an address-oblivious algorithm. Furthermore, such a protocol is also highly fragile, although it leads to efficient information spreading (as pointed out in the star graph example above).

The random phone call model was thoroughly studied in the literature starting with the work of Frieze and Gimmet [17] and following by Pittel [33] who proved an upper bound of $O(\log n)$ rounds for `push` in the complete graph. Demers et al. [9] considered both `push` and `pull` as a simple and decentralized way to disseminate information in a network and studied their rate of progress. Finally, Karp et al. [23] gave a detailed analysis for this model. They used `push&pull` to optimize the message complexity and showed the robustness of the scheme. They proved that, while using only `push` the communication overhead is $\Omega(n \log n)$, their algorithm only requires $O(n \log\log n)$ message transmissions by having a running time of $O(\log n)$, even under arbitrary oblivious failures. Moreover they proved that any address-oblivious algorithm (that selects neighbors uniformly at random) will require $\Omega(n \log\log n)$ message transmissions.

1.1 Our contribution

We are given a complete graph, in which the nodes act in synchronous rounds. We consider the same assumptions as in the random phone call model: the algorithm needs to be distributed, address-oblivious and it can select neighbors at random. In addition we use the fact that if an address of a node u (e.g., its IP address) is known by some other node v, then v can call directly on the address of u.

This slight addition leads to a significant improvement in the running time (i.e., the number of rounds to spread a message of a node to all other nodes in the network) from $O(\log n)$ to $O(\sqrt{\log n})$, w.h.p., but still keeps the algorithm robust. Furthermore, a node may fail (at the beginning or during the algorithm is executed) with probability $O(1/2^{\sqrt{\log n}})$, independently. The main result of the paper also deals with the so called bit communication complexity, i.e., the number of bits sent over the edges of the network during the execution of the algorithm.

Theorem 1 *At the end of the algorithm Jumping–Push–Pull (JPP), all but $O(F)$ nodes are informed w.h.p.,*[1] *where F is the number of failed nodes (as described above). The algorithm has running time $O(\sqrt{\log n})$ and produces a* **bit** *communication complexity of $O(n(\log^{3/2} n + b \cdot \log\log n))$, w.h.p., where b is the bit length of the message.*

Clearly, if there are no failures (i.e., $F = 0$), then all nodes become informed in the number of rounds given in Theorem 1. As mentioned, we inform all nodes in $O(\sqrt{\log n})$ rounds vs. $O(\log n)$ rounds achieved by the algorithm of Karp et al. Our message complexity is $O(n\sqrt{\log n})$ compared to $O(n \log\log n)$ and if the rumor is of bit length $b = \Omega(\frac{\log^{3/2} n}{\log\log n})$ both of the algorithms bit complexity is $\Omega(b \cdot n \log\log n)$. Moreover, if there are $\Omega(n)$ messages to be distributed in the network, then the first term in the expression describing the bit communication complexity is amortized over the total number of message transmissions (cf. [23]), and we obtain the same communication overhead as in [23].

Few words on the basic idea of the algorithm are in place. In a nutshell our approach has two phases: first we build an infrastructure, a virtual topology, that is efficient for `push&pull`. Second, we perform a simple `push&pull` on the virtual topology. The running time is the combination of both these tasks. For example, constructing a random star would be preferable since the second phase will then take only a constant number of rounds, but as it turns out the cost of the first phase, in this case, is too high. Interestingly, our algorithm results in balancing these two phases where each task requires $O(\sqrt{\log n})$ rounds. Instead of a star with a single leader we build a virtual topology with about random $n/2^{\sqrt{\log n}}$ leaders and each leader is connected to about $2^{\sqrt{\log n}}$ nodes we call *connectors* (a node is either a leader or a connector). Each connector is then linked to two leaders after a process of pointer jumping [28] . This simple 2-level hierarchy results in a very efficient information spreading. Leaders are a source of fast `pull` mechanism and connectors are essential for fast spreading among leaders using `push`. Our approach was motivated by similar phenomena in social networks [2,16] (see the related work section for a more detailed description of these results).

Journal version update Motivated by the conference version of this paper [1], Haeupler and Malkhi [21] improved our bound and presented an elegant algorithm that solves the problem we study here in $O(\log\log n)$ rounds together with a matching lower bound. Nevertheless we think our work

[1] In this paper with high probably or w.h.p. is with probability at least $1 - n^{-1-\Omega(1)}$.

contributes to the understanding of the gossiping process and may be useful in extending the model to general graphs.

2 Preliminaries—rumor spreading

Let $G(V, E)$ be an undirected graph, with V the set of nodes and E the set of edges. Let $n = |V|$ and $m = |E|$. For $v \in V$, let $N(v) = \{u \in V \mid (vu) \in E\}$ the set of neighbors of v and $d(v) = |N(v)|$ the degree of v. Initially a single arbitrary node holds a rumor (i.e., a token) of size b bits; then the process of rumor-spreading (or gossiping) progresses in synchronous *rounds*. At each round, each node v selects a single *communication partner*, $u \in N(v)$ from its neighbors and v calls u. The method by which v choses u is called the *gossip algorithm*. The algorithm is called *address-oblivious* if v's state in round t does not depend on the addresses of its communication partners at time t. Meaning, any decision about if, how and what to send in the current round is made before the current round. Nevertheless, v's state can still depend on the addresses of its communication partners from previous rounds [23].

Randomized gossip is maybe the most basic address-oblivious algorithm, in particular, when the communication partners are selected uniformly at random the process is known as *uniform gossip*. A well studied such case is the *random phone call model* [23] where G is the complete graph and u is selected u.a.r from $V \setminus \{v\}$. Upon selecting a communication partner the *gossip protocol* defines the way and which information is transferred between v and u. Three basic options are considered to deliver information between communication partners: `push`, `pull` and `push&pull`. In `push` the calling node, v, sends a message to the called node u, in `pull` a message is only transferred the other way (if the called node, u, has what to send) and in `push&pull` each of the communication partners sends a message to the node at the other end of the edge. The content of the messages is defined by the protocol and can contain only the rumor (in the simplest case) or additional information like counters or state information (e.g., like in [23]).

After selecting the graph (or graph model), the gossip algorithm and protocol, the main metrics of interest are the dissemination time and the message complexity. Namely how many rounds and messages are needed until all vertices are informed[2] (on average or with high probability), even under node failures. The *bit complexity* is also a metric of interest and counts the *total* number of bits sent during the dissemination time. This quantity is a bit more involved since it depends also on b (the size of the rumor) and messages at different phases of the algorithms may have different sizes.

[2] A call, in which no data is sent (e.g., the rumor, or a pointer), is not considered as a message.

A *pointer jumping* is a classical operation from parallel algorithm design [28] where the destination of your next round pointer is the pointer at which your current pointer points to. Our algorithm uses pointer jumping by sending the addresses (i.e., pointers) of previous communication partners to current partners (see Sect. 4 for a detailed description).

3 Related work

Besides the basic random phone call model, gossip algorithms and rumor spreading were generalized in several different ways. The basic extension was to study *uniform gossip* (i.e., the called partner is selected uniformly at random from the neighbors lists) on graphs other than the clique. Feige et al. [15] studied randomized broadcast in networks and extended the result of $O(\log n)$ rounds for `push` to different types of graphs like hypercubes and random graphs models. Following the work of Karp et al. [23], and in particular in recent years the `push&pull` protocol was studied intensively, both to give tight bounds for general graphs and to understand its performance advantages on specific families of graphs. A lower bound of $\Omega(\log n)$ for uniform gossip on the clique can be concluded from [35] that studies the sequential case. We are not aware of a lower bound for general, address-oblivious `push&pull`.

Recently Giakkoupis [18] proved an upper bound for general graphs as a function of the *conductance*, ϕ, of the graph, which is $O(\phi^{-1} \log n)$ rounds. Since the conductance is at most a constant, this bound cannot lead to a value of $o(\log n)$, but is tight for many graphs. Doerr et al. [10] studied information spreading on a known model of social networks and showed for the first time an upper bound which is $o(\log n)$ for a family of natural graphs. They proved that while uniform gossip with `push&pull` results in $\Theta(\log n)$ rounds in preferential attachment graphs, a slightly improved version where nodes are not allowed to repeat their last call results in a spreading time of $O(\frac{\log n}{\log \log n})$. A similar idea was previously used in [3,14] to reduce the message complexity of `push&pull` in random graphs. Fountoulakis et al. [16] considered spreading a rumor to all but a small ϵ-fraction of the population. For random power law graphs [7] they proved that `push&pull` informs all but an ϵ-fraction of the nodes in $O(\log \log n)$ rounds. Their proof relies on the existence of many *connectors* (i.e., nodes with low degree connected to high degree nodes) which amplify the spread of the rumor between high degree nodes, and this influenced our approach; in some sense our algorithm tries to imitate the structure of the social network they studied.

Another line of research was to study `push&pull` (as well as `push` and `pull` separately) but not under the uniform gossip model. Censor-Hillel et al. [5], gave an algorithm for all-to-all dissemination in arbitrary graphs which eliminates the dependency on the conductance. For unlimited message sizes (essentially you can send everything you know), their randomized algorithm informs all nodes in $O(D + \text{polylog}(n))$ rounds where D is the graph diameter; clearly this is tight for many graphs. Quasirandom rumor spreading was first offered by Doerr et al. [11, 12] and showed to outperform the randomized algorithms in some cases (see also [4] for a study of the message complexity of quasirandom rumor spreading). Most recently Haeupler [20] proposed a completely deterministic algorithm that spreads a rumor with $2(D + \log n) \log n$ rounds (but also requires unlimited message size).

In a somewhat different model (but similar to ours), where nodes can contact any address as soon as they learn about it, Harchol-Balter et al. [22] considered the problem of resource discovery (i.e., learning about all nodes in the graph) starting from an arbitrary graph. They used a form of one hop pointer jumping with `push&pull` and gave an upper bound of $O(\log^2 n)$rounds for their algorithm. In [26, 27] resource discovery in both–the deterministic and the asynchronous case–were studied and improved bounds were presented.

The idea of first building a virtual structure (i.e.; topology control) and then do gossip on top of this structure is not novel and a similar idea was presented by Melamed and Keidar [31].

Another source of influence to our work was the work on pointer jumping with `push&pull` in the context of efficient construction of peer-to-peer networks [30] and on computing minimum spanning tress [29].

4 Jumping–push–pull in $O(\sqrt{\log n})$-time

First, we present the algorithm, which disseminates a rumor by `push&pull` in $O(\sqrt{\log n})$ time, w.h.p. Then, we analyze our algorithm, show its corectness, and prove the runtime bound.

4.1 Algorithm—rumor spreading with pointer jumping

First, we provide a high-level overview of our algorithm. At the beginning, a message resides on one of the nodes, and the goal is to distribute this message (or rumor) to every node in the network. We assume that each node has a unique address (which can e.g. be its IP-address), and every node can select a vertex uniformly at random from the set of all nodes (i.e., like in the random phone call model). Additionally, a node can store a constant number of addresses, out of which it can call one in a future round. In our algorithm a node decides in each round whether it chooses an address uniformly at random or from the pool of the addresses stored before the current round. In general, one may allow the nodes to combine these two abilities (e.g., choose a node at random or use previous knowledge from a probability distribution over both) to select the next communication partner.

In our analysis, we assume for simplicity that every node knows n exactly. However, a slightly modified version of our algorithm also works if the nodes have an estimate of $\log n$, which is correct up to some constant factor. We discuss this case in Sect. 5.

The algorithm consists of five main *phases* and these phases may contain several *rounds* of communication. Basically there are two types of nodes in the algorithm, which we call *leaders* and *connectors*. The algorithm is defined in the following way:

Phase 0 Each informed node performs `push` in every step of this phase. The phase consists of $c \log\log n$ steps, where c is some suitable constant. According to e.g. [23], the message is contained in at least $\log^2 n$ many nodes at the end of this phase, w.h.p.

Phase 1 Each node flips a coin to decide whether it will be a leader, with probability $1/2^{\sqrt{\log n}}$, or a connector, with probability $1 - 1/2^{\sqrt{\log n}}$.

Phase 2 Each connector chooses leaders by preforming five sub-phases of pointer jumping, each for $c\sqrt{\log n}$ rounds. At the end, all but a constant fraction of connectors will have at least 2 leader addresses stored with high probability. Every such connector keeps exactly 2 leader addresses (chosen uniformly at random) and forgets all the others. A detailed description of this phase is given below.

Phase 3 Each connector opens in each round of this phase a communication channel to a randomly chosen node from the list of leaders received in the previous phase. However, once a connector receives the message, it only transmits once in the next round using `push` communication to its other leader. The leaders send the message in each round over all incoming channels during the whole phase (i.e., the leaders send the message by `pull`). The length of this phase is $c\sqrt{\log n}$ rounds.

Phase 4 Every node performs the usual `push&pull` (i.e., median counter algorithm according to [23]) for $c\sqrt{\log n}$ rounds. All informed nodes are considered to be in state B_1 at the beginning of this phase (cf. [23]). For a description of the median counter algorithm of [23] see the proof of Theorem 1.

The second phase needs some clarification: it consists of 5 sub-phases in which connectors chose leaders. In each sub-phase, every connector performs so called pointer-jumping [28] for $c\sqrt{\log n}$ rounds, where c is some large constant. The leaders do not participate in pointer jumping, and when contacted by a connector, they let it know that it has reached a leader. The pointer jumping sub-phase works as follow: in the first round every connector chooses a node uniformly at random, and opens a communication channel to it. Then, each (connector or leader) node, which has incoming communication channels, sends its address by `pull` to the nodes at the other end of these channels. In each round $i > 1$ of this sub-phase, every connector calls on the address obtained in step $i-1$, and opens a channel to the corresponding node. Then, every node transmits over each incomming channel the address obtained in step $i-1$ to the node(s) that opened the channel(s). Clearly, at some time t each node stores only the address received in the previous step $t-1$ of the current sub-phase, and the addresses stored at the end of the previous sub-phases.

If in some sub-phase a connector v does not receive a leader address at all, then it forgets the address stored in the last step of this sub-phase. In this case we say that v is "black" in this sub-phase. The idea of using connectors to amplify the information propagation in graphs has already been used in e.g. [16].

From the description of the algorithm it follows that its running time is $O(\sqrt{\log n})$. In the next section we show that every node becomes informed with probability $1-n^{-1-\Omega(1)}$.

4.2 Analysis of the algorithm

For our analysis we assume the following failure model. Each node may fail (before or during the execution of the algorithm) with some probability $O(1/2^{\sqrt{\log n}})$. This implies that e.g. $n^{1-\epsilon}$ nodes may fail in total, where $\epsilon > 0$ can be any small constant. If a node fails, then it does not participate in any pointer- or message-forwarding process. Moreover, we assume that the other nodes do not realize that a node has failed, even if they contact it directly. That is, all nodes which contact (directly or by pointer-jumping) a failed node in some sub-phase are considered to be what we call *useless* (i.e., failed).

First, we give a high-level overview of our proofs. Basically, we do not consider Phases 0 and 1 in the analysis; the resulting properties on the set of informed nodes are straightforward, and have already been discussed in e.g. [23]. Thus, we know that at the end of Phase 0, the rumor is contained in at least $\log^2 n$ nodes, and at the end of Phase 1 there are $n/2^{\sqrt{\log n}} \cdot (1 \pm o(1))$ leaders, w.h.p. Lemma 1 analyzes Phase 2. We show that most of the connectors will point to a leader after a sub-phase, w.h.p. To show this, we bound the probability that for a node v, the choices of the nodes in the first step of this sub-phase lead to a cycle of connectors. If such a cycle occurs, then after performing pointer jumping for $c\sqrt{\log n}$ steps, v will point to a node on this cycle. Since the probability that a node is on such a cycle is relatively low, and we have in total 5 sub-phases, which are run independetly, we conclude that there may be at most two sub-phases in which an (arbitrary but fixed) connector ends up on a cycle. Hence, each connector will point to a leader, after at least 2 sub-phases as long as no node failures occur. At this point we do not take node failures into account. However, we will consider node failures when the dissemination procedure is analyzed.

In Lemma 2, we bound the number of nodes pointing to the same leader. For this, we consider the layers of nodes, which are at distance 1, 2, etc. from an arbitrary but fixed leader u after the first step of a sub-phase. We derive an upper bound on the total number of layers, and then bound the growth of a layer i compared to the previous layer $i-1$ by standard balls into bins techniques. From this, we obtain an upper bound on the number of nodes pointing to u at the end, which is polynomial in $2^{\sqrt{\log n}}$, w.h.p.

In Lemma 3 we show that most of the connectors share a leader address at the end of a sub-phase with $\Omega(2^{\sqrt{\log n}}/\log n)$ many connectors, w.h.p. Here, we start to consider node failures too. To show this, we compute the expected length of the path from a connector to a leader after the first step of a sub-phase. However, since these distances are not independent, we apply Martingale techniques to show that for most nodes these distances occur with high probability.

Lemma 4 analyzes then the growth in the number of informed nodes within two steps of Phase 3. What we basically show is that after any two steps, the number of informed nodes is increased by a factor of $2^{\sqrt{\log n}/2}$, w.h.p., and most of the newly informed nodes are connected to a (second) leader, which is not informed yet. Thus, most connectors which point to these leaders are also not informed. These will become informed two steps later.

The main theorem then uses the fact that at the end of Phase 3 a $2^{7\sqrt{\log n}}$-th fraction of the nodes is informed, w.h.p. Then, we can apply the algorithm of [23] to inform all nodes within additional $O(\sqrt{\log n})$ steps, w.h.p.

Now we start with the details. In the first lemma we consider only one single sub-phase and assume that there are no node failures.

For every connetor v, let $r(v)$ be the choice of v in the first round of a (fixed) sub-phase. Furthermore, let $R(v)$ be the set of nodes which can be reached by node v using (directed) edges of the form $(u, r(u))$ only. That is, a node u is in $R(v)$ iff there exist some nodes $u_1, \ldots, u_k$ such that $u_1 = r(v)$, $u_{i+1} = r(u_i)$ for any $i \in \{1, \ldots, k-1\}$, and $u = r(u_k)$.

We say that $R(v)$ has a cycle, if there is an integer k and nodes $w_1, \ldots, w_k \in R(v)$ exist such that $w_i = r(w_{i-1})$ for any $i \in \{2, \ldots, k\}$ and $w_1 = w_k$. Clearly, if there are no node failures, then only one of the following cases may occur: either a leader u exists with $u \in R(v)$, or $R(v)$ has a cycle. This holds, since every node w in $R(v)$ has exactly one outgoing (directed) edge $(w, r(w))$, and if no cycle emerges, then we obtain a directed path whose end u must be a leader, since otherwise there would be a further edge $(u, r(u))$ (note that only leaders do not choose any nodes at the beginning of a sub-phase). We prove the following lemma.

Lemma 1 *For an arbitrary but fixed connector v, the set $R(v)$ has a cycle with probability $O\left(\frac{2^{2\sqrt{\log n}}\log^2 n}{n}\right)$. Furthermore, the size of $R(v)$ is $|R(v)| = O(2^{\sqrt{\log n}}\log n)$, w.h.p., and $|R(v)| = O(2^{\sqrt{\log n}})$, with constant probability.*

Before the proof we give some definitions that will be used also in future proofs. Let $P(v)$ be a directed path $(v, u_1, \dots, u_k)$, where $u_1 = r(v)$, $u_{i+1} = r(u_i)$ for any $i \in \{1, \dots, k-1\}$, and $u_i \neq u_j, v$ for any $i, j \in \{1, \dots, k\}$, $i \neq j$. Then, $r(u_k) \in \{v, u_1, \dots, u_{k-1}\}$ with probability $k/(n-1)$. Let this event be denoted by A_k. Furthermore, let B_k be the event that $r(u_k)$ is not a leader (B_1 is the event that neither $r(v)$ nor $r(u_1)$ is a leader). Let L be the set of leaders.

Proof (of Lemma 1) Since communication partners are selected independently we have for $k \le n - |L|$

$$Pr[\overline{A_k} \wedge B_k \mid \overline{A_1} \wedge B_1 \dots \overline{A_{k-1}} \wedge B_{k-1}] = \frac{n-1-|L|-k}{n-1}$$

and

$$Pr[\overline{A_1} \wedge B_1] = \frac{n-1-|L|}{n-1} \cdot \frac{n-|L|-2}{n-1}.$$

Note that for all $k \ge n - |L|$

$$Pr[\overline{A_k} \wedge B_k \wedge \overline{A_1} \wedge B_1 \dots \overline{A_{k-1}} \wedge B_{k-1}] = 0$$

since the number of connectors is at most $n - |L|$ and v is assumed to be a connector too. Simple application of Chernoff bounds imply that $|L| = n(1 \pm o(1))/2^{\sqrt{\log n}}$, w.h.p. We condition on the event that this bound holds on $|L|$, and obtain for $k = c \cdot 2^{\sqrt{\log n}}\log n$ with c being some large constant

$$\begin{aligned}
&Pr[\overline{A_1} \wedge B_1] \cdot Pr[\overline{A_2} \wedge B_2 \mid \overline{A_1} \wedge B_1] \cdot \dots \cdot \\
&\quad \cdot Pr[\overline{A_k} \wedge B_k \mid \overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{k-1}} \wedge B_{k-1}] \\
&\quad \le \Pi_{i=1}^{k} \frac{n-|L|-i-1}{n-1} && (1)\\
&\quad \le \left(1 - \frac{|L|}{n-1}\right)^{c2^{\sqrt{\log n}}\log n} && (2)\\
&\quad \le \left(1 - \frac{1-o(1)}{2^{\sqrt{\log n}}}\right)^{c2^{\sqrt{\log n}}\log n} && (3)\\
&\quad \le n^{-3-\Omega(1)}, && (4)
\end{aligned}$$

whenever c is large enough. The last inequality follows from $(1-1/x)^x \le 1/e$ for all $x \ge 1$. Since $Pr[|R(v)| = k]$ with $k > c2^{\sqrt{\log n}}\log n$ is smaller than $Pr[|R(v)| = c2^{\sqrt{\log n}}\log n]$, we obtain that the size of $R(v)$ is at most $c \cdot 2^{\sqrt{\log n}}\log n$, w.h.p. Applying Inequality (4) with $k = c \cdot 2^{\sqrt{\log n}}$, we obtain that the size of $R(v)$ is at most $c \cdot 2^{\sqrt{\log n}}$, with some constant probability tending to 1 as c tends to ∞.

Now we prove that

$$Pr[R(v) \text{ contains a cycle}] = O\left(\frac{2^{2\sqrt{\log n}}\log^2 n}{n}\right).$$

We know that

$$Pr[A_i \mid \overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{i-1}} \wedge B_{i-1}] = \frac{i}{n-1}.$$

Then, $R(v)$ has a cycle, with probability less than

$$\begin{aligned}
&\sum_{i=1}^{n-|L|-1} Pr[A_i \mid \overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{i-1}} \wedge B_{i-1}] \\
&\quad \cdot Pr[\overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{i-1}} \wedge B_{i-1}] \\
&\le \sum_{i=1}^{c2^{\sqrt{\log n}}\log n} Pr[A_i \mid \overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{i-1}} \wedge B_{i-1}] + \\
&\quad \sum_{i=c2^{\sqrt{\log n}}\log n+1}^{n-|L|-1} Pr[\overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{i-1}} \wedge B_{i-1}] \\
&\le \sum_{i=1}^{c2^{\sqrt{\log n}}\log n} \frac{i}{n-1} + O(n^{-2-\Omega(1)}) \\
&\le \frac{(c2^{\sqrt{\log n}}\log n)^2}{n} + O(n^{-2-\Omega(1)}).
\end{aligned}$$

As already shown, if $i > c2^{\sqrt{\log n}}\log n$, then $Pr[\overline{A_1} \wedge B_1 \wedge \dots \wedge \overline{A_{i-1}} \wedge B_{i-1}] = O(n^{-2-\Omega(1)})$ if c is large enough. □

We can also show the following upper bound on the number of connectors sharing the same leader address. This bound also holds in the case of node failures, since failed nodes can only decrease the number of connectors sharing the same leader address.

Lemma 2 *At the end of a sub-phase each connector shares the same leader address with at most $O(2^{3.1\sqrt{\log n}})$ other connectors, w.h.p.*

Proof For any set S of nodes, let $r(S) = \{v \in V \mid r(v) \in S\}$. We model the parallel process of choosing nodes in the first round of a fixed sub-phase by the following sequential process (that is, the first round of the sub-phase is modeled by the whole sequence of steps of the sequential process). In the first step of the sequential process, all connectors choose a random node. We keep all edges between $(u, r(u))$ with $r(u) \in L$, and release all other edges. Let L_1 denote the set of nodes u with $r(u) \in L$. In the ith step, we let each node of $V \setminus \cup_{j=0}^{i-1} L_j$ choose a node from the set $V \setminus \cup_{j=0}^{i-2} L_j$ uniformly at random, where $L_0 = L$. Recall that the nodes are not allowed to choose themselves. Then, L_i is the set of nodes u with $r(u) \in L_{i-1}$, and all edges $(u, r(u))$ (generated in this step) with $r(u) \notin L_{i-1}$ are released.

Note that the sequential process produces an edge distribution on the nodes of the graph which stochastically dominates the edge distribution produced by the parallel process, since in the sequential process no cycles can occur. If now $S \subset L_{i-1}$, then the probability for a node $v \in V \setminus \cup_{j=0}^{i-1} L_j$ to choose a node in S is $|S|/(|V \setminus \cup_{j=0}^{i-2} L_j| - 1)$. In order to derive an upper bound on the number of nodes choosing a node in S, we model the process by a balls into bins game, in which $|V \setminus \cup_{j=0}^{i-1} L_j|$ balls are thrown into $(|V \setminus \cup_{j=0}^{i-2} L_j| - 1)/|S|$ bins uniformly at random. According to Theorem 1 of [34] the number of nodes v with $r(v) \in S$ is at most $|S| + O(\log n + \sqrt{|S| \log n})$, w.h.p.

Similar to the definition of L_i, for a leader u the nodes v with $r(v) = u$ are in set $L_1(u)$, the nodes v with $r(r(v)) = u$ are in set $L_2(u)$, and generally, the nodes v with $r(v) \in L_{i-1}(u)$ define the set $L_i(u)$.

Then, according to the arguments above (i.e., setting $S = L_i(u)$), we have $|L_{i+1}(u)| = |L_i(u)| + O(\log n + \sqrt{|L_i(u)| \log n})$, w.h.p. We assume now that $|L_1(u)| = \Theta(\log n)$ (from [34] we may conclude that $|L_1(u)| = O(\log n)$, w.h.p.). Then, for any $i \leq c \cdot 2^{\sqrt{\log n}} \log n$, we assume the highest growth for $|L_{i+1}(u)|$, i.e., $|L_{i+1}(u)| = |L_i(u)| + O(\sqrt{|L_i(u)| \log n})$, where c is some constant. This recursion yields $|L_{i+1}(u)| \leq c(i+1)^2 \log n$, if c is large enough. Then, $|L_{c \cdot 2^{\sqrt{\log n}} \log n}(u)| < c^3 2^{2\sqrt{\log n}} \log^3 n$. Since $|R(v)| = O(2^{\sqrt{\log n}} \log n)$ for any v (cf. Lemma 1), and assuming that $|L_i(u)| \leq c i^2 \log n$ for each i, we obtain the claim. □

Let us fix a sub-phase. We allow now node failures (i.e., each node may fail with some probability $O(1/(2^{\sqrt{\log n}}))$, and prove the following lemma.

Lemma 3 *There are cn connectors, where $c > 0$ is a constant, which store the addresses of at least two leaders, and each of these leader addresses is shared by at least $\Omega\left(\frac{2^{\sqrt{\log n}}}{\log n}\right)$ connectors, w.h.p.*

Proof First, we consider the case in which no node failures are allowed. Then, we extend the proof.

We have shown in Lemma 1 that the length of a path $(v, u_1, \ldots, u_k, u)$ from a node v to a leader u is $O(2^{\sqrt{\log n}} \log n)$, w.h.p., where $u_1 = r(v)$, $u_i = r(u_{i-1})$ for any $i \in \{2, \ldots, k\}$, and $u = r(u_k)$. Let u be a leader, and let $L_i(u)$ be the set of connectors which have distance i from u after a certain (arbitrary but fixed) sub-phase of the second phase. Furthermore, let $L_i(L) = \cup_{u \in L} L_i(u)$. For our analysis, we model the process of choosing nodes in the first step of this sub-phase by a sequential process (similar to the proof of the previous lemma), in which first v chooses a node, then $r(v)$ chooses a node, then $r(r(v))$ chooses a node, etc. In step i of this sequential process the i-th node u_{i-1} on the path $P(v)$ chooses a node. For some $i = O(2^{\sqrt{\log n}}/\log n)$ we have

$$Pr[v \notin \cup_{j=1}^{i} L_j(L) \mid \overline{A_1} \wedge \cdots \wedge \overline{A_{i-1}}] \geq \left(1 - \frac{|L|}{n-i-1}\right)^i .$$

Since $L = O(n/2^{\sqrt{\log n}})$, it follows that a node has a path (or a cycle) $P(v)$ of length $\Omega(2^{\sqrt{\log n}}/\log n)$ with probability $1 - o(1)$.

Additionally, observe that, as long as no node failures occur, $R(v)$ either contains a cycle or $v \in \cup_{j=1}^{n-1} L_j(L)$. According to Lemma 1, $R(v)$ has a cycle with probability $O\left(\frac{2^{2\sqrt{\log n}} \log^2 n}{n}\right)$. Thus, $Pr[v \in \cup_{j=1}^{n-1} L_j(L)] = 1 - O(2^{2\sqrt{\log n}} \log^2 n/n)$ and the number of nodes satisfying this property is $n(1 - o(1))$, w.h.p.

So we obtain that, given $R(v) \cap L \neq \emptyset$ a node has a path of length $\Omega(2^{\sqrt{\log n}}/\log n)$ to a leader with probability $1 - o(1)$. Hence, the expected number of such nodes is n(1-o(1)).

Now we consider node failures. A node v is considered *useless*, if it fails (as described at the beginning each node fails with probability $O(1/2^{\sqrt{\log n}})$), or there is a node in $R(v)$, which fails. Since $|R(v)| = O(2^{\sqrt{\log n}})$ with constant probability, there is a node in $R(v)$ that fails with at most some constant probability. However, these probabilities are not independent. Nevertheless, the expected number of nodes, which will not be *useless* **and** have a path of length $\Omega\left(\frac{2^{\sqrt{\log n}}}{\log n}\right)$ to a leader, is $\Theta(n)$.

Now, consider the following martingale. Let $v_1, \ldots, v_{n-|L|}$ denote the connectors. In step j, we reveal the directed edges and nodes from node v_j to all nodes in $R(v_j)$ obtained from a (fixed) sub-phase. That is, let X be the random variable for the number of nodes which are not *useless* and have a path of length at least $c' 2^{\sqrt{\log n}}/\log n$ to a leader, where $c' > 0$ is some small constant. We know that $E[X] = \Theta(n)$. Furthermore, let $X_0 = E[X]$ and define X_i as the conditional expectation of X conditioned by the knowledge of the sets $R(v_1), \ldots, R(v_i)$ together with all edges of the form $(u, r(u))$ where $u \in \cup_{j=1}^{i} R(v_j)$.

Clearly, $X_0, \ldots, X_{n-|L|}$ is a martingale, and assuming that $|R(v_j)| = O(2^{\sqrt{\log n}} \log n)$ for all v_j, this martingale satisfies the $O(2^{\sqrt{\log n}} \log n)$-Lipschitz condition (since $|X_i - X_{i-1}| = O(2^{\sqrt{\log n}} \log n)$). Thus, applying the Azuma–Hoeffding inequality [32], we obtain that $\Theta(n)$ nodes are connected to a leader by a path of length $\Omega\left(\frac{2^{\sqrt{\log n}}}{\log n}\right)$ and will not be *useless*, w.h.p.

We use the following observation to show that all the connectors of on a path to a leader share the same leader address.

Observation 1 *If in an arbitrary but fixed sub-phase of the second phase $R(v) \cap L \neq \emptyset$ for some connector v, then v stores the address of a leader u at the end of this phase,w.h.p.*

This observation is a simple application of the pointer jumping algorithm (that all connectors execute in this sub-phase) on a directed path of length $|R(v)|$. For a directed path of length n, $\log n$ executions of pointer jumping will be sufficient for the first node to point the last node.

According to Lemma 1, $|R(v)| = O(2^{\sqrt{\log n}} \log n)$, w.h.p. so $c\sqrt{\log n}$ executions will be sufficient for every node on the path to point to the same leader.

To conclude the proof of the lemma, a $\Theta(n)$ fraction of the nodes store at the end of the second phase the addresses of at least two leaders, and such a connector shares each of these leader addresses with $\Omega(2^{\sqrt{\log n}}/\log n)$ other connectors, w.h.p. □

Now we concentrate on the third phase. Consider the set of *good* connectors: connectors that stored at least two and at most 5 different leader addresses and each leader address stored by it is shared with at least $\Omega(2^{\sqrt{\log n}}/\log n)$ other connectors. From Lemma 3, there are cn good connectors with high probability. Out of good connectors, let $\tilde{C}$ be the set of nodes v with the following additional property. The first time a leader of v receives the message, v will contact this leader in the next step, pulls the message, and in the next step it will push the message to the other leader.

Note that a good connector v will have this property with a constant probability (since it has a constant number of leaders), independently of the other nodes. Therefore, the total number of nodes in $\tilde{C}$ is $\Theta(n)$, w.h.p.

Now we have the following observation.

Observation 2 *Let C_i be the set of nodes which store the same (arbitrary but fixed) leader address after a certain subsphase, and assume that $|C_i| = \Omega(2^{\sqrt{\log n}}/\log n)$. Then, $|C_i \cap \tilde{C}| = \Theta(|C_i|)$, w.h.p.*

The proof of this observation follows from the fact that if two nodes share the same leader address after a certain subphase, then each of these nodes will share with constant probability a leader address obtained in some other subphase with at least $\Omega(2^{\sqrt{\log n}}/\log n)$ other connectors. However, the events (to share the leader address obtained from another subphase with at least $\Omega(2^{\sqrt{\log n}}/\log n)$ other connectors) for two arbitrary but fixed nodes is not independet. Let now C_j be some set of nodes pointing to a leader address, which contains a node $v \in C_i$. Since $|C_i|, |C_j| = O(2^{3.1\sqrt{\log n}})$ w.h.p. (see Lemma 2), there will be with probability at least $1 - n^{-2}$ at most 4 nodes in $C_i \cap C_j$. Conditioning on this, we define for the nodes of $C_i \cap \tilde{C}$ the following martingale. Let $C_i = \{v_1, \ldots, v_{|C_i|}\}$ and let X the random variable, which describes $|C_i \cap \tilde{C}|$. Clearly, $E[X] = \Theta|C_i|$. Furthermore, $X_0 = E[X]$, and define X_i as the conditional expectation of X, conditioned by the knowledge of the sets pointing to the leaders of the nodes $v_1, \ldots, v_i$. However, since only 4 nodes from C_i share the set C_j of some other leader, $|X_i - X_{i-1}| \leq 4$, and the martingale satisfies the 4-Lipschitz condition, which leads to the statement of the observation.

Now we are ready to show the following lemma.

Lemma 4 *After the third phase the number of informed nodes is at least $\frac{n}{2^{7\sqrt{\log n}}}$, w.h.p.*

Proof For a node $v \in \tilde{C}$, let $C_v^{(1)}$ be the set of nodes, which store the same leader address as the first leader of v, let $C_v^{(2)}$ be defined respectively with the second leader address of v (obtained in the same sub-phases of the second phase), and for which we have $|C_v^{(1)}|, |C_v^{(2)}| = \Omega(2^{\sqrt{\log n}}/\log n)$. We know that each node has exactly 2 leader addresses. Since after Phase 0 at least $\log^2 n$ nodes are informed w.h.p., we may assume that at the beginning of this phase a node $w \in \tilde{C}$ is informed, and w pushes the message exactly once. That is, after two steps all nodes of $C_w^{(j)} \cap \tilde{C}$ are informed, where j is either 1 or 2 (we may assume w.l.o.g. that $j = 1$). Furthermore, we assume that these are the only nodes which are informed after the second step.

Now, we show by induction that the following holds. After $2i$ steps, the number of informed nodes $I(i)$ in $\tilde{C}$ is at least $\min\{2^{\sqrt{\log n}\cdot i/2}, n/2^{7\sqrt{\log n}}\}$, w.h.p. Furthermore, there is a partition of the set $\{C_v^{(j)} \cap \tilde{C} \mid v \in I(i),\ j \in \{1, 2\}\}$, into the sets $E^{(j)}(i)$ and $F^{(j)}(i)$, where $E^{(j)}(i)$ are the sets $C_v^{(j)} \cap \tilde{C}$ with $|C_v^{(j)} \cap \tilde{C} \cap I(i)| = O(\log n)$, and $F^{(j)}(i)$ are the sets $C_v^{(j)} \cap \tilde{C}$ with $C_v^{(j)} \cap \tilde{C} \cap I(i) = C_v^{(j)} \cap \tilde{C}$. Roughly speaking, the sets belonging to $E^{(j)}(i)$ contain some nodes, which have just been informed in the last time step, and most of the nodes from these sets are still uninformed. If now these nodes perform `push`, and in the next step the nodes of the sets in $E^{(j)}(i)$ a `pull`, then these nodes become informed as well.

Our assumption is that the number of sets $E_v^{(j)}(i)$ is $\Omega(|I(i)|/\log n)$, w.h.p. This obviously holds before the first or after the second step.

Assume that the induction hypothesis holds after step $2i$. We are going to show that it also holds after step $2(i + 1)$. Clearly, if U is some set of nodes which have the same leader address after an arbitrary but fixed subphase of the second phase, where $|U| = \Omega(2^{\sqrt{\log n}}/\log n)$, then we have $|U \cap \tilde{C}| = \Theta(|U|)$, w.h.p. (see Observation 2).

On the other hand, there are at least $\Omega(n/2^{3.1\sqrt{\log n}})$ such sets U with $U \notin \cup_{j=1,2} F^{(j)}(i)$, w.h.p., since the largest set we can obtain has size $O(2^{3.1\sqrt{\log n}})$, w.h.p. (cf. Lemma 2). According to our induction hypothesis, at least $\Omega(|I(i)|/\log n)$ and at most $O(|I(i)|)$ of these sets are elements of $E^{(j)}(i)$, where $v \in I(i)$.

Clearly, a node $v \in \tilde{C} \setminus I(i)$ will be in at most one of these sets, w.h.p. Since any of these sets accomodates at least $\Theta(2^{\sqrt{\log n}}/\log n)$ nodes from $\tilde{C}$, w.h.p., the number of informed nodes increases within two steps by at least a factor of $\Theta(2^{\sqrt{\log n}}/\log^2 n) \gg 2^{\sqrt{\log n}/2}$, which leads to $|I(i+1)| \geq 2^{\sqrt{\log n}\cdot(i+1)/2}$, w.h.p. The induction step can be performed as long as $|I(i)| \leq n/2^{7\sqrt{\log n}}$. Now we concentrate on the distribution of these nodes among the sets $U \notin \{E_v^{(j)}(i) \mid v \in I(i),\ j \in \{1,2\}\}$. Note that each such node belongs to two sets; one of these sets is an element of $E_v^{(j)}(i)$ for some $v \in I(i)$, while the other one is not. Since the total number of nodes in some set of $E^{(j)}(i)$ is $O(2^{3.1\sqrt{\log n}})$, w.h.p., we have $|I(i+1)| = O(2^{3.1\sqrt{\log n}} \cdot |I(i)|) = O(n/2^{3.9\sqrt{\log n}})$. As argued above, there are at least $\Omega(n/2^{3.1\sqrt{\log n}})$ sets U with $U \notin \{F_v^{(j)}(i+1) \mid v \in I(i+1),\ j \in \{1,2\}\}$, w.h.p., where U is some set of nodes which have the same leader address after an arbitrary but fixed subphase of the second phase, and $|U| = \Omega(2^{\sqrt{\log n}}/\log n)$. Thus, a node $v \in (I(i+1) \setminus I(i)) \cap \tilde{C}$ is assigned to a fixed such U with probability $O(1/|I(i+1)|)$. Therefore, none of the sets $E_v^{(j)}(i+1)$ will accomodate more than $O(\log n)$ nodes from $(I(i+1) \setminus I(i)) \cap \tilde{C}$, w.h.p. [34], and the claim follows. □

Now we are ready to prove our main theorem, which also compares the communication overhead of the usual `push&pull` algorithm of [23] to our algorithm. Note that the **bit** communication complexity of [23] w.r.t. one rumor is $O(nb \cdot \log\log n)$, w.h.p., where b is the bit length of that rumor. We should also mention here that in [23] the authors assumed that messages (so called updates in replicated databases) are frequently generated, and thus the cost of opening communication channels amortizes over the cost of sending messages through these channels. If in our scenario messages are frequently generated, then we may also assume that the cost of the pointer jumping phase is negligible compared to the cost of sending messages, and thus the communication overhead in our case would also be $O(nb \log\log n)$. In our theorem, however, we assume that one message has to be distributed, and sending the IP-address of a node through a communication channel is $O(\log n)$. Also, opening a channel without sending messages generates an $O(\log n)$ communication cost.

Theorem 1 *At the end of the JPP algorithm, all but O(F) nodes are informed w.h.p., where F is the number of failed nodes as described above. The algorithm has running time $O(\sqrt{\log n})$ and produces a* **bit** *communication complexity of $O(n(\log^{3/2} n + b \cdot \log\log n))$, w.h.p., where b is the bit length of the message.*

Proof In the fourth phase we apply the *median counter* algorithm presented in [23].

For the sake of completeness, we describe this algorithm here as given in [23]. There, each node can be in a state called A, B, C, or D. State B is further subdivided in substates B_1, …, $B_{ctr_{\max}}$, where $ctr_{\max} = O(\log\log n)$ is some suitable integer. At every round each node selects uniformly at random a communication partner and executes both `push` and `pull`. The rules are as follows:

– If a node v in state A receives the rumor only from nodes in state B, then it switches to state B_1. If v obtains the rumor from a state C node, then it switches to state C.
– If a node v in state B_i communicates with more nodes in some state B_j with $j \geq i$ than with nodes in state A or $B_{j'}$ with $j' < i$, then v switches to state B_{i+1}. If v gets the rumor from a state C node, then it switches to state C.
– A node in state C sends the rumor for $O(\log\log n)$ further steps. Then, it switches to state D and stops sending the rumor.

At the beginning of Phase 4 of the JPP algorithm, all informed nodes will be considered in state B_1 and all uninformed nodes in state A. Recall that at the end of the third phase, there are at least $n/2^{7\sqrt{\log n}}$ informed nodes, w.h.p. (cf. Lemma 4). Also, the communication overhead in JPP w.r.t. the rumor is $O(n \cdot b)$ in the third phase, since each connector transmits at most twice the *message*, and the number of leaders is bounded by $O(n/2^{\sqrt{\log n}})$, w.h.p.

Theorem 3.1 of [23] bounds the number of rounds as well as the message complexity of the median counter algorithm. That is, Karp et al. show that the algorithm produces $O(n \log\log n)$ message transmissions and finishes in $O(\log n)$ rounds w.h.p. At the beginning of Phase 4, we denote by $I(t_0)$ the set of informed nodes, and assume that all nodes are in state B_1, where $|I(t_0)| \geq n/2^{7\sqrt{\log n}}$, w.h.p. In the proof of Theorem 3.1 (Case 2), Karp et al. show that "there are no players whose counters will be increased more than some $c \log\log n$ time" as long as the number of informed nodes is less than $n/\log^2 n$, w.h.p.[3] As they mention later, this holds no matter whether node failures are allowed or not. To reach $\Omega(n/\log^2 n)$ informed nodes starting from $\Omega(n/2^{7\sqrt{\log n}})$ requires $O(c\sqrt{\log n})$ rounds since (as shown in [23]) this is what is called an *exponential growth phase* for which $|I(i+1)| > (1+\epsilon)|I(i)|$, w.h.p. Furthermore, in the errorless case the number of steps needed to inform all nodes–once $n/\log^2 n$ nodes are informed–is $O(\log\log n)$ (see beginning of the analysis of Case 2 in the proof of Theorem 3.1 [23]). Thus setting the counter $ctr_{\max}$ to some $c' \log\log n$ with c' being large enough, no node will ever enter state C before all nodes are informed and the number of rounds until all nodes are informed is $O(\sqrt{\log n})$.

Therefore, we may assume that at the time step when all nodes are informed for the first time, the nodes are in some

[3] Note that in [23] the authors consider arbitrary node weights, which are $1/n$ for all nodes in our case.

states B_j with $j \leq ctr_{\max}$. If the nodes are allowed to be in state C for $2c' \log\log n$ steps, then in each of the next rounds the smallest j-value in B_j increases by one. This holds since if at the beginning of some time step, a node is in some state B_j, and all other nodes are in some state $B_{j'}$ or C with $j' \geq j$, then all the B_j-nodes switch their state to B_{j+1} or C in this step. Thus, after $c' \log\log n$ steps, all nodes will be in state C. After additional $2c' \log\log n$ steps all nodes are finally in state D, and they stop transmitting any messages.

If node failures are allowed, then Karp et al. show that until the time step in which $n - O(n/\log n)$ nodes are in state C or D, the total number of message transmissions is $O(n \log\log n)$, w.h.p., and "the remaining error-free players can only cause $O(\log n)$ messages each". This implies that in the case when node failures may occur, the total number of message transmissions will not exceed $O(n \log\log n)$, w.h.p., where each message contains b bits.

Now we are able to count the bit complexity produced in Phase 4. The total number of bits transmitted through the communication channels—as long as less than $n/\log^2 n$ nodes are informed—is at most $n \cdot b/\log n$, since the whole phase requires at most $O(\log n)$ steps, w.h.p., and in each step at most $bn/\log^2 n$ bits are transmitted through the channels. Then, as explained above, after additional $O(\log\log n)$ rounds all nodes are in state D, and in each of these rounds there can be at most nb bits transmitted in total. This leads to a bit communication complexity (for *messages*) of $O(nb \cdot \log\log n)$, w.h.p.

The communication overhead w.r.t. the addresses sent by the nodes in the pointer jumping phase is upper bounded by $O(n\sqrt{\log n} \cdot \log n)$, where $\sqrt{\log n}$ stands for the number of rounds in the second phase, while the $\log n$ term describes the bit size of a message (an address is some polynomial in n). □

5 Discussion–non-exact case

As mentioned in Sect. 4.1, a modified version of our algorithm also works if the nodes only have an estimate of $\log n$, which is accurate up to some constant factor. In this case, we introduce some dummy sub-phases between any two phases and any sub-phases of Phase 2. Now, for a node v the length of the i'th sub-phase of Phase 2 will be $\rho^{2i} c\sqrt{\log n_v}$, and between sub-phases i and $i+1$, there will be a dummy sub-phase of length $\rho^{2i+1} c\sqrt{\log n_v}$. Here n_v is the estimate of n at node v. Accordingly, the dummy sub-phase between Phases 1 and 2 will have length $\rho c\sqrt{\log n_v}$, between Phases 2 and 3 length $\rho^{11} c\sqrt{\log n_v}$, and between 3 and 4 length $\rho^{13} c\sqrt{\log n_v}$. The length of Phase 3 will be $\rho^{12} c\sqrt{\log n_v}$, and that of Phase 4 will be $\rho^{14} c\sqrt{\log n_v}$. Here ρ will be a large constant, such that $\rho^i \gg \sum_{j=0}^{i-1} \rho^j$ for any $i < 15$. Furthermore,

$$\sum_{j=0}^{i} \rho^j c \min_{v \in V} \sqrt{\log n_v} \gg \sum_{j=0}^{i-1} \rho^j c \max_{v \in V} \sqrt{\log n_v} + c \max_{v \in V} \sqrt{\log n_v},$$

where $i \in \{1, \ldots, 15\}$.

The role of the dummy sub-phases is to synchronize the actions of the nodes. That is, no node will enter a phase or sub-phase before the last node leaves the previous phase or sub-phase. Accordingly, no node will leave a phase or a sub-phase, before the last node enters this phase or sub-phase. Moreover, the whole set of nodes will be together for at least $c\sqrt{\log n}$ steps in every phase or sub-phase. This ensures that all the phases and sub-phases of the algorithm will work correctly, and lead to the results we have derived in the previous section. Note that, however, the communication overhead might increase to some value $O(n(\log^{3/2} n + b\sqrt{n}))$.

References

1. Avin, C., Elsässer, R.: Faster rumor spreading: breaking the logn barrier. In: Proceedings of the 27th International Symposium on Distributed Computing–DISC 2013, pp. 209–223. Springer, Berlin (2013)
2. Avin, C., Lotker, Z., Pignolet, Y.-A., Turkel, I.: From caesar to twitter: structural properties of elites and rich-clubs. CoRR abs/1111.3374 (2012)
3. Berenbrink, P., Elsässer, R., Friedetzky, T.: Efficient randomised broadcasting in random regular networks with applications in peer-to-peer systems. In: Proceedings of the 27th ACM Symposium on Principles of Distributed Computing , pp. 155–164 (2008)
4. Berenbrink, P., Elsässer, R., Sauerwald, T.: Communication complexity of quasirandom rumor spreading. Algorithmica **72**(2), 467–492 (2015)
5. Censor-Hillel, K., Haeupler, B., Kelner, J., Maymounkov, P.: Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. In: Proceedings of the 44th ACM Symposium on Theory of Computing, pp. 961–970 (2012)
6. Chaintreau, A., Fraigniaud, P., Lebhar, E.: Opportunistic spatial gossip over mobile social networks. In: Proceedings of the 1st Workshop on Online Social Networks , pp. 73–78 (2008)
7. Chung, F., Lu, L.: Connected components in random graphs with a given degree expected sequence. Ann. Comb. **6**, 125–145 (2002)
8. Deb, S., Médard, M., Choute, C.: Algebraic gossip: a network coding approach to optimal multiple rumor mongering. IEEE Trans. Inf. Theory **52**(6), 2486–2507 (2006)
9. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing , pp. 1–12 (1987)
10. Doerr, B., Fouz, M., Friedrich, T.: Social networks spread rumors in sublogarithmic time. In: Proceeding of the 43rd Annual ACM Symposium on Theory of Computing , pp. 21–30 (2011)
11. Doerr, B., Friedrich, T., Sauerwald, T.: Quasirandom rumor spreading. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 773–781 (2008)
12. Doerr, B., Friedrich, T., Sauerwald, T.: Quasirandom rumor spreading: expanders, Push vs. Pull and Robustness. In: Proceedings of

the 36th International Colloquium on Automata, Languages and Programming, pp. 366–377 (2009)
13. Elsässer, R., Sauerwald, T.: On the runtime and robustness of randomized broadcasting. In: Proceedings of the 17th International Symposium on Algorithms and Computation, pp. 349–358 (2006)
14. Elsässer, R., Sauerwald, T.: The power of memory in randomized broadcasting. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 218–227 (2008)
15. Feige, U., Peleg, D., Raghavan, P., Upfal, E.: Randomized broadcast in networks. Random Struct. Algorithms **1**(4), 447–460 (1990)
16. Fountoulakis, N., Panagiotou, K., Sauerwald, T.: Ultra-fast rumor spreading in social networks. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1642–1660 (2012)
17. Frieze, A.M., Grimmett, G.R.: The shortest-path problem for graphs with random arc-lengths. Discrete Appl. Math. **10**(1), 57–77 (1985)
18. Giakkoupis, G.: Tight bounds for rumor spreading in graphs of a given conductance. In: 28th International Symposium on Theoretical Aspects of Computer Science, pp. 57–68 (2011)
19. Gurevich, M., Keidar, I.: Correctness of gossip-based membership under message loss. SIAM J. Comput. **39**(8), 3830–3859 (2010)
20. Haeupler, B.: Simple, fast and deterministic gossip and rumor spreading. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 705–716 (2013)
21. Haeupler, B., Malkhi, D.: Optimal gossip with direct addressing. In: Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing, New York, NY, PODC '14, pp. 176–185. ACM (2014)
22. Harchol-Balter, M., Leighton, T., Lewin, D.: Resource discovery in distributed networks. In: Proceedings of the 18th Annual ACM symposium on Principles of Distributed Computing, pp. 229–237 (1999)
23. Karp, R., Schindelhauer, C., Shenker, S., Vöcking, B.: Randomized rumor spreading. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 565–574 (2000)
24. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 482–491 (2003)
25. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
26. Kutten, S., Peleg, D.: Asynchronous resource discovery in peer-to-peer networks. Comput. Netw. **51**(1), 190–206 (2007)
27. Kutten, S., Peleg, D., Vishkin, U.: Deterministic resource discovery in distributed networks. Theory Comput. Syst. **36**(5), 479–495 (2003)
28. Leighton, F.T.: Introduction to Parallel Algorithms and Architectures. Morgan Kaufmann, San Francisco (1992)
29. Lotker, Z., Patt-Shamir, B., Pavlov, E., Peleg, D.: Minimum-weight spanning tree construction in o (log log n) communication rounds. SIAM J. Comput. **35**(1), 120–131 (2005)
30. Mahlmann, P., Schindelhauer, C.: Distributed random digraph transformations for peer-to-peer networks. In: Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 308–317 (2006)
31. Melamed, R., Keidar, I.: Araneola: a scalable reliable multicast system for dynamic environments. In: Proceedings Third IEEE International Symposium on Network Computing and Applications, 2004 (NCA 2004), pp. 5–14. IEEE (2004)
32. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, New York (2005)
33. Pittel, B.: On spreading a rumor. SIAM J. Appl. Math. **47**(1), 213–223 (1987)
34. Raab, M., Steger, A.: "Balls into bins"—a simple and tight analysis. In: Proceedings of the RANDOM/APPROX. pp. 159–170 (1998)
35. Sauerwald, T.: On mixing and edge expansion properties in randomized broadcasting. Algorithmica **56**(1), 51–88 (2010)