SIAM J. COMPUT.
Vol. 41, No. 3, pp. 455–483

ALMOST-OPTIMAL GOSSIP-BASED AGGREGATE COMPUTATION*

JEN-YEU CHEN† AND GOPAL PANDURANGAN‡

Abstract. Motivated by applications to modern networking technologies, there has been interest in designing efficient gossip-based protocols for computing aggregate functions. While gossip-based protocols provide robustness due to their randomized nature, reducing the message and time complexity of these protocols is also of paramount importance in the context of resource-constrained networks such as sensor and peer-to-peer networks. We present provably time-optimal efficient gossip-based algorithms for aggregate computation with almost optimal message complexity. Given an n-node network, our algorithms guarantee that all the nodes can compute the common aggregates (such as Max, Min, Average, Sum, and Count) of their values in optimal $O(\log n)$ time and using $O(n \log \log n)$ messages. Our result improves on the algorithm of Kempe, Dobra, and Gehrke [*Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2003, pp. 482–491] that is time-optimal but uses $O(n \log n)$ messages, as well as on the algorithm of Kashyap et al. [*Proceedings of Symposium on Principles of Database Systems*, 2006, pp. 308–317] that uses $O(n \log \log n)$ messages but is not time-optimal (takes $O(\log n \log \log n)$ time). Furthermore, we show that our algorithms can be used to improve gossip-based aggregate computation in sparse communication networks, such as in peer-to-peer networks. The main technical ingredient of our algorithm is a technique called *distributed random ranking (DRR)* that can be useful in other applications as well. DRR gives an efficient distributed procedure to partition the network into a forest of (disjoint) trees of small size. Since the size of each tree is small, aggregates within each tree can be efficiently obtained at their respective roots. All the roots then perform a uniform gossip algorithm on their local aggregates to reach a distributed consensus on the global aggregates. Our algorithms are non-address-oblivious. In contrast, we show a lower bound of $\Omega(n \log n)$ on the message complexity of any address-oblivious algorithm for computing aggregates. This shows that non-address-oblivious algorithms are needed to obtain significantly better message complexity. Our lower bound holds regardless of the number of rounds taken or the size of the messages used. Our lower bound is the first nontrivial lower bound for gossip-based aggregate computation and also gives the first formal proof that computing aggregates is strictly harder than rumor spreading in the address-oblivious model.

Key words. gossip-based protocols, aggregate computation, distributed randomized protocols, probabilistic analysis, lower bounds

AMS subject classifications. 68M14, 68M11, 68M12, 68Q25, 68W40, 68W15, 68W20, 68P20

DOI. 10.1137/100793104

1. Introduction.

1.1. Background and previous work. Aggregate statistics (e.g., Average, Count, Max, Min, and Sum) are significantly useful for many applications in networks [4, 5, 8, 11, 14, 16, 18, 30]. These statistics have to be computed over data stored at individual nodes. For example, in a peer-to-peer network, the average number of files stored at each node or the maximum size of files exchanged between

*Received by the editors April 22, 2010; accepted for publication (in revised form) January 9, 2012; published electronically May 3, 2012. A preliminary version of this paper was published in the *Proceedings of the* 22*nd ACM Symposium on Parallelism in Algorithms and Architectures*, Greece, 2010.

http://www.siam.org/journals/sicomp/41-3/79310.html

†Department of Electrical Engineering, National DongHwa University, ShouFeng, Hualien 97401, Taiwan, ROC (jenyeu@ieee.org). This author's work was supported in part by Taiwan NSC grant NSC-100-2221-E-259-026.

‡Division of Mathematical Sciences, Nanyang Technological University, Singapore 637371, and Department of Computer Science, Brown University, Providence, RI 02912 (gopalpandurangan@gmail.com). This author's work was supported in part by Nanyang Technological University grant M58110000, Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 2 grant MOE2010-T2-2-082, US NSF grant CCF-1023166, and a grant from the United States-Israel Binational Science Foundation (BSF).

nodes is an important statistic needed by system designers for optimizing overall performance [28, 31]. Similarly, in sensor networks, knowing the average or maximum remaining battery power among the sensor nodes is a critical statistic. Many research efforts have been dedicated to developing scalable and distributed algorithms for aggregate computation. Among them, gossip-based algorithms [3, 5, 7, 13, 14, 17, 21, 23, 26, 29] have recently received significant attention because of their simplicity of implementation, scalability to large network size, and robustness to frequent network topology changes. In a gossip-based algorithm, each node exchanges information with a randomly chosen communication partner in each round. The randomness inherent in the gossip-based protocols naturally provides robustness, simplicity, and scalability [12, 13]. We refer to [12, 13, 14] for a detailed discussion on the advantages of gossip-based computation over centralized and deterministic approaches and their attractiveness to emerging networking technologies such as peer-to-peer, wireless, and sensor networks. This paper focuses on designing efficient gossip-based protocols for aggregate computation that have low message and time complexity. This is especially useful in the context of resource-constrained networks such as sensor and wireless networks, where reducing message and time complexity can yield significant benefits in terms of lowering congestion and lengthening node lifetimes.

Much of the early work on the gossip algorithm focused on using randomized communication for rumor propagation (a *broadcasting* problem) [6, 12, 27]. In particular, Karp et al. [12] gave a rumor spreading algorithm (for spreading a single message throughout a network of n nodes) that takes $O(\log n)$ communication rounds and $O(n \log \log n)$ messages. It is easy to establish that $\Omega(\log n)$ rounds are needed by any gossip-based rumor spreading algorithm. (This bound also holds for gossip-based aggregate computation.) They also showed that any rumor spreading algorithm needs at least $\Omega(n \log \log n)$ messages for a class of randomized gossip-based algorithms referred to as *address-oblivious* algorithms [12]. Informally, an algorithm is called address-oblivious if the decision to send a message to its communication partner in a round does not depend on the partner's address. The algorithm of Karp et al. is address-oblivious. For non-address-oblivious algorithms, they show a lower bound of $\omega(n)$ messages if the algorithm is allowed only $O(\log n)$ rounds.

Kempe, Dobra, and Gehrke [14] were the first to present randomized gossip-based algorithms for computing aggregates. They analyzed a gossip-based protocol for computing sums, averages, quantiles, and other aggregate functions. In their scheme for estimating average, each node selects another random node to which it sends half its value; a node on receiving a set of values just adds them to its own halved value. Their protocol takes $O(\log n)$ rounds and uses $O(n \log n)$ messages to converge to the true average in a n-node network. Their protocol is address-oblivious. The work of Kashyap et al. [13] was the first to address the issue of reducing the message complexity of gossip-based aggregate protocols, even at the cost of increasing the time complexity. They presented an algorithm that significantly improves message complexity over the protocol of Kempe, Dobra, and Gehrke. Their algorithm uses only $O(n \log \log n)$ messages but is not time-optimal—it runs in $O(\log n \log \log n)$ time. Their algorithm achieves this $O(\log n / \log \log n)$ factor reduction in the number of messages by randomly clustering nodes into groups of size $O(\log n)$, selecting a representative for each group, and then having the group representatives gossip among themselves. Their algorithm is not address-oblivious. For other related work on gossip-based protocols, see [5, 13] and the references therein.

1.2. Our contributions. We present improved gossip-based algorithms for computing various aggregate functions. Given an n-node network, our algorithms guar-

TABLE 1.1
DRR-gossip vs. other gossip-based algorithms.

Algorithm	Time complexity	Message complexity	Address oblivious
Efficient gossip [13]	$O(\log n \log\log n)$	$O(n \log\log n)$	no
Uniform gossip [14]	$O(\log n)$	$O(n \log n)$	yes
DRR-gossip (this paper)	$O(\log n)$	$O(n \log\log n)$	no

antee that all the nodes can compute the common aggregates (Min/Max, Average, Sum, Count, etc.) of their values in optimal $O(\log n)$ time and using $O(n \log\log n)$ messages. Our result (cf. Table 1.1) improves on the algorithm of Kempe, Dobra, and Gehrke [14] (called "uniform gossip") that is time-optimal but uses $O(n \log n)$ messages as well as on the algorithm of Kashyap et al. [13] (called "efficient gossip") that uses $O(n \log\log n)$ messages but is not time-optimal (takes $O(\log n \log\log n)$ time).

Our algorithms use a simple scheme called *distributed random ranking (DRR)* that gives an efficient distributed protocol to partition a network into a forest of disjoint trees of $O(\log n)$ size. Since the size of each tree is small, aggregates within each tree can be efficiently obtained at their respective roots. All the roots then perform a uniform gossip algorithm on their local (tree) aggregates to reach a distributed consensus on the global aggregates. Our idea of forming trees and then doing gossip among the roots of the trees is similar to the idea of Kashyap et al. The main novelty is that our DRR technique gives a simple and efficient distributed way of decomposing the network into disjoint trees (groups) which takes only $O(\log n)$ rounds and $O(n \log\log n)$ messages. This leads to a simpler and faster algorithm than that of [13]. The paper [26] proposes the following heuristic: divide the network into clusters (called the "bootstrap phase"), aggregate the data within the clusters—these are aggregated in a small subset of nodes within each cluster called clusterheads; the clusterheads then use the gossip algorithm of Kempe, Dobra, and Gehrke to do inter-cluster aggregation; and finally, the clusterheads will disseminate the information to all the nodes in the respective clusters. It is not clear in [26] how to efficiently implement the bootstrap phase of dividing the network into clusters. Also, only numerical simulation results are presented in [26] to show that their approach gives better complexity than the algorithm of Kempe, Dobra, and Gehrke. It is mentioned *without proof* that their approach can take $O(n \log\log n)$ messages and $O(\log n)$ time. Hence, to the best of our knowledge, our work presents the first rigorous protocol that provably shows these bounds.

Our second contribution is analyzing gossip-based aggregate computation in sparse networks. In sparse topologies such as P2P networks, point-to-point communication between all pairs of nodes (as assumed in gossip-based protocols) may not be a reasonable assumption. On the other hand, a small number of neighbors in such networks makes it feasible to send one message simultaneously to all neighbors in one round: in fact, this is a standard assumption in the distributed message passing model [25]. We show how our DRR technique leads to improved gossip-based aggregate computation in such (arbitrary) sparse networks, e.g., P2P network topologies such as Chord [31]. The improvement relies on a key property of the DRR scheme that we prove: the *height* of each tree produced by DRR in any *arbitrary* graph is bounded by $O(\log n)$ w.h.p.[1] In Chord, for example, we show that DRR-gossip takes $O(\log^2 n)$ time w.h.p.

[1]Throughout the paper, "with high probability (w.h.p.)" means "with probability at least $1 - 1/n^{\alpha}$ for some constant $\alpha > 0$."

and $O(n \log n)$ messages. In contrast, uniform gossip gives $O(\log^2 n)$ rounds and $O(n \log^2 n)$ messages.

Our algorithm is non-address-oblivious, i.e., some steps use addresses to decide which partner to communicate in a round. The time complexity of our algorithm is optimal. The message complexity is within a factor $o(\log \log n)$ of the optimal, because Karp et al. [12] showed a lower bound of $\omega(n)$ for any non-address-oblivious rumor spreading algorithm that operates in $O(\log n)$ rounds. (Computing aggregates is at least as hard as rumor spreading—cf. section 2.)

Our third contribution is a nontrivial lower bound of $\Omega(n \log n)$ on the message complexity of any address-oblivious algorithm for computing aggregates. This lower bound holds regardless of the round complexity or the size of the messages (i.e., even assuming that nodes can send arbitrarily long messages). Our result shows that non-address-oblivious algorithms (such as ours) are needed to obtain a significant improvement in message complexity. We note that this bound is significantly larger than the $\Omega(n \log \log n)$ messages shown by Karp et al. [12] for rumor spreading. Thus, our result also gives the first formal proof that computing aggregates is strictly harder than rumor spreading in the address-oblivious model. Another implication of our result is that the algorithm of Kempe, Dobra, and Gehrke [14] is asymptotically message optimal for the address-oblivious model.

1.3. Related work and comparison. It is worthwhile to distinguish between the problem of gossip-based aggregate computation and the general gossiping problem (introduced by Demers et al. [6]) in the random phone call model [12]. In the gossiping problem, each node has to disseminate its initial information (a message or a value) to all the other nodes. To achieve this goal, in each step every node exchanges its acquired information with a randomly chosen neighbor. Rumor spreading (also sometimes called *broadcasting*) can be considered a simpler version of gossiping, where only one (source) node needs to disseminate its initial information to all the other nodes. Clearly the gossiping problem is solved when every node successfully finishes broadcasting its value. As mentioned in the previous section, Karp et al. [12] presented a rumor spreading algorithm that takes $O(n \log n \log n)$ messages and $O(\log n)$ communication rounds.

Note that the requirement for accomplishing gossiping is the *total information exchange* among all nodes. This is a somewhat stronger requirement than computing aggregates. Clearly, an algorithm that does gossiping can be easily used to compute aggregates or any other function of the values (i.e., the initial information) at the n nodes, since in the end every node would know the values of all the other nodes. However, a standard assumption in gossiping is that messages received by a node (and its initial message) can be combined (without message-shorten processing such as fusion or aggregation) and subsequently transmitted as a single message; hence, message sizes can be quite large, say, up to $\Theta(n)$ bits. In contrast, in aggregate computation, typically only small-size (logarithmic) messages are allowed (as used in this paper and in previous literature [13, 14]). Thus, algorithms for the gossiping problem do not directly lead to efficient aggregate computation. Also, in this context, we mention the work in [1, 9] that manages to obtain *fault-tolerant* gossip algorithms with $O(n)$ message complexity and logarithmic time complexity by allowing nodes to send more than one message per round, e.g., a node can send $O(n)$ messages in a round. However, in this paper as well as in previous literature on the random phone call model [12, 13, 14], we allow a node to send only one or a constant number of small-size messages per round.

The recent work of Berenbrink et al. [2] studies the general gossiping problem in the random phone call model of Karp et al. [12]. In this model, Berenbrink et al. [2] present a lower bound of $\Omega(n \log n)$ message complexity for any $O(\log n)$-time randomized gossiping algorithm with probability $1 - o(1)$. This can be seen as a separation result between broadcasting (i.e., rumor spreading) and gossiping in the random phone call model. Our lower bound (i.e., $\Omega(n \log n)$) also applies to the gossiping problem (total information exchange), since our lower bound proof also implies the same lower bound for the gossiping problem in the random phone call model. (Our proof bounds the message complexity of sending $n^{\Omega(1)}$ messages to $\Omega(n)$ nodes.) However, there are differences between the two lower bounds (i.e., ours versus Berenbrink et al. [2]). First, our model (for the lower bound result) is somewhat less general than the definition of "address-oblivious" as given in [12] (which is also the model used in Berenbrink et al. [2]), where a node's decision to communicate with its partner (chosen randomly) can depend on the addresses of the partners chosen in the previous communication rounds. In our case, the decision of whether a node sends a message to its communication partner (which is chosen randomly) does not depend on the partner's address nor on the knowledge of the *addresses* of the partners seen in previous rounds. However, the decision to communicate may depend on the value(s) acquired by the node until the current point in the execution or other factors (e.g., a node may choose not to communicate for a long period of time). Second, our result can be considered stronger in the sense that it does not assume any bound on the running time of the algorithm (although the nodes have less information about the system when making random choices).

1.4. Probabilistic preliminaries. We use Doob martingale extensively in our analysis [19]. Let $X_0, \ldots, X_n$ be *any* sequence of random variables and let Y be any random variable with $E[|Y|] < \infty$. Define the random variable $Z_i = E[Y|X_0, \ldots, X_i]$, $i = 0, 1, \ldots, n$. Then $Z_0, Z_1, \ldots, Z_n$ form a Doob martingale sequence.

We use the martingale inequality known as Azuma's inequality, stated as follows [19]. Let $X_0, X_1, \ldots$ be a martingale sequence such that for each k,

$$|X_k - X_{k-1}| \leq c_k,$$

where c_k may depend on k. Then $\forall t \geq 0$ and any $\lambda > 0$,

$$\Pr(|X_t - X_0| \geq \lambda) \leq 2e^{-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}}. \tag{1.1}$$

We also use the following variant of the Chernoff bound from [24], which works in the case of dependent indicator random variables that are correlated as defined below.

LEMMA 1 (see [24]). *Let $Z_1, Z_2, \ldots, Z_s \in \{0, 1\}$ be random variables such that for all l and for any $S_{l-1} \subseteq \{1, \ldots, l-1\}$, $\Pr(Z_l = 1 | \bigwedge_{j \in S_{l-1}} Z_j = 1) \leq \Pr(Z_l = 1)$. Then for any $\delta > 0$, $\Pr(\sum_{l=1}^{s} Z_l \geq \mu(1+\delta)) \leq (\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu$, where $\mu = \sum_{l=1}^{s} E[Z_l]$.*

1.5. Paper organization. The rest of this paper is organized as follows. The network model is described in section 2, followed by sections where each phase of the DRR-gossip algorithm is introduced and analyzed separately. The DRR-gossip algorithms to compute Average and Max are summarized in section 3.4. Computing other aggregates is described in section 3.5. Section 4 applies DRR-gossip to sparse networks. A lower bound on the message complexity of any address-oblivious algorithm for computing aggregates is presented and proved in section 5. We conclude in section 6.

2. Model. The network $G(V)$ consists of a set V of n nodes; each node $i \in V$ has a data value denoted by v_i. All the node values form a value vector $\mathbf{v} = [v_i]$. The goal is to compute aggregate functions such as Min, Max, Sum, Average, etc., over $\mathbf{v}$ in a distributed manner. We next describe the model used in our paper.

The nodes communicate in discrete time-steps referred to as *rounds*. As in prior work on this problem [12, 13], we assume that rounds are synchronized and all nodes can communicate simultaneously in a given round. Each node can communicate with every other node. In a round, each node can choose a communication partner independently and uniformly at random. As in previous work [12, 14], choosing a random partner is taken as a primitive, i.e., no assumptions are made on how this is implemented. Note that the above previous work assumes a point-to-point communication model between all pairs of nodes, i.e., the underlying graph model is complete. (Presumably it is easy to select a random partner in a complete graph model, since a node has to simply sample a random edge out of itself.) Our algorithm detailed in section 3 assumes the same model. In section 4, we consider a more realistic communication model where the underlying graph can be an arbitrary (in particular, sparse) graph. We refer to section 4 for details.

A node i is said to *call* a node j if i chooses j as a communication partner. Once a call is established, we assume that information can be exchanged in both directions along the link. Two types of messages can be identified based on the direction of the message: A *push* message is one that is sent by a calling node to the called node, and a *pull* message is the message obtained by the calling node from the called node. In one round, a node can call only *one* other node. We assume that nodes have unique addresses. Furthermore, we assume that nodes initially have only local knowledge, i.e., they start off with knowing only their own address and data value. They do not have any information about other nodes' addresses or data values. (However, we assume that they know n—the total number of nodes.) The length of a message is limited to $O(\log n + \log s)$, where s is the range of values. It is important to limit the size of messages used in aggregate computation, as communication bandwidth is often a costly resource in distributed settings. All the above assumptions are also used in prior work [13, 14].

The above model is similar to (but not the same as) the *random phone call* model as defined in Karp et al. [12]. This model was first used by Karp et al. to study rumor spreading via gossip and has subsequently been used (possibly with some changes) to study other computation problems via gossip such as aggregation. The differences are partly due to the fact that we are dealing with different problems: rumor spreading versus aggregate computation. (Note that rumor spreading can be thought of as a special case of aggregate computation—we can adapt an algorithm for computing the Max to do rumor spreading, e.g., by appending the rumor node with a large value. Hence, rumor spreading is not harder than computing aggregates.) One difference is that in Karp et al., it is assumed that there is a special node that starts with a rumor (which is implicit in their model), whereas in our model, there is no such special (or leader) node. However, the main difference is that while the algorithm of Karp et al. follows an *address-oblivious* random phone call, ours does not. In an address-oblivious random phone call model, a node's decision to communicate in a round does not depend on its currently chosen partner's address. (The partner itself is chosen randomly in each round.) However, it can depend on the addresses of its partners chosen in the previous rounds [12]. In contrast, in our model, while most of the rounds involve choosing a partner in a random fashion and communicating in an address-oblivious fashion, some steps do not. In these steps, a node chooses a partner

not randomly but based on its address (which it gets to know during the course of the algorithm). In our algorithm such a step arises only in Phase 3 (cf. section 3.3).

Similar to the algorithms of [13, 14], our algorithm can tolerate the following two types of failures: (i) some fraction of nodes may crash initially, and (ii) links are lossy and messages can get lost. Thus, while nodes cannot fail once the algorithm has started, communication can fail with a certain probability δ. Without loss of generality, $1/\log n < \delta < 1/8$ (or some similar small constant): Larger values of δ, require only $O(1/\log(1/\delta))$ repeated calls to bring down the probability below $1/8$, and smaller values only make it easier to prove our claims.

3. DRR-gossip algorithms. Our algorithm, henceforth called *DRR-gossip*, proceeds in phases. In Phase 1, every node runs the DRR scheme to construct a forest of (disjoint) trees. In Phase 2, each tree computes its local aggregate (e.g., sum or maximum) by a Convergecast process; the local aggregate is obtained at the root. Finally, in Phase 3, all the roots utilize a suitably modified version of the uniform gossip algorithm of Kempe, Dobra, and Gehrke [14] to obtain the global aggregate. Finally, if necessary, the roots forward the global aggregate to other nodes in their trees. We describe each phase in the following sections. We start with the description of the algorithm of DRR.

3.1. Phase 1: DRR. The DRR algorithm is as follows (cf. Algorithm 1). Every node $i \in V$ chooses a rank independently and uniformly at random from $[0, 1]$. (Equivalently, each node can choose a rank uniformly at random from $[1, n^3]$ which leads to the same asymptotic bounds; however, choosing from $[0, 1]$ leads to a smoother analysis, e.g., allows use of integrals.) Each node i then samples up to $\log n - 1$ random nodes sequentially (one in each round) until it finds a node of higher rank to connect to. If none of the $\log n - 1$ sampled nodes have a higher rank, then node i becomes a "root." Since every node except root nodes connects to a node with higher rank, there is no cycle in the graph. Thus, this process results in a collection of disjoint trees which together constitute a forest $\mathbb{F}$.

In the following two theorems, we show the upper bounds of the number of trees and the size of each tree produced by the DRR algorithm; these are critical in bounding the time complexity of DRR-gossip.

THEOREM 1 (number of trees). *The number of trees produced by the DRR algorithm is $O(n/\log n)$ w.h.p.*

Proof. Assume that ranks have already been assigned to the nodes. All ranks are distinct with probability 1. Number the nodes according to the order statistic of their ranks: the ith node is the node with the ith smallest rank. Let the indicator random variable X_i take a value of 1 if the ith smallest node is a root and 0 otherwise. Let $X = \sum_{i=1}^{n} X_i$ be the total number of roots. The ith smallest node becomes a root if all the nodes that it samples have rank smaller than or equal to itself, i.e., $\Pr(X_i = 1) = \left(\frac{i}{n}\right)^{\log n - 1}$. Hence, by linearity of expectation, the expected number of roots (and thus trees) is

$$E[X] = \sum_{i=1}^{n} \Pr(X_i = 1) = \sum_{i=1}^{n} \left(\frac{i}{n}\right)^{\log n - 1}$$
$$= \Theta\left(\int_1^n \left(\frac{i}{n}\right)^{\log n - 1} d\,i\right) = \Theta\left(\frac{n}{\log n}\right).$$

ALGORITHM 1. $\mathbb{F} = \mathrm{DRR}(G)$.

```
foreach node i ∈ V do
    choose rank(i) independently and uniformly at random from [0, 1]
    set found = FALSE // higher ranked node not yet found
    set parent(i)=NULL // initially every node is a root node
    set k = 0 // number of random nodes probed
    repeat
        sample a node u independently and uniformly at random from V and
        get its rank
        if rank(u) > rank(i) then
            set parent(i) = u
            set found = TRUE
            set k = k + 1
        end
    until found == TRUE or k > log n − 1
    if found == TRUE then
        send a connection message including its identifier, i, to its parent node
        parent(i)
    end
    Collect the connection messages and accordingly construct the set of its
    children nodes, Child(i)
    if Child(i) = ∅ then
        become a leaf node
    else
        become an intermediate node
    end
end
```

Note that X_is are independent (but not identically distributed) random variables, since the probability that the ith smallest ranked node becomes the root depends only on the $\log n - 1$ random nodes that it samples and independent of the samples of the rest of the nodes. Thus, applying a Chernoff bound [19], we have $\Pr(X > 6E[X]) \leq 2^{E[X]} = o(1/n)$. □

THEOREM 2 (size of a tree). *The number of nodes in every tree produced by the DRR algorithm is at most $O(\log n)$ w.h.p.*

Proof. We show that the probability that a tree of size $\Omega(\log n)$ is produced by the DRR algorithm goes to zero asymptotically. Fix a set S of $k = c \log n$ nodes for some sufficiently large positive constant c. We first compute the probability that a (spanning) tree of size k is constructed over this set of k nodes by the DRR algorithm. For the sake of analysis, we direct tree edges as follows: a tree edge (i, j) is directed from node i to node j if $\mathrm{rank}(i) < \mathrm{rank}(j)$, i.e., i connects to j. Without loss of generality, fix a permutation of S: $(s_1, \ldots, s_\alpha, \ldots, s_\beta, \ldots, s_k)$, where $\mathrm{rank}(s_\alpha) > \mathrm{rank}(s_\beta), 1 \leq \alpha < \beta \leq k$. This permutation induces a directed spanning tree on S in the following sense: s_1 is the root and any other node s_α $(1 < \alpha \leq k)$ connects to a node in the totally (strictly) ordered set $\{s_1, \ldots, s_{\alpha-1}\}$ (as fixed by the above permutation). For convenience, we denote the event that a node s connects to a node on a directed tree T as $s \triangleright T$. Note that $s \triangleright T$ indicates that s's rank is less than the

rank of the node to which it connects. Also, we denote the event of a directed spanning tree T_h being induced on the totally (strictly) ordered set $\{s_1, s_2, \ldots, s_\alpha, \ldots, s_h\}$ as $\omega(T_h)$. The node s_α can only connect to its preceding nodes in the ordered set. Note that the directed spanning tree T_h is not necessarily unique over the h nodes. Clearly, the event $\omega(T_h)$ and the event $(\omega(T_1) \cap (s_2 \triangleright T_1) \cap (s_3 \triangleright T_2) \cap \cdots \cap (s_h \triangleright T_{h-1}))$ are equivalent.[2] As a special case, $\omega(T_1)$ is the event of the induced directed tree T_1 containing only the root node s_1 and $\Pr(\omega(T_1)) = 1$. We are interested in the event $\omega(T_k)$, i.e., a directed spanning tree T_k is constructed on the set S of k nodes in the above fashion. In the following, we bound the probability of the event $\omega(T_k)$ occurring

$$\begin{aligned}\Pr(\omega(T_k)) &= \Pr\left(\omega(T_1) \cap (s_2 \triangleright T_1) \cap (s_3 \triangleright T_2) \cap \cdots \cap (s_k \triangleright T_{k-1})\right) \\ &= \Pr(\omega(T_1)) \Pr(s_2 \triangleright T_1 |\, \omega(T_1)) \Pr(s_3 \triangleright T_2 |\, \omega(T_2)) \ldots \\ &\quad \ldots \Pr(s_k \triangleright T_{k-1} |\, \omega(T_{k-1})). \end{aligned} \tag{3.1}$$

To bound each of the terms in the product, we use the principle of deferred decisions [19, 22]: when a new node is sampled (i.e., for the first time) we assign it a random rank. For simplicity, we assume that each node sampled is a new node—this does not change the asymptotic bound, since there are now only $k = O(\log n)$ nodes under consideration and each node samples at most $O(\log n)$ nodes. This assumption allows us to use the principle of deferred decisions to assign random ranks without worrying about sampling an already sampled node. Below we bound the conditional probability $\Pr(s_\alpha \triangleright T_{\alpha-1} |\, \omega(T_{\alpha-1}))$ for any $2 \le \alpha \le k$ as follows. Let $r_q = \text{rank}(s_q)$ be the rank of node s_q, $1 \le q \le \alpha$; then

$$\begin{aligned}&\Pr(s_\alpha \triangleright T_{\alpha-1} |\, \omega(T_{\alpha-1})) \\ &\quad \le \int_0^1 \int_0^{r_1} \int_0^{r_2} \cdots \int_0^{r_{\alpha-1}} \sum_{h=1}^{\log n - 1} \left(\frac{\alpha - 1}{n}\right) r_\alpha^{h-1}\, dr_\alpha \ldots dr_1.\end{aligned}$$

The explanation for the above bound is as follows. Since $T_{\alpha-1}$ is a directed spanning tree on the first $\alpha-1$ nodes, and s_α connects to $T_{\alpha-1}$, we have $r_1 > r_2 > \cdots > r_{\alpha-1} > r_\alpha$. Hence r_1 can take any value between 0 and 1, r_2 can take any value between 0 and r_1, and so on. This is captured by the respective ranges of the integrals. The term inside the integrals is explained as follows. There are at most $\log n - 1$ attempts for node s_α to connect to any one of the first $\alpha - 1$ nodes. Suppose it connects at the hth attempt. Then, the first $h - 1$ attempts should have sampled nodes whose rank are less than r_α. Hence, we have the term r_α^{h-1}. The term $(\alpha-1)/n$ is the probability that s_α samples (and hence connects to) any one of the first $\alpha - 1$ nodes at the hth attempt. Simplifying the right-hand side, we have

$$\begin{aligned}&\Pr\left(s_\alpha \triangleright T_{\alpha-1} |\, \omega(T_{\alpha-1})\right) \\ &\quad \le \frac{\alpha - 1}{n} \int_0^1 \int_0^{r_1} \int_0^{r_2} \cdots \int_0^{r_{\alpha-1}} \sum_{h=1}^{\log n - 1} r_\alpha^{h-1}\ dr_\alpha \ldots dr_1 \\ &\quad \le \frac{\alpha - 1}{n} \left(\frac{0!}{\alpha!} + \frac{1!}{(\alpha+1)!} + \frac{2!}{(\alpha+2)!} + \cdots + \frac{(\log n)!}{(\log n + \alpha)!}\right).\end{aligned}$$

[2] Events A and B in a random experiment are said to be equivalent if the probability of the symmetric difference is 0: $\Pr((A \backslash B) \cup (B \backslash A)) = \Pr(A \backslash B) + \Pr(B \backslash A) = 0$. Also, equivalent events have the same probability: if A and B are equivalent, then $\Pr(A) = \Pr(B)$.

The above expression is bounded by $\frac{b}{n}$, where $0 < b < 1$ if $\alpha > 2$ and $0 < b \leq (1 - \frac{1}{\log n+2})$ if $\alpha = 2$. Besides, $\Pr(T_1) \leq \frac{1}{\log n}$ (cf. Theorem 1); hence, (3.1) is bounded by $\left(\frac{b}{n}\right)^{k-1} \frac{1}{\log n}$.

Using the above, the probability that a tree of size $k = c \log n$ is produced by the DRR algorithm is bounded by

$$\begin{aligned}\binom{n}{k} k! \left(\frac{b}{n}\right)^{k-1} \frac{1}{\log n} &\leq \frac{(ne)^k}{k^k} O(\sqrt{k}) \frac{k^k}{e^k} \left(\frac{b}{n}\right)^{k-1} \frac{1}{\log n} \\ &\leq \frac{c' \cdot n}{\log^{\frac{1}{2}} n} \cdot b^{k-1} \\ &\leq \frac{c'}{b^2 \log^{\frac{1}{2}} n} \cdot n^{1+c \log b} = o(1/n)\end{aligned}$$

if c sufficiently large. □

Complexity of Phase 1: The DRR Algorithm.

THEOREM 3. *The message complexity of the DRR algorithm is $O(n \log \log n)$ w.h.p. The time complexity is $O(\log n)$ rounds.*

Proof. Let $d = \log n - 1$. Fix a node i. Its rank is chosen uniformly at random from $[0, 1]$. The expected number of nodes sampled before a node i finds a higher ranked node (or else, all d nodes will be sampled) is computed as follows. The probability that exactly k nodes will be sampled is $\Theta(\frac{1}{k+1}\frac{1}{k})$, since the last node sampled should be the highest ranked node and i should be the second highest ranked node (w.h.p., all the nodes sampled will be unique). Hence the expected number of nodes probed is $\sum_{k=1}^{d} \Theta(k \frac{1}{k+1}\frac{1}{k}) = O(\log d)$ and the expected number of messages exchanged by node i is $O(\log d)$. By linearity of expectation, the expected total number of messages exchanged by all nodes is $O(n \log d) = O(n \log \log n)$.

To show concentration, we set up a Doob martingale [19] as follows. Let X denote the random variable that counts the total number of nodes sampled by all nodes. $E[X] = O(n \log d)$. Assume that ranks have already been assigned to the nodes. Number the nodes according to the order statistic of their ranks: the ith node is the node with the ith smallest rank. Let the indicator random variable Z_{ik} $(1 \leq i \leq n,\ 1 \leq k \leq d)$ indicate whether the kth sample by the ith *smallest ranked* node succeeded (i.e., it found a higher ranked node). If it succeeded, then $Z_{ij} = 1\ \forall j \leq k$ and $Z_{ij} = 0\ \forall j > k$. Thus, $X = \sum_{i=1}^{n} \sum_{k=1}^{d} Z_{ik}$. Then the sequence $X_0 = E[X], X_1 = E[X|Z_{11}], \ldots, X_{nd} = E[X|Z_{11}, \ldots, Z_{nd}]$ is a Doob martingale. Note that $|X_\ell - X_{\ell-1}| \leq d$ $(1 \leq \ell \leq nd)$ because fixing the outcome of a sample of one node affects only the outcomes of other samples made by the same node and not the samples made by other nodes. Applying Azuma's inequality, for a positive constant ϵ we have

$$\Pr(|X - E[X]| \geq \epsilon n) \leq 2 \exp\left(-\frac{\epsilon^2 n^2}{2n(\log n)^3}\right) = o(1/n).$$

The time complexity is immediate since each node probes at most $O(\log n)$ nodes in as many rounds. □

3.2. Phase 2: Convergecast and Broadcast. In the second phase of our algorithm, the local aggregate of each tree is obtained at the root by the Convergecast algorithm—an aggregation process starting from leaf nodes and proceeding upward along the tree to the root node. For example, to compute the local max/min, all leaf

ALGORITHM 2. $\mathbf{cov}_{\max}$ = Convergecast-max($\mathbb{F}$,$\mathbf{v}$).

Input: The forest $\mathbb{F}$, and the value vector $\mathbf{v}$ over all nodes in $\mathbb{F}$
Output: The local Max aggregate vector $\mathbf{cov}_{\max}$ over all the roots
foreach *leaf node* **do** send its value to its parent
foreach *intermediate node* **do**
- collect values from its children
- compare collected values with its own value
- update its value to the maximum amid all and send the maximum to its parent.

end
foreach *root node* z **do**
- collect values from its children
- compare collected values with its own value
- update its value to the local maximum value $\mathbf{cov}_{\max}(z)$.

end

ALGORITHM 3. $\mathbf{cov}_{sum}$ = Convergecast-sum($\mathbb{F}$,$\mathbf{v}$).

Input: The forest $\mathbb{F}$ and the value vector $\mathbf{v}$ over all nodes in $\mathbb{F}$
Output: The local Sum aggregate vector $\mathbf{cov}_{sum}$ over all the roots.
Initialization: every node i stores a row vector $(v_i, w_i = 1)$ including its value v_i and a size count w_i
foreach *leaf node* $i \in \mathbb{F}$ **do**
- send its parent a message containing the vector $(v_i, w_i = 1)$
- reset $(v_i, w_i) = (0, 0)$.

end
foreach *intermediate node* $j \in \mathbb{F}$ **do**
- collect messages (vectors) from its children
- compute and update $v_j = v_j + \sum_{k \in Child(j)} v_k$, and $w_j = w_j + \sum_{k \in Child(j)} w_k$, where $Child(j) = \{j$'s children nodes$\}$
- send computed (v_j, w_j) to its parent
- reset its vector $(v_j, w_j) = (0, 0)$ when its parent successfully receives its message.

end
foreach *root node* $z \in \tilde{V}$ **do**
- collect messages (vectors) from its children
- compute the local Sum aggregate $\mathbf{cov}_{sum}(z, 1) = v_z + \sum_{k \in Child(z)} v_k$, and the size count of the tree $\mathbf{cov}_{sum}(z, 2) = w_z + \sum_{k \in Child(z)} w_k$, where $Child(z) = \{z$'s children nodes$\}$.

end

nodes simply send their values to their parent nodes. An intermediate node collects the values from its children, compares them with its own value, and sends its parent node the max/min value among all received values and its own. A root node can then obtain the local max/min value of its tree.

After the Convergecast process, each root broadcasts its address to all other nodes in its tree via the tree links. This process proceeds from the root down to the leaves via

the tree links (these two-way links were already established during Phase 1.) At the end of this process, all nonsroot nodes know the identity (address) of their respective roots.

Complexity of Phase 2: Convergecast and Broadcast.

THEOREM 4. *The message complexity of the Convergecast and Broadcast algorithms in Phase* 2 *is* $O(n)$ *w.h.p. The time complexity is* $O(\log n)$ *rounds.*

Proof. Every node except the root nodes needs to send a message to its parent in the upward aggregation process of the Convergecast algorithms. Thus the message complexity is $O(n)$. Since each node can communicate with at most one node in one round, the time complexity is bounded by the size of the tree. (This is the reason for bounding size and not just height.) Since the tree size (hence, tree height also) is bounded by $O(\log n)$ (cf. Theorem 2) the time complexity of Convergecast and Broadcast is $O(\log n)$. Moreover, the number of roots is at most $O(n/\log n)$ by Theorem 1; the number of nonroot nodes each of which will receive a Broadcast message is $n - O(n/\log n)$. Thus, the message complexity for Broadcast is also $O(n)$. ∎

ALGORITHM 4. $\hat{\mathbf{x}}_{\max} = \text{Gossip-max}(G, \mathbb{F}, \tilde{V}, \mathbf{y})$.

Initialization: every root $i \in \tilde{V}$ is set the initial value $x_{0,i} = y(i)$ from the input $\mathbf{y}$.

/* To compute Max, $x_{0,i} = y(i) = \mathbf{cov}_{\max}(i)$; To compute the largest tree size (used in computing $Average$), $x_{0,i} = y(i) = \mathbf{cov}_{sum}(i, 2)$. */

Gossip procedure:

for *t from 1 to* $O(\log n)$ *rounds* **do**

- Every root $i \in \tilde{V}$ independently and uniformly at random, selects a node in V and sends the selected node a message containing its current value $x_{t-1,i}$.
- Every node $j \in V - \tilde{V}$ forwards any received messages to its root.
- Every root $i \in \tilde{V}$
  - — collects messages and compares the received values with its own value
  - — updates its current value $x_{t,i}$, which is also the $\hat{\mathbf{x}}_{max,t}(i)$, node i's current estimate of Max, to the maximum among all received values and its own.

end

Sampling procedure:

for *t from 1 to* $\frac{1}{c}\log n$ *rounds* **do**

- Every root $i \in \tilde{V}$ independently and uniformly at random selects a node in V and sends each of the selected nodes an inquiry message.
- Every node $j \in V - \tilde{V}$ forwards any received inquiry messages to its root.
- Every root $i \in \tilde{V}$, upon receiving inquiry messages, sends the inquiring roots its value.
- Every root $i \in \tilde{V}$, updates $x_{t,i}$, i.e. $\hat{\mathbf{x}}_{max,t}(i)$, to the maximum value it inquires.

end

3.3. Phase 3: Gossip. In the third phase, all roots of the trees compute the global aggregate by performing the gossip algorithm on the overlaying graph $\tilde{G} = clique(\tilde{V})$, where $\tilde{V} \subseteq V$ is the set of roots and $|\tilde{V}| = m = O(n/\log n)$

ALGORITHM 5. $\hat{\mathbf{x}}_{ru} = \text{Data-spread}(G, \mathbb{F}, \tilde{V}, x_{ru})$.

Initialization: A root node $i \in \tilde{V}$ which intends to spread its value x_{ru}, $|x_{ru}| < \infty$ sets $x_{0,i} = x_{ru}$. All the other nodes j set $x_{0,j} = \perp$.
Run gossip-max$(G, \mathbb{F}, \tilde{V}, \mathbf{x}_0)$ on the initialized values.

ALGORITHM 6. $\hat{\mathbf{x}}_{ave} = \text{Gossip-ave}(G, \mathbb{F}, \tilde{V}, \mathbf{cov}_{sum})$.

Initialization: Every root $i \in \tilde{V}$ sets a vector $(s_{0,i}, g_{0,i}) = \mathbf{cov}_{sum}(i)$, where $s_{0,i}$ and $g_{0,i}$ are the local sum of values and the size of the tree rooted at i, respectively.
for *t from 1 to* $O(\log m + \log(1/\epsilon))$ *rounds* **do**
Every root node $i \in \tilde{V}$ independently and uniformly at random selects a node in V and sends the selected node a message containing a row vector $(s_{t-1,i}/2, g_{t-1,i}/2)$.
Every node $j \in V - \tilde{V}$ forwards any received messages to the root of its tree.
Let $A_{t,i} \subseteq \tilde{V}$ be the set of roots whose messages reach root node i at round t. Every root node $i \in \tilde{V}$ updates its row vector by
$s_{t,i} = s_{t-1,i}/2 + \sum_{j \in A_{t,i}} s_{t-1,j}/2$,
$g_{t,i} = g_{t-1,i}/2 + \sum_{j \in A_{t,i}} g_{t-1,j}/2$.
Every root node $i \in \tilde{V}$ updates its estimate of the aggregate *Average* by $\hat{\mathbf{x}}_{ave,t}(i) = \hat{x}_{ave,t,i} = s_{t,i}/g_{t,i}$.
end

(cf. Theorem 1). Since the trees are formed randomly, roots do not know each other's addresses. However, the gossip procedure on the roots (i.e., the nodes of $\tilde{G}$) can be implemented as described below.

In each round of the gossip procedure, every root independently and uniformly at random selects a node in V (i.e., calls a node of the graph G) to send its message. If the selected node is another root then the task is completed. If not, the selected node will *forward* the received message to the root of its tree—here is where we use a non-address-oblivious communication. We note that all nodes know their root's address at the end of Phase 2 (cf. section 3.2). Thus, to traverse through an edge of $\tilde{G}$, a message needs at most two hops of G.

Algorithm 4, Gossip-max, and Algorithm 6, Gossip-ave (which is a modification of the Push-Sum algorithm of [14]), compute Max and Average, respectively. (Other aggregates such as Min, Sum, and Count can be calculated by a suitable modification, as mentioned later in section 3.5.) Note that in addition to the gossip procedure, the Gossip-max algorithm needs a sampling procedure in which every root inquires values from other roots. Similar to the gossip procedure, the sampling procedure requires the selected node, if it is a not a root itself, to forward the inquiry message to the root of its tree. The inquiry message contains the inquiring root's address so that the inquired root can send back its value accordingly.

Algorithm 5, Data-spread, a modification of Gossip-max, can be used by a root node to spread its value. If a root needs to spread a particular value over the network, it sets this value as its initial value and all other roots set their initial value to $-\infty$ or the bottom value $\perp$ of the system.

3.3.1. Performance of Gossip-max and Data-spread. Let m denote the number of roots. By Theorem 1, we have $m = |\tilde{V}| = O(n/\log n)$, where $n = |V|$. Karp et al. [12] show that all m nodes of a complete graph can know a particular rumor (e.g., the Max in our application) in $O(\log m) = O(\log n)$ rounds with high probability by using their Push algorithm (a prototype of our Gossip-max algorithm) with uniform selection probability. Similar to the Push algorithm, Gossip-max needs $O(m \log m) = O(n)$ messages for all roots to obtain Max if the selection probability is uniform, i.e., $1/m$. However, in the implementation of the Gossip-max algorithm on the forest, the root of a tree is selected with a *probability proportional to its size (number of nodes in the tree)*. Hence, the selection probability is not uniform. In this case, we can only guarantee that after the gossip procedure of the Gossip-max algorithm, a portion of the roots including the root of the largest tree will possess the Max. After the gossip procedure, roots can sample $O(\log n)$ other roots to confirm and update, if necessary, their values and reach consensus on the global maximum, Max. In the following, we discuss the performance of the Gossip-max algorithm procedure by procedure.

Gossip procedure. We first show the following theorem for the gossip procedure of the Gossip-max algorithm.

THEOREM 5. *After the gossip procedure of the Gossip-max algorithm, at least* $\Omega(\frac{c \cdot n}{\log n})$ *roots obtain the global maximum, Max, w.h.p., where* $n = |V|$ *and* $0 < c < 1$ *is a constant.*

Proof. As per our failure model, a message may fail to reach the selected root node with probability ρ (which is at most 2δ, since failure may occur either during the initial call to a nonroot node or during the forwarding call from the nonroot node to the root of its tree). For convenience, we call those roots that know the Max value (the global maximum) the max-roots and those that do not the non-max-roots.

Let R_t be the number of max-roots in round t. Our proof is in two steps. We first show that, w.h.p., $R_t > 4 \log n$ after $8 \log n/(1-\rho)$ rounds of Gossip-max. If $R_0 > 4 \log n$, then the task is completed. Consider the case when $R_0 < 4 \log n$. Since the initial number of max-roots is small in this case, the probability that a max-root selects another max-root is small. Similarly, the probability that two or more max-roots select the same root is also small. So, in this step, w.h.p. a max-root will select a non-max-root to send out its gossip message. If the gossip message successfully reaches the selected non-max-root, R_t will increase by 1. Let X_i denote the indicator of the event that a gossip message i from some max-root successfully reaches the selected non-max-root. We have $Pr(X_i = 1) = (1-\rho)$. Then $X = \sum_{i=1}^{8 \log n/(1-\rho)} X_i$ is the minimal number of max-roots after $8 \log n/(1-\rho)$ rounds. Clearly, $E[X] = 8 \log n$. Here we conservatively assume the worst situation that initially there is only one max-root and at each round only one max-root selects a non-max-root. So X is the minimal number of max-roots after $8 \log n/(1-\rho)$ rounds. For clarity, let $\tilde{n} = 8 \log n/(1-\rho)$. Define a Doob martingale sequence $Z_0, Z_1, \ldots, Z_{\tilde{n}}$ by setting $Z_0 = E[X]$ and, for $1 \le i \le \tilde{n}$, $Z_i = E[X|X_1, \ldots, X_i]$. It is clear that $Z_{\tilde{n}} = X$ and, for $1 \le i \le \tilde{n}$, $|Z_i - Z_{i-1}| \le 1$.

Applying Azuma's inequality and setting $\epsilon = 1/2$,

$$\begin{aligned}
\Pr(|X - E[X]| \ge \epsilon E[X]) &= \Pr(|Z_{\tilde{n}} - Z_0| \ge \epsilon E[X]) \\
&\le 2 \exp\left(-\frac{\epsilon^2 E[X]^2}{2 \sum_{i=1}^{\tilde{n}} 1^2}\right) = 2 \exp\left(-\frac{\epsilon^2 E[X]^2}{2(\frac{8 \log n}{1-\rho})}\right) \\
&= 2 \exp\left(\log n^{-(1-\rho)}\right) = 2 \cdot n^{-(1-\rho)},
\end{aligned}$$

where ρ could be arbitrarily small. Without loss of generality, let $\rho < 1/4$; then $\Pr(|X - E[X]| \geq \epsilon E[X]) \leq 2 \cdot n^{-\frac{3}{4}}$. Hence, w.h.p., after $8 \log n/(1-\rho) = O(\log n)$ rounds, $R_t \geq R_0 + X > R_0 + \frac{1}{2}E[X] = R_0 + 4 \log n > 4 \log n$.

In the second step of our proof, we lower bound *the increasing rate* of R_t when $R_t > 4 \log n$. In each round, there are R_t messages sent out from max-roots. Let Y_i denote the indicator of an event that such a message i from a max-root successfully reaches a non-max-root. $Y_i = 0$ when either of the following events happens: (1) the message i fails in routing to its destination; (2) the message i is sent to another max-root, although it successfully travels over the network; the probability of this event is at most $((1-\rho)R_t \log n)/n$ since w.h.p. the size of a tree is $O(\log n)$ (cf. Theorem 2); (3) the message i and at least one other message are destined to a same non-max-root. Compared to the probability (which is at most $((1-\rho)R_t \log n)/n)$ that exactly one other message arrives at the same non-max-root picked by message i, the probabilities that two or more other messages choose the same non-max-root as message i are negligibly small (less than $((1-\rho)(R_t \log n)/n)^2)$. Therefore, we consider only the case in which two messages are destined to a same non-max root. We also conservatively exclude both these messages on their possible contributions to the increase of R_t. Thus, this event happens with probability at most $((1-\rho)R_t \log n)/n$.

Applying the union bound [19],

$$\Pr(Y_i = 0) \leq \rho + \frac{2(1-\rho)R_t \log n}{n}.$$

Since $R_t \leq \frac{cn}{\log n}$ for any constant $0 < c < 1$ (otherwise, the task is completed), $\Pr(Y_i = 0) \leq \rho + 2c(1-\rho) = c' + (1-c')\rho$, where $c' = 2c < 1$ is a constant that is suitably fixed so that $c' + (1-c')\rho < 1$. Consequently, we have $\Pr(Y_i = 1) > (1-c')(1-\rho)$ and $E[Y] = \sum_{i=1}^{R_t} E[Y_i] > (1-c')(1-\rho)R_t$.

Applying Azuma's inequality as before,

$$\begin{aligned}\Pr(|Y - E[Y]| > \epsilon E[Y]) &< 2\exp\left(-\frac{\epsilon^2 E[Y]^2}{2R_t}\right) \\ &< 2\exp\left(-\frac{\epsilon^2(1-c')^2(1-\rho)^2 R_t}{2}\right).\end{aligned}$$

Since in this step, w.h.p. $R_t > 4 \log n$, and $(1-c')^2(1-\rho)^2 > 0$, setting $\epsilon = \frac{1}{2}$ and $\alpha = O(1)$, we obtain

$$\Pr(Y < \frac{1}{2}(1-c')(1-\rho)R_t) < 2 \cdot n^{-\alpha}.$$

Thus, w.h.p., $R_{t+1} > R_t + \frac{1}{2}(1-c')(1-\rho)R_t = \beta R_t$, where $\beta = 1 + \frac{1}{2}(1-c')(1-\rho) > 1$. Therefore, w.h.p., after $(8 \log n/(1-\rho) + \log_\beta n) = O(\log n)$ rounds, at least $\Omega(\frac{c \cdot n}{\log n})$ roots will have the Max. □

Sampling procedure. From Theorem 5, after the gossip procedure, there are $\Omega(\frac{cn}{\log n}) = \Omega(cm)$, $0 < c < 1$, roots with the Max value. For all roots to reach consensus on Max, they sample each other as in the sampling procedure. It is possible that the root of a larger tree will be sampled more frequently than the roots of smaller trees. However, this nonuniformity is an advantage, since the roots of larger trees would have obtained Max (in the gossip procedure) with higher probability due to this same nonuniformity. Hence, in the sampling procedure, a root without Max can obtain Max with higher probability by this nonuniform sampling. Thus, we have the following theorem.

THEOREM 6. *After the sampling procedure of the Gossip-max algorithm, all roots know the Max value, w.h.p.*

Proof. After the sampling procedure, the probability that none of the roots possessing the Max is sampled by a root not knowing the Max is at most $\left(\frac{m-cm}{m}\right)^{\frac{1}{c}\log n} < \frac{1}{n}$. Thus, after the sampling procedure, with probability at least $1-\frac{1}{n}$, all the roots will know the Max. □

Complexity of the Gossip-max and Data-spread algorithms.

THEOREM 7. *The Gossip-max algorithm totally takes $O(\log n)$ rounds and $O(n)$ messages for all the roots in the network to reach consensus on Max.*

Proof. In the Gossip-max algorithm, the gossip procedure takes $O(\log n)$ rounds and $O(m \log n) = O(n)$ messages. The sampling procedure takes $O(\frac{1}{c}\log n)$=$O(\log n)$ rounds and $O(\frac{m}{c}\log n) = O(n)$ messages. To sum up, the Gossip-max algorithm totally takes $O(\log n)$ rounds and $O(n)$ messages for all the roots in the network to reach consensus on Max. □

The complexity of the Data-spread algorithm is the same as the Gossip-max algorithm.

3.3.2. Performance of Gossip-ave. When the uniformity assumption holds in gossip (i.e., in each round, nodes are selected uniformly at random), it has been shown in [14] that on an m-clique with probability at least $1-\delta'$, Gossip-ave (uniform push-sum in [14]) needs $O(\log m + \log\frac{1}{\epsilon} + \log\frac{1}{\delta'})$ rounds and $O(m(\log m + \log\frac{1}{\epsilon} + \log\frac{1}{\delta'}))$ messages $\forall m$ (root) nodes to reach consensus on the average within a relative error of at most ϵ. When uniformity does not hold, the performance of uniform gossip will depend on the distribution of selection probability. In the efficient gossip algorithm [13], it is shown that the root being selected with the largest probability will have the global average, Average, in $O(\log m + \log\frac{1}{\epsilon})$ rounds. Here, we show in Theorem 8 that the same upper bound holds for our Gossip-ave algorithm, namely, the root of the largest tree will have Average after $O(\log m + \log\frac{1}{\epsilon})$ rounds of the gossip procedure of Gossip-ave algorithm. In this bound, $m = O(n/\log n)$ is the number of roots (obtained from the DRR algorithm) and the relative error $\epsilon = n^{-\alpha}$, $\alpha > 0$.

THEOREM 8. *W.h.p., there exists a time $T_{ave} = O(\log m + \alpha \log n) = O(\log n)$, $\alpha > 0$, such that for all time $t \geq T_{ave}$, the relative error of the estimate of Average on the root of the largest tree, z, is at most $\frac{2}{n^{\alpha}-1}$, where the relative error is $\frac{|\hat{x}_{ave,t,z} - x_{ave}|}{|x_{ave}|}$ and the value of Average is $x_{ave} = \frac{\sum_i v_i}{n}$.*

(This theorem will be proved later.)

We recall that the Gossip-ave algorithm works on the graph $\tilde{G} = clique(\tilde{V})$, where $\tilde{V} \subseteq V$ is the set of roots and $|\tilde{V}| = m = O(n/\log n)$. To prove Theorem 8, we need some definitions as in [14]. We define a m-tuple contribution vector $\mathbf{y}_{t,i}$ such that $s_{t,i} = \mathbf{y}_{t,i} \cdot \mathbf{x} = \sum_j y_{t,i,j} x_j$ and $w_{t,i} = \|\mathbf{y}_{t,i}\|_1 = \sum_j y_{t,i,j}$, where $y_{t,i,j}$ is the jth entry of $\mathbf{y}_{t,i}$ and x_j is the initial value at root node j, i.e., $x_j = \mathbf{cov}_{sum}(j)$, the local aggregate of the tree rooted at node j computed by Convergecast-sum. $\mathbf{y}_{0,i} = e_i$, the unit vector with the ith entry being 1. Therefore, $\sum_i y_{t,i,j} = 1$, and $\sum_i w_{t,i} = m$. When $\mathbf{y}_{t,i}$ is close to $\frac{1}{m}\mathbf{1}$, where $\mathbf{1}$ is the vector with all entries 1, the approximate value of Average at time t, $\hat{x}_{ave,t,i} = \frac{s_{t,i}}{g_{t,i}}$ is close to the true value of Average, x_{ave}. Note that $w_{t,i}$, which is different from $g_{t,i}$, is a dummy parameter borrowed from [14] to characterize the diffusion speed.

In our Gossip-ave algorithm, we set $g_{0,i}$ to be the size of the root i's tree. The algorithm then computes the estimate of Average directly by $\hat{x}_{ave,t,i} = s_{t,i}/g_{t,i}$. If we set a dummy weight $w_{t,i}$, whose initial value $w_{0,i} = 1\ \forall i \in \tilde{V}$, the algorithm performs in the same manner: every node works on a triplet $(s_{t,i}, g_{t,i}, w_{t,i})$ and computes $\hat{x}_{ave,t,i} = \frac{(s_{t,i}/w_{t,i})}{(g_{t,i}/w_{t,i})}$. $(s_{t,i}/w_{t,i})$ is the estimate of the average local sum on a root and

$g_{t,i}/w_{t,i}$ is the estimate of the average size of a tree. Their relative errors are bounded in the same way as follows.

The relative error in the contributions (with respect to the diffusion effect of gossip) at node i at time t is $\Delta_{t,i} = \max_j |\frac{y_{t,i,j}}{\|\mathbf{y}_{t,i}\|_1} - \frac{1}{m}| = \|\frac{\mathbf{y}_{t,i}}{\|\mathbf{y}_{t,i}\|_1} - \frac{1}{m} \cdot \mathbf{1}\|_\infty$. The potential function $\Phi_t = \sum_{i,j}(y_{t,i,j} - \frac{w_{t,i}}{m})^2$ is the sum of the variance of the contributions $y_{t,i,j}$. Let z be the root of the largest tree.

To prove Theorem 8, we need some auxiliary lemmas.

LEMMA 2 (geometric convergence of Φ). *The conditional expectation*

$$E[\Phi_{t+1}|\Phi_t = \phi] = \frac{1}{2}\left(1 - \sum_{i\in\tilde{V}} P_i^2\right)\phi < \frac{1}{2}\phi,$$

where $P_i = (1-\delta)\frac{g_i}{n}$ is the probability that the root node i is selected by any other root node, g_i is the size of the tree rooted at node i, δ is the probability that a message fails to reach its destined root node, and n is the total number of nodes in the network.

Proof. This proof is generalized from [14]. The difference is that the selection probability, P_i, is no longer uniform but depends on the tree size, g_i. P_i is the probability that root i is selected by any other root and $\sum_{i\in\tilde{V}} P_i^2$ is the probability that two roots select the same root. The conditional expectation of potential at round $t+1$ is

$$\begin{aligned}
&E[\Phi_{t+1}|\Phi_t = \phi] \\
&= \frac{1}{2}\phi + \frac{1}{2}\sum_{i,j,k}\left(y_{i,j} - \frac{w_i}{m}\right)\left(y_{k,j} - \frac{w_k}{m}\right)P_i \\
&\quad + \frac{1}{2}\sum_{j,k}\sum_{k'\neq k}\left(y_{k,j} - \frac{w_k}{m}\right)\left(y_{k',j} - \frac{w_{k'}}{m}\right)\sum_{i\in\tilde{V}} P_i^2 \\
&= \frac{1}{2}\phi + \frac{1}{2}\sum_{i,j,k}\left(y_{i,j} - \frac{w_i}{m}\right)\left(y_{k,j} - \frac{w_k}{m}\right)P_i \\
&\quad + \frac{\sum_{i\in\tilde{V}} P_i^2}{2}\sum_{k,j,k'}\left(y_{k,j} - \frac{w_k}{m}\right)\left(y_{k',j} - \frac{w_{k'}}{m}\right) \\
&\quad - \frac{\sum_{i\in\tilde{V}} P_i^2}{2}\sum_{k,j}\left(y_{k,j} - \frac{w_k}{m}\right)^2 \\
&= \frac{1}{2}\left(1 - \sum_{i\in\tilde{V}} P_i^2\right)\phi \\
&\quad + \frac{1}{2}\sum_{i,j}\left(P_i + \sum_{i\in\tilde{V}} P_i^2\right)\left(y_{i,j} - \frac{w_i}{m}\right)\sum_k\left(y_{k,j} - \frac{w_k}{m}\right) \\
&= \frac{1}{2}\left(1 - \sum_{i\in\tilde{V}} P_i^2\right)\phi < \frac{1}{2}\phi.
\end{aligned}$$

The last equality follows from the fact that

$$\sum_k\left(y_{k,j} - \frac{w_k}{m}\right) = \sum_k y_{k,j} - \sum_k \frac{w_k}{m} = 1 - 1 = 0. \qquad \square$$

LEMMA 3. *There exists a $\tau = O(\log m)$ such that after $\forall t > \tau$ rounds of Gossip-ave, $w_{t,z} \geq 2^{-\tau}$ at z, the root of the largest tree.*

Proof. In the case that the selection probability is uniform, it has been shown in [14] that on an m-clique, with probability at least $1 - \frac{\delta'}{2}$, after $4 \log m + \log 2\delta'$ rounds, a message originating from any node (through a random walk on the clique) would have visited all nodes of the clique. When the distribution of the selection probability is not uniform, it is clear that a message originating from any node must have visited the node with the highest selection probability after a certain number of rounds that is greater than $4 \log m + \log 2\delta'$ with probability at least $1 - \frac{\delta'}{2}$. □

From the previous two lemmas, we derive the following lemma.

LEMMA 4 (diffusion speed of gossip-ave). *With probability at least* $1 - \delta'$*, there exists a time* $T_{ave} = O(\log m + \log \frac{1}{\epsilon} + \log \frac{1}{\delta'})$ *such that* $\forall t \geq T_{ave}$*, the contributions at* z*, root of the largest tree, is nearly uniform, i.e.,* $\max_j |\frac{y_{t,z,j}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m}| = \|\frac{\mathbf{y}_{t,i}}{\|\mathbf{y}_{t,i}\|_1} - \frac{1}{m} \cdot \mathbf{1}\|_\infty \leq \epsilon$.

Proof. By Lemma 2, we obtain that $E[\Phi_t] < (m-1)2^{-t} < m2^{-t}$, as $\Phi_0 = (m-1)$. By Lemma 3, we set $\tau = 4 \log m + \log \frac{2}{\delta'}$ and $\hat{\epsilon}^2 = \epsilon^2 \cdot \frac{\delta'}{2} \cdot 2^{-2\tau}$. Then after $t = \log m + \log \frac{1}{\hat{\epsilon}}$ rounds of Gossip-ave, $E[\Phi_t] \leq \hat{\epsilon}$. By Markov's inequality [19], with probability at least $1 - \frac{\delta'}{2}$, the potential $\Phi_t \leq \epsilon^2 \cdot 2^{-2\tau}$, which guarantees that $|y_{t,i,j} - \frac{w_{t,i}}{m}| \leq \epsilon \cdot 2^{-\tau}$ for all the root nodes i.

To have $\max_j |\frac{y_{t,z,j}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m}| \leq \epsilon$, we need to lower bound the weight of node z. From Lemma 3, $w_{t,z} = \|\mathbf{y}_{t,z}\|_1 \geq 2^{-\tau}$ with probability at least $1 - \frac{\delta'}{2}$. Note that Lemma 3 applies only to z, the root of the largest tree. (A root node of a relatively small tree may not be selected often enough to have such a lower bound on its weight.) Using the union bound, we obtain, with probability at least $1 - \delta'$, $\max_j |\frac{y_{t,z,j}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m}| \leq \epsilon$. □

Now we are ready to prove Theorem 8.

Proof of Theorem 8. For convenience, we restate Theorem 8: *W.h.p., there exists a time* $T_{ave} = O(\log m + \alpha \log n) = O(\log n)$*,* $\alpha > 0$*, such that for all time* $t \geq T_{ave}$*, the relative error of the estimate of Average on the root of the largest tree,* z*, is at most* $\frac{2}{n^\alpha - 1}$*, where the relative error is* $\frac{|\hat{x}_{ave,t,z} - x_{ave}|}{|x_{ave}|}$*, and the value of Average is* $x_{ave} = \frac{\sum_i v_i}{n}$.

Proof. From Lemma 4, with probability at least $1 - \delta'$, it is guaranteed that after $T_{ave} = O(2 \log m + \log \frac{1}{\epsilon} + \log \frac{1}{\delta'})$ rounds of Gossip-ave, at z, the root of the largest tree, $\|\frac{\mathbf{y}_{t,i}}{\|\mathbf{y}_{t,i}\|_1} - \frac{1}{m} \cdot \mathbf{1}\|_\infty \leq \frac{\epsilon}{m}$. Let both $\epsilon = n^{-\alpha}$ and $\delta' = n^{-\alpha}$, $\alpha > 0$; then $T_{ave} = O(2 \log m + 2\alpha \log n) = O(\log n)$.

Using Hölder's inequality [10, 22], we obtain

$$\begin{aligned}
\frac{\left|\frac{s_{t,z}}{w_{t,z}} - \frac{1}{m}\sum_j x_j\right|}{\left|\frac{1}{m}\sum_j x_j\right|} &= \frac{\left|\frac{\mathbf{y}_{t,z} \cdot \mathbf{x}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m} \cdot \mathbf{1} \cdot \mathbf{x}\right|}{\left|\frac{1}{m}\sum_j x_j\right|} = m \cdot \frac{\left|\left(\frac{\mathbf{y}_{t,z}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m} \cdot \mathbf{1}\right) \cdot \mathbf{x}\right|}{\left|\sum_j x_j\right|} \\
&\leq m \cdot \frac{\|\frac{\mathbf{y}_{t,z}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m} \cdot \mathbf{1}\|_\infty \cdot \|\mathbf{x}\|_1}{\left|\sum_j x_j\right|} \\
&\leq \epsilon \cdot \frac{\sum_j |x_j|}{\left|\sum_j x_j\right|}.
\end{aligned}$$

When all x_j have the same sign, we have $\frac{\left|\frac{s_{t,z}}{w_{t,z}} - \frac{1}{m}\sum_j x_j\right|}{\left|\frac{1}{m}\sum_j x_j\right|} \leq \epsilon$. Further, we need to bound the relative error of Average. Without loss of generality, let the *true* average

of the sum of values in a tree be positive, i.e., $s_{ave} = \frac{1}{m}\sum_j x_j > 0$, and, by definition, the *true* average of the size of a tree is also positive, i.e., $g_{ave} = \frac{1}{m}\sum_j g_j = \frac{n}{m} > 0$. Therefore, the global average of all values in the network, Average, is $x_{ave} = \frac{s_{ave}}{g_{ave}}$. Since $|\frac{s_{t,z}}{w_{t,z}} - s_{ave}| \le \epsilon s_{ave}$ and $|\frac{g_{t,z}}{w_{t,z}} - g_{ave}| \le \epsilon g_{ave}$ we obtain

$$\hat{x}_{ave,tz} = \frac{s_{t,z}}{g_{t,z}} = \frac{\left(\frac{s_{t,z}}{w_{t,z}}\right)}{\left(\frac{g_{t,z}}{w_{t,z}}\right)} \in \left[\frac{1-\epsilon}{1+\epsilon}\frac{s_{ave}}{g_{ave}}, \quad \frac{1+\epsilon}{1-\epsilon}\frac{s_{ave}}{g_{ave}}\right].$$

Set $\epsilon' = c\epsilon$, where $c = \frac{2}{(1-\epsilon)} > 2$ is bounded when $\epsilon < 1$. (For example, if $\epsilon \le 10^{-2}$, then $c = 2.0\bar{2}$ and $\epsilon' = \frac{200}{99}\epsilon$.) We set $\epsilon = n^{-\alpha}$, and then $\epsilon' = \frac{2}{n^\alpha - 1} \approx 2\epsilon$. Thus, with probability at least $1 - \frac{1}{n^\alpha}$, the relative error at z is

$$\frac{|\hat{x}_{ave,t,z} - x_{ave}|}{|x_{ave}|} \le \epsilon'$$

after at most $O(\log m + 2\alpha \log n) = O(\log n)$ rounds of Gossip-ave algorithm.

The above assumption that all x_j have the same sign is just for complexity analysis but not for the execution of the Gossip-ave algorithm. The Gossip-ave algorithm works well without any assumption on the values of roots. In the following, we further relax this assumption and show that the upper bound on the running time is also valid when x_j are not all of the same sign.

Let $\gamma = \|\mathbf{x}\|_1 \ne 0$ and $\mathbf{x}' = \mathbf{x} + 2\gamma \cdot \mathbf{1} > \mathbf{0}$, i.e., all $x'_j > 0$ have the same sign. It is obvious that the average aggregate of the $\mathbf{x}'$ is a simple offset of the average aggregate of the $\mathbf{x}$, i.e., $x'_{ave} = x_{ave} + 2\gamma$. Proceeding through the same data exchanging scenario in each round of the Gossip-ave algorithm on $\mathbf{x}$ and $\mathbf{x}'$, after t rounds, at root node z, we have the relationship between the two corresponding estimates of the average aggregates on $\mathbf{x}'$ and $\mathbf{x}$: $\hat{x}'_{ave,\,t,\,z} = \hat{x}_{ave,\,t,\,z} + 2\gamma$. The desired related error is $\frac{|\hat{x}_{ave,\,t,\,z} - x_{ave}|}{|x_{ave}|} \le \epsilon'$. Let $\gamma = O(n^\alpha)$ and a stricter threshold $\tilde{\epsilon} = \epsilon' \frac{|x_{ave}|}{|x_{ave}+2\gamma|} < \epsilon'$. As all x'_j are of the same sign, w.h.p. at least $1 - \frac{1}{\delta'}$, after $t = O(\log n + \log\frac{1}{\tilde{\epsilon}} + \log\frac{1}{\delta'}) = O(\log n)$ rounds,

$$\frac{|\hat{x}'_{ave,\,t,\,z} - x'_{ave}|}{|x'_{ave}|} = \frac{|(\hat{x}_{ave,\,t,\,z} + 2\gamma) - (x_{ave} + 2\gamma)|}{|x_{ave} + 2\gamma|} \le \tilde{\epsilon} = \epsilon' \frac{|x_{ave}|}{|x_{ave} + 2\gamma|}.$$

From the above equation, we conclude that

$$\frac{|\hat{x}_{ave,\,t,\,z} - x_{ave}|}{|x_{ave}|} \le \epsilon'.$$

That is, running the Gossip-ave algorithm on an arbitrary vector $\mathbf{x}$, with probability at least $1 - \frac{1}{\delta'}$, after $t = O(\log n + \log\frac{1}{\tilde{\epsilon}} + \log\frac{1}{\delta'}) = O(\log n)$ rounds, the relative error of the estimate of the average aggregate is less than $\epsilon' = \frac{2}{n^\alpha - 1} = O(n^{-\alpha})$. ∎

Evaluating the performance of the Gossip-ave algorithm using the criterion of relative error causes a problem when $x_{ave} = 0$, whereas the Gossip-ave algorithm works well when $x_{ave} = 0$. In this case, using absolute error criterion, i.e., $|\hat{x}_{ave,\,t,\,z} - x_{ave}| = |\hat{x}_{ave,\,t,\,z}| \le \epsilon'$, is more suitable. Here, we would show that the upper bound of the running time of Theorem 8 is also valid for the case that $x_{ave} = 0$ and the performance is assessed under the absolute error criterion $|\hat{x}_{ave,\,t,\,z}| \le \epsilon'$. By a similar

technique as in the above proof, choose an offset constant $\gamma = O(n^\alpha) > 1$ such that $\mathbf{x}' = \mathbf{x} + \gamma \cdot \mathbf{1} > \mathbf{0}$, i.e., all $x'_j > 0$ are with the same sign. Also, let $\tilde{\epsilon} = \frac{\epsilon'}{|x'_{ave}|} = \frac{\epsilon'}{\gamma} < \epsilon'$. Proceeding through the same data exchanging scenario in each round of the Gossip-ave algorithm on $\mathbf{x}$ and $\mathbf{x}'$, w.h.p. at least $1 - \frac{1}{\delta'}$, after $t = O(\log n + \log \frac{1}{\tilde{\epsilon}} + \log \frac{1}{\delta'}) = O(\log n)$ rounds,

$$\frac{|\hat{x}'_{ave,\,t,\,z} - x'_{ave}|}{|x'_{ave}|} = \frac{|(\hat{x}_{ave,\,t,\,z} + \gamma) - \gamma|}{\gamma} \leq \tilde{\epsilon} = \frac{\epsilon'}{\gamma}.$$

From the above equation, we have that $|\hat{x}_{ave,\,t,\,z}| \leq \epsilon'$. This concludes the mapping relationship between the relative error criterion and the absolute error criterion.

Complexity of the Gossip-ave algorithm.

THEOREM 9. *The Gossip-ave algorithm takes $O(\log n)$ rounds and $O(n)$ messages for the root of the largest tree to obtain Average within a relative error of at most $\frac{2}{n^\alpha - 1}$, $\alpha > 0$.*

Proof. From Theorem 8, the Gossip-ave algorithm needs $O(\log m + \alpha \log n) = O(\log n)$ rounds and hence $mO(\log n) = O(n)$ messages for the root of the largest tree to obtain the global average aggregate, Average, within a relative error of at most $\frac{2}{n^\alpha - 1}$, $\alpha > 0$. □

3.4. DRR-gossip algorithms. Putting together our results from the previous subsections, we present Algorithm 7, DRR-gossip-max, and Algorithm 8, DRR-gossip-ave, for computing Max and Average, respectively.

The DRR-gossip-ave algorithm is more involved than the DRR-gossip-max algorithm. Unlike the Gossip-max algorithm which ensures that all the roots will have Max w.h.p., the Gossip-ave algorithm only guarantees that the root of the largest tree in terms of tree size will have the Average w.h.p. To ensure that all the roots have Average w.h.p., after the Gossip-ave algorithm, the root of the largest tree has to spread out its estimate by using the Data-spread algorithm where the root of the largest tree sets its estimate, the Average, computed by the Gossip-ave algorithm, as the data to be spread out. Therefore, every root needs to know in advance whether it is the root of the largest tree. To achieve this, the Gossip-max algorithm is executed beforehand on tree sizes which are obtained from the Convergecast-sum algorithm. (Note that the Gossip-max procedure in the DRR-gossip-max algorithm is executed on the local maximums computed by the Convergecast-max algorithm.) Every root could compare the maximum tree size obtained from the Gossip-max algorithm with the size of its own tree to recognize whether it is the root of the largest tree. (Note that the Gossip-max algorithm and the Gossip-ave algorithm cannot be executed simultaneously, since the Gossip-ave algorithm does not have the sampling procedure as in the Gossip-max algorithm.) Finally, every root then broadcasts the Average obtained from the Data-spread algorithm to all its tree members.

ALGORITHM 7. DRR-gossip-max.

Run $DRR(G)$ to obtain the forest $\mathbb{F}$ and $\tilde{V}$.
Run Convergecast-max($\mathbb{F}$,$\mathbf{v}$) to obtain local Maximum of every tree $\mathbf{cov}_{\max}$.
Run Gossip-max(G, $\mathbb{F}$, $\tilde{V}$, $\mathbf{cov}_{\max}$) to obtain global Maximum Max at all the roots.
Every root node broadcasts the Max to all nodes in its tree.

ALGORITHM 8. DRR-gossip-ave.

Run DRR(G) algorithm to obtain the forest $\mathbb{F}$ and $\tilde{V}$.
Run Convergecast-sum($\mathbb{F}$, $\mathbf{v}$) algorithm to obtain the local sum of values of every tree $\mathbf{cov}_{sum}(*, 1)$ and the size of every tree $\mathbf{cov}_{sum}(*, 2)$.
Run Gossip-max(G, $\mathbb{F}$, $\tilde{V}$, $\mathbf{cov}_{sum}(*, 2)$) algorithm on the size of trees to find the root of the largest tree. At the end of this step, a root z will know that it is the one with the largest tree size.
Run Gossip-ave(G, $\mathbb{F}$, $\tilde{V}$, $\mathbf{cov}_{sum}$) algorithm for the root z of the largest tree to obtain Average.
Run Data-spread(G, $\mathbb{F}$, $\tilde{V}$, Average) algorithm—the root z of the largest tree uses its average estimate, i.e., Average, as the value to spread.
Every root broadcasts its value to all the nodes in its tree.

Complexity of the DRR-gossip algorithms.

THEOREM 10. *The DRR-gossip algorithms take $O(\log n)$ rounds and $O(n)$ messages for all nodes to reach a consensus on the aggregates Max and Average.*

Proof. To conclude from the previous sections, the time complexity of the DRR-gossip algorithms is $O(\log n)$ since each of all phases needs $O(\log n)$ rounds. The message complexity is dominated by the DRR algorithm in the Phase I which needs $O(n \log \log n)$ messages. □

3.5. Computing other aggregates. In addition to Max and Average, we show how to compute other aggregates. First, Min and Max are essentially the same. Also, it is straightforward to adapt our average computation algorithm to compute the Sum and the Count. To compute the Sum, the Gossip-ave algorithm can be modified in the following way. In the stage of initialization, all the roots except one (say, r) set their weight to zero, i.e., $g_{0,i} = 0\ \forall i \neq r$; the root r sets its weight to one, i.e., $g_{0,r} = 1$. Performing the Gossip-ave algorithm after this modification, in the end, $g_{t,i}$ will converge to $1/m$, where m is the number of roots (and also the number of trees), and the estimate $\hat{x}_{ave,t,i} = s_{t,i}/g_{t,i} = m \cdot s_{t,i}$ at every node will converge to the global Sum because the $s_{t,i}$ will converge to the average of all the local sums each of which is the sum of all values in a tree. To avoid the possible faulty division by zero, all the initial weights can be offset by some predetermined positive constant c, i.e., $g_{0,i} = c\ \forall i \neq r$ and $g_{0,r} = c+1$ for the root r. In the end, all the weights will converge to $c + (1/m)$ and so the value m can be retrieved to obtain the Sum aggregate. Note that by offsetting the initial weights, the estimate $\hat{x}_{ave,t,i}$ will not converge to the global Sum and hence need not be computed. Instead, the value m needs to be first retrieved from the weight $g_{t,i} = c + (1/m)$ and then be used to compute the Sum aggregate, $m \cdot s_{t,i}$.

To compute the Count aggregate, nodes simply set their values to one if they are with some desired property and zero if not. Then by performing the algorithm for computing Sum, we can obtain the Count of nodes with some desired property. Other more involved aggregates such as linear synopses and quantiles can also be computed by modifying the gossip algorithm in the way as shown in [14].

4. Sparse networks: Local-DRR algorithm. In sparse networks, a small number of neighbors makes it feasible for each node to send messages to all of its neighbors simultaneously in one round. In fact, this is a standard assumption in the tradi-

tional message passing distributed computing model [25]. (Here it is assumed messages sent to different neighbors in one round can all be different.) We show how DRR-gossip can be used to improve gossip-based aggregate computation in such networks.

We assume that in a round of time, a node of an arbitrary undirected graph can communicate directly only with its immediate neighbors (i.e., nodes that are connected directly by an edge). Thus, nodes have only local knowledge. They know only their immediate neighbors and have no knowledge of the global topology. (Note that in previous sections, any two nodes can communicate with each other in a round under a complete graph model.) Thus, on such a communication model, we have a variant of the DRR algorithm, called the *Local-DRR algorithm*, where a node only exchanges rank information with its immediate neighbors. Each node chooses a random rank in $[0, 1]$ as before. Then each node connects to its highest-ranked neighbor (i.e., the neighbor which has the highest rank among all its neighbors). A node that has the highest rank among all its neighbors will become a root. Since every node, except root nodes, connects to a node with higher rank, there is no cycle in the graph. Thus this process results in a collection of disjoint trees. As shown in Theorem 11 below, the key property is that the *height* of each tree produced by the Local-DRR algorithm on an *arbitrary* graph is bounded by $O(\log n)$ w.h.p. This enables us to bound the time complexity of Phase 2 of the DRR-gossip algorithm, i.e., Convergecast and Broadcast, on an arbitrary graph by $O(\log n)$ w.h.p.

THEOREM 11. *On an arbitrary undirected graph, all the trees produced by the Local-DRR algorithm have a height of at most* $O(\log n)$ *w.h.p.*

Proof. Fix any node u_0. We first show that the path from u_0 to a root is at most $O(\log n)$ long, w.h.p. Let $u_1, u_2, \ldots$ be the successive ancestors of u_0, i.e., u_1 is the parent of u_0 (i.e., u_0 connects to u_1), u_2 is the parent of u_1, and so on. (Note that $u_1, u_2, \ldots$ are all null if u_0 itself is the root.) Define the complement value to the rank of u_i as $C_i := 1 - \text{rank}(u_i)$, $i \geq 0$. The main thrust of the proof is to show that the sequence C_i, $i \geq 0$, decreases geometrically w.h.p. We adapt a technique used in [20].

For $t \geq 0$, let I_t be the indicator random variable for the event that a root has not been reached after t jumps, i.e., $u_0, u_1, \ldots, u_t$ are not roots. We need the following lemma.

LEMMA 5. $\forall t \geq 1$ *and* $\forall z \in [0, 1]$, $E[C_{t+1} I_t | C_t I_{t-1} = z] \leq z/2$.

Proof. We can assume that $z \neq 0$; since $C_{t+1} \leq C_t$ and $I_t \leq I_{t-1}$, the lemma holds trivially if $z = 0$. Therefore, we have $I_{t-1} = 1$ and $C_t = z > 0$. We focus on the node u_t. Denote the set of neighbors of node u_t by U; the size of U is at most $n - 1$. Let Y be the random variable denoting the number of "unexplored" nodes in set U, i.e., those that do not belong to the set $\{u_0, u_1, \ldots, u_{t-1}\}$. If $Y = 0$, then u_t is a root and hence $C_{t+1} I_t = 0$. We will prove that $\forall d \geq 1$,

$$E[C_{t+1} I_t | ((C_t I_{t-1} = z) \wedge (Y = d))] \leq z/2. \tag{4.1}$$

Showing the above is enough to prove the lemma, because if the lemma holds conditional on all positive values of d, it also holds unconditionally. For convenience, we denote the left-hand side of (4.1) as Γ.

Fix some $d \geq 1$. In all arguments below, we condition on the event "$(C_t I_{t-1} = z) \wedge (Y = d)$." Let $v_1, v_2, \ldots, v_d$ denote the d unexplored nodes in U. If $\text{rank}(v_i) < \text{rank}(u_t)$ $\forall i$ $(1 \leq i \leq d)$, then u_t is a root and hence $C_{t+1} I_t = 0$. Therefore, conditioning on the value $y = \min_i C_i = \min_i (1 - \text{rank}(v_i)) \leq z$, and considering the d possible

values of i that achieve this minimum, we get

$$\Gamma = d \int_0^z y(1-y)^{d-1} dy.$$

Evaluating the above yields

$$\Gamma = \frac{1-(1-z)^d(1+zd)}{(d+1)}.$$

We can show that the right-hand side of the above is at most $z/2$ by a straightforward induction on d. □

Using Lemma 5, we now prove Theorem 11.

We have $E[C_1 I_0] \leq E[C_1] \leq 1$. Hence by Lemma 5 and an induction on t yields that $E[C_t I_{t-1}] \leq 2^{-t}$. In particular, letting $T = 3 \log n$, where c is some suitable constant, we get $E[C_T I_{T-1}] \leq n^{-3}$.

Now, suppose $u_T = u$ and that $C_T I_{T-1} = z$. The degree of node u is at most n; for each of these nodes v, $\Pr(\text{rank}(v) > \text{rank}(u)) = \Pr(1 - \text{rank}(v) < 1 - \text{rank}(u)) = \Pr(1 - \text{rank}(v) < z) = z$. Thus the probability that u is not a root is at most nz; more formally, $\forall z, \Pr(I_T = 1 | C_T I_{T-1} = z) \leq nz$. So,

$$\Pr(I_T = 1) \leq \log n E[C_T I_{T-1}] \leq n/n^3 = 1/n^2.$$

Hence, w.h.p., the number of hops from any fixed note to the root is $O(\log n)$. By the union bound, the statement holds for all nodes w.h.p. □

Similar to Theorem 1, we can bound the number of trees produced by the Local-DRR algorithm on an arbitrary sparse graph.

THEOREM 12. *Let G be an arbitrary connected undirected graph having n nodes. Let $d_i = O(n^{1/4-\epsilon})$ be the degree of node i, $1 \leq i \leq n$, where $\epsilon > 0$ is some fixed constant. Then the number of trees produced by the Local-DRR algorithm is $O(\sum_{i=1}^n \frac{1}{d_i+1})$ w.h.p. In particular, if $d_i = d$ $\forall i$, then the number of trees is $O(n/d)$ w.h.p.*

Proof. Let the indicator random variable X_i take the value of 1 if node i is a root and 0 otherwise. Let $X = \sum_{i=1}^n X_i$ be the total number of roots. $\Pr(X_i = 1) = 1/(d_i+1)$ since this is the probability its value is the highest among all its d_i neighbors. Hence, by linearity of expectation, the expected number of roots (hence, trees) is

$$E[X] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n \frac{1}{d_i+1} \geq n/d_{avg},$$

where d_{avg} is the average degree of the graph. To show concentration, we cannot directly use a standard Chernoff bound since X_i are not independent (connections are not independently chosen, but fixed by the underlying graph).

We use Azuma's inequality to show concentration. Let $X_1, \ldots, X_n$ be the ranks chosen by the nodes. Let $f = f(X_1, \ldots, X_n)$ be the number of roots. Set up a Doob martingale sequence: $Z_i = E[f | X_0, \ldots, X_i]$, $i = 0, 1, \ldots, n$. We have $Z_0 = E[f]$ and $Z_n = f$. We note that $|Z_i - Z_{i-1}| < d_i$, since changing the rank of the ith node can affect the status of only the ith node and its neighbors. Applying Azuma's inequality,

$$\Pr(|Z_n - Z_0| > t) \leq 2e^{-t^2/(2\sum_{i=1}^n d_i^2)}.$$

Setting $t = \Omega(n/d_{avg})$ and using the assumption that d_{avg} and d_i is $O(n^{1/4-\epsilon})$ gives the high probability result. □

We make two assumptions regarding the network communication model: (1) as mentioned earlier, a node can send a message simultaneously to all its neighbors (i.e., nodes that are connected directly by an edge) in the same round; (2) there is a routing protocol which allows any node to communicate with a *random* node in the network in $O(T)$ rounds and using $O(M)$ messages w.h.p. Assumption (1) is standard in distributed computing literature [5, 25]. As for Assumption (2), there are well-known techniques for sampling a random node in a network, e.g., using random walks (e.g., [32]) or using special properties of the underlying topology, e.g., as in P2P topologies such as Chord [15]. Under the above assumptions, we obtain the performance of DRR-gossip using the Local-DRR algorithm on sparse graphs in the following theorem.

THEOREM 13. *On a d-regular graph $G(V, E)$, where $|V| = n$ and $d = O(n/\log n)$, the time complexity of the DRR-gossip algorithms is $O(\log n + T \log \frac{n}{d})$ w.h.p. by using the Local-DRR algorithm and a routing protocol running in $O(T)$ rounds and $O(M)$ messages (w.h.p.) between a gossip pair; the corresponding message complexity is $O(|E| + \frac{n}{d} M \log \frac{n}{d})$ w.h.p.*

Proof. Phase 1 (Local-DRR) takes $O(1)$ time, since each node can find its largest ranked neighbor in constant time (assumption (1)) and needs $O(|E|)$ messages in total (since at most two messages travel through an edge). Phase 2 (Convergecast and Broadcast) takes $O(\log n)$ time (by Theorem 5 and assumption (1)) and $O(n)$ messages. Phase 3 (uniform gossip) takes $O(T \log \frac{n}{d})$ time (assumption (2)) and needs $O(\frac{n}{d} M \log \frac{n}{d})$ messages (assumption (2) and Theorem 12). □

We can apply the above theorem to Chord [31] and have the following theorem.

THEOREM 14. *In Chord, the DRR-gossip algorithms, with the Local-DRR algorithm in Phase 1, take $O(\log^2 n)$ rounds and $O(n \log n)$ messages for all nodes to reach consensus on aggregates w.h.p.*

Proof. Each node in Chord has a degree $d = O(\log n)$. Chord admits an efficient (nontrivial) protocol (cf. [15]) which satisfies assumption (2) with $T = O(\log n)$ and $M = O(\log n)$ (both in expectation, which is sufficient here). Hence Theorem 13 shows that DRR-gossip takes $O(\log^2 n)$ time and $O(n \log n)$ messages w.h.p. □

In contrast, the straightforward uniform gossip [14] gives $O(T \log n) = O(\log^2 n)$ rounds and $O(M \cdot n \log n) = O(n \log^2 n)$ messages w.h.p. for all nodes of Chord to reach consensus.

5. Lower bound for address-oblivious algorithms. We conclude by showing a nontrivial lower bound result on gossip-based aggregate computation: any address-oblivious algorithm for computing aggregates requires $\Omega(n \log n)$ messages, *regardless of the number of rounds or the size of the (individual) messages*. We assume the address-oblivious random phone call model: i.e., communication partners are chosen randomly (without depending on their addresses). As mentioned in section 1.1, we assume that a node's decision to communicate with its partner (which is chosen randomly) does not depend on the partner's address nor on the knowledge of the *addresses* of the partners seen in previous rounds. However, the decision to communicate can depend on the value(s) acquired by the node until the current point in the execution or other factors and thus each node may act differently in each round (e.g., some may always choose to communicate and other may remain silent for a long time.)

The following theorem gives a lower bound for computing the Max aggregate. The argument can be adapted for other aggregates as well.

THEOREM 15. *Any address-oblivious algorithm that computes the maximum value, Max, in an n-node network needs $\Omega(n \log n)$ messages w.h.p. (regardless of the number of rounds).*

Proof. We lower bound the number of messages exchanged between nodes before a large fraction of the nodes correctly knows the (correct) maximum value. Without loss of generality, we assume that nodes can send messages that are arbitrarily long since a restricted message length will only increase the number of messages that need to be sent. (The bound will hold regardless of this assumption.) Further, without loss of generality, we assume that a node can send a list of all node addresses and the corresponding node values learned so far (without any aggregation). For any node i to have correct knowledge of the maximum, it should somehow know the values at all other nodes. (Otherwise, an adversary—who knows the random choices made by the algorithm—can always make sure that the maximum is at a node which is not known by i.) There are two ways that i can learn about another node j's value: (1) directly: i gets to know j's value by communicating with j directly (at the beginning, each node knows only about its own value); and (2) indirectly: i gets to know j's value by communicating with a node $w \neq j$ which has a knowledge of j's value. Note that w itself may have learned about j's value either directly or indirectly.

Let v_i be the (initial) value associated with node i, $1 \leq i \leq n$. We will assume that all values are *distinct*. By the adversary argument, the requirement is that at the end of any algorithm, on the average, at least half the nodes should know (in the above direct or indirect way) all the v_i, $1 \leq i \leq n$. Otherwise, the adversary can make any value that is not known to more than half the nodes, the maximum. We want to show that the number of messages needed to satisfy the above requirement is at least $cn \log n$ for some (small) constant $c > 0$. In fact, we show something stronger: at least $cn \log n$ (for some small $c > 0$) messages are needed if we require even $n^{\Omega(1)}$ values to be known to at least $\Omega(n)$ nodes.

We define a stage (consisting of one or more rounds) as follows. Stage 1 starts with round 1. If stage t ends in round j, then stage $t+1$ starts in round $j+1$. Thus, it remains to describe when a stage ends. We distinguish sparse and dense stages. A sparse stage contains at most ϵn messages (for a suitably chosen small constant $\epsilon > 0$, fixed later in the proof). The length of these stages is maximized, i.e., a sparse stage ends in a round j if adding round $j+1$ to the stage would result in more than ϵn messages. A dense stage consists of only one round containing more than ϵn messages. Observe that the number of messages during the stages 0 to j is at least $(j-1)\epsilon n/2$ because any pair of consecutive stages contains at least ϵn messages by construction. (We note that this type of separation into dense and spare stages is also done in Karp et al. [12] for their lower bound of rumor spreading in their model.)

Let $S_i(t)$ be the set of *nodes* that know v_i at the beginning of stage t. At the beginning of stage 1, $|S_i(1)| = 1$ $\forall 1 \leq i \leq n$.

At the beginning of stage t, we call a value *typical* if it is known by at most $6^t \log n$ nodes (i.e., $|S_i(t)| \leq 6^t \log n$) *and* it was typical at the beginning of all stages prior to t. All values are typical at the beginning of stage 1. Let k_t denote the number of typical values at the beginning of stage t.

The proof of the theorem follows from the following claim. (Constants specified will be fixed in the proof; we do not try to optimize these values.)

Claim. At the beginning of stage t, at least $(1/6)^t n$ values are typical w.h.p. $\forall t \leq \delta \log n$ for a fixed positive constant δ.

The above claim will imply the theorem since at the end of stage $t = \delta \log n$, $|S_i(t)| \leq o(n)$ for at least $n^{\Omega(1)}$ values, i.e., at least $n^{\Omega(1)}$ values are not yet known to $1 - o(1)$ fraction of the nodes after stage $t = \delta \log n$. (The adversary can make any of these $n^{\Omega(1)}$ values the maximum to ensure that any algorithm fails.) Hence the number of messages needed is at least $\Omega(n \log n)$.

We prove the above claim by induction: We show that if the claim holds at the beginning of a stage, then it holds at the end of the stage. We show this whether the stage is dense or sparse, and thus we have two cases.

Case 1. The stage is dense. A dense stage consists of only one round with at least ϵn messages. Fix a typical value v_i. Let $U_i(t) = V - S_i(t)$, i.e., the set of nodes that do not know v_i at the beginning of stage t. For $1 \leq k(i) \leq |U_i(t)|$, let $x_{k(i)}$ denote the indicator random variable that denotes whether the $k(i)$th of these nodes gets to know the value v_i in this stage. Let $X_i(t) = \sum_{k(i)=1}^{|U_i(t)|} x_{k(i)}$. Let u be a node that does not know v_i. u can get to know v_i either by calling a node that knows the value or being called by a node that knows the value. The probability it gets to know v_i by calling is at most $6^t \log n/n$ and the probability that it gets called by a node knowing the value is at most $6^t \log n/n$ (this quantity is $o(1)$, since $t \leq \delta \log n$ and δ is sufficiently small). Hence the total probability that it gets to know v_i is at most $2 \cdot 6^t \log n/n$. Thus, the expected number of nodes that get to know v_i in this stage is $E[X_i(t)] = \sum_{k(i)=1}^{|U_i(t)|} \Pr\{x_{k(i)} = 1\} \leq 2 \cdot 6^t \log n$. The variables $x_{k(i)}$ are not independent but are negatively correlated in the sense of Lemma 1 (the variant of the Chernoff bound described in section 1.4) by which we have

$$\Pr(X_i(t) > 5 \cdot 6^t \log n) = \Pr(X_i(t) > (1 + 3/2) \cdot 2 \cdot 6^t \log n) \leq 1/n^2.$$

By the union bound, w.h.p., at most $5 \cdot 6^t$ new nodes get to know each typical value. Thus, w.h.p. the total number of nodes knowing a typical value (for every such value) in this stage is at most $6^t \log n + 5 \cdot 6^t \log n = 6^{t+1} \log n$, thus satisfying the induction hypothesis. It also follows that a typical value at the beginning of a dense phase remains typical at the end of the phase, i.e., $k_{t+1} = k_t$ w.h.p.

Case 2. The stage is sparse. By definition, there are at most ϵn messages in a sparse stage. Each of these messages can be a push or a pull. (We recall that a push message is one that is sent by a calling node to the called node and a pull message is the message obtained by the calling node from the called node.) A sparse stage may consist of multiple rounds.

Fix a typical value v_i. W.h.p, there are at most $6^t \log n$ nodes that know this typical value at the beginning of this stage. Using pull messages, since the origin is chosen uniformly at random, the probability that one of these nodes is contacted is at most $1/n(\epsilon n) = \epsilon$. Hence the expected number of messages sent by nodes knowing v_i is at most $\epsilon 6^t \log n$. Let $Y_i(t)$ be the random variable that corresponds to the number of new nodes that get to know v_i (using pull messages). Thus $E[Y_i(t)]$ is at most $\epsilon 6^t \log n$. The high probability bound can be shown as earlier by using Lemma 1 (the variant of the Chernoff bound),

$$\Pr(Y_i(t) > 3\epsilon \cdot 6^t \log n) = \Pr(X_i(t) > (1 + 2) \cdot 6^t \log n) \leq 1/n^{1.5},$$

where we now set $\epsilon = 1/6$. By the union bound, w.h.p., no typical value is known to more than $0.5 \cdot 6^t \log n$ nodes using pull messages.

We next consider the effect of push messages. We focus on values that are typical at the beginning of this stage. We show that w.h.p. at least a constant fraction of the typical values remain typical at the end of this phase. As defined earlier, let k_t be the number of such typical values. In this stage, at most ϵn nodes are involved in pushing—let this set be Q. Consider a random typical value x. Since a typical value is known by at most $6^t \log n$ nodes and destinations are uniformly randomly chosen, the probability that x is known to a node in Q is at most $\frac{6^t \log n}{n}$. Hence

the expected number of times that x will be pushed by set Q is at most $\epsilon 6^t \log n$. Using a Chernoff bound, as above, it follows that w.h.p., x is pushed no more than $3\epsilon 6^t \log n = 0.5 \cdot 6^t \log n$ times (since $\epsilon = 1/6$). Since pulling only results in at most $0.56^t \log n$ new nodes knowing x (as shown in the above paragraph), the total number of new nodes that get to know x is at most $6^t \log n$ w.h.p. (via both pulling and pushing). However, for x to become nontypical, it should be known to at least $5 \cdot 6^t \log n$ new nodes in this stage. Thus, w.h.p., a random typical value remains typical at the end of the phase. Hence, w.h.p., at least some constant fraction of the nodes (say, at least 1/6) remain typical at the end of this phase. □

6. Concluding remarks. We presented an almost-optimal gossip-based protocol for computing aggregates that takes $O(n \log \log n)$ messages and $O(\log n)$ rounds. We also showed how our protocol can be applied to improve performance in networks with a fixed underlying topology. The main technical ingredient of our approach is a simple distributed randomized procedure called DRR to partition a network into trees of small size. The improved bounds come at the cost of sacrificing address-obliviousness. However, as we show in our lower bound, this is necessary if we need to break the $\Omega(n \log n)$ message barrier. This is also the first nontrivial lower bound on the message complexity of aggregate computation. Our bounds raise other interesting open questions: (1) Are $\Omega(n \log \log n)$ messages a lower bound for gossip-based aggregate computation in the non-address-oblivious model? (2) Is $\Omega(n \log n)$ the tight threshold for address-oblivious protocols? and (3) What is the exact threshold for non-address-oblivious protocols?

Our lower bound assumed a restricted address-oblivious model, i.e., a node's decision to communicate with its partner does not depend on the partner's address nor on the knowledge of the *addresses* of the partners seen in previous rounds. It would be interesting to prove a lower bound (or explore algorithms with better message complexity than $\Omega(n \log n)$) without this restriction.

Another question concerns our failure model, which is the same as the one assumed by previous work [13, 14]. The paper by Karp et al. [12] presents protocols that are resilient to a constant fraction of the participating nodes crashing during the execution. By contrast, the current paper assumes a simpler crash model: nodes may crash initially, and some constant fraction of the messages may be dropped. We note that the work of Karp et al. tackles rumor spreading, a simpler problem. However, it will be nice to duplicate our results for the stronger crash model. It may be possible to adapt DRR to work in this model.

Another interesting direction is to see whether the DRR technique can be used to obtain improved bounds for other distributed computing problems.

REFERENCES

[1] D. Alistarh, S. Gilbert, R. Guerraoui, and M. Zadimoghaddam, *How efficient can gossip be? (on the cost of resilient information exchange)*, in Proceedings of the 37th International Colloquium on Automata, Languages and Programming, vol. 2, 2010, pp. 115–126.

[2] P. Berenbrink, J. Czyzowicz, R. Elsässer, and L. Gasieniec, *Efficient information exchange in the random phone-call model*, in Proceedings of the 37th International Colloquium on Automata, Languages and Programming, vol. 2, (2010), pp. 127–138.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, *Randomized gossip algorithms*, IEEE Trans. Inform. Theory, 52 (2006), pp. 2508–2530.

[4] J.-Y. Chen and J. Hu, *Analysis of distributed random grouping for aggregate computation on wireless sensor networks with randomly changing graphs*, IEEE Trans. Parallel Distributed Syst., 19 (2008), pp. 1136–1149.

[5] J.-Y. Chen, G. Pandurangan, and D. Xu, *Robust aggregate computation in wireless sensor network: Distributed randomized algorithms and analysis*, IEEE Trans. Parallel Distributed Syst., 17 (2006), pp. 987–1000.
[6] 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 Proceedings of the ACM Symposium on Principles of Distributed Computing, 1987, pp. 1–12.
[7] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, *Geographic gossip: Efficient aggregation for sensor networks*, in Proceedings of the ACM-IEEE International Conference on Information Processing in Sensor Networks, 2006, pp. 69–76.
[8] J. Gao, L. Guibas, N. Milosavljevic, and J. Hershberger, *Sparse data aggregation in sensor networks*, in Proceedings of the ACM-IEEE International Conference on Information Processing in Sensor Networks, 2007, pp. 430–439.
[9] S. Gilbert and D. R. Kowalski, *Distributed agreement with optimal communication complexity*, in Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms, Austin, TX, 2010, pp. 965–977.
[10] G. H. Hardy, J. E. Littlewood, and G. Polya, *Inequalities*, Cambridge University Press, Cambridge, UK, 1952.
[11] M. Jelasity, A. Montresor, and O. Babaoglu, *Gossip-based aggregation in large dynamic networks*, ACM Trans. Comput. Syst., 23 (2005), pp. 219–252.
[12] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking, *Randomized rumor spreading*, in Proceedings of the IEEE Annual Symposium on Foundations of Computer Science, 2000, pp. 565–574.
[13] S. Kashyap, S. Deb, K. V. M. Naidu, R. Rastogi, and A. Srinivasan, *Efficient gossip-based aggregate computation*, in Proceedings on Principles of Database Systems, 2006, pp. 308–317.
[14] D. Kempe, A. Dobra, and J. Gehrke, *Gossip-based computation of aggregate information*, in Proceedings of the IEEE Annual Symposium on Foundations of Computer Science, 2003, pp. 482–491.
[15] V. King, S. Lewis, J. Saia, and M. Young, *Choosing a random peer in chord*, Algorithmica, 49 (2007), pp. 147–169.
[16] B. Krishnamachari, D. Estrin, and S. B. Wicker, *The impact of data aggregation in wireless sensor networks*, in International Workshop on Distributed Event-Based Systems, 2002, pp. 575–578.
[17] P. Kyasanur, R. R. Choudhury, and I. Gupta, *Smart gossip: An adaptive gossip-based broadcasting service for sensor networks*, in Proceedings of IEEE Conference on Mobile Ad-Hoc and Sensor Systems, 2006, pp. 91–100.
[18] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, *Tag: A tiny aggregation service for ad-hoc sensor networks*, SIGOPS Oper. Syst. Rev., 36 (2002), pp. 131–146.
[19] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, Cambridge, UK, 2005.
[20] R. Morselli, B. Bhattacharjee, M. A. Marsh, and A. Srinivasan, *Efficient lookup on unstructured topologies*, in Proceedings of the ACM Symposium on Principles of Distributed Computing, 2005, pp. 77–86.
[21] D. Mosk-Aoyama and D. Shah, *Computing separable functions via gossip*, in Proceedings of the ACM Symposium on Principles of Distributed Computing, 2006, pp. 113–122.
[22] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.
[23] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, *Synopsis diffusion for robust aggregation in sensor networks*, in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp. 250–262.
[24] A. Panconesi and A. Srinivasan, *Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds*, SIAM J. Comput., 26 (1997), pp. 350–368.
[25] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, Philadelphia, 2000.
[26] R. Di Pietro and P. Michiardi, *Brief announcement: Gossip-based aggregate computation: Computing faster with non address-oblivious schemes*, in Proceedings of the ACM Symposium on Principles of Distributed Computing, 2008, p. 442; also available online from http://www.eurecom.fr/∼michiard/downloads/podc08_a_ext.pdf.
[27] B. Pittel, *On spreading a rumor*, SIAM J. Appl. Math., 47 (1987), pp. 213–223.
[28] A. I. T. Rowstron and P. Druschel, *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems*, in Proceedings of Middleware, 2001, pp. 329–350.

[29] R. Sarkar, X. Zhu, and J. Gao, *Hierarchical spatial gossip for multi-resolution representation in sensor network*, in Proceedings of the International Conference on Information Processing in Sensor Networks, 2007, pp. 420–429.

[30] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, *Medians and beyond: New aggregation techniques for sensor networks*, in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp. 239–249.

[31] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, in Proceedings of the Annual Conference of the Special Interest Group on Data Communication, 2001, pp. 149–160.

[32] M. Zhong and K. Shen, *Random walk based node sampling in self-organizing networks*, SIGOPS Oper. Syst. Rev., 40 (2006), pp. 49–55.