

# Almost-Optimal Gossip-Based Aggregate Computation

Jen-Yeu Chen\*

Gopal Pandurangan†

## Abstract

Motivated by applications to modern networking technologies, there has been tremendous 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 efficient gossip-based algorithms for aggregate computation that are both time optimal and almost message-optimal. Given a  $n$ -node network, our algorithms guarantee that all the nodes can compute the common aggregates (such as Min, Max, Count, Sum, Average, Rank etc.) 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 et al. [8] that is time-optimal, but uses  $O(n \log n)$  messages as well as on the algorithm of Kashyap et al. [7] that uses  $O(n \log \log n)$  messages, but is not time-optimal (takes  $O(\log n \log \log n)$  time). Our algorithm uses a technique called as *distributed random ranking* that 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.

---

\*Department of Electrical Engineering, National DongHwa University, Hualien 97401, Taiwan. E-mail: jenyue@ieee.org

†Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA. E-mail: gopal@cs.purdue.edu

# 1 Introduction

## 1.1 Background and Previous Work

Aggregate statistics (e.g., Average, Max/Min, Sum, and, Count etc.) are significantly useful for many applications in networks [2, 5, 8, 9, 11, 18]. 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 nodes is an important statistic needed by system designers for optimizing overall performance [16, 19]. 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 [1, 2, 4, 7, 8, 10, 12, 14, 17] 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 [6, 7]. We refer to [7, 8, 6] 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 gossip focused on using randomized communication for rumor propagation [3, 15, 6]. In particular, Karp et al. [6] 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 [6]. 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. Karp et al.'s algorithm 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 et al. [8] 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 of 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. [7] 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 over the message complexity of the protocol of Kempe et al. 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 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, we refer to [7, 2] and the references therein.

## 1.2 Our Results

In this paper, we present efficient gossip-based algorithms for computing various aggregate functions that improves upon previous results. Given a  $n$ -node network, our algorithms guarantee that all the nodes can compute the common aggregates (such as Min, Max, Count, Sum, Average, Rank etc.) 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 et al. [8] that is time-optimal, but uses  $O(n \log n)$  messages as well as on the algorithm of Kashyap et al. [7] 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 as *distributed random ranking (DRR)* that gives an efficient distributed protocol to partition the 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 [7]. 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 and the message complexity is within a factor  $o(\log \log n)$  of the optimal. This is because, Karp et al [6] 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.)

Our second contribution is 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 (such as ours) are needed to get a significant improvement in message complexity.

Our algorithm, henceforth called as *DRR-gossip*, proceeds in phases. In phase one, every node runs the DRR scheme to construct a forest of (disjoint) trees. Within each tree, in phase two, the local aggregate (e.g., sum or maximum) of the tree is computed by a Convergecast process and obtained at the root of the tree. Finally in phase three, all the roots utilize a suitably modified version of the uniform gossip algorithm of Kempe et al. [8] to obtain the global aggregate. Finally, if necessary, the roots forward the global aggregate along the tree to other nodes in their trees.

## 1.3 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 whole DRR-gossip algorithm is summarized in Section 6. In Section 7, we establish a lower bound for the address oblivious model. We conclude in Section 8.

## 2 Model

The network consists of a set  $V$  of  $n$  nodes; each node  $i \in V$  has a data value denoted by  $v_i$ . The goal is to compute aggregate functions such as Min, Max, Sum, Average etc., of the node values.

The nodes communicate in discrete time-steps referred to as *rounds*. As in prior work on this problem [6, 7], we assume that communication 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. A node  $i$  is said to *call* a node  $j$  if  $i$  chooses  $j$  as a communication partner. (This is known as the *random phone call* model [6].) Once a call is established, we assume that information can be exchanged in both directions along the link. In one round, a node can call only *one* other node. (If multiple nodes attempt to communicate with a node, then the connection is queued up.) We assume that nodes have unique addresses. 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

---

**Algorithm 1:**  $\mathbb{F} = \text{DRR}(G)$ 

---

```
foreach node  $i \in V$  do
  choose  $\text{rank}(i)$  independently and uniformly at random from  $[0, 1]$  ;
  set  $\text{found} = \text{FALSE}$  // higher ranked node not yet found ;
  set  $\text{parent}(i) = \text{NULL}$  // 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  $\text{rank}(u) > \text{rank}(i)$  then
      set  $\text{parent}(i) = u$ ;
      set  $\text{found} = \text{TRUE}$ ;
      set  $k = k + 1$ ;
    end
  until  $\text{found} == \text{TRUE}$  or  $k < \log n - 1$  ;
  if  $\text{found} == \text{TRUE}$  then
    send a connection message including its identifier,  $i$ , to its parent node  $\text{parent}(i)$ ;
  end
  Collect the connection messages and accordingly construct the set of its children nodes,  $\text{Child}(i)$ ;
  if  $\text{Child}(i) = \emptyset$  then
    become a leaf node;
  else
    become an intermediate node;
  end
end
```

---

the above assumptions are also used in prior work [8, 7]. Our algorithm can tolerate the following types of failures (similar to the algorithms of [8, 7]). In particular, we assume 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  $\delta$ . Without loss of generality,  $1/\log n < \delta < 1/8$ : Larger values of  $\delta$ , requires only  $O(1/\log(1/\delta))$  repeated calls to bring down the call probability below  $1/8$ , and smaller values only make it easier to prove our claims.

Throughout the paper, “with high probability (whp)” means “with probability at least  $1 - 1/n^\alpha$ , for some  $\alpha > 0$ ”.

### 3 Phase I: Distributed Random Ranking

#### 3.1 The DRR algorithm

The DRR algorithm is as follows. 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  $[1, n^3]$ ; 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) till it finds a node of higher rank to connect to. If none of the  $\log n - 1$  sampled nodes have higher rank then  $i$  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  $\mathbb{F}$ . Algorithm 1 gives the pseudo-code of the DRR algorithm.

We next analyze 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.

### 3.2 Number of Trees

**Theorem 1** *The number of trees produced by the DRR algorithm is  $O(n/\log n)$  whp.*

*Proof:* The probability that a node  $i$  becomes a root is  $\Theta(1/\log n)$ , because this is the probability that its rank is highest among all the  $\log n - 1$  nodes probed. Hence, by linearity of expectation, the expected number of roots (and thus, trees) is  $O(n/\log n)$ .

To show tight concentration, we use the following argument. Assume that ranks have already been assigned to the nodes. Number the nodes according to the order statistic of their ranks: the  $i$ th node is the node with the  $i$ th smallest rank. Let the indicator random variable  $X_i$  take the value of 1 if the  $i$ th smallest node is a root and 0 otherwise. (Note that probability that  $X_n = 1$  is 1 and the probability that  $X_0 = 1$  is 0.) Let  $X = \sum_{i=1}^n X_i$  be the total number of roots. As shown above,  $E[X] = O(n/\log n)$ . Clearly,  $X_i$ s are independent random variables, since the probability that the  $i$ th smallest ranked node becomes the root depends only on the  $\log n - 1$  nodes that it samples and independent of the samples of the rest of the nodes. Applying a Chernoff's bound [13] we have:

$$\Pr(X > 6E[X]) \leq 2^{E[X]} = o(1/n).$$

■

### 3.3 Tree Size

**Theorem 2** *The number of nodes in every tree produced by the DRR algorithm is at most  $O(\log n)$  whp.*

*Proof:* We bound that the probability that a tree of size  $O(\log n)$  is produced by the DRR algorithm. Fix a set  $S$  of  $k = c \log n$  nodes, for some sufficiently large positive constant  $c$ . We first compute the probability that this set of  $k$  nodes form a tree. For the sake of analysis, we will direct tree edges as follows: a tree edge  $(i, j)$  is directed from node  $i$  to node  $j$  if  $\text{rank}(j) < \text{rank}(i)$ , i.e.  $i$  connects to  $j$ . W.l.o.g., fix a permutation of  $S$ :  $(s_1, \dots, s_\alpha, \dots, s_\beta, \dots, s_k)$  where  $\text{rank}(s_\alpha) > \text{rank}(s_\beta)$ ,  $1 \leq \alpha < \beta \leq k$ . This permutation naturally induces a spanning directed tree on  $S$  in the following fashion:  $s_1$  is the root and any other node  $s_\alpha$  ( $1 < \alpha \leq k$ ) connects to a node in the totally strict ordered set  $\{s_1, \dots, s_{\alpha-1}\}$  (as fixed by the above permutation). For convenience, we denote the event that a node  $s$  connects to any node on a directed tree,  $T$ , as  $s \rightarrow T$ . Note that  $s \rightarrow T$  implies that  $s$ 's rank is less than that of any node on the tree  $T$ . Also, we denote a directed spanning tree induced on the strictly totally ordered set  $\{s_1, s_2, \dots, s_\alpha, \dots, s_h\}$  as  $T_h$  where a node  $s_\alpha$  can only connects to its preceding nodes in the ordered set. As a special case,  $T_1$  is the induced directed tree containing only the root node  $s_1$ . In the following, we bound the probability of  $T_k$  happening:

$$\begin{aligned} \Pr(T_k) &= \Pr(T_1 \cap (s_2 \rightarrow T_1) \cap (s_3 \rightarrow T_2) \cap \dots \cap (s_k \rightarrow T_{k-1})) \\ &= \Pr(T_1) \Pr(s_2 \rightarrow T_1 | T_1) \Pr(s_3 \rightarrow T_2 | T_2) \dots \Pr(s_k \rightarrow T_{k-1} | T_{k-1}). \end{aligned} \quad (1)$$

To bound each of the terms in the product, we use the principle of deferred decisions: 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.

$$\begin{aligned}
& \Pr(s_\alpha \rightarrow T_{\alpha-1} | T_{\alpha-1}) \\
& \leq \int_0^1 \int_0^{r_1} \int_0^{r_2} \cdots \int_0^{r_{\alpha-1}} \sum_{h=0}^{\log n - 1} \binom{\alpha - 1}{n} r_\alpha^h dr_\alpha \cdots dr_1 \\
& = \frac{\alpha - 1}{n} \int_0^1 \int_0^{r_1} \int_0^{r_2} \cdots \int_0^{r_{\alpha-1}} [1 + r_\alpha + r_\alpha^2 + \cdots + r_\alpha^{\log n - 1}] dr_\alpha \cdots dr_1 \\
& = \frac{\alpha - 1}{n} \left( \frac{0!}{\alpha!} + \frac{1!}{(\alpha + 1)!} + \frac{2!}{(\alpha + 2)!} + \cdots + \frac{(\log n)!}{(\log n + \alpha)!} \right),
\end{aligned}$$

where  $r_q = \text{rank}(s_q)$  is the rank of node  $s_q$ ,  $1 \leq q \leq \alpha$ . The above expression is bounded by  $\frac{a}{n}$ , where  $0 < a < 1$  if  $\alpha > 2$  and  $0 < a \leq (1 - \frac{1}{\log n + 2})$  if  $\alpha = 2$ . Besides,  $\Pr(T_1) \leq \frac{1}{\log n}$  (cf. Theorem 1); hence, the equation (1) is bounded by  $(\frac{a}{n})^{k-1} \frac{1}{\log n}$ .

Using the above, the probability that a tree of size  $k = c \log n$  is produced by the DRR algorithm is bounded by  $\binom{n}{k} k! (\frac{a}{n})^{k-1} \frac{1}{\log n} \leq \frac{(ne)^k}{k^k} O(\sqrt{k}) \frac{k^k}{e^k} (\frac{a}{n})^{k-1} \frac{1}{\log n} \leq \frac{c' \cdot n}{\log^{\frac{1}{2}} n} \cdot a^{k-1} = o(1/n)$ , if  $c$  sufficiently large.  $\blacksquare$

### 3.4 Complexity of the DRR algorithm

**Theorem 3** *The message complexity of the DRR algorithm is  $O(n \log \log n)$  whp. 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 (whp, all the nodes sampled will be unique). Hence the expected number of nodes probed is

$$\sum_{k=1}^d \Theta(k \frac{1}{k+1} \frac{1}{k}) = O(\log d).$$

Hence the number of messages exchanged by node  $i$  is  $O(\log d)$ . By linearity of expectation, the 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 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  $i$ th node is the node with the  $i$ th smallest rank. Let the indicator r.v.  $Z_{ik}$  ( $1 \leq i \leq n$ ,  $1 \leq k \leq d$ ) indicate whether the  $k$ th sample by the  $i$ th *smallest ranked* node succeeded or not (i.e., it found a higher ranked node). If it succeeded then  $Z_{ij} = 1$  for all  $j \leq k$  and  $Z_{ij} = 0$  for all  $j > k$ . Thus  $X = \sum_{i=1}^n \sum_{k=1}^d Z_{ik}$ . Then the sequence  $X_0 = E[X]$ ,  $X_1 = E[X|Z_{11}]$ ,  $\dots$ ,  $X_{nd} = E[X|Z_{11}, \dots, Z_{nd}]$  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  $\epsilon$  we have:

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

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

## 4 Phase II: Convergecast and Broadcast

In the second phase of our algorithm, the local aggregate of each tree will be 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 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 then can obtain the local max/min value of its tree. Provided in the Appendix A, Algorithm 7 and Algorithm 8 are the pseudo-codes of the Convergecast-max algorithm and the Convergecast-sum algorithm, respectively. In the pseudo-codes, the input  $\mathbf{v} \in M_{n,1}$ , where  $M_{x,y}$  is the  $x \times y$  matrix, representing the value vector over all nodes in  $V$ ; the output  $\mathbf{cov}_{max} \in M_{m,1}$  and  $\mathbf{cov}_{sum} \in M_{m,2}$  are the computed aggregates at root nodes, where  $m$  is the number of root nodes.

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 non-root nodes know the identity (address) of their respective roots.

### Complexity of Phase II

Every node except the root nodes needs to send a message to its parent in the upward aggregation process of the Convergecast algorithms. So 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 the 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, as the number of roots is at most  $O(n/\log n)$  by Theorem 1, the message complexity for broadcast is also  $O(n)$ .

## 5 Phase III: Gossip

In the third phase, all roots of the trees then compute the global aggregate by performing the uniform gossip algorithm on the graph  $\tilde{G} = \text{clique}(\tilde{V})$ , where  $\tilde{V} \subseteq V$  is the set of roots and  $|\tilde{V}| = m$ .

The idea of uniform gossip is as follows. Every root independently and uniform at random selects an node to send its message. If the selected node is another root then the task is completed. If not, then the selected node needs to forward the received message to its root. (Thus, to traverse through an edge of  $\tilde{G}$ , a message needs at most two hops of  $G$ .)

The algorithm 2, Gossip-max, and the algorithm 4, Gossip-ave, a modification from the Push-Sum algorithm of [8, 7], compute the *Max* and *Ave* aggregates respectively (other aggregates such as Min, Sum etc. can be calculated by a suitable modification). Note that there is no sampling procedure in the Gossip-ave algorithm. The algorithm 3, Data-spread, a modification of Gossip-max, is used by a root node to spread its value. If a root needs to spread a particular value over the network, it sets the particular value as its initial value of the Gossip-max algorithm and all the other roots set their initial value to minus infinity.

### 5.1 Performance of the Gossip-max and Data-spread Algorithms

Let  $m$  denote the number of the root nodes. By Theorem 1, we have  $m = |\tilde{V}| = O(n/\log n)$  where  $n = |V|$ . Karp, et al. [6], have proved that all the  $m$  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, the Gossip-max algorithm needs  $O(m \log m) = O(n)$  messages for all the roots to obtain *Max* if the selection probability is uniformly  $1/m$ . However, in the implementation of the Gossip-max algorithm on the ranking tree forest, the root of a tree is selected with *the probability proportional to the size of its ranking tree*. The uniformity of the selection probability does not hold here, although the fluctuation of the tree size may be small. 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 gossip

---

**Algorithm 2:**  $\hat{\mathbf{x}}_{max} = \text{Gossip-max}(G, \mathbb{F}, \tilde{V}, \mathbf{y})$ 

---

**Initialization:** every root  $i \in \tilde{V}$  is of 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  $Ave$ ,  $x_{0,i} = y(i) = \mathbf{cov}_{sum}(i, 2)$ . \*/

**gossip procedure:**

**for**  $t=1 : O(\log n)$  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 amid all received values and its own.

**end**

**sampling procedure:**

**for**  $t=1 : \frac{1}{c} \log n$  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**

---

---

**Algorithm 3:**  $\hat{\mathbf{x}}_{ru} = \text{Data-spread}(G, \mathbb{F}, \tilde{V}, x_{ru})$ 

---

**Initialization:** A root node  $i \in \tilde{V}$  who intends to spread its value  $x_{ru}$ ,  $|x_{ru}| < \infty$  setups  $x_{0,i} = x_{ru}$ .

All the other nodes  $j$  setup  $x_{0,j} = -\infty$ .

Run gossip-max( $G, \mathbb{F}, \tilde{V}, \mathbf{x}_0$ ) by the initialized values.

---

---

**Algorithm 4:**  $\hat{\mathbf{x}}_{ave} = \text{Gossip-ave}(G, \mathbb{F}, \tilde{V}, \mathbf{cov}_{sum})$ 

---

**Initialization:** Every root  $i \in \tilde{V}$  setups 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**  $t = 1 : O(\log m + \log(1/\epsilon))$  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 ranking 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 global average by  $\hat{\mathbf{x}}_{ave,t}(i) = \hat{x}_{ave,t,i} = s_{t,i}/g_{t,i}$ .

**end**

---

procedure, roots can sample a  $O(\log n)$  number of other roots to confirm and update, if necessary, their values and reach the consensus on  $Max$ .

### Gossip Procedure

We have the following theorem for the Gossip procedure.

**Theorem 4** *Running the Gossip-max algorithm on  $\tilde{G} = \text{clique}(\tilde{V})$  of  $G(V, E)$ , after the gossip procedure, whp, at least  $\Omega(\frac{cn}{\log n})$  root nodes already have the global maximum,  $Max$ , where  $n = |V|$  and  $0 < c < 1$ .*

We refer to the Appendix for the proof.

### Sampling Procedure

From Theorem 4, after the gossip procedure, there are  $\Omega(\frac{cn}{\log n}) = \Omega(cm)$ ,  $0 < c < 1$  nodes with the  $Max$ . For all the roots to reach the consensus on  $Max$ , all the roots then sample each other as in the sampling procedure. It is possible that the root of a larger ranking tree will be sampled more frequently than the roots of smaller trees. However, this non-uniformity is an advantage, since the roots of larger ranking trees could have obtained the  $Max$  in the gossip procedure with higher probability due to this same non-uniformity. In the sampling procedure, a root without  $Max$  can obtain  $Max$  with higher probability by this non-uniform sampling.

**Theorem 5** *Running the Gossip-max algorithm on the overlay complete graph  $\tilde{G} = \text{clique}(\tilde{V})$  of  $G(V, E)$ , after the sampling procedure, whp, all the roots will know the  $Max$ .*

*Proof:* After the sampling procedure, the probability that none of the max-roots being sampled by a non-max-root is at most  $(\frac{m-cm}{m})^{\frac{1}{c} \log n} < \frac{1}{n}$ . Thus, after the sampling procedure, with probability at least  $1 - \frac{1}{n}$ , all the roots will know the  $Max$ . ■

### 5.1.1 Complexity of the Gossip-max and the Data-spread algorithms

The gossip procedure takes  $O(\log n)$  rounds and  $O(m \log n) = O(\frac{n}{\log n} \log n) = O(n)$  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, this phase totally takes  $O(\log n)$  rounds and  $O(n)$  messages for all the roots in a network to reach consensus on  $Max$ .

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

### 5.2 Performance of the Gossip-ave Algorithm

In the case that the uniformity of gossip is held, it has been shown in [8] that on an  $m$ -clique with probability at least  $1 - \delta'$ , Gossip-ave (uniform push-sum in [8]) 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 for all the  $m$  nodes to reach consensus on the global average within a relative error at most  $\epsilon$ . When the uniformity does not hold, the performance of uniform gossip will depend on the distribution of the selection probability. It has been shown in [7] that in their efficient gossip algorithm the node being selected with the largest probability will have the global average,  $Ave$ , in  $O(\log m + \log \frac{1}{\epsilon})$  rounds. In this paper, we prove that the same upper bound holds for our Gossip-ave algorithm, namely, the root of the largest tree will have  $Ave$  after  $O(\log m + \log \frac{1}{\epsilon})$  rounds of the gossip procedure of the Gossip-ave algorithm. Note that unlike the Gossip-max algorithm, the Gossip-ave algorithm has no sampling procedure. In this bound,  $m = O(n/\log n)$  is the number of roots obtained from the DRR algorithm and the relative error  $\epsilon = n^{-\alpha}$ ,  $\alpha > 0$ . Thus, the root of the largest tree will have  $Ave$  after  $O(\log m + \log \frac{1}{\epsilon}) = O(\log n)$  rounds of the gossip procedure. The upper bound of the running time of the Gossip-ave algorithm is in the following theorem.

**Theorem 6** *Whp, 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 aggregate on the root of the largest ranking 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 global  $Ave$  is  $x_{ave} = \sum_j x_j$ , when all  $x_j$  have the same sign. (We can always offset the values to have the same sign.)*

We refer to the Appendix for the proof.

### Complexity of Gossip-ave

The Gossip-ave algorithm needs  $O(\log m + \log \frac{1}{\epsilon}) = O(\log n)$  rounds and  $m \cdot O(\log n) = O(n)$  messages for the root of the largest tree to have the global average aggregate,  $Ave$ , within a relative error at most  $\frac{2}{n^{\alpha}-1}$ ,  $\alpha > 0$ .

## 6 DRR-gossip Algorithms

Putting together our results from the previous sections, we present Algorithm 5, the DRR-gossip-max algorithm, and Algorithm 6, the DRR-gossip-ave algorithm, for computing  $Max$  and  $Ave$ , respectively. In the DRR-gossip-max algorithm, after the Gossip-max procedure, all the roots will know  $Max$  whp. If necessary, a root then broadcasts  $Max$  to its tree members, requiring  $O(\log n)$  rounds and  $O(n)$  messages since the size of a tree is at most  $O(\log n)$  nodes by Theorem 2 and the number of roots is  $O(n/\log n)$  by Theorem 1.

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  $Max$  whp, the Gossip-ave algorithm only guarantees that the root of the largest tree will have the  $Ave$  whp. After the Gossip-ave algorithm, the root of the largest tree has to spread out the  $Ave$  by the Data-spread algorithm. Hence, every root needs to know in advance if 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.) In the end, every root broadcasts  $Ave$  obtained from the Data-spread algorithm to all its tree members.

---

### Algorithm 5: DRR-gossip-max

---

Run  $DRR(G)$  to obtain  $\mathbb{F}$ , the ranking tree forest.  
 Run Convergecast-max( $\mathbb{F}, \mathbf{v}$ ).  
 Run Gossip-max( $G, \mathbb{F}, \tilde{V}, \mathbf{cov}_{max}$ ).  
 Every root node broadcasts the  $Max$  to nodes in its ranking tree.

---



---

### Algorithm 6: DRR-gossip-ave

---

Run  $DRR(G)$  algorithm to obtain  $\mathbb{F}$ , the ranking tree forest.  
 Run Convergecast-sum( $\mathbb{F}, \mathbf{v}$ ) algorithm.  
 Run Gossip-max( $G, \mathbb{F}, \tilde{V}, \mathbf{cov}_{sum}(*, 2)$ ) algorithm on the sizes of ranking trees to find the root of the largest tree. At the end of this phase, 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.  
 Run Data-spread( $G, \mathbb{F}, \tilde{V}, Ave$ ) algorithm—the root of the largest ranking tree uses its average estimate, i.e.,  $Ave$ , as the value to spread.  
 Every root broadcasts its value to all the nodes in its ranking tree.

---

### 6.1 The complexity of DRR-gossip algorithms

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

## 7 Lower Bound for Address-Oblivious Algorithms

We show that 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 random phone call model: i.e., communication partners are chosen randomly (without depending on their addresses). The following theorem gives a lower bound for computing the Min aggregate. The argument can be adapted for other aggregates as well.

**Theorem 7** *Any address-oblivious algorithm that computes the Minimum value,  $Min$ , in a  $n$ -node network needs  $\Omega(n \log n)$  messages whp.*

*Proof:* Fix any arbitrary node  $i \in V$ . We lower bound the number of messages (transmissions) exchanged between nodes before  $i$  correctly knows the minimum value,  $Min$ , among all nodes. Suppose nodes can send messages that are arbitrary long. (The bound will hold regardless of this assumption.) Without loss of generality, we will assume that a node can send a list of all node addresses and the corresponding node values learned so far (without any aggregation). For  $i$  to have correct knowledge of the minimum, it should somehow know the values at all other nodes. (Otherwise, an adversary — who may know the random choices made by the nodes — can always make sure that the minimum 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) direct way:  $i$  gets to know  $j$ 's value by communicating with  $j$  directly ; and (2) indirect way:  $i$  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 (hence the definition is recursive). Also, at the beginning of the algorithm, since each node knows only about its own value, only the direct way applies.

The main idea of the proof is to appeal to a coupon collector's problem (e.g., [13]) to bound the number of messages needed by  $i$  to know all the other  $n - 1$  values. In a coupon collector setting, there are  $n$  types of coupons, and in each step, each coupon can be generated independently and uniformly at random from all the  $n$  types. It is known that the number of steps needed to collect all the  $n$  coupon types is  $\Theta(n \log n)$  whp [13]. Here we have  $n$  node values in place of  $n$  coupons. The main difference between the coupon collector's setting and here is that in one round,  $i$  can get many different new node values by accessing some node  $w$ . However, each of these different values would have involved a separate message between some two nodes. In other words, if  $i$  gets to know  $f$  different new node values in a round by accessing  $w$ , then each of these  $f$  values would be attributed to at least  $f$  different transmissions that would be needed to bring these values to  $w$ . On the other hand, if  $i$  learns nothing new in accessing  $w$ , then one transmission is wasted (the transmission between  $i$  and  $w$ ). Randomly permute the  $f$  different ( $f \geq 1$ ) node values learnt by  $i$  (some of these values might be new to  $i$  and others already known to  $i$ ) in round  $t$  ( $t \geq 1$ ); let these be denoted by  $X_1^t, \dots, X_f^t$ . Since calls are made randomly, for all  $\alpha$ ,  $\Pr(X_\alpha^t = j) = 1/n$  for any particular node (value)  $j$ . But the random variables are not independent; however, the dependence only increases the probability of accessing a new item:

(1) if  $X_\alpha^t$  is a new item (i.e., not known by  $i$  before), then  $\Pr(X_\alpha^t | past) = 1/n$ . This follows because the probability of this new item reaching  $w$  is  $1/n$ . This can be formally shown by induction on the number of steps needed by the item to reach  $w$ .

(2) if  $X_\alpha^t$  is an old item, then  $\Pr(X_\alpha^t | past) \geq 1/n$ . This can be again shown by induction on the number of steps needed by the item to reach  $w$ .

Thus, the number of messages needed stochastically dominates the number of steps needed in the coupon collector's problem. Hence the claimed lower bound follows. ■

## 8 Concluding Remarks

We presented a gossip-based protocol for computing aggregates that takes  $O(n \log \log n)$  messages and  $O(\log n)$  rounds. Our protocol is non-address oblivious. We also showed that  $\Omega(n \log n)$  messages are needed by any address-oblivious algorithm. An interesting question is to establish whether  $\Omega(n \log \log n)$  messages is a lower bound for gossip-based aggregate computation (in the non-address oblivious random phone call model) if only  $O(\log n)$  rounds are allowed.

## References

- [1] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Trans. on Netw.*, 14(SI):2508–2530, 2006.

- [2] J.-Y. Chen, G. Pandurangan, and D. Xu. Robust aggregate computation in wireless sensor network: distributed randomized algorithms and analysis. In *IEEE Trans. on Paral. and Dist. Sys. (TPDS)*, pages 987–1000, Sep. 2006.
- [3] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM.
- [4] Alexandros G. Dimakis, Anand D. Sarwate, and Martin J. Wainwright. Geographic gossip: efficient aggregation for sensor networks. In *IPSN*, pages 69–76, 2006.
- [5] Jie Gao, Leonidas Guibas, Nikola Milosavljevic, and John Hershberger. Sparse data aggregation in sensor networks. In *IPSN*, 2007.
- [6] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *FOCS*, page 565, 2000.
- [7] Srinivas Kashyap, Supratim Deb, K. V. M. Naidu, Rajeev Rastogi, and Anand Srinivasan. Efficient gossip-based aggregate computation. In *PODS*, pages 308–317, 2006.
- [8] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. FOCS*, 2003.
- [9] B. Krishnamachari, D. Estrin, and S. Wicker. Impact of data aggregation in wireless sensor networks. In *DEBS*, 2002.
- [10] Pradeep Kyasanur, Romit Roy Choudhury, and Indranil Gupta. Smart gossip: An adaptive gossip-based broadcasting service for sensor networks. In *MASS*, 2006.
- [11] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.
- [12] Damon Mosk-Aoyama and Devavrat Shah. Computing separable functions via gossip. In *PODC*, pages 113–122, 2006.
- [13] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [14] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys*, 2004.
- [15] Boris Pittel. On spreading a rumor. *SIAM J. Appl. Math.*, 47(1):213–223, 1987.
- [16] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.
- [17] R. Sarkar, X. Zhu, and J. Gao. Hierarchical spatial gossip for multi-resolution representation in sensor network. In *IPSN*, 2007.
- [18] Nisheeth Shrivastava, Chiranjeev Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys*, 2004.

- [19] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

## Appendices

### A Convergecast algorithms

---

**Algorithm 7:**  $\text{cov}_{max} = \text{convergecast-max}(\mathbb{F}, \mathbf{v})$

---

**Input:** the ranking forest  $\mathbb{F}$ , and the value vector  $\mathbf{v}$  over all nodes in  $\mathbb{F}$

**Output:** the local *Max* aggregate vector  $\text{cov}_{max}$  over 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  $\text{cov}_{max}(z)$ .

**end**

---



---

**Algorithm 8:**  $\text{cov}_{sum} = \text{Convergecast-sum}(\mathbb{F}, \mathbf{v})$

---

**Input:** the ranking forest  $\mathbb{F}$  and the value vector  $\mathbf{v}$  over all nodes in  $\mathbb{F}$

**Output:** the local *Ave* aggregate vector  $\text{cov}_{sum}$  over 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  $v_j = v_j + \sum_{k \in \text{Child}(j)} v_k$ , and  $w_j = w_j + \sum_{k \in \text{Child}(j)} w_k$ , where  $\text{Child}(j) = \{j\text{'s children nodes}\}$

- 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  $\text{cov}_{sum}(z, 1) = v_z + \sum_{k \in \text{Child}(z)} v_k$ , and the size count of the tree  $\text{cov}_{sum}(z, 2) = w_z + \sum_{k \in \text{Child}(z)} w_k$ , where  $\text{Child}(z) = \{z\text{'s children nodes}\}$ .

**end**

---

## B The proof of Theorem 4

*Proof:* As per our failure model, a message may fail to reach the selected root node with probability  $\rho$  (which is at most  $2\delta$ , since failure may occur either during the initial call to a non-root node or during the forwarding call from the non-root node to the root of its tree). For convenience, we call those roots who know the *Max* value (the global Maximum) as the max-roots and those who do not as 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, whp,  $R_t > 4 \log n$  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 chance that a max-root selects another max-root is small. Similarly, the chance that two or more max-roots select the same root is also small. So, in this step, whp 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, the  $R_t$  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.

Applying Azuma's inequality [13] and setting  $\epsilon = 1/2$ :

$$\begin{aligned} Pr(|X - E[X]| > \epsilon E[X]) &< 2 \exp\left(-\frac{\epsilon^2 E[X]^2}{2\left(\frac{8 \log n}{1 - \rho}\right)}\right) \\ &< 2 \exp\left(-\frac{\frac{1}{4} E[X]^2}{16 \log n}\right) = 2 \exp(-\log n) = 2 \cdot n^{-1}. \end{aligned}$$

Hence, with probability at least  $1 - \frac{2}{n}$ , after  $8 \log n / (1 - \rho) = O(\log n)$  rounds,  $R_t > \frac{1}{2} E[X] = 4 \log n$ .

In the second step of our proof, we find the *lower bound of 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. The  $Y_i = 0$  when one of the following event happens. (1) The message  $i$  fails in routing to its destination in probability  $\rho$ . (2) The message  $i$  destined to another max-root although it successfully travels over the network with probability  $(1 - \rho)$ . The probability of this event is at most  $\frac{(1 - \rho) R_t \log n}{n}$  since whp the size of a ranking tree is  $O(\log n)$  (cf. Theorem 2). (3) The message  $i$  and at least one another message are destined to the same non-max-root. As the probability three or more messages are destined to a same node is very small, we only consider the case that two messages select the same non-max-root. We also conservatively exclude both two messages on their possible contributions to the increase of  $R_t$ . This event happens with the probability at most  $\frac{(1 - \rho) R_t \log n}{n}$ .

Applying union bound [13],

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

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,

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

Since in this step, whp  $R_t > 4 \log n$ , 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, whp,  $R_{t+1} > R_t + \frac{1}{2}(1 - c')(1 - \rho)R_t = \beta R_t$ , where  $\beta = 1 + \frac{1}{2}(1 - c')(1 - \rho) > 1$ . Therefore, whp, after  $(8 \log n / (1 - \rho) + \log_{\beta} n) = O(\log n)$  rounds, at least  $\Omega(\frac{cn}{\log n})$  roots will have the *Max*. ■

## C The proof of Theorem 6

To prove Theorem 6, we need some definitions as in [8]. 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  $j$ -th 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 ranking tree rooted at node  $j$  computed by Convergecast-sum.  $\mathbf{y}_{0,i} = \mathbf{e}_i$ , the unit vector with the  $i$ -th 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 of *Ave*,  $\hat{x}_{ave,t,i} = \frac{s_{t,i}}{g_{t,i}}$ , is close to the true average  $x_{ave}$ . Note that  $w_{t,i}$  which is different from  $g_{t,i}$  is a dummy parameter borrowed from [8] to characterize the diffusion speed.

In our Gossip-ave algorithm, we set  $g_{0,i}$  to be the size of the root  $i$ 's ranking 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 as that 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 ranking 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  $\Delta_{t,i} = \max_j \left| \frac{y_{t,i,j}}{\|\mathbf{y}_{t,i}\|_1} - \frac{1}{m} \right| = \left\| \frac{y_{t,i,j}}{\|\mathbf{y}_{t,i}\|_1} - \frac{1}{m} \right\|_{\infty}$ . Also, a potential function

$$\Phi_t = \sum_{i,j} \left( y_{t,i,j} - \frac{w_{t,i}}{m} \right)^2$$

is the sum of the variance of the contributions  $y_{t,i,j}$ . We name the root of the largest ranking tree the node  $z$ .

To prove Theorem 6, we further need some auxiliary lemmas.

**Lemma 8 (geometric convergence of  $\Phi$ )** *The conditional expectation  $E[\Phi_{t+1} | \Phi_t = \phi] = \frac{1}{2}(1 - \sum_{i \in \tilde{V}} P_i^2)\phi < \frac{1}{2}\phi$ , where  $P_i = (1 - \delta)\frac{g_i}{n}$  is the probability that the root node  $i$  is selected by other root nodes,  $g_i$  is the size of ranking tree rooted at node  $i$ ,  $\delta$  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 [8]. The difference is that the selection probability,  $P_i$ , is not uniform any more but depends on the tree size,  $g_i$ . The  $P_i$  is the probability that root  $i$  is selected by any other roots 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

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

■

**Lemma 9** *There exists a  $\tau = O(\log m)$  such that after  $\forall t > \tau$  rounds of Gossip-ave,  $w_{t,z} \geq 2^{-\tau}$  at the root  $z$  of the largest ranking tree.*

*Proof:* In the case that the selection probability is uniform, it has been shown in [8] 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 theorem.

**Theorem 10 (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 the root,  $z$ , of the largest ranking tree is nearly uniform, i.e.,  $\| \frac{y_{t,z,y}}{\|y_{t,z}\|_1} - \frac{1}{m} \|_\infty \leq \epsilon$ .*

*Proof:* By Lemma 8, we obtain that  $E[\Phi_t] < (m-1)2^{-t} < m2^{-t}$ , as  $\Phi_0 = (m-1)$ . By Lemma 9, 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  $t = \log m + \log \frac{1}{\epsilon}$  rounds of Gossip-ave,  $E[\Phi_t] \leq \hat{\epsilon}$ . By Markov's Inequality, 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 the goal  $\max_j \left| \frac{y_{t,z,y}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m} \right| \leq \epsilon$ , we need the lower bound on the weight of node  $z$ . From Lemma 9,  $w_{t,z} = \|\mathbf{y}_{t,z}\|_1 \geq 2^{-\tau}$  with probability at least  $1 - \frac{\delta'}{2}$ . Note that Lemma 9 only applies for the root node  $z$  of the largest ranking tree. (A root node of a relatively small ranking tree may not be selected often enough to have such lower bound on its weight.) Taking union bound of the probability, we obtain, with probability at least  $1 - \delta'$ ,  $\max_j \left| \frac{y_{t,z,y}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m} \right| \leq \epsilon$ .  $\blacksquare$

Now we are ready to prove Theorem 6.

**Proof of Theorem 6**

*Proof:* From Theorem 10, with probability at least  $1 - \delta'$ , it is guaranteed that after  $T_{ave} = O(\log m + \log \frac{1}{\epsilon} + \log \frac{1}{\delta'})$  rounds of Gossip-ave, at the root  $z$  of the largest tree,  $\left\| \frac{y_{t,z,y}}{\|\mathbf{y}_{t,z}\|_1} - \frac{1}{m} \right\|_\infty \leq \epsilon$ . Let both  $\epsilon = n^{-\alpha}$  and  $\delta' = n^{-\alpha}$ ,  $\alpha > 0$ , then  $T_{ave} = O(\log m + 2\alpha \log n) = O(\log n)$ .

Apply Hölder's Inequality, we obtain

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

When  $x_j$  are all with 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 the *Ave*. Let  $s_{ave} > 0$ ,  $g_{ave} > 0$ <sup>1</sup> be the *true* average of the sum of values in a ranking tree and the true average of the size of a ranking tree, respectively. The global average *Ave* is  $x_{ave} = \frac{s_{ave}}{g_{ave}}$ . Since  $\left| \frac{s_{t,z}}{w_{t,z}} - s_{ave} \right| \leq \epsilon s_{ave}$  and  $\left| \frac{g_{t,z}}{w_{t,z}} - g_{ave} \right| \leq \epsilon g_{ave}$  we obtain

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

Set  $\epsilon' = c\epsilon$ , where  $c = \frac{2}{(1-\epsilon)} > 2$  is bounded when  $\epsilon < 1$ . (For example, if  $\epsilon \leq 10^{-2}$ , then  $c = 2.0\bar{2}$  and  $\epsilon' = \frac{200}{99}\epsilon$ .) Typically, 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 the root  $z$  of the largest ranking tree is

$$\frac{|\hat{x}_{ave,tz} - x_{ave}|}{|x_{ave}|} \leq \epsilon',$$

by  $O(\log m + 2\alpha \log n) = O(\log n)$  rounds of Gossip-ave algorithm.  $\blacksquare$

<sup>1</sup>The  $g_{ave} > 0$  by definition. We can always offset values to have  $s_{ave} > 0$ .