

# Locality-Based Aggregate Computation in Wireless Sensor Networks

Jen-Yeu Chen, *Member, IEEE*, Gopal Pandurangan, *Member, IEEE*, and  
Jianghai Hu, *Member, IEEE*

## Abstract

The design of efficient gossip-based protocols for computing aggregate functions in wireless sensor network has received considerable attention recently. A primary reason for this surge in interest is their robustness due to their randomized nature. In addition, because of the inherent simplicity in their design, gossip-based protocols are suitable to be implemented on sensor nodes with limited computing capability and battery power. In the context of energy-constrained sensor networks, it is of paramount importance to reduce the message and time complexity of gossip-based protocols. In this paper, we present DRR-gossip, an energy-efficient and robust aggregate computation algorithm in sensor networks. Exploiting locality, the Distributed Random Ranking (DRR) algorithm first builds a forest of disjoint local trees over the sensor network. The height of each tree is small, so that the aggregates of the tree nodes can be quickly obtained at the root of the tree. All the roots then perform a uniform gossip algorithm on their respective aggregates to reach a distributed consensus on the global aggregate. We prove that the DRR-gossip algorithm requires  $O(n)$  messages and  $O(\frac{n^{3/2}}{\log^{1/2} n})$  transmissions to obtain aggregates on a random geometric graph. This reduces the energy consumption by at least a factor of  $\frac{1}{\log n}$  over the standard uniform gossip algorithm. Experiments validate the theoretical results and show that DRR-gossip needs much less transmissions than other gossip-based schemes.

## Index Terms

wireless sensor networks; aggregate computation; gossip; distributed algorithm; randomized algorithm; random geometric graph

Jen-Yeu Chen is with the Department of Electrical Engineering, National Dong Hwa University, Taiwan, R.O.C. E-mail: jenyeu@ieee.org. Gopal Pandurangan is with the Department of Computer Science, Purdue University, West Lafayette, USA. E-mail: gopal@cs.purdue.edu. Jianghai Hu is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, USA. E-mail: jianghai@purdue.edu.

## I. INTRODUCTION

Aggregate statistics such as Average, Max/Min, Sum and Count have been significantly useful for many applications in sensor networks [2], [4], [9], [10], [14], [21]. Many research efforts have been dedicated to developing scalable and distributed algorithms for the computation of aggregates in sensor networks. Among them gossip-based algorithms [1], [2], [3], [8], [9], [12], [15], [18], [20] have recently received much attention because of their simplicity for implementation, scalability to large network size, and robustness to frequent network topology changes. In a standard gossip algorithm, every node computes and updates the approximate of the target aggregate iteratively. In an iteration, every node randomly selects another node to transmit the information of its current approximate. As a result, each node's approximate will gradually approach the correct aggregate. In the end, all the approximates will reach a consensus, within some accuracy threshold, on the target aggregate.

There may be substantial energy waste in a standard gossip algorithm. It has been shown in [1] that the neighboring gossip algorithm<sup>1</sup>, where a node randomly selects a neighboring node to exchange information, needs the same order of radio transmissions,  $O(n^2)$ , as the plain flooding on a Poisson random geometric graph. In [9] another version of gossip, named uniform gossip, allows that every node uniformly at random selects another node in the network to send its data. Multi-hop routing is necessary for uniform gossip. To reach the global consensus on aggregate, the uniform gossip needs  $O(\log n)$  rounds,  $O(n \log n)$  messages and  $O(D \cdot n \log n)$  transmissions in the worst case, where  $D$  is the diameter of the network. Recently, some novel gossip-based algorithms have been proposed to improve the performance of a standard gossip algorithm. The geographic gossip of [3] adopts the neighboring gossip scheme but allows two remote nodes to exchange information by geographic routing, improving the cost of transmissions to  $O(n^{3/2} \log^{3/2} n)$  from the neighboring gossip's  $O(n^2)$ . The efficient gossip in [8], an improvement of uniform gossip scheme where groups are formed before applying uniform gossip, needs a slightly longer time  $O(\log n \log \log n)$ , but reduces the messages and transmissions to  $O(n \log \log n)$  and  $O(D \cdot n \log n)$ , respectively. A different approach, the hierarchical spacial gossip [20], using ODI-synopsis and restricting the communications locally, needs a longer  $O(\text{poly}(\log n))$  time but requires only  $O(n \text{poly}(\log n))$  transmissions.

In this paper, a novel approach named the DRR-gossip algorithm is presented. Exploiting locality, a Distributed Random Ranking algorithm, DRR, is used to first build a forest of (disjoint) localized trees

<sup>1</sup>The neighboring gossip is called the fastest gossip in [1] as its diffusion speed can be optimized by carefully setting the selection probability.

over the sensor network. The height of a tree is small, thus the aggregates of its nodes can be quickly obtained at its root. All the roots then perform the uniform gossip algorithm on their respective aggregates to reach distributed consensus on the global aggregate. Compared with the standard uniform gossip, our DRR-gossip algorithm requires  $O(n)$  messages and  $O(\frac{n^{3/2}}{\log^{1/2} n})$  transmissions to reach consensus, reducing the energy consumption by at least a factor of  $1/\log n$  on a Poisson random geometric graph. The DRR-gossip algorithm is inspired from the efficient gossip ([8]) idea of reducing the number of nodes participating in the gossip process in order to decrease the numbers of messages and transmissions for achieving consensus. Further, our DRR-gossip algorithm takes advantage of the locality of the trees to further decrease the number of messages to  $O(n)$  while still maintains a running time of  $O(\log n)$ , whereas the efficient gossip algorithm needs  $O(\log n \log \log n)$  time and  $O(n \log \log n)$  messages.

The DRR-gossip algorithm proceeds in phases. In phase one, all the sensor nodes run the DRR algorithm to construct a forest of disjoint ranking trees. In phase two, within each ranking tree, the local aggregate (e.g. sum or maximum) of a ranking tree is computed by a convergecast process and obtained at the root of the tree. In phase three, all the roots of the ranking trees utilize a suitably modified version of the uniform gossip algorithm [9] to obtain the global aggregate. Finally, if necessary, a root can forward the global aggregate along the tree branches to all the other nodes of the tree.

The rest of this paper is organized as follows. The network model is described in Section II 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 VI. Simulation results are provided in Section VII. We finally conclude in Section VIII.

## II. NETWORK MODEL

A wireless sensor network is abstracted as a connected undirected graph  $G(V, E)$  with all the sensor nodes as the set of vertices  $V$  and all the bi-directional wireless communication links as the set of edges  $E$ . This underlying graph can be arbitrary depending on the deployment of sensor nodes. Let each sensor node  $i$  be associated with an initial observation or measurement value denoted as  $v_i \in \mathbb{R}$ . The assigned values over all vertices form a vector  $\mathbf{v}$ . The goal is to compute aggregate functions such as average, sum, max, min etc. on the vector of values  $\mathbf{v}$ .

For ease of understanding, we present and analyze the DRR-gossip algorithm in synchronized rounds, even though the algorithm is not required to be synchronized. All the sub-procedures work in asynchronous setting though the analysis is more involved and needs to be altered.

The DRR-gossip algorithm is an application layer algorithm to which the underlying layers should be

transparent. Thus, we do not assume any particular MAC and PHY layers for message transmissions but assume that all the nodes of the undirected connected graph  $G$  can simultaneously send a message to their one-hop neighbors in a “step” of the algorithm<sup>2</sup>. In phase one and phase two of the DRR-gossip algorithm, all messages require only local one-hop transmissions whereas in phase three messages may require multi-hop transmissions. The time for a gossip message to reach its destination is a “round” of the gossip algorithm. Thus, in the phase three, the gossip algorithm proceeds in rounds, each of which contains several steps. The length of a message is limited to  $O(\log n + \log s)$ , where  $s$  is the range of measured values on sensor nodes.

### III. PHASE I: DISTRIBUTED RANDOM RANKING

#### A. The DRR algorithm

The DRR algorithm is implemented in the following way. On an undirected connected graph  $G = (V, E)$ , every node  $i \in V$  *uniformly at random* generates a random value,  $rank(i) \in [0, 1]$ , named the rank<sup>3</sup> of node  $i$ , and then sends its rank to all its (one-hop) neighboring nodes. Each node compares the ranks it receives with its own and then connects to the node of the highest rank among all of its neighbors and itself. If a tie between two ranks happens, nodes can break the tie by their identifiers. Through these connections, many *disjoint* “local ranking trees” are established on the graph. These trees together constitute a forest  $\mathbb{F}$ , which is a subgraph of  $G$ , i.e.,  $\mathbb{F}(V', E') \subseteq G(V, E)$ , where  $V' = V$  and  $E' \subseteq E$ . If a node is of the highest rank among its neighbors and itself, it becomes the root of a tree. Since every node, except the root nodes, connects to a node with a higher rank, there is no loop in the constructed graph, the forest  $\mathbb{F}$ . The pseudo-code of the DRR algorithm is provided in Algorithm 1.

An example to illustrate the result of the DRR algorithm is shown in Fig. 1. The Poisson random geometric graph shown in Fig. 1(a) has 400 nodes in a unit square. After running the DRR algorithm, the constructed forest of ranking trees is shown in Fig. 1(b).

In this paper we focus our analysis on Poisson random geometric graphs, a useful and popular graph model for sensor networks [5], [17], though the analysis can be extended to general graphs. A Poisson

<sup>2</sup>In wireless MAC layer, two neighboring nodes can not send messages simultaneously. One node needs to send its message after the other one finishes. In application layer, we define the necessary time period for nodes to successfully finish their transmissions as one “step” of the algorithm. Obviously, the time period of a step will vary according to the underlying MAC layer.

<sup>3</sup>The rank of a node  $i$ ,  $rank(i)$ , is independent of its value  $v_i$ .

---

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


---

```

1 foreach node  $i \in V$  do
2   independently and uniformly at random generate a rank  $rank(i) \in [0, 1]$ ;
3   send  $rank(i)$  to all its neighboring nodes  $j \in \mathbb{N}(i)$  where  $\mathbb{N}(i) = \{j | (i, j) \in E\}$ ;
4   collect the ranks from its neighbors and compare them with its own rank. Let
       $\psi = \max_{j \in \mathbb{N}(i)} rank(j)$ ;
5   if  $\psi > rank(i)$  then
6     set  $parent(i) = \operatorname{argmax}_{j \in \mathbb{N}(i)} rank(j)$ ;
7   else
8     set  $parent(i) = \text{NULL}$ , and become a root node.
9   end
10  send a connection message including its identifier,  $i$ , to its parent node  $parent(i)$ ;
11  collect the connection messages and accordingly construct the set of its children nodes,
       $Child(i)$ ;
12  if  $Child(i) = \emptyset$  then
13    become a leaf node;
14  else
15    become an intermediate node.
16  end
17 end

```

---

random geometric graph is denoted by  $G(n, r(n)) = G(V, E)$ , where  $|V| = n$  and the radius  $r(n) = \Omega((\frac{\log n}{n})^{1/2})$  to maintain connectivity [5].

### B. The expected number of ranking trees and the locality of a ranking tree

To find the expected number of ranking trees, i.e., the expected number of roots, generated by the DRR algorithm on a Poisson random geometric graph, we need the following lemma.

**Lemma 1:** At the end of the DRR algorithm, a node  $i \in V$  of degree  $d_i$  in graph  $G$  becomes the root of a local ranking tree with the probability  $1/(d_i + 1)$ .

Due to space limitation, we refer to appendix for the proof.

The following theorem shows the expected number of roots in a Poisson random geometric graph.

**Theorem 2:** On a Poisson random geometric graph  $G(n, r)$ , the expected number of ranking trees constructed by the DRR algorithm is  $m = O(\frac{n}{\log n})$ .

*Proof:* Given a Poisson random geometric graph  $G(n, r)$ , an  $n$ -tuple random variable  $\vec{d} = (d_1, d_2, \dots, d_n)$ , where  $d_i$  is the degree of node  $i \in V$ , is used to represent the degrees of all nodes of a possible instance of the random geometric graph  $G$ . The degree  $d_i$  is a random variable determined by the 2-D homogeneous Poisson point process with intensity  $n$ .

Let  $Y_i$  be the indicator for the event that the node  $i \in V$  becomes a root and  $Y = \sum_{i \in V} Y_i$  be the random variable indicating the total number of roots after running the DRR algorithm. Given  $\vec{d}$ , the expected number of roots is then

$$E[Y | \vec{d}] = \sum_{i \in V} E[Y_i | \vec{d}] = \sum_{i \in V} \frac{1}{d_i + 1}.$$

So, letting  $\gamma = (\sum_{k=0}^n e^{-\lambda} \frac{\lambda^k}{k!})^{-1} = \Theta(1)$ , the expected number of roots of the local ranking trees is

$$\begin{aligned} E[Y] &= E \left[ E[Y | \vec{d}] \right] = \sum_{i \in V} E \left[ \frac{1}{d_i + 1} \right] \\ &= \gamma n \cdot \sum_{k=0}^n \frac{1}{k+1} \cdot e^{-\lambda} \frac{\lambda^k}{k!} \\ &= \frac{\gamma n}{\lambda} \cdot e^{-\lambda} \sum_{k=0}^n \frac{\lambda^{k+1}}{(k+1)!} \\ &= \frac{\gamma n}{\lambda} \cdot e^{-\lambda} \left( e^\lambda - 1 - o\left(\frac{\lambda^{n+2}}{(n+2)!}\right) \right) \\ &= \frac{\gamma n}{\lambda} \left( 1 - e^{-\lambda} \left( 1 + o\left(\frac{\lambda^{n+2}}{(n+2)!}\right) \right) \right), \end{aligned}$$

where  $\lambda = n \cdot \pi r^2 = \Omega(\log n)$  is the expected degree of a node in the Poisson random geometric graph. In the above, the third equality follows from the fact that every node has the same degree distribution determined by a 2-D Poisson point process. Thus, running Distributed Random Ranking on a Poisson random geometric graph with the number of nodes  $n$ , the expected number of roots  $E[Y]$  is  $O(\frac{n}{\log n})$ , i.e.,  $E[Y] \leq \frac{\gamma n}{\log n} (1 - \frac{1}{n}) = O(\frac{n}{\log n})$ . ■

### Locality of a ranking tree

To show the locality of a ranking tree, we need to bound the maximal height of a local ranking tree in the Poisson random geometric graph  $G(n, r)$ . To do this, some definitions and lemmas are needed.

**Definition 3:** An  $h$ -hop path  $\{i_1, i_2, \dots, i_{h+1}\}$  in graph  $G(V, E)$  is an ordered sequence of  $h + 1$  nodes where each node is only encountered once (i.e., the multitude of any node in a path equals to one)

and any two consecutive nodes are adjacent to each other, i.e.,  $(i_k, i_{k+1}) \in E$ , for  $1 \leq k \leq h$ .

**Definition 4:** An  $h$ -hop *potential ranking path*  $\{i_1, i_2, \dots, i_{h+1}\}$  in a graph  $G(V, E)$  is an  $h$ -hop path where each node is only adjacent, in the graph  $G$ , to its preceding node and its following node, i.e.,  $(i_k, i_{k+1}) \in E$ , for  $1 \leq k \leq h$  and  $(i_k, i_{k+s}) \notin E$ , for  $1 \leq k \leq h-1$  and  $1 < s \leq h+1-k$ .

**Definition 5:** A *rooted tree path* is a simple path contained entirely within a ranking tree, with the starting node the root of the ranking tree.

**Lemma 6:** Any rooted tree path is a potential ranking path.

Due to space limitation, we refer to appendix for the proof.

Note that a potential ranking path is not necessarily a rooted tree path whereas a rooted tree path has to be a potential ranking path.

**Lemma 7:** Given a path of length  $h$  in a Poisson random geometric graph, the probability that it is a potential ranking path of length  $h$  is  $O((\frac{2}{3})^{h-1})$ .

Due to space limitation, we refer to appendix for the proof.

**Theorem 8 (Locality of ranking trees):** On a Poisson random geometric graph  $G(n, r)$ , w.h.p.<sup>4</sup>, the height of the ranking trees constructed by the DRR algorithm is bounded by  $O(\log n)$ .

*Proof:* We show that the probability that an  $h$ -hop path, starting from a root, is a rooted tree path is asymptotically close to zero when  $h = \Theta(\log n)$ . First, this path is a potential ranking path with probability  $O((2/3)^{h-1})$ . Secondly, conditioning on that it is a potential ranking path, this  $h$ -hop potential ranking path is an  $h$ -hop rooted tree path only if all the nodes' ranks on this path are in a strictly decreasing order, which happens with probability  $1/(h+1)!$ . Let  $h+1 = \Theta(\log n)$ . The probability that an  $h$ -hop path, starting from a root, is a rooted tree path is at most  $\frac{1}{(h+1)!} \cdot O((\frac{2}{3})^{h-1}) = o((\frac{2}{3})^{2h}) = o((\frac{2}{3})^{2 \log n}) = o(n^{-\alpha})$ ;  $\alpha > 0$ .

We further show that the *expected number* of  $(\log n)$ -hop rooted tree paths,  $Z$ , is asymptotically close to zero. On a Poisson random geometric graph, the total number of  $h$ -hop paths starting from a root is  $O(\log^h n)$ , w.h.p., since the degree of a node is bounded by  $O(\log n)$  w.h.p., [5]. Letting  $h = c \cdot \log n$

<sup>4</sup>The term with high probability, w.h.p., means with probability at least  $1 - O(n^{-\alpha})$  for some positive constant  $\alpha$ .

and using Stirling's approximation,

$$\begin{aligned} Z &= O\left(\frac{1}{(h+1)!} \cdot \left(\frac{2}{3}\right)^{h-1} \cdot \log^h n\right) \\ &= O\left(\frac{(\log n)^{c \log n} (2/3)^{c \log n}}{(2\pi c \log n)^{1/2} (c \log n/e)^{c \log n}}\right) \\ &= O\left(\frac{(2e/3c)^{c \log n}}{(2\pi c \log n)^{1/2}}\right) = O\left(\frac{n^{-\alpha}}{(2\pi c \log n)^{1/2}}\right), \end{aligned}$$

where  $\alpha = c(\log(\frac{3c}{2}) - 1)$ . By suitably fixing  $c$ , we have  $\alpha > 1$  and that  $Z$  is asymptotically close to zero. Hence the probability that a tree is of height  $O(\log n)$ , i.e., the probability that there exists a rooted tree path of length  $O(\log n)$  is asymptotically close to zero. Therefore, w. h. p. the height of a local ranking tree is bounded by  $O(\log n)$ . ■

### C. Advantages of the DRR algorithm

The DRR algorithm has the following advantages: (1) locality; (2) robustness to adversary; and (3) easy load balancing. First, from the above Theorem 8, a message originating from a leaf of a local ranking tree needs at most  $O(\log n)$  relays to arrive at the root of the tree. Local aggregates of a ranking tree can be quickly obtained at the root node with low cost of messages and transmissions. Secondly, consider a deterministic approach that constructs local trees rooted at some pre-determined roots. An adversary can easily locate a root node and destroy it since the root's location is fixed. In contrast, the DRR algorithm has the robustness against the adversary since the roots are randomly distributed in the network. Also, running the DRR algorithm periodically to change roots can balance the transmission load over nodes in the network.

### D. Complexity of the DRR algorithm

In step 2 of the algorithm, every node needs to broadcast its rank to its neighbors. This costs  $O(|V| = n)$  messages in a round since every node uses wireless transmission. In step 4, all the nodes except the roots of local ranking trees have to send a message to their parent nodes, requiring totally  $O(|V| = n)$  messages in a round. Totally in these two steps, the message complexity is  $O(n)$ . By Theorem 8 the time complexity is  $O(\log n)$ .

## IV. PHASE II: CONVERGE-CAST

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 the leaf nodes and proceeding upward

---

**Algorithm 2:**  $\text{conv}_{max} = \text{Converge-cast-max}(\mathbb{F}, \mathbf{v})$ 


---

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

**Output:** the local *Max* aggregate vector  $\text{conv}_{max}$  over roots

```

1 foreach leaf node do send its value to its parent;
2 foreach intermediate node do
3   - collect values from its children;
4   - compare collected values with its own value;
5   - update its value to the maximum amid all and send the maximum to its parent.
6 end
7 foreach root node  $z$  do
8   - collect values from its children;
9   - compare collected values with its own value;
10  - update its value to the local maximum value  $\text{conv}_{max}(z)$ .
11 end

```

---

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. The pseudo-code of the Converge-cast-max algorithm and the Converge-cast-sum algorithm are provided in Algorithm 2 and Algorithm 3, respectively. In the pseudo-codes, the input  $\mathbf{v} \in M_{n,1}$ , where  $M_{x,y}$  is the  $x \times y$  matrix, represents the value vector over all nodes in  $V$ ; the output  $\text{conv}_{max} \in M_{m,1}$  and  $\text{conv}_{sum} \in M_{m,2}$  are the computed aggregates at root nodes, where  $m$  is the number of root nodes.

#### A. Complexity of Converge-cast

Every node except the root nodes needs to send a message to its parent in the upward aggregation process of the Converge-cast algorithms. So the message complexity and transmission complexity are both  $O(n)$ . It directly follows from Theorem 8 that the time complexity is  $O(\log n)$ .

### V. PHASE III: GOSSIP ON THE OVERLAY COMPLETE GRAPH

In the third phase, all roots of the local ranking trees compute the global aggregate by performing the uniform gossip algorithm on an abstract overlaying graph  $\tilde{G} = \text{clique}(\tilde{V})$ , where  $\tilde{V} \subseteq V$  is the set

---

**Algorithm 3:**  $\text{conv}_{sum} = \text{Converge-cast-sum}(\mathbb{F}, \mathbf{v})$ 


---

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

**Output:** the local *Ave* aggregate vector  $\text{conv}_{max}$  over roots.

- 1 **Initialization:** every node  $i$  stores a row vector  $(v_i, w_i = 1)$  including its value  $v_i$  and a size count  $w_i$
  - 2 **foreach** leaf node  $i \in \mathbb{F}$  **do**
  - 3     - send its parent a message containing the vector  $(v_i, w_i = 1)$
  - 4     - reset  $(v_i, w_i) = (0, 0)$ .
  - 5 **end**
  - 6 **foreach** intermediate node  $j \in \mathbb{F}$  **do**
  - 7     - collect messages (vectors) from its children
  - 8     - 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  $\text{Child}(j) = \{j\text{'s children nodes}\}$
  - 9     - send computed  $(v_j, w_j)$  to its parent
  - 10    - reset its vector  $(v_j, w_j) = (0, 0)$  when its parent successfully receives its message.
  - 11 **end**
  - 12 **foreach** root node  $z$  **do**
  - 13    - collect messages (vectors) from its children
  - 14    - compute the local sum aggregate  $\text{conv}_{sum}(z, 1) = v_z + \sum_{k \in \text{Child}(z)} v_k$ , and the size count of the local ranking tree  $\text{conv}_{sum}(z, 2) = w_z + \sum_{k \in \text{Child}(z)} w_k$ , where  $\text{Child}(z) = \{z\text{'s children nodes}\}$ .
  - 15 **end**
- 

of roots and  $|\tilde{V}| = m$ . To traverse through an edge of  $\tilde{G}$ , a message may need a multi-hop relay, i.e., several one-hop transmissions, on the original physical network graph  $G$ .

Exploiting the abundant research results on routing algorithms on sensor networks (e.g., [23], [22], [11]), roots can communicate with each other by using one of these routing algorithms. In particular, the (greedy) geographic routing [6], [3], [19], [13] by which a source node can route its message to the node closest to a chosen position (coordinate) without knowing the identifier of the destination node is suitable for the roots to perform uniform gossip since a root node does not know where and who the other roots are. We will discuss this in more detail in Section VI-B1.

The idea of uniform gossip is as follows. Every root independently and uniform at random selects a node to send (or say, gossip) 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 by multi-hop relaying upward its ranking tree. Thus, in a sense, each ranking tree becomes a super node and all the super nodes form a complete graph,  $\tilde{G}$ .

The algorithm 4, Gossip-max, and the algorithm 6, Gossip-ave (a modification of the Push-Sum algorithm of [9], [8]), compute the *Max* and *Ave* aggregates respectively. Note that there is no sampling procedure in the Gossip-ave algorithm.

The algorithm 5, Data-spread, is a modification of the Gossip-max algorithm for a root node to spread its value. If a root needs to spread a particular value over the network, it sets that particular value as its initial value of the Gossip-max algorithm and all the other roots set their initial values to minus infinity.

#### A. Performance of the Gossip-max and Data-spread algorithms

Let  $m$ ,  $n$  denote the number of the root nodes and the number of all the nodes of a sensor network, respectively. We have  $m = |\tilde{V}| = O(n/\log n)$  and  $n = |V|$ . Karp et al. [7] gave a rumor spreading algorithm (for spreading a data item, such as *Max*, throughout a network of  $n$  nodes) that takes  $O(\log n)$  communication rounds and  $O(n \log \log n)$  messages. Their Push algorithm, a prototype of our Gossip-max algorithm, uses uniform selection probability, i.e., each node is chosen with equal 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 ranking tree is selected with a *probability proportional to the size of its associated ranking tree*. The uniformity of the selection probability does not hold here even though 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 the gossip procedure, roots can sample an  $O(\log n)$  number of other roots to confirm and update, if necessary, their values to reach the consensus on *Max*.

To analyze the performance of the Gossip-max algorithm, we assume that a message may fail to reach the selected root node with probability  $\rho$  while it travels through the network. The message may fail at the routing stage or at the forwarding stage along the ranking tree toward the root node. But the probability to fail at the latter stage is typically much smaller than at the former one as the height of a local ranking tree is relatively much smaller than the average hop-count of a routing path. We set  $\rho = 1 - (1 - \rho_1)(1 - \rho_2) \approx \rho_1$ , where  $\rho_1$ ,  $\rho_2$  are the failure probabilities at the above two stages

---

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


---

- 1 **Initialization:** every root  $i \in \tilde{V}$  is of the maximum value  $x_{0,i} = \text{conv}_{max}(i)$  computed from the Converge-cast-max algorithm.
  - 2 **gossip procedure:**
  - 3 **for**  $t=1 : O(\log n)$  **rounds do**
  - 4   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  $x_{t-1,i}$ .
  - 5   Every node  $j \in V - \tilde{V}$  forwards any received messages along its ranking tree to its root.
  - 6   Every root  $i \in \tilde{V}$
  - 7     - collects messages and compares the received values with its own value
  - 8     - updates its current value  $x_{t,i}$ , node  $i$ 's current estimate of  $Max$ , to the maximum amid all received values and its own.
  - 9 **end**
  - 10 **sampling procedure:**
  - 11 **for**  $t=1 : \frac{1}{c} \log n$  **rounds do**
  - 12   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.
  - 13   Every node  $j \in V - \tilde{V}$  forwards any received inquiry messages to its root.
  - 14   Every root  $i \in \tilde{V}$ , upon receiving inquiry messages, sends the inquiring roots its value.
  - 15   Every root  $i \in \tilde{V}$ , updates  $x_{t,i}$ , i.e.  $\hat{\mathbf{x}}_{max,t}(i)$ , to the maximum value it inquires.
  - 16 **end**
- 

---

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


---

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

respectively. For convenience, we call those roots which have already known the  $Max$  as max-roots and those which have not as the non-max-roots.

1) *Gossip procedure:*

---

**Algorithm 6:**  $\hat{\mathbf{x}}_{ave} = \text{Gossip-ave}(G, \mathbb{F}, \tilde{V}, \text{conv}_{sum})$

---

- 1 **Initialization:** Every root  $i \in \tilde{V}$  sets up a vector  $(s_{0,i}, g_{0,i}) = \text{conv}_{sum}(i)$ , where  $s_{0,i}$  and  $g_{0,i}$  are the local sum of values and the size of the local ranking tree rooted at  $i$ , respectively.
  - 2 **for**  $t = 1 : O(\log m + \log(1/\epsilon))$  **rounds do**
  - 3     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  $(s_{t-1,i}/2, g_{t-1,i}/2)$ .
  - 4     Every node  $j \in V - \tilde{V}$  forwards any received messages to the root of its ranking tree.
  - 5     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
  - 6          $s_{t,i} = s_{t-1,i}/2 + \sum_{j \in A_{t,i}} s_{t-1,j}/2$ ,
  - 7          $g_{t,i} = g_{t-1,i}/2 + \sum_{j \in A_{t,i}} g_{t-1,j}/2$ .
  - 8     Every root node  $i \in \tilde{V}$  updates its estimate of the global average by
  - $\hat{\mathbf{x}}_{ave,t}(i) = \hat{x}_{ave,t,i} = s_{t,i}/g_{t,i}$ .
  - 9 **end**
- 

**Theorem 9:** Running the Gossip-max algorithm on  $\tilde{G} = \text{clique}(\tilde{V})$  of  $G(V, E)$  will result in, w.h.p., at least  $\Omega(\frac{c \cdot n}{\log n})$  root nodes having the global maximum,  $Max$ , where  $n = |V|$  and  $0 < c < 1$  is a constant.

*Proof:* 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 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, 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  $Z_0, Z_1, \dots, Z_{\tilde{n}}$  by setting  $Z_0 = E[X]$ , and, for  $1 \leq i \leq \tilde{n}$ ,  $Z_i = E[X | X_1, \dots, X_i]$ . It is clear

that  $Z_{\tilde{n}} = X$  and, for  $1 \leq i \leq \tilde{n}$ ,  $|Z_i - Z_{i-1}| \leq 1$ .

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

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

where  $\rho$  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 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 a message  $i$  from a max-root successfully reaches a non-max-root.  $Y_i = 0$  when either of the following events happen: (1) The message  $i$  fails in routing to its destination; (2) The message  $i$  is sent to another max-root, even though it successfully travels over the network. The probability of this event is at most  $\frac{(1-\rho)R_t \log n}{n}$  since w.h.p. the size of a ranking tree is  $O(\log n)$ . (3) The message  $i$  and at least another message are destined to the same non-max-root. As the probability of three or more messages destining to the same node is much smaller, we only consider the case that two messages select the same non-max-root. We conservatively exclude both two messages on their possible contributions to the increase of  $R_t$ . This event happens with a probability at most  $\frac{(1-\rho)R_t \log n}{n}$ .

Applying the union bound [16],

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

Since  $R_t \leq \frac{cn}{\log n}$  for any constant  $0 < c < 1/2$  (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, 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  $\Omega(\frac{cn}{\log n})$  roots will have the *Max*. ■

2) *Sampling procedure*: From Theorem 9, 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 the larger ranking trees could have obtained the *Max* in the gossip procedure with a higher probability due to this same non-uniformity. In the sampling procedure, a root without *Max* can obtain *Max* with higher probability through this non-uniform sampling.

**Theorem 10**: Running the Gossip-max algorithm on the overlay complete graph  $\tilde{G} = \text{clique}(\tilde{V})$  of  $G(V, E)$ , after the sampling procedure, w.h.p., all the roots will know the *Max*.

*Proof*: After the sampling procedure, the probability that none of the max-roots is 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, any root node will know the *Max* with probability at least  $1 - \frac{1}{n}$ . ■

3) *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, the Gossip-max algorithm in total takes  $O(\log n)$  rounds and  $O(n)$  messages for all the roots in a network to reach consensus on *Max* w.h.p.

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

## B. Performance of the Gossip-ave algorithm

In the case that the uniformity of gossip is held, it has been shown in [9] that on an  $m$ -clique with probability at least  $1 - \delta'$ , Gossip-ave (uniform push-sum in [9]) 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 with 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 [8] that by using the 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 the following, 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.

We need some definitions as in [9]. We define an  $m$ -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.,  $x_j = \mathbf{conv}_{sum}(j)$ , the local aggregate of the ranking tree rooted at node  $j$  computed by Converge-cast-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 [9] 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

$$\hat{x}_{ave,t,i} = \frac{(s_{t,i}/w_{t,i})}{(g_{t,i}/w_{t,i})},$$

where  $(s_{t,i}/w_{t,i})$  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 as  $z$ .

For the upper bound of the running time of the Gossip-ave algorithm, we have the following theorem.

**Theorem 11:** 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 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.

To prove this theorem we need some auxiliary lemmas. Due to the space limitation, we refer to appendix for their proofs.

**Lemma 12 (geometric convergence of  $\Phi$ ):** The conditional expectation  $E[\Phi_{t+1} | \Phi_t = \phi] = \frac{1}{2}(1 - \sum_{i \in \bar{V}} P_i^2)\phi < \frac{1}{2}\phi$ , 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.

**Lemma 13:** 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.

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

**Theorem 14 (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.,  $\|\frac{y_{t,z,y}}{\|y_{t,z}\|_1} - \frac{1}{m}\|_\infty \leq \epsilon$ , where  $\delta' > 0$  and  $\epsilon > 0$  are constants.

*Proof:* By Lemma 12, we obtain that  $E[\Phi_t] < (m-1)2^{-t} < m2^{-t}$ , as  $\Phi_0 = (m-1)$ . By Lemma 13, 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 the Markov 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  $\max_j |\frac{y_{t,z,y}}{\|y_{t,z}\|_1} - \frac{1}{m}| \leq \epsilon$ , we need the lower bound on the weight of node  $z$ . From Lemma 13,  $w_{t,z} = \|y_{t,z}\|_1 \geq 2^{-\tau}$  with probability at least  $1 - \frac{\delta'}{2}$ . Note that Lemma 13 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  $1 - \delta'$ ,  $\max_j |\frac{y_{t,z,y}}{\|y_{t,z}\|_1} - \frac{1}{m}| \leq \epsilon$ . ■

Now we are ready to prove Theorem 11.

### Proof of Theorem 11

*Proof:* From Theorem 14, 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,  $\|\frac{y_{t,z,y}}{\|y_{t,z}\|_1} - \frac{1}{m}\|_\infty \leq \epsilon$ . Let both  $\epsilon = n^{-\alpha}$  and  $\delta' = n^{-\alpha}$ ,  $\alpha > 0$ . Then  $T_{ave} = O(\log m + 2\alpha \log n) = O(\log n)$ .

Applying Hölder's inequality, we obtain

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

When  $x_j$  are all of 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* aggregate. Let  $s_{ave} > 0$  and  $g_{ave} > 0$ <sup>5</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 the Gossip-ave algorithm. ■

Hence 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.

## VI. DRR-GOSSIP ALGORITHMS

Putting together our results from the previous sections, we present Algorithm 7, the DRR-gossip-max algorithm, and Algorithm 8, 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* w.h.p. If necessary, a root then broadcasts *Max* to nodes within its tree.

The DRR-gossip-ave algorithm is more complicated 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

<sup>5</sup>By definition,  $g_{ave} > 0$ ; w.l.o.g., we can offset values to have  $s_{ave} > 0$ .

only guarantees that the root of the largest tree will have the *Ave* w.h.p. 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 members within its tree.

---

**Algorithm 7:** DRR-gossip-max

---

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

---

**Algorithm 8:** DRR-gossip-ave

---

- 1 Run  $DRR(G)$  to obtain  $\mathbb{F}$ , the ranking tree forest.
  - 2 Run Convergecast-sum( $\mathbb{F}, \mathbf{v}$ ) algorithm.
  - 3 Run Gossip-max( $G, \mathbb{F}, \tilde{V}, \mathbf{conv}_{sum}(*, 2)$ ) 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.
  - 4 Run Gossip-ave( $G, \mathbb{F}, \tilde{V}, \mathbf{conv}_{sum}$ ) algorithm.
  - 5 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.
  - 6 Every root broadcasts its value to all the nodes in its ranking tree.
- 

*A. Robustness and correctness under link failures*

A salient property of the DRR-gossip algorithm is its robustness against link failures. As long as the ranking tree forest is constructed, a link failure will neither incur any re-construction of the ranking trees nor stop the gossiping process from aggregate computation. There is no overhead on tree re-construction

and re-starting the computation. All the algorithms in each phase, namely, DRR, Converge-cast, and Gossip, can continue even though link failures happen.

During phase one (DRR algorithm), a node that fails to connect to its parent can just try its neighboring node of the second highest rank. A parent node which can not receive messages from one of its child nodes after a timeout can omit that child node. The child node then will try to connect to another node with lower rank or become a root. After a ranking tree is formed, in phases two and three, a child node which fails to connect to its parent node due to link failure simply becomes a new root having all its offspring in its new tree. In practice, the forest of ranking trees can be periodically renewed to adapt to a new network graph due to link failures or repairs. Another benefit of the periodical renewal of the ranking trees is load balancing. Root nodes have heavier work loads than other nodes. A renewal of ranking forest can change the set of roots in the network.

The correctness of the DRR-gossip-max under link failures is clear since the *Max* will eventually stay in some root no matter when and where the link failure happens as long as the graph  $G$  remains connected. For the DRR-gossip-ave, we reset the value vector  $(v_j, w_j) = (0, 0)$  in the Converge-cast-sum algorithm to ensure the mass conservation of Gossip-ave algorithm. In this way, a new root produced from a link failure and joining in the middle of the computation of Gossip-ave will not affect the computation results.

### B. Performance of the DRR-gossip algorithms

In previous sections we have showed the message and time complexities for each phase of the DRR-gossip algorithms. For the last step of the DRR-gossip algorithms, all the roots broadcast their values to nodes in their trees for a global consensus. This will take  $O(n)$  messages and  $O(\log n)$  time. To sum up, the DRR-gossip algorithm will take  $O(\log n)$  time and  $O(n)$  messages. In addition to the time and message complexities, it is often more important to know the transmission complexity. We discuss the transmission complexity in the following section.

1) *Complexity of transmissions:* In previous sections, we have discussed the time and message complexities for each phase. However, unlike in phase one and phase two, a message in phase three for gossip employs multi-hop transmissions. The routing cost,  $R(n)$ , i.e., the number of multi-hop transmissions for a message to reach its destination, is decided by the underlying routing mechanism which is not specified in this paper. Using shortest-path routing algorithms,  $R(n)$  is bounded by the diameter,  $D(G)$ , of the network graph  $G$ . In a Poisson random geometric graph,  $D(G) = n^{\frac{1}{2}}$ . There are many routing algorithms in the literature that can be applied. Among them the greedy geographic routing algorithm [3], [6] is

TABLE I  
 DRR-GOSSIP VS. OTHER GOSSIP-BASED ALGORITHMS ON POISSON RANDOM GEOMETRIC GRAPH.

Algorithm	time (steps)	messages	transmissions
neighboring gossip [1]	$O(n^2)$	$O(n^2)$	$O(n^2)$
geographic gossip [3]	$O(n \log^2 n \cdot R(n))$	$O(n \log n)$	$O(n \log^2 n \cdot R(n))$
uniform gossip [9]	$O(\log n \cdot R(n))$	$O(n \log n)$	$O(n \log n \cdot R(n))$
efficient gossip [8]	$O(\log n \log \log n \cdot R(n))$	$O(n \log \log n)$	$O(n \log \log n \cdot R(n))$
DRR-gossip [this paper]	$O(\log n \cdot R(n))$	$O(n)$	$O(n \cdot R(n))$

On a Poisson random geometric graph  $G(n, r(n))$ ,  $R(n) = 1/r(n) = O((n/\log n)^{1/2})$  by geographic routing.

especially suitable for DRR-gossip algorithms in that it does not require the sender to know the identifier of the receiver [3], [19]. On a Poisson random geometric graph  $G(n, r(n))$ , where  $r(n) = \Omega((\frac{\log n}{n})^{1/2})$ , it has been shown in [3] that a gossip message needs  $R(n) = O(\frac{1}{r(n)})$  steps (transmissions) to reach the selected node by the greedy geographic routing. Since the number of transmissions in the first two phases is  $O(n)$  which is dominated by the number of transmissions in phase three, employing the greedy geographic routing in phase three, the DRR-gossip algorithms need  $O(n \cdot R(n)) = O(\frac{n^{3/2}}{\log^{1/2} n})$  transmissions for all the nodes in the network to obtain the global aggregates.

2) *Comparison to other representative gossip-based algorithms:* To compare the DRR-gossip algorithms with other representative gossip-based algorithms, we provide a table for their performances on a Poisson random geometric graph. The neighboring gossip [1] is a variant of the standard uniform gossip: nodes only communicate with its one-hop direct neighbors. All the messages in neighboring gossip requires only local transmissions as the DRR algorithm. For both uniform gossip [9] and efficient gossip [8], every message requires  $R(n)$  multi-hop transmissions. However, for DRR-gossip, only messages in phase three need multi-hop routing. In the other two phases, each message requires only one “local one-hop transmission.”

It is clear that, in all performance categories, the DRR-gossip-algorithm is at least equal to or better than all the other representative gossip-based algorithms.

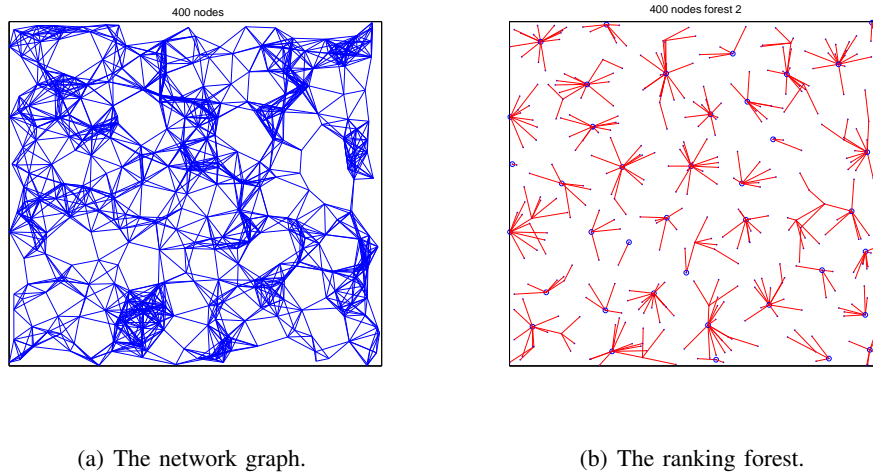


Fig. 1. An instance of the Poisson random geometric graph of size 400 nodes and an instance of the ranking forest formed on it by the DRR algorithm.

## VII. SIMULATIONS

We study the performance of the proposed algorithms by simulations on several instances of the Poisson geometric graph of various sizes and various connection topologies. In the simulation, every algorithm is executed 100 times to get the average performance. The accuracy criterion, i.e. the relative error of the average aggregate, is set to  $\epsilon = 1\%$ .

We present the comparison of four different gossip-based algorithms on *Ave* computation: (A) a variant of the DRR-gossip-ave algorithm, called DRR-gossip-ave (all meet), where Gossip-ave runs until all root nodes satisfy the accuracy criterion—hence, Data-spread is not needed after Gossip-ave; (B) our DRR-gossip-ave, the Algorithm 8 in section VI; (C) a variant of the uniform gossip algorithm (called neighboring uniform gossip) where a node uniformly at random selects one of its one-hop neighbors to gossip with; (D) uniform gossip [9] on the complete graph of  $n$  nodes. Since each algorithm has its own definition of “a round,” we only compare them by the total number of transmissions as an indicator of energy cost. For fair comparison, except the neighboring uniform gossip, which does not apply multi-hop routing, all the other three algorithms have been conservatively adjusted by the routing factor  $R(n) = (n/\log n)^{\frac{1}{2}}$ . The transmissions for DRR, Converge-cast-sum and the downward broadcasting from root nodes to tree members are also included in the results of (A) and (B). We assume wireless transmissions for the DRR algorithms.

Fig. 2 shows that the DRR-gossip-ave algorithm saves a tremendous number of transmissions, compared to the uniform gossip and the neighboring uniform gossip. Also in Fig. 2, the DRR-gossip-ave is compared

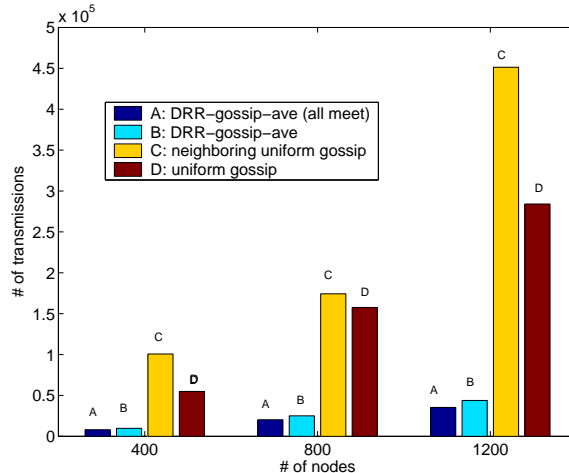


Fig. 2. The numbers of transmissions for the computation of aggregate *Ave* by different algorithms on Poisson random geometric graphs with various numbers of nodes.

to a centralized version of the DRR-gossip-ave, the DRR-gossip-ave (all meet), which is impractical but provided here for comparison. In the simulation of this variant version, an outside supervisor acts as a centralized authority that knows all the roots' values. (The dreadful overhead of collecting all roots' values to the centralized authority in every round is not included.) When all the roots satisfy the accuracy criterion, its Gossip-ave is then stopped. Note that the DRR-gossip-ave, Algorithm 8, by contrast, is executed in a totally distributed manner. A root cannot know whether the other roots and itself have already met the accuracy criterion or not. Thus, the Gossip-ave in DRR-gossip-ave runs a certain number of rounds, i.e.,  $O(\log n)$ , and then stops. Data-spread is executed afterward to guarantee that all the roots have the *Ave*. Simulation results show that our fully distributed DRR-gossip-ave algorithm utilizes about the same number of transmissions as the centralized version DRR-gossip-ave (all meet).

## VIII. CONCLUSION

In this paper, we presented a set of DRR-gossip algorithms to compute aggregates in a wireless sensor network. By the DRR –Distributed Random Ranking– algorithm local trees of small height, bounded by  $O(\log n)$ , are constructed so that the local aggregates can be quickly obtained at the roots of the ranking trees by Converge-cast. Then the root nodes perform gossip to obtain the global aggregate. With a smaller number of nodes participating in the gossip process, our DDR-gossip algorithms converge as fast as the standard uniform gossip algorithm but requires less messages and transmissions. Our analyses show that the proposed DRR-gossip algorithms require  $O(\log n)$  rounds,  $O(n)$  messages, and  $O(\frac{n^{3/2}}{\log^{1/2} n})$

transmissions on a Poisson random geometric graph, reducing the energy consumption by a factor of  $\frac{1}{\log n}$  over the standard uniform gossip algorithm.

As future work, our analyses on Poisson Random geometric graphs can be extended to general graphs. The locality approach can be extended to include more than one-hop neighbors (e.g., all nodes within a  $k$ -hop neighborhood can be considered during the tree building process). This can make DRR-gossip work more efficiently in sparse graphs such as grid and linear array. The analysis of our algorithms on a time-varying graph and the improvement by considering data correlation are also among future extensions.

## REFERENCES

- [1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Trans. on Netw.*, vol. 14, no. SI, pp. 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)*, Sep. 2006, pp. 987–1000.
- [3] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: efficient aggregation for sensor networks," in *IPSN*, 2006, pp. 69–76.
- [4] J. Gao, L. Guibas, N. Milosavljevic, and J. Hersherberger, "Sparse data aggregation in sensor networks," in *IPSN*, 2007.
- [5] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity," in *CDC*, 1998.
- [6] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *MobiCom*, 2000.
- [7] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *FOCS*, 2000, p. 565.
- [8] S. Kashyap, S. Deb, K. V. M. Naidu, R. Rastogi, and A. Srinivasan, "Efficient gossip-based aggregate computation," in *PODS*, 2006, pp. 308–317.
- [9] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. FOCS*, 2003.
- [10] B. Krishnamachari, D. Estrin, and S. Wicker, "Impact of data aggregation in wireless sensor networks," in *DEBS*, 2002.
- [11] S. Kwon and N. B. Shroff, "Paradox of shortest path routing for large multi-hop wireless networks," in *INFOCOM*, 2007.
- [12] P. Kyasanur, R. R. Choudhury, and I. Gupta, "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks," in *MASS*, 2006.
- [13] S. Lee, B. Bhattacharjee, and S. Banerjee, "Efficient geographic routing in multihop wireless networks," in *ACM Mobihoc*, 2005.
- [14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.
- [15] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in *PODC*, 2006, pp. 113–122.
- [16] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.
- [17] S. Muthukrishnan and G. Pandurangan, "The bin-covering technique for thresholding random geometric graph properties," in *ACM SODA*, 2005.
- [18] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *SenSys*, 2004.

- [19] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *MobiCom*, 2003.
- [20] R. Sarkar, X. Zhu, and J. Gao, "Hierarchical spatial gossip for multi-resolution representation in sensor network," in *IPSN*, 2007.
- [21] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks," in *SenSys*, 2004.
- [22] T. Stathopoulos, M. Lukac, D. McIntire, J. Heidemann, D. Estrin, and W. Kaiser, "End-to-end routing for dual-radio sensor networks," in *INFOCOM*, 2007.
- [23] D. Tian and N. Georganas, "Energy efficient routing with guaranteed delivery in wireless sensor networks," in *WCNC*, 2003.

## APPENDIX A

### THE PROOF OF LEMMA 1

*Proof:* Every node  $i$  independently and uniformly at random generates a rank  $rank(i) \in [0, c]$ ,  $c > 0$ . (We set  $c = 1$  in the DRR algorithm.) The probability for a node  $i$  with degree  $d_i$  in the graph  $G$  to become a root is

$$p_i = \int_0^c \left(\frac{1}{c}\right) \left(\frac{x}{c}\right)^{d_i} dx = \frac{1}{(d_i + 1)}.$$

■

## APPENDIX B

### THE PROOF OF LEMMA 6

*Proof:* Suppose a node  $i_k$  is the  $k$ -th node on a rooted tree path starting from its root and adjacent to  $i_{k-2}$ , the  $(k-2)$ -th node on the rooted tree path. Since  $i_{k-2}$ 's rank is higher than  $i_{k-1}$ 's,  $i_k$ 's parent node should be  $i_{k-2}$ , a contradiction. ■

## APPENDIX C

### THE PROOF OF LEMMA 7

*Proof:* We prove by induction on  $h$ . When  $h = 1$ , the lemma trivially holds. When  $h = 2$ , let the path be  $\{i_1, i_2, i_3\}$ . For this path to be a potential ranking path,  $i_3$  needs to be in the radio coverage area of  $i_2$  but not in the radio coverage area of  $i_1$ . The probability that  $i_3$  is in such an area is at most  $(\frac{\pi r^2}{3} + \frac{\sqrt{3}}{2} r^2) / \pi r^2 < 2/3 = (2/3)^{(h-1)}$ . When  $h > 2$ , let the path be  $\{i_1, \dots, i_h, i_{h+1}\}$ . For this path to be a potential ranking path, first, the path  $\{i_1, \dots, i_h\}$  needs to be a potential ranking path with probability at most  $(2/3)^{(h-1)}$ . Secondly, the node  $i_{h+1}$  needs to be in the radio coverage area of  $i_h$  but not in the radio coverage area of any precedent nodes, which happens with probability less than  $2/3$ . Thus the probability that an  $h$ -hop path is a potential ranking path is less than  $(2/3)^{(h-1)} \cdot (2/3) = (2/3)^h$ . ■

## APPENDIX D

## THE PROOF OF LEMMA 12

*Proof:* This proof is generalized from [9]. 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

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

■

## APPENDIX E

## THE PROOF OF LEMMA 13

*Proof:* In the case that the selection probability is of uniform distribution, it has been shown in [9] that on an  $m$ -clique with probability at least  $1 - \frac{\delta'}{2}$  after  $4 \log m + \log 2\delta'$  rounds, where  $\delta' > 0$  is a constant, a message originating from any node (through a random walk on the clique) will visit 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  $1 - \frac{\delta'}{2}$ . ■