Efficient Gossip-Based Aggregate Computation

Srinivas Kashyap*

University of Maryland
raaghav@cs.umd.edu

Supratim Deb, K. V. M. Naidu,
Rajeev Rastogi, Anand Srinivasan

Bell Labs Research India, Bangalore
{supratim, naidukvm,
rastogi, anands}@lucent.com

ABSTRACT

Recently, there has been a growing interest in gossip-based protocols that employ randomized communication to ensure robust information dissemination. In this paper, we present a novel gossip-based scheme using which all the nodes in an n-node overlay network can compute the common aggregates of MIN, MAX, SUM, AVERAGE, and RANK of their values using $O(n \log\log n)$ messages within $O(\log n \log\log n)$ rounds of communication. To the best of our knowledge, ours is the first result that shows how to compute these aggregates with high probability using only $O(n \log\log n)$ messages. In contrast, the best known gossip-based algorithm for computing these aggregates requires $O(n \log n)$ messages and $O(\log n)$ rounds. Thus, our algorithm allows system designers to trade off a small increase in round complexity with a significant reduction in message complexity. This can lead to dramatically lower network congestion and longer node lifetimes in wireless and sensor networks, where channel bandwidth and battery life are severely constrained.

1. INTRODUCTION

Many large-scale distributed applications require aggregate statistics (e.g., MIN, MAX, SUM, AVERAGE) to be computed over data stored at individual nodes. For example, in peer-to-peer systems [18, 20], the average number of files stored at each peer node or the maximum size of files exchanged between nodes is an important input to system designers for optimizing overall performance. Similarly, in sensor networks [17, 12], disseminating individual readings of temperature or humidity among all the sensor nodes, besides being too expensive, may also be unnecessary, and aggregates like MAX or AVERAGE may suffice in most cases. And finally, in a wireless network monitoring application using software probes deployed on mobile handsets to monitor performance, a service provider may be more interested in the abnormal measurements recorded by the probes like unusually low signal strength or atypically high application response times.

Depending on the application, the aggregate computation procedure must satisfy some of the following requirements.

- *Scale to a large number of nodes.* P2P systems and sensor networks can have millions of participating nodes. The procedure should be able to handle such massively distributed applications, and overall computation times should increase gradually and smoothly as new nodes join.

- *Be robust in the presence of failures.* Link and node reliability can be expected to be poor in wireless and sensor networks. Thus, the procedure must be resilient to node failures and message loss.

- *Incur low communication overhead.* Wireless links typically have low bandwidths, and in sensor networks, nodes have limited battery lives. As a result, the computation process should involve only a small number of message transmissions.

While the exact trade-off between the aforementioned requirements is not completely understood, an important question is whether it is possible to design efficient algorithms satisfying a sizable subset of the above-mentioned requirements.

For example, consider a centralized approach in which each node transmits its value to a central coordinator that then computes the aggregate. Clearly, this is extremely efficient in terms of message overhead. However, it is lacking in terms of scalability and reliability since the coordinator can quickly become a bottleneck and is a single point of failure. Similarly, though alternate approaches based on propagating aggregate computation up the nodes of a deterministic tree solve the scalability issue [8, 21], they are still susceptible to node and link failures.

To overcome the above-mentioned scalability and reliability problems, several researchers have proposed decentralized gossip-based schemes for computing various aggregates in overlay networks [11, 15, 9, 5, 4]. In gossip-based protocols, each node exchanges information with a randomly-chosen communication partner in each round. By their very nature, gossip-based schemes are robust; they are resilient to message failures as well as node failures, thus making them ideally suited for P2P, wireless and sensor networks with potentially poor link-reliability.

*This work was done while the author was visiting Bell Labs Research India, Bangalore.

PODS'06, June 26–28, 2006, Chicago, Illinois, USA.

Much of the early gossip work focused on using randomized communication to propagate a single message throughout a network of n nodes [7, 16, 10]. More recently, Kempe *et al.* [11] presented the first set of analytical results on computation of aggregate functions using randomized gossip. They analyzed a simple gossip-based protocol for computing sums, averages, quantiles and other aggregate functions. In their scheme for estimating averages, 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. Kempe *et al.* showed that these values converge to the true average in $O(\log n)$ rounds resulting in $O(n \log n)$ messages.

In this paper, we address the following question: *is it possible to reduce the message complexity of aggregate (max, sum, average, rank of an element) computation schemes from $O(n \log n)$ while relaxing the number of rounds to slightly exceed* $\log n$? We present a novel scheme to compute MIN, MAX, SUM, AVERAGE and RANK using $O(\log n \log\log n)$ rounds of communication and $O(n \log\log n)$ messages. To the best of our knowledge, ours is the first result that computes these various aggregates in a network with probabilistic link and node failures using only $O(n \log\log n)$ messages. Thus, compared to previous work [11], our scheme achieves a significant reduction in communication overhead at the cost of only a modest increase in the number of rounds. This can yield significant benefits in terms of lowering congestion and lengthening node lifetimes in bandwidth and energy-constrained environments like wireless and sensor networks.

Our algorithms achieve 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 representatives for each group, and then having the group representatives gossip among themselves. It is interesting to note that Karp *et al.* [10] proved that a single message cannot be spread in a network using less than $n \log\log n$ message exchanges for a class of algorithms referred to as *address-oblivious* algorithms. Although our algorithm is not "strictly" address-oblivious, this lower-bound result indicates that it might be hard to reduce the message complexity further without substantially increasing the number of rounds.

The rest of this paper is organized as follows. In Section 2, we present the underlying assumptions of our gossip framework. Section 3 describes prior related work. In Section 4, we describe our schemes for computing the various aggregates. Finally, we conclude in Section 5.

2. MODEL

The network consists of a set V of n nodes; each node $u \in V$ has a value denoted by $val(u)$. We are interested in computing aggregate functions like MIN, MAX, SUM, AVERAGE, RANK etc. of the node values.

The nodes communicate in discrete time-steps referred to as *rounds*. As in prior work on this problem [11, 10, 4], we assume that communication rounds are synchronized, and that all nodes can communicate simultaneously during a given round. The communication graph can be either *push-based* or *pull-based*. In the push-based model, a node selects a communication partner at random, and transmits information. A node can transmit to only one node in a round. In the pull-based model, a node chooses a communication partner at random and requests information. Thus, in this model, a node can receive from only one node in a round.

We assume that each node can communicate with every other node; each node chooses a communication partner independently and uniformly at random. A node u is said to *call* a node v if u chooses v as a communication partner. Once a call is established, we assume that information can be exchanged in both directions along the link.

Message sizes are bounded by $O(\log n + \log q)$ bits, where $\{1 \ldots q\}$ is the range of values at the nodes. The values at the nodes do not change during the execution of a query. When multiple nodes attempt to communicate with a node, then a connection is either queued up or rejected (if the queue size is already sufficiently large).

We assume the failure model of [11]. In particular, we assume two types of failures: **(i)** some fraction of the nodes may crash initially, and **(ii)** links are lossy and messages may get lost. Thus, while nodes cannot crash during the execution of the algorithm, communication can fail (either due to lossy links or due to initial node crashes) with a certain probability δ. W.l.o.g., we assume that δ is some constant such that $\frac{1}{\log n} < \delta < \frac{1}{8}$. Our results can also be proved without this assumption. In particular, for larger values of δ, we can make $O(1/\log(1/\delta))$ repeated calls to bring down the call failure probability below $\frac{1}{8}$. On the other hand, call failure probabilities lower than $\frac{1}{\log n}$ only make it easier to prove our claims.

We consider two query models: one in which only a single node that initiates the query is interested in knowing the result, and another in which all nodes need to know the query answer.

3. RELATED WORK

Randomized gossip-based schemes for spreading a single message update in a network date back to the work on epidemic algorithms by Demers *et al.* [7]. Initial work on spreading a single message using randomized gossip [7, 16] essentially show that a single message can be spread in a network of n nodes in $O(\log n)$ rounds and $O(n \log n)$ message transfers. Karp *et al.* [10] presented an improved algorithm and showed that even when a δ fraction of the nodes and messages can fail adversarially, all but a $O(\delta)$ fraction of the nodes in the network will have the message within $O(\log n)$ rounds and using only $O(n \log\log n)$ messages. They also gave lower bounds for the problem of single message dissemination.

Several works have considered the problem of deterministic in-network aggregation using trees [8, 21]. As shown in [11] and [4], such approaches are not resilient to node and message failures.

Kempe *et al.* [11] used gossip to compute aggregates of a distributed collection of values. They presented schemes that compute the sum and average of a distributed collection of values in $O(\log n)$ rounds and $O(n \log n)$ messages. They also extended the scheme to compute rank, select random samples, quantiles (using $O(\log^2 n)$ rounds and $O(n \log^2 n)$ messages) and several other aggregate functions. Our work aims to save an $O(\log n/\log\log n)$ factor in the number of messages used to compute these aggregates while giving up a $O(\log\log n)$ factor in the number of rounds.

Boyd *et al.* [4] considered non-uniform gossip where the probability of node i communicating with its neighbor j is P_{ij}. Their proposed algorithm is different from the standard uniform gossip model in that it considers an asynchronous

setting and in each asynchronous clock tick, it finds a random matching between the vertices. These vertices then average and update their values. They also presented a distributed scheme to find the optimal communication probabilities for each pair of vertices to ensure that the gossip algorithm converges at the fastest rate.

Finally, in [5], the authors employ a gossip-based approach to compute aggregates in a wireless sensor network setting. They present an algorithm with better performance based on a property of wireless transmissions where all nodes within the radio range can hear a broadcast.

4. OUR SCHEME

In this section, we describe our various schemes; the scheme for computing MAX is detailed in Section 4.2, and the schemes for computation of SUM, AVERAGE and RANK are described in Section 4.3. However, before delving into the details of our approach, we discuss some of our initial attempts that did not work.

4.1 Simple Approaches (that do not work)

In this subsection, we discuss two approaches that are simple, but have high message complexity. These will motivate the need for a more sophisticated solution.

Repeated rumor-spreading. Computation of MAX is in some sense similar to the rumor-spreading problem considered in [10]; however, we cannot simply invoke their results to spread the MAX value throughout the network because each node in the network holds a potentially important piece of information. Thus, naively running the rumor-spreading algorithm of [10] by considering each node's information as a rumor imposes a communication cost of $O(n^2 \log\log n)$.

Random query trees. Another attempt to compute MAX is to gossip for $O(\log\log n)$ rounds, which will result in the MAX spreading to $O(\log^p n)$ nodes (for some constant p) with only $O(n \log\log n)$ work. Then, the query node selects two nodes at random from the set of all nodes, and each selected node then repeats this process. A selected node marks itself to ensure it is not picked again. This construction goes on until the tree has $O(n/\log^{p-1} n)$ nodes. Once this happens the nodes will aggregate values up the tree and the query node will have the MAX w.h.p. [1] after $O(\log n)$ rounds. This is because the probability that the tree does not contain the MAX is at most $(1-\frac{\log^p n}{n})^{n/\log^{p-1} n} \leq 1/n$.

However, consider the communication complexity of this scheme. Consider the penultimate level of the tree. We have about $n/c\log^{p-1} n$ nodes at this level. Each of them needs to contact two *new* neighbors in the current round. The probability of a failed call for a node at this level is $\frac{1}{c\log^{p-1} n}$. To ensure this is $O(n^{-\alpha})$, each node has to draw $O(\log^p n/\log\log n)$ random samples. This means that, at the last level alone, the scheme has communication complexity $O(\frac{n}{\log^{p-1} n} * \frac{\log^p n}{\log\log n}) = O(n\log n/\log\log n)$. Note that we are ignoring the fact that there might be collisions between the samples chosen at the same level. This however only makes the tree smaller and our argument stronger. Further, if we consider message failures as well, the communication overhead will be even higher.

[1] w.h.p. = with probability at least $1 - O(n^{-\alpha})$ for some constant $\alpha > 0$.

4.2 Computation of MAX

We first describe the idealized version of our scheme to give the key intuition, and subsequently present the more practical version.

Intuition

Suppose that, incurring zero cost, we can divide all the nodes into $\frac{n}{\log n}$ groups, each of size $\log n$, and each group having a fixed root (let us call the roots *red* nodes and the others *blue* nodes). Now each red node can determine the MAX in its group, for example, by sequentially getting values from all nodes in the group and computing MAX. Note that this will need $O(\log n)$ rounds and $O(\log n)$ messages per group. Let us call this phase as *Grouping*.

Next, the red nodes gossip among themselves to compute MAX. For this, one can use the scheme in [11] to compute MAX for m nodes in $O(\log m)$ rounds and $O(m \log m)$ messages. Since, there are $\frac{n}{\log n}$ red nodes involved in gossiping, we get a complexity of $O(\log n)$ rounds and $O(n)$ messages. We refer to this phase as *Gossip*.

Finally, the red nodes propagate the MAX in their own groups, which has complexity similar to the Grouping phase. We call this phase as *Sampling/Propagation*. Essentially, if any node wishes to know MAX, it can do sampling, but, if all the nodes wish to know MAX, then few nodes can do sampling followed by propagation to the other nodes.

Note that this gives us an *ideal* scheme with $O(\log n)$ rounds and $O(n)$ messages. However, in order to achieve it in the presence of node and message failures, the Grouping phase must be performed in a distributed manner . Deterministic grouping is not possible due to initial node failures, which can potentially result in a red node failure if chosen deterministically. Therefore, we propose a randomized strategy to form the groups. Each node decides to be a red node independently with probability $\frac{1}{\log n}$. This gives roughly $O(\frac{n}{\log n})$ red nodes to start with. Now, the red nodes start forming groups by randomly contacting/being contacted by other nodes. This group formation happens in two phases: first *Push* and then *Pull*. Essentially, the Push and Pull phases simulate the *ideal* case, but in a distributed manner. Below, we describe these further in the context of our overall algorithm, and also argue that it is necessary to do both push and pull for making every node part of some group.

Overview of the Algorithm

With the above intuition in place, we are now ready to provide an overview of our scheme. The scheme consists of four phases: Push, Pull, Gossip and Sampling/Propagating.

1. **Push:** Initially all nodes are unmarked (no color assigned). Roughly $\frac{n}{\log n}$ decide to be red nodes. Each red node makes $O(\log n \log\log n)$ requests for other nodes to join. Each non-red node accepts one of the join requests randomly and marks itself blue. Also, each successful join updates the value at the red node to the max of the two values. At the end of this phase, at most $O(\frac{n}{\log n})$ nodes remain unmarked and each red node knows the MAX of its current group. The complexity of this phase is $O(\log n \log\log n)$ rounds and $O(n \log\log n)$ messages. Clearly, at the end of this phase, group size is at most $O(\log n \log\log n)$.

2. **Pull:** Each remaining unmarked node makes $O(\log n)$ calls and joins the first group it successfully calls (*i.e.*, the call is not rejected by the red node of the group). In each round, a red node accepts at most $O(\log \log n)$ calls and drops the remaining calls, so that its group size is at most $O(\log n \log \log n)$ (we will see in a moment why this is needed). At the end of this phase, all nodes are marked with some color w.h.p., and thus, part of some group. Also, each red node knows the MAX of its current group. The complexity of this phase is $O(\log n)$ rounds and $O(n)$ messages.

3. **Gossip:** Once grouping is done, the red nodes start the Gossip phase where they gossip among themselves. Since calls are made uniformly and randomly, a red node might end up calling a blue node. We can easily fix this by having the blue node return its parent red node to the caller so that it can be called in the subsequent round. Note that this does not make much difference to the protocol except for increasing the call failure probability. Also, since groups are not of the same size and the probability of a red node receiving a call is directly proportional to its group size, this is no longer uniform gossip, as considered in [11]. Using the fact that the group sizes are $O(\log n \log \log n)$, we show that at the end of this Gossip phase (after each red node has made $O(\log n)$ calls), the number of nodes with the MAX is $\Omega(n/(\log n \log \log n))$. The complexity of this phase is $O(\log n \log \log n)$ rounds and $O(n)$ messages.

4. **Sampling/Propagation:** Any node that wishes to compute MAX requests $\log n$ other nodes to "sample" $O(\log n \log \log n)$ nodes for the MAX. The maximum of these $O(\log^2 n \log \log n)$ samples is then reported as the MAX. We show that this is just the right number of samples for a node to get the MAX w.h.p. The message complexity of this phase is $\text{poly}(\log n)$.

   The preceding shows how any node interested in MAX can do sampling to obtain the MAX. If the MAX needs to be propagated to all the nodes, roughly $\Theta(\log n)$ nodes (this can be achieved by each node tossing a coin with probability $c \log n/n$) decide to be propagators and sample the max values. The result is then propagated to all the nodes using a modified version of the rumor spreading algorithm of [10]. This requires $O(\log n)$ rounds and $O(n \log \log n)$ messages.

Note that we have two phases (a Push phase followed by a Pull phase) during the group construction. This is necessary to keep the message complexity low; as observed in [10], push or pull applied alone to contact the unmarked nodes can result in excessive message transmissions. Basically, push is more efficient initially when a large number of unmarked nodes need to be contacted, while pull works better towards the end when fewer unmarked nodes remain. It is also important that the scheme for the group construction ensures that none of the constructed groups are too large. Otherwise, a single red node can receive a large number of simultaneous calls during the Gossip phase resulting in an increased number of rounds.

Description of the Algorithm

We now describe each of the four phases in greater detail. In each phase, communication between nodes takes place in rounds. Although, strictly speaking, each node is allowed to exchange information with only one partner in a given round, for convenience, in certain phases, we allow nodes to communicate with multiple nodes in a round. However, while calculating the total number of rounds for a phase, we count the multiple communications involving a single node as separate rounds. For instance, in Phase 2, a node can exchange information with $O(\log \log n)$ nodes (while dropping any extra requests) in a single round. Thus, even though Phase 2 has only $O(\log n)$ rounds, we compute the time complexity of Phase 2 as $O(\log n \log \log n)$ rounds.

In our analysis, we use several well-known results to bound the tail probability of a random variable (*e.g.*, Chernoff bounds, Azuma's inequality). These are included in the Appendix for easy reference.

Phase 1 Push

1: Each node independently decides to remain active with probability $1/\log n$ or else becomes inactive.
2: Let A be the set of all nodes that decide to remain active.
3: Each $u \in A$ marks itself *red*.
4: **for** $\frac{\log n \log \log n}{1-\delta}$ rounds **do**
5: Each $u \in A$ selects a node v independently and uniformly at random from the set of all nodes.
6: **for all** v that are unmarked **do**
7: v selects a node u at random from the red nodes that contacted v.
8: u and v exchange values and each stores the value $\max(\text{val}(u), \text{val}(v))$.
9: v points to u and marks itself *blue*.
10: **end for**
11: **end for**
12: All marked (*red* and *blue*) nodes become inactive.

Lemma 1. *The number of unmarked nodes at the end of Phase 1 is $\Theta(n/\log n)$ w.h.p.*

Proof. After step 1 of Phase 1, using standard Chernoff-bound type arguments, we have $n/\log n \pm \sqrt{n}$ *red* nodes w.h.p. Each of these *red* nodes tries to contact $\frac{\log n \log \log n}{1-\delta}$ nodes independently and uniformly at random (one connection attempt per round). Hence, the total number of connection attempts is $\frac{n \log \log n}{1-\delta}\left(1 \pm \frac{\log n}{\sqrt{n}}\right)$ w.h.p. Fix an unmarked node u. We now determine the probability that u receives none of these messages.

At each step, this can happen because of two reasons: either the connection failed, or the connection succeeded, but went to a different node. Let X_i be an indicator random variable (r.v.) such that X_i is 1 if node i was not contacted in Phase 1 by a red node and 0 otherwise. Then, the number of unmarked nodes at the end of Phase 1 is given by $X = \sum_{i=1}^{n} X_i$.

Let c denote the total number of connection requests. As shown above, w.h.p., we have the following.

$$\frac{n \log \log n}{1-\delta}\left(1 - \frac{\log n}{\sqrt{n}}\right) \leq c \leq \frac{n \log \log n}{1-\delta}\left(1 + \frac{\log n}{\sqrt{n}}\right)$$

Now, the probability that u is not contacted in any of the c

Phase 2 Pull

Let $B \subset V$ denote the set of all *unmarked* nodes.
for $\frac{2\log n}{\log(1/8\delta)}$ rounds **do**
  Each $u \in B$ selects a node v independently and uniformly at random from the set of all nodes.
  for all v that are *unmarked* **do**
    Drop all requests.
  end for
  for all v that are marked *blue* **do**
    Drop all but $\frac{4}{\delta(1-\delta)}\log\log n$ requests from nodes in B.
    for all $u \in B$ that contacted v, and are not dropped **do**
      v sends u a pointer to its *red* parent w.
      u contacts w.
    end for
  end for
  for all v that are marked *red* **do**
    Drop all but $\frac{4}{\delta(1-\delta)}\log\log n$ requests from nodes in B (including the requests forwarded by the *blue* nodes in the group).
    for all $u \in B$ that contacted v, and are not dropped **do**
      u points to v and marks itself *blue*.
      u and v exchange values.
    end for
  end for
end for

connection requests is given by the following.

$$\begin{aligned}\Pr[X_i = 1] &= (\delta + (1-\delta)(1-1/n))^c \\ &\leq \left(1 - \frac{1-\delta}{n}\right)^{\frac{n}{1-\delta}\log\log n\left(1-\frac{\log n}{\sqrt{n}}\right)} \\ &\leq e^{-\log\log n\left(1-\frac{\log n}{\sqrt{n}}\right)} \\ &= (\log n)^{-\left(1-\frac{\log n}{\sqrt{n}}\right)} \\ &\leq 2/\log n\end{aligned}$$

Similarly, we obtain that

$$\Pr[X_i = 1] \geq \left(1 - \frac{1-\delta}{n}\right)^{\frac{n}{1-\delta}\log\log n\left(1+\frac{\log n}{\sqrt{n}}\right)}$$

Since $e^t(1 - t^2/n) \leq (1 + t/n)^n$, we have

$$\begin{aligned}\Pr[X_i = 1] &\geq e^{-\log\log n\left(1+\frac{\log n}{\sqrt{n}}\right)}\left(1 - \frac{2(1-\delta)\log\log n}{n}\right) \\ &= \log n^{-\left(1+\frac{\log n}{\sqrt{n}}\right)}\left(1 - \frac{2(1-\delta)\log\log n}{n}\right) \\ &\geq \frac{1}{4\log n}\end{aligned}$$

Hence, it follows that $E[X] \in \Theta(n/\log n)$ and applying Azuma's inequality, we have

$$\Pr[|X - E[X]| > \epsilon E[X]] \leq 2e^{-\frac{\epsilon^2 n^2}{2n\log^2 n}}$$

Finally, using Lemma 4 (in Appendix) and setting $\epsilon = \frac{\log^{3/2} n}{\sqrt{n}}$, it follows that $\Pr[X > (1+\epsilon)2n/\log n] \leq 1/n^{\Omega(1)}$ and

Phase 3 Gossip

Let A be the set of all *red* nodes.
for $\left(\frac{3\log n}{(1-\delta)^2} + \log_{\frac{17}{16}} n\right)$ rounds **do**
  Each node in A selects a node independently and uniformly at random from the set of all nodes.
  for all *blue* nodes (v) that are contacted **do**
    Drop all but $\frac{2}{\delta(1-\delta)}\log\log n$ requests from nodes.
    for all *red* nodes (u) whose requests have not been dropped **do**
      v passes on its parent w's address to u.
      u then contacts v's *red* parent w.
    end for
  end for
  for all *red* nodes (v) that are contacted **do**
    Drop all but $\frac{2}{\delta(1-\delta)}\log\log n$ requests from nodes.
    for all *red* nodes (u) whose requests have not been dropped **do**
      u and v compare values. The node with the smaller value replaces its value with the higher one.
    end for
  end for
end for

$\Pr[X < (1-\epsilon)n/4\log n] \leq 1/n^{\Omega(1)}$. Because $0 < \epsilon < \frac{1}{2}$, it follows that $\frac{n}{8\log n} < X < \frac{4n}{\log n}$ w.h.p. □

Corollary 1. *The number of blue nodes at the end of Phase 1 is $\Theta(n - n/\log n)$ w.h.p.*

Proof. Follows directly from Lemma 1 by noting that the number of red nodes is $\Theta(n/\log n)$ w.h.p. □

Lemma 2. *Every node that was unmarked at the end of Phase 1 attaches itself to a red node by the end of Phase 2 w.h.p.*

Proof. Consider the event that an unmarked node fails to attach to a red node at the end of its $2\log n/\log(1/8\delta)$ calls. An unmarked node's call fails to attach it to a red node iff one of the following five bad events happen.

1. The call fails. This event occurs with probability δ.
2. The call succeeds but lands in another unmarked node. By Lemma 1, the number of unmarked nodes is at most $4n/\log n$. Therefore, the probability of this event is at most $4(1-\delta)/\log n$.
3. The call succeeds and lands in a blue node v. However, the subsequent call to attach to v's red parent fails. By Corollary 1, the number of blue nodes is at most $n - n/8\log n$. Hence, the probability of this bad event is at most $\delta\,(1-\delta)\left(1 - \frac{1}{8\log n}\right)$.
4. The call lands in a red node that has already received $\frac{4}{\delta(1-\delta)}\log\log n$ connections, and is dropped as a result. The probability of this event is at most the probability that a red node receives more than $\frac{4}{\delta(1-\delta)}\log\log n$ connections. The latter can be bounded from above as follows. Consider any red node r, and let g denote its group size. Then, for every round of the pull phase, because at most $\frac{4n}{\log n}$ unmarked nodes participate, the

probability that r receives more than $\frac{4}{\delta(1-\delta)}\log\log n$ is equivalent to the probability of getting $\frac{4}{\delta(1-\delta)}\log\log n$ successes in $\frac{4n}{\log n}$ binomial trials, each with success probability at most $\frac{g(1-\delta)^2}{n}$. The expected number of successes is at most $\frac{4g(1-\delta)^2}{\log n} < 4(1-\delta)\log\log n$. From Markov's inequality, the probability that the number of connections Y is greater than $\frac{4}{\delta(1-\delta)}\log\log n$ equals $\Pr[Y > \frac{4}{\delta(1-\delta)}\log\log n] \leq \delta(1-\delta)^2 \leq \delta$.

5. The call lands in a blue node that has already received $\frac{4}{\delta(1-\delta)}\log\log n$ connections, and is dropped as a result. This is similar to the previous case (for red nodes) but with success probability at most $\frac{1-\delta}{n}$. Thus, the probability of this bad event is at most $\delta(1-\delta)^2 \leq \delta$.

Putting all this together and observing that we have δ as some constant greater than $1/\log n$ and less than $1/8$, we obtain the following:

$$\begin{aligned}
&\Pr[\text{Node fails to attach}]\\
&\leq \left(\delta + \tfrac{4(1-\delta)}{\log n} + \delta(1-\delta)\left(1 - \tfrac{1}{8\log n}\right) + 2\delta\right)^{\frac{2\log n}{\log(1/8\delta)}}\\
&\leq (\delta + 4\delta(1-\delta) + \delta(1-\delta) + 2\delta)^{2\log n/\log(1/8\delta)}\\
&\leq (8\delta)^{2\log n/\log(1/8\delta)}\\
&= \frac{1}{n^2}
\end{aligned}$$

Applying a union bound over all the unmarked nodes, it follows that the probability of a node remaining unmarked is at most $\frac{1}{n\log n}$. In other words, every node that was unmarked at the end of Phase 1 attaches itself to a *red* node by the end of Phase 2 w.h.p. □

Remark. At the end of Phase 2, all the group sizes are $O(\log n \log\log n)$. This follows because the first phase runs for $O(\log n\log\log n)$ rounds, and in each round, a red node contacts at most one node. Further, the second phase runs for $O(\log n)$ rounds, and in each round, a red node contacts at most $O(\log\log n)$ nodes.

LEMMA 3. *At the end of Phase 3, at least* $\Omega\left(\frac{(1-\delta)n}{\log n\log\log n}\right)$ *nodes have the max value w.h.p.*

PROOF. A call is now equivalent to at most two successive calls (if the call lands in a blue node, then we have to make another call to connect to its red parent). Define the new call failure probability as $\delta' = 1-(1-\delta)^2$. Let ϕ_t be the number of *red* nodes with the max value at the end of round t of Phase 3. Let ϕ_0 denote the number of red nodes with the MAX before the start of Phase 3.

We first prove that $\phi_\tau > (1-\sqrt{2/3})\log n$ after $\tau = \frac{3\log n}{1-\delta'}$ rounds. If $\phi_0 > \log n$, we are done. Consider the case when $\phi_0 < \log n$. Since $\phi_0 < \log n$, if a call does not fail, then it will contact a new node w.h.p. Therefore we are only interested in finding the number of rounds before $\log n$ successful calls are made. Define $X_i = 1$ as the probability that call i from the max node succeeds. Let $X = \sum_{i=1}^{(3\log n)/1-\delta'} X_i$ be the total number of successful calls in $\frac{3\log n}{1-\delta'}$ rounds. The probability that a call succeeds is $\Pr[X_i = 1] = 1-\delta'$. Hence, $E[X] = 3\log n$.

Applying Azuma's inequality and setting $\epsilon = \sqrt{2/3}$:

$$\begin{aligned}
\Pr[|X - E[X]| > \epsilon E[X]] &< 2\exp\left(-\frac{\epsilon^2 E[X]^2}{2\log n}\right)\\
&= 2\exp\left(-\frac{9\epsilon^2\log n}{6}\right)\\
&= 2\exp(-\log n)\\
&= 2/n
\end{aligned}$$

Thus, w.h.p., at the end of $\tau = \frac{3\log n}{1-\delta'}$ rounds, we have $\phi_\tau > \left(1-\sqrt{2/3}\right)\log n$. Once this happens, as we show below, we enter an exponential growth phase and in the next $O(\log n)$ rounds, about $O(\frac{n}{\log n\log\log n})$ red nodes will have the MAX.

Let us re-number the rounds to simplify the expressions. Let us number the first round with at least $(1-\sqrt{2/3})\log n$ red nodes having the MAX as round 0 (note that such a round exists based on previous arguments). Now let us compute the value of ϕ_{t+1} given that in the current round ϕ_t red nodes have the MAX. We have $\phi_0 > \left(1-\sqrt{2/3}\right)\log n$. Since we have ϕ_t red nodes with MAX, there will be at least ϕ_t messages containing MAX in the current round (ignoring pull transmissions). In the rest of this proof, we only consider these messages containing MAX.

Let X_i be the indicator r.v. that denotes whether message i (containing MAX) is successful or not: a message is successful if the call succeeds and the MAX reaches a node that has not already been informed about the MAX. Then, $X = \sum_{i=1}^{\phi_t} X_i$ is the number of successful messages. We also have $1-\delta' = (1-\delta)^2$.

A message can fail iff one of the following happens.

1. Either the first or the second call fails. This event occurs with probability δ'.

2. Recall that some connections are dropped if a node receives more than $\frac{2}{\delta(1-\delta)}\log\log n$ connections. The probability of this event at the first node (a blue node) is at most $\delta(1-\delta)$ (the proof is very similar to the proof in Lemma 2 for the case where a blue node receives more than $\Omega(\log\log n)$ calls), while the probability of this event at the second contacted node (red node that is the parent of the blue node in the first call) is at most $\delta(1-\delta') < \delta(1-\delta)$. Thus, the total probability is at most $2\delta(1-\delta)$.

3. Both calls succeed, but the group contacted already contains the MAX. The probability of this event occurring is at most $\frac{(1-\delta')\phi_t\log n\log\log n}{n}$, because the maximum group size of a red node is $\frac{\log n\log\log n}{1-\delta}$ (by construction) and there are at most ϕ_t red nodes with MAX.

4. Both calls succeed, but the group contacted is simultaneously contacted by another node with the MAX. In this case, we conservatively assume that this message is wasted; in other words, if two messages reach the same node, then both messages are considered unsuccessful. This event occurs with probability at most $\frac{(1-\delta')\phi_t\log n\log\log n}{n}$.

Thus, the probability that a message is wasted can be upper-bounded as follows.

$$\Pr[X_i = 0] \leq \delta' + 2\delta(1-\delta) + \frac{2(1-\delta')\phi_t \log n \log\log n}{n}$$

Since we have $\phi_t < \frac{n}{16 \log n \log\log n}$,

$$\begin{aligned}
\Pr[X_i = 0] &\leq 1 - (1-\delta)^2 + 2\delta(1-\delta) + \tfrac{(1-\delta)^2}{8} \\
&= 1 - \tfrac{7}{8}(1-\delta)^2 + 2\delta(1-\delta) \\
&\leq \tfrac{1}{8} + \tfrac{15\delta}{4} \\
\Pr[X_i = 1] &\geq 1 - (\tfrac{1}{8} + \tfrac{15\delta}{4}) \\
&= \tfrac{7}{8} - \tfrac{15\delta}{4} \\
E[X] &\geq \left(\tfrac{7}{8} - \tfrac{15\delta}{4}\right)\phi_t
\end{aligned}$$

Since $\delta < 1/8$, the above expression implies that, in expectation, the number of nodes with MAX grows exponentially from one round to the next. To make this claim w.h.p., we apply Azuma's inequality to show a sharp concentration result for the expected number of successful messages.

$$\Pr[|X - E[X]| > \epsilon E[X]] < 2\exp\left(-\frac{\epsilon^2 (E[X])^2}{2\phi_t}\right)$$

Since $E[X] \geq \left(\frac{7}{8} - \frac{15\delta}{4}\right)\phi_t$, we have

$$\Pr[|X - E[X]| > \epsilon E[X]] < 2\exp\left(-\tfrac{\epsilon^2}{2}\left(\tfrac{7}{8} - \tfrac{15\delta}{4}\right)^2 \phi_t\right)$$

Since $\phi_t > \left(1 - \sqrt{2/3}\right)\log n$ and $\left(\frac{7}{8} - \frac{15\delta}{4}\right)^2 > 0$, setting $\epsilon = 1/2$, we obtain

$$\Pr\left[X < \tfrac{1}{2}\left(\tfrac{7}{8} - \tfrac{15\delta}{4}\right)\phi_t\right] < \frac{2}{n^{O(1)}}$$

Hence, w.h.p., the number of red nodes with MAX in round $t+1$ is

$$\begin{aligned}
\phi_{t+1} &> \phi_t + \tfrac{1}{2}\left(\tfrac{7}{8} - \tfrac{15\delta}{4}\right)\phi_t \\
&= \frac{23 - 30\delta}{16}\phi_t \\
&> \tfrac{17}{16}\phi_t
\end{aligned}$$

The last step above follows since $\delta < 1/8$. Thus, w.h.p., after $\left(\frac{3\log n}{1-\delta'} + \log_{\frac{17}{16}} n\right)$ rounds, at least $\Omega\left(\frac{(1-\delta)n}{\log n \log\log n}\right)$ red nodes will have the MAX. □

THEOREM 1. *Any node can compute the value of MAX using $O(n \log\log n)$ messages and $O(\log n \log\log n)$ rounds of communication.*

PROOF. From Lemma 3, it follows that at least $\frac{cn}{\log n \log\log n}$ red nodes (for some $c > 0$) have the MAX. Once Phases 1, 2 and 3 are complete, any node that is interested in MAX requests $\log n$ other nodes to "sample" the MAX. Each of these $\log n$ nodes then successfully samples $\frac{2}{c}\log n \log\log n$ nodes, and returns the maximum observed value to the querying node. The probability that none of the nodes with the MAX is sampled is at most $(1 - \frac{c}{\log n \log\log n})^{\frac{2}{c}\log^2 n \log\log n} < \frac{1}{n^2}$. Thus, the maximum of all the values obtained from the delegated nodes is the actual maximum w.h.p.

To bound the total number of communication rounds, observe that the number of communication rounds in each phase is $O(\log n \log\log n)$. This can be easily seen from the description of each phase.

To bound the total number of messages, we note that the number of messages in Phase 1 is $O(n \log\log n)$. By Lemma 1, the number of unmarked nodes that take part in Phase 2 is $\Theta(n/\log n)$, and hence the message complexity of Phase 2 is $O(n)$. The message complexity of Phase 3 is $O(n)$ because the number of red nodes is $\Theta(n/\log n)$, and each red node makes $O(\log n)$ calls. The message complexity of sampling is simply the number of samples that need to be drawn and is therefore $O\left(\log^2 n \log\log n\right)$. Thus, the overall message complexity is $O(n \log\log n)$. □

COROLLARY 1. *All nodes can compute the value of MAX using $O(n \log\log n)$ messages and $O(\log n \log\log n)$ rounds of communication.*

PROOF. If all nodes want MAX, then we first perform Phases 1, 2 and 3 as before. At the end of Phase 3, each node decides to be a propagator of MAX with probability $2\log n/n$. It can be shown that there will be $\Theta(\log n)$ propagators w.h.p. Next, each propagator gets MAX by sampling $\log n$ other nodes as described in the first paragraph of the proof of Theorem 1. After this, applying the result from [10], the propagator nodes can disseminate MAX to $\Omega((1-\delta)n)$ nodes using $O(n \log\log n)$ messages and $O(\log n)$ rounds.

After this, we can form groups as in Phases 1, 2, but with red nodes chosen from the nodes that already have MAX. As every node belongs to some group, every node also has MAX. Essentially, Phase 1 is performed with a minor modification. Only nodes who have MAX perform step 1 of Phase 1. Since δ is a constant and at least $\Omega((1-\delta)n)$ nodes have MAX, we still have $\Theta(n/\log n)$ red nodes at the end of this step. The rest of the Phase 1 proceeds as before. Phase 2 is unchanged. Since at the end of these two phases, each node has successfully communicated with a red node, all nodes have MAX. Phases 1 and 2 together take $O(\log n \log\log n)$ rounds and $O(n \log\log n)$ messages as seen above. □

4.3 Computation of SUM, AVERAGE, RANK

We now extend our MAX computation scheme to estimate the sum and average of node values. The key idea behind our algorithm is the following. First, groups of nodes are formed using Phases 1, 2 of the MAX computation algorithm. As before, the group heads will be referred to as the *red* nodes and the other nodes will be referred to as the *blue* nodes. During group formation, every *red* node maintains the size of its group, and the sum of the values within its group. Next, the group heads use the MAX computation algorithm to compute the maximum group size (for reasons that will become clear soon). Finally, the *red* nodes use the gossip-based Push-Sum algorithm in [11] to compute the average of node values. As we will show later, owing to the distinct group sizes, one can only ensure that the true-average resides in the *red* node with the largest group size. As the nodes already compute the largest group size, the *red* node with the largest group knows its identity and hence also knows that the value it has at the end of the protocol is the true-average (with a small relative error).

Algorithm 4 is a modified version of the Push-Sum protocol in [11]. Let $A = \{r_1 \ldots r_m\}$ be the set of red nodes after Phases 1, 2 of MAX computation. In our version, the Push-Sum protocol computes the average of the set of values $x_{r_1}, \ldots, x_{r_m}$ for only the red nodes. Note that every other node will be child of one of the red nodes, and in the modified protocol, any call to these nodes will be forwarded to its parent in A.

Algorithm 4 Modified-Push-Sum($x_{r_1} \dots x_{r_m}$)

1: $s_{u,t}$ is node u's value in round t. $s_{u,0} = x_u$ for each $u \in A$.
2: $w_{u,t}$ is node u's weight in round t. $w_{u,0} = 1$ for each $u \in A$.
3: **for** $O(\log m + \log(1/\epsilon))$ rounds **do**
4: Each node $u \in A$ independently and uniformly at random calls a node $v \in \{1 \dots n\}$. If v is a blue node, u is directed to the group head of v in the subsequent round. Note that the group head is a red node from A.
5: Let $Y_{v,t}$ be the set of nodes that called v in round t.
6: $s_{v,t} = s_{v,t-1}/2 + \sum_{u \in Y_{v,t}} s_{u,t-1}/2$.
7: $w_{v,t} = w_{v,t-1}/2 + \sum_{u \in Y_{v,t}} w_{u,t-1}/2$.
8: The current estimate of the average at node v is $s_{v,t}/w_{v,t}$.
9: **end for**

Algorithm 5 Compute-Average

1: Form groups as in Phases 1, 2 of MAX computation. Let $A = \{r_1, \dots, r_m\}$ be the set of red nodes, and for each red node $u \in A$, let g_u denote its group size (that is, the size of red node u and all the blue nodes attached to it).
2: Use the MAX finding scheme presented earlier to find the maximum size group for all the *red* nodes. Since all *red* nodes can find MAX, red node r can determine that it has the maximum sized group (break ties using node ids in the messages used for MAX computation).
3: Let $y_u = \sum_{v \in \text{group(u)}} val(v)$.
4: All the *red* nodes compute g_{avg} using Modified-Push-Sum($g_{r_1}, \dots, g_{r_m}$).
5: All the *red* nodes compute y_{avg} using Modified-Push-Sum($y_{r_1}, \dots, y_{r_m}$).
6: Node r computes the average $\hat{\mu} = y_{avg}/g_{avg}$, and communicates it to all nodes using the rumor-spreading scheme in [10].

Let α denote the true average of the x_{r_i}'s. Further, let r be the red node with the maximum group size and x_{avg} denote the average computed at node r.

THEOREM 2. *At the end of Algorithm 4, $|x_{avg} - \alpha|/\alpha \leq \varepsilon$ for any $\varepsilon > 0$. Furthermore, the total number of messages is $O(m(\log m + \log \frac{1}{\varepsilon}))$ and the number of rounds is $O(\log m + \log \frac{1}{\varepsilon})$.*

The proof of Theorem 2 is along similar lines as the proof in [11]. We will need some definitions as in [11] followed by a couple of key lemmas. Let $\vec{v}_{u,t}$ be a vector for node u in round t, of which the z^{th} component $v_{u,z,t}$ denotes the fraction of node z's value that is currently part of u's sum $s_{u,t}$ in round t. Thus, the sum at node u in round t, $s_{u,t} = \sum_z v_{u,z,t} x_z$. As shown in [11], the invariant $\sum_u v_{u,z,t} = 1 \; \forall z$ holds for Algorithm 4 as well. Similar to [11], define the following potential function for round t.

$$\Phi_t = \sum_{u,z} (v_{u,z,t} - \tfrac{w_{u,t}}{m})^2$$

Above, $m = \Theta(n/\log(n))$ is the number of *red* nodes, and $w_{u,t} = \sum_z v_{u,z,t}$. We now state the following lemma which is a variant of a similar result proved in [11].

LEMMA 4.1. *The following holds:*

$$E[\Phi_{t+1} | \Phi_t = \phi] = \tfrac{1}{2}(1 - \textstyle\sum_{u \in A} p_u^2)\phi \;, \qquad (1)$$

where $p_u = (1-\delta)^2 g_u / \sum_{v \in A} g_v$.

PROOF. This proof is very similar to the proof in [11]. The only difference is that p_u, the probability with which a node u is called in a round, is no longer uniform, but is skewed based on the group size g_u. □

We also need the following lemma which is somewhat different from [11].

LEMMA 4.2. *There exists a $\tau \in O(\log m)$ such that after $\tau' > \tau$ rounds of execution of Modified-Push-Sum, $w_{r,\tau'} \geq 2^{-\tau}$ w.h.p.*

PROOF. In [11], the above result is proved for all nodes and not just the node r with the maximum group size. The proof in [11] relies on the fact that when every node is contacted with uniform probability, each node receives a fraction of every other node's value after τ rounds. In our case, since the distribution is skewed, we cannot guarantee the lower bound on weight for each and every node. However, it is easy to see that the red node with the largest group size has a higher probability of being contacted. Thus, we can show that it receives a fraction of every other node's value after τ rounds, and thus its weight satisfies the lower bound. □

PROOF OF THEOREM 2. The proof is along the lines of [11], and employs the results of Lemmas 4.1 and 4.2. Due to Lemma 4.1, we get $\Phi_t \leq m2^{-t}$ (intuitively, Φ_t decreases by a constant factor in each round and is m when $t = 0$). By choosing $t = \log m + 2\log \frac{1}{\varepsilon} + 2\tau$ (τ as in lemma 4.2), we can show that $\Phi_t \leq \varepsilon^2 2^{-2\tau}$ for any $\varepsilon > 0$. This implies that $|v_{r,z,t} - \frac{w_{r,t}}{m}| \leq \varepsilon 2^{-\tau}$ for all z, or in other words $|\frac{v_{r,z,t}}{w_{r,t}} - \frac{1}{m}| \leq \frac{\varepsilon 2^{-\tau}}{w_{r,t}}$. Lemma 4.2 gives us the required lower bound of $w_{r,t} \geq 2^{-\tau}$ to have $|\frac{v_{r,z,t}}{w_{r,t}} - \frac{1}{m}| \leq \varepsilon$. Note that $\frac{v_{r,z,t}}{w_{r,t}}$ represents the contribution of z's value at r and w.h.p. this is approximately $\frac{1}{m}$ for all nodes. This implies that after $t \in O(\log m + \log \frac{1}{\varepsilon})$ rounds, the average x_{avg} computed at r has relative error at most ϵ. □

Algorithm 5 uses our Modified-Push-Sum procedure to compute y_{avg} and g_{avg}, the average group value and average group size, respectively. Let $\hat{\mu} = \frac{y_{avg}}{g_{avg}}$ be the estimate of average as computed by the red node r with the largest group, and let μ be the actual average of all the values $val(u)$ across nodes that did not fail at the beginning.

Our main result of this subsection is the following.

THEOREM 3. *At the end of Algorithm 5, $|\hat{\mu} - \mu|/\mu \leq 3\epsilon$ for any $\epsilon > 0$. Furthermore, the total number of messages is $O(n(\log\log n + \log \frac{1}{\epsilon}))$ and the number of rounds is $O(\log n \log\log n + \log \frac{1}{\epsilon})$.*

PROOF. It is easy to see that y_{avg} and g_{avg} are computed with relative error at most ϵ at node r (follows from Theorem 2). This implies that the relative error in the computation of $\hat{\mu} = y_{\text{avg}}/g_{\text{avg}}$ is at most 3ϵ. The message complexity of Modified-Push-Sum among $m = \Theta(n/\log n)$ *red* nodes is simply $O(m(\log m + \log \frac{1}{\varepsilon})) = O(n + n\log \frac{1}{\varepsilon})$ and the time complexity is $O(\log m + \log \frac{1}{\varepsilon}) = O(\log n + \log \frac{1}{\varepsilon})$. Thus the

message and time complexity are dominated by the formation of groups, and are $O(n \log\log n)$ and $O(\log n \log\log n)$, respectively. □

COROLLARY 2. *All nodes can find the true average of node values using $O(\log n \log\log n)$ rounds of communication and $O(n \log\log n)$ total messages.*

Given the average, the sum can be computed by just multiplying the average group value y_{avg} by the number of groups (which can be estimated using Modified-Push-Sum with only $x_r = 1$ and the remaining x_is equal to 0). Using this algorithm for computing the sum, computing the rank of a given value is also straightforward. The value x whose rank needs to be computed can be disseminated to all the nodes, and every node now keeps a new value which is 1 if its original value is less than x and 0 otherwise. Computing the sum of these new values gives the rank of x. Since dissemination and sum computation can be done in $O(\log n \log\log n)$ rounds and $O(n \log\log n)$ messages, we have the following corollary.

COROLLARY 3. *The rank of any value (the value is ranked among the values for nodes that did not fail) can be computed using $O(\log n \log\log n)$ rounds of communication and $O(n \log\log n)$ total messages.*

5. CONCLUSION

In this paper, we presented a novel gossip-based scheme for computing common aggregates like MIN, MAX, SUM, AVERAGE and RANK of node values using $O(n \log\log n)$ messages and in $O(\log n \log\log n)$ rounds of communication. To the best of our knowledge, this is the first result to show that these aggregates can be computed with high probability using only $O(n \log\log n)$ messages. Thus, compared to the previously best known results for distributed aggregate computation by Kempe et al., our scheme significantly reduces the communication overhead (a factor of $O(\log n / \log\log n)$) while causing only a modest increase (a factor of $O(\log\log n)$) in the number of rounds.

We conjecture that our results achieve the lower bound for message complexity in the gossip model since previous work by Karp et al. showed that even simpler problems like rumor dissemination require at least $\Omega(n \log\log n)$ messages regardless of the number of rounds. Formally deriving the lower bounds for message and time complexity for aggregate computation using gossip-style communication remains a topic for future work.

6. REFERENCES

[1] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. In *Proc. 27th ACM STOC*, pages 238–247, 1995.

[2] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proc. ACM SIGMOD*, 2003.

[3] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. In *Technical report, Computer Science Dept., Stanford University*, 2003.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *Proc. IEEE INFOCOM*, 2005.

[5] J. Chen, G. Pandurangan, and D. Xu. Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis. In *Proc. 4th IPSN*, 2005.

[6] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Proc. ACM SIGMOD*, pages 25–36, 2005.

[7] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. 6th ACM PODC*, pages 1–12, 1987.

[8] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proc. Conf. on Dependable Systems and Networks*, pages 433–442, 2001.

[9] M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proc. 24th ICDCS*, pages 102–109, 2004.

[10] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proc. 41st IEEE FOCS*, pages 565–574, 2000.

[11] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. 44th IEEE FOCS*, pages 482–491, 2003.

[12] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc networks. In *Proc. 5th OSDI*, 2002.

[13] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. ACM SIGMOD*, 2003.

[14] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, USA, 1995.

[15] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. 2nd ACM SenSys*, pages 250–262, 2004.

[16] B. Pittel. On spreading a rumor. *SIAM J. Applied Math.*, 47(1):213–223, February 1987.

[17] G. Pottie and W. Kaiser. Wireless integrated network sensors. *CACM*, 43:51–58, 2000.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, 2001.

[19] V. Stemann. Parallel balanced allocations. In *Proc. ACM SPAA*, pages 261–269, 1996.

[20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, pages 149–160, 2001.

[21] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM TOCS*, May 2003.

[22] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proc. 1st CIDR*, 2003.

APPENDIX

A. TECHNICAL DESIDERATA

We present some of the existing results in probability that we use extensively in our proofs. We use the following inequality to apply the method of bounded differences.

THEOREM 4 (AZUMA'S INEQUALITY [14]). *Let $Y_0, Y_1, \ldots$ be a martingale sequence such that for each k,*

$$|Y_k - Y_{k-1}| \leq c_k$$

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

$$\Pr[|Y_t - Y_0| \geq \lambda] \leq 2exp\left(-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}\right)$$

Since the method of bounded differences is applied frequently, we briefly revisit it here. The following content is from [14]. Let $X_1, \ldots, X_n$ be *any* sequence of random variables. Let $f(X_1, \ldots, X_n)$ be some function defined over these random variables. The function f is said to satisfy the Lipschitz condition if an arbitrary change in the value of any one argument of the function does not change the value of the function by more than 1. The sequence of random variables $Y_0 = E[f(X_1, \ldots, X_n)]$, $Y_i = E[f(X_1, \ldots, X_n)|X_1, \ldots, X_i]$ and $Y_n = f(X_1, \ldots, X_n)$ forms a martingale sequence. If f is Lipschitz, then for $1 \leq i \leq n$, $|Y_i - Y_{i+1}| \leq 1$. This condition is clearly satisfied for the sum of indicator random variables. We can then use Azuma's inequality to bound the probability that a sum of indicator random variables deviates from the expected value of that sum.

THEOREM 5 (CHERNOFF BOUNDS [14]). *Let X be a sum of independent and identically distributed 0/1 random variables. Let μ denote the expected value of X. Then we have:*

1. $\Pr[X \leq (1-\epsilon)\mu] \leq exp\left(-\frac{\mu\epsilon^2}{2}\right)$ *for all* $0 < \epsilon < 1$.
2. $\Pr[X \geq (1+\epsilon)\mu] \leq exp\left(-\frac{\mu\epsilon^2}{3}\right)$ *for all* $0 < \epsilon < 1$.
3. $\Pr[X \geq (1+\epsilon)\mu] \leq exp\left(-\frac{\mu\epsilon^2}{2+\epsilon}\right)$ *for all* $\epsilon > 1$.

From the statement above, we can infer that $\Pr[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\mu\epsilon}{2}\right)$ for all $\epsilon \geq 2$.

We also use the following simple fact frequently in the proofs.

LEMMA 4. *If $\mu_1 \leq \mu \leq \mu_2$, then:*

1. $\Pr[X > (1+\epsilon)\mu_2] \leq \Pr[X > (1+\epsilon)\mu] \leq \Pr[X > (1+\epsilon)\mu_1]$.
2. $\Pr[X < (1-\epsilon)\mu_1] \leq \Pr[X < (1-\epsilon)\mu] \leq \Pr[X < (1-\epsilon)\mu_2]$.

PROOF. For any two events A and B if $A \Rightarrow B$, we have $\Pr[A] \leq \Pr[B]$:

$$\begin{aligned} Pr[\text{A and B}] = \Pr[B|A]\Pr[A] &= \Pr[A|B]\Pr[B] \\ \Pr[A] &= \Pr[A|B]\Pr[B] \\ \Pr[A] &\leq \Pr[B] \end{aligned}$$

The lemma follows since $(X > (1+\epsilon)\mu_2) \Rightarrow (X > (1+\epsilon)\mu)$ and $(X > (1+\epsilon)\mu) \Rightarrow (X > (1+\epsilon)\mu_1)$. Similarly, we have $(X < (1-\epsilon)\mu_1) \Rightarrow (X < (1-\epsilon)\mu)$ and $(X < (1-\epsilon)\mu) \Rightarrow (X < (1-\epsilon)\mu_2)$. □