Gossip-Based Computation of Aggregate Information

David Kempe[*], Alin Dobra, and Johannes Gehrke[†]
Department of Computer Science, Cornell University
Ithaca, NY 14853, USA
{kempe,dobra,johannes}@cs.cornell.edu

Abstract

Over the last decade, we have seen a revolution in connectivity between computers, and a resulting paradigm shift from centralized to highly distributed systems. With massive scale also comes massive instability, as node and link failures become the norm rather than the exception. For such highly volatile systems, decentralized gossip-based protocols are emerging as an approach to maintaining simplicity and scalability while achieving fault-tolerant information dissemination.

In this paper, we study the problem of computing aggregates with gossip-style protocols. Our first contribution is an analysis of simple gossip-based protocols for the computations of sums, averages, random samples, quantiles, and other aggregate functions, and we show that our protocols converge exponentially fast to the true answer when using uniform gossip.

Our second contribution is the definition of a precise notion of the speed with which a node's data diffuses through the network. We show that this diffusion speed is at the heart of the approximation guarantees for all of the above problems. We analyze the diffusion speed of uniform gossip in the presence of node and link failures, as well as for flooding-based mechanisms. The latter expose interesting connections to random walks on graphs.

1. Introduction

Over the last decade, we have seen a revolution in connectivity between computers, and a resulting paradigm shift from centralized computation to highly distributed systems. For example, large-scale peer-to-peer (P2P) networks with millions of servers are being used or designed for distributed information storage and retrieval [9, 30, 32], and advances in hardware are leading to the augmentation of our physical environment with sensor networks consisting of hundreds of thousands of small sensor nodes [24, 28, 35]. Applications for such large-scale distributed systems have three salient properties that distinguish them from traditional centralized or small-scale distributed systems.

First, the dynamics of large-scale distributed systems are often significantly different. For example, in P2P networks, individual machines are often under the control of a large number of heterogeneous users who may join or leave the network at any time. Sensor networks often involve the deployment in inhospitable or inaccessible areas that are naturally under high stress (for example in battlefields or inside larger devices). Individual sensors may fail at any time, and the wireless network that connects them is highly unreliable. Thus, with massive distribution comes massive instability; consequently, the system as a whole must be highly fault-tolerant, as node and link failures or temporary communication disruptions are the norm rather than the exception.

Second, due to the large number of nodes and the volatility of the system, any reliance on central coordination will limit the system's scalability. *Gossip-based* (or *epidemic*) protocols are emerging as an important communication paradigm. In gossip-based protocols, each node contacts one or a few nodes in each round (usually chosen at random), and exchanges information with these nodes. The dynamics of information spread bear a resemblance to the spread of an epidemic [5, 10], and lead to high fault-tolerance and "self-stabilization" [8, 10, 34]. Gossip-based protocols usually do not require error recovery mechanisms, and thus enjoy a large advantage in simplicity, while often incurring only moderate overhead compared to optimal deterministic protocols, such as the construction of data dissemination trees. The guarantees obtained from gossip are usually probabilistic in nature; they achieve high stability under stress and disruptions, and scale gracefully to a huge number of nodes. In comparison, traditional techniques have absolute guarantees, but are unstable or fail to make progress during periods of even modest disruption.

Third, due to the large scale of the system, the values of

[*]Supported by an NSF Graduate Research Fellowship
[†]Supported by NSF Grants IIS-0133481 and CCR-0205452, and by gifts from Microsoft and Intel.

aggregate functions over the data in the whole network (or a large part of it) are often more important than individual data at nodes [17, 24, 34]. For example, in a sensor network with temperature sensors, we are often more interested in the average or median temperature measured by all sensors in an area rather than the single measurement at an individual sensor. In a sensor network with acoustic and vibration sensors, we may want to find out to which extent events of especially large noise or vibration are spatially or temporally correlated. In a P2P system, we may be interested in the total number of files, the average size of files stored, or quantiles about the amount of free space on the machines' disks. At the same time, communication bandwidth is often a scarce resource in decentralized settings, so the computation of aggregates should involve only small messages. In particular, any protocol collecting all local data at one given node will create communication bottlenecks, or a message implosion at that node.

Motivated by these considerations, we study the following class of *Node Aggregation* problems: in a network of n nodes, each node i holds a value x_i (or a set M_i of values), and the goal is to compute some aggregate function of these values (such as sums, averages, quantiles, etc.) in a decentralized and fault-tolerant fashion, while using small messages only. The Node Aggregation problem was recently defined formally by Bawa et al. [7], who restrict their attention to sums, averages, minima, and maxima. They define several natural notions of "validity" of a result in the presence of node failures, and show that "practical validity" (the weakest notion) is the only one that can be achieved under adversarial crash failures. They also present protocols for aggregation based mostly on building trees.

Here, we extend the study of aggregation beyond sums and averages, and show how to use gossip-based, completely decentralized protocols to compute random samples, quantiles, and answers to several other aggregate database queries in a decentralized fashion. We posit a weaker failure model than Bawa et al. [7], and obtain simple protocols for all of the above problems. We can show that all of our protocols converge to the true answer exponentially fast.

The Push-Sum protocol

Our first contribution is a simple and natural protocol Push-Sum for computing sums or averages of values at the nodes of a network. At all times t, each node i maintains a *sum* $s_{t,i}$, initialized to $s_{0,i} := x_i$, and a *weight* $w_{t,i}$, initialized to $w_{0,i} := 1$. At time 0, it sends the pair $(s_{0,i}, w_{0,i})$ to itself, and in each subsequent time step t, each node i follows the protocol given as Algorithm 1.

We show that with probability at least $1 - \delta$, the relative error in the approximation of the average has dropped to within ε, in at most $O(\log n + \log \frac{1}{\varepsilon} + \log \frac{1}{\delta})$ rounds

Algorithm 1 Protocol Push-Sum

```
1: Let $\{(\hat{s}_r, \hat{w}_r)\}$ be all pairs sent to $i$ in round $t-1$
2: Let $s_{t,i} := \sum_r \hat{s}_r$, $w_{t,i} := \sum_r \hat{w}_r$
3: Choose a target $f_t(i)$ uniformly at random
4: Send the pair $(\frac{1}{2}s_{t,i}, \frac{1}{2}w_{t,i})$ to $f_t(i)$ and $i$ (yourself)
5: $\frac{s_{t,i}}{w_{t,i}}$ is the estimate of the average in step $t$
```

(Section 3). Notice also that the lengths of all messages are bounded by the largest number of bits to encode the x_i, plus the number of rounds that the protocol has run.

If we are interested in computing the sum instead of the average, then we only need to apply a small change: instead of all nodes starting with weight $w_{0,i} = 1$, only one node (for instance the one at which the query was inserted) starts with weight 1, while all others start with weight 0. We than obtain exactly the same kind of approximation guarantees.

Push-Sum is a very natural protocol, yet the proof of the approximation guarantee is non-trivial and relies crucially on a useful property we term *mass conservation*: the average of all sums $s_{t,i}$ is always the correct average, and the sum of all weights $w_{t,i}$ is always n. For many natural protocols violating this property (for instance, Pull-based protocols), it is not difficult to verify that they cannot *converge* to the true results, in the sense that with some constant probability (possibly depending on n, but not the time t), the approximation stays bounded away from the true average. We will elaborate more on this issue in the full version.

Diffusion Speeds

The analysis of Push-Sum builds on an understanding of the *diffusion speed* of Uniform Gossip, characterizing how fast a value originating with any one node diffuses through the network. This notion is made precise in Section 2, although we hasten to add here that it does not in general coincide with the "broadcast time" [6, 18, 29]— the time it takes to disseminate a message to all nodes using point-to-point communication. Push-Sum is generic with respect to the underlying mechanism for communication, and its convergence speed corresponds in a precise sense to the diffusion speed of the communication mechanism.

We believe that this correspondence is of interest in itself, as the choice of communication mechanism will depend strongly on the actual network and its physical implementation. In sensor networks or P2P networks of relatively low degree, it may be easily feasible for a node to send a message to all of its neighbors at once, but point-to-point connectivity may be hard to achieve (in particular for sensor networks, where nodes usually use radio broadcasts). Other networks may support the abstraction of point-to-point communication, but the number of messages that a node can send in a round is limited, so that Uniform Gossip is preferred. Our approach permits us to design and analyze protocols independently of the actual communication mechanism; the convergence speed will be determined by the diffusion speed of the mechanism.

Hence, we analyze the diffusion speed not only for Uniform Gossip, but also for several other communication mechanisms. In particular, we analyze the impact of node failures and message loss on the diffusion speed of Uniform Gossip. In addition, we show that the diffusion speed for flooding techniques corresponds in a precise sense to the mixing time of a random walk on the network. Thus, we obtain good diffusion speeds for flooding on many P2P network architectures, which are known to possess good expansion properties.

Protocols for other problems

Building on ideas from the protocol Push-Sum and the notion of diffusion speeds, we design protocols for several more complex types of queries. Specifically, we show how to extend the analysis in a relatively straightforward way to answer many kinds of aggregate queries in databases [3, 4, 11, 13, 14, 15, 33], essentially any query that can be approximated well using linear synopses.

A somewhat more elaborate analysis shows that using only small messages and few rounds of the underlying communication mechanism, we can compute good random samples from among the union of all values held by the nodes. This, in turn, permits us to design a fast decentralized algorithm for computing quantiles (Section 4).

Related Work

Previously, several systems have been proposed that combine gossip-based communication with an explicit hierarchy on the nodes that allows for more easy aggregation [17, 34]. These approaches have been observed to scale well in practice, but require the maintenance of an explicit tree on nodes, and the election of leaders within subtrees.

The dissemination time of gossip distributions, and the time to broadcast one value to all nodes, has been studied in the past, see [18] for a survey. In particular, Frieze and Grimmett, and Pittel [16, 27] give precise constants in the $O(\log n)$ upper bound for Uniform Gossip. Feige et al. [12] consider random broadcasting on random graphs and hypercubes. Ravi [29] and Bar-Noy et al. [6] study approximation algorithms for the (NP-complete) problem of optimal broadcasting. Karp et al. consider tradeoffs between the number of rounds of gossip and the number of message duplicates that are sent [20].

The impact of message size restrictions on the ability to solve distributed computation tasks is investigated in [21]. It is shown that for the problems of locating the closest copy of a resource or building an approximate minimum spanning tree, different gossip distributions exhibit qualitatively different behavior when restricted to small messages.

2. Diffusion Speeds

We define a notion of *diffusion speed*, which lets us characterize precisely how quickly values originating with multiple sources diffuse evenly through a network, for a given communication mechanism.

Recall that in the basic Push-Sum protocol, each node chooses some other node uniformly at random, and passes on half of its sum and weight, keeping the other half. We generalize this idea to other communication mechanisms as follows: each node i, in each round t, chooses a non-negative *share* $\alpha_{t,i,j}$ for each node j, such that $\sum_j \alpha_{t,i,j} = 1$, and sends an $\alpha_{t,i,j}$ fraction of its sum and weight to each j. The choice of shares may be deterministic or randomized, and may or may not depend on the time t. We identify the communication mechanism with the shares $(\alpha_{t,i,j})_{t,i,j}$.

To track the diffusion of a node i's value under a given communication mechanism, we define — solely for the purpose of analysis — the following vector-based version of the protocol. Each node i locally maintains an n-dimensional *contribution vector* $\mathbf{v}_{t,i}$. Initially, it sends the vector $\mathbf{e}_i$ (the vector with 1 in the i-coordinate, and 0 in all others) to itself. In all subsequent rounds, the protocol is:

Algorithm 2 Protocol Push-Vector

```
1: Let $\{\hat{\mathbf{v}}_r\}$ be all vectors sent to $i$ in round $t-1$
2: Let $\mathbf{v}_{t,i} := \sum_r \hat{\mathbf{v}}_r$
3: Choose shares $\alpha_{t,i,j}$ for all nodes $j$
4: Send $\alpha_{t,i,j} \cdot \mathbf{v}_{t,i}$ to each $j$
```

The sums and weights in the Push-Sum protocol can be expressed in terms of contribution vectors as $s_{t,i} = \mathbf{v}_{t,i} \cdot \mathbf{x} = \sum_j v_{t,i,j} \cdot x_j$, and $w_{t,i} = \|\mathbf{v}_{t,i}\|_1 = \sum_j v_{t,i,j}$. Therefore, if $\mathbf{v}_{t,i}$ is (close to) a multiple of the all-1 vector $\mathbf{1}$, then $\frac{s_{t,i}}{w_{t,i}}$ is (close to) the true average, in a sense to be made precise in Section 3.

This motivates characterizing the diffusion speed of the communication mechanism by the speed with which the contribution vectors converge to multiples of the $\mathbf{1}$ vector. We define the relative error at node i at time t to be $\Delta_{i,t} = \max_j |\frac{v_{t,i,j}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n}| = \|\frac{\mathbf{v}_{t,i}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n} \cdot \mathbf{1}\|_\infty$. We say that $T = T(\delta, n, \varepsilon)$ is (an upper bound on) the *diffusion speed* of the mechanism defined by the distribution on shares $\alpha_{t,i,j}$ if $\max_i \Delta_{i,t} \leq \varepsilon$ with probability at least $1-\delta$, at all times $t \geq T(\delta, n, \varepsilon)$. That is, the relative errors in the contributions at all nodes i are bounded by ε.[1]

[1]Although our framework is presented with decentralized communica-

2.1. Uniform Gossip

In this section, we characterize the diffusion speed of Uniform Gossip.

Theorem 2.1 *The diffusion speed of Uniform Gossip is $T_U(\delta, n, \varepsilon) = O(\log n + \log \frac{1}{\varepsilon} + \log \frac{1}{\delta})$. Thus, with probability at least $1-\delta$, there is a time $t = O(\log n + \log \frac{1}{\varepsilon} + \log \frac{1}{\delta})$ such that the contributions at all times $t' \geq t$ and all nodes i are nearly uniform, i.e. $\max_j |\frac{v_{t',i,j}}{\|\mathbf{v}_{t',i}\|_1} - \frac{1}{n}| \leq \varepsilon$.*

In order to prove this theorem, first notice that the property of *mass conservation* mentioned in the introduction now translates to the following

Proposition 2.2 (Mass conservation) *Under the protocol Push-Vector with Uniform Gossip, at any time t, the sum of all of j's contributions at all nodes i is $\sum_i v_{t,i,j} = 1$, and hence the sum of all weights is $\sum_i w_{t,i} = n$.*

The proof of Theorem 2.1 is based on studying the potential function $\Phi_t = \sum_{i,j}(v_{t,i,j} - \frac{w_{t,i}}{n})^2$, the sum, over all i, of the variance of the contributions $v_{t,i,j}$ (as a function of j). The following Lemma guarantees geometric convergence of the absolute errors, by showing that Φ drops to less than half its previous value in expectation.

Lemma 2.3 *The conditional expectation of Φ_{t+1} is $\mathrm{E}[\Phi_{t+1} \mid \Phi_t = \phi] = (\frac{1}{2} - \frac{1}{2n})\phi$.*

Proof. Suppose we are given all contributions $v_{i,j} = v_{t,i,j}$ and weights $w_i = w_{t,i}$ at time t, as well as the calling assignment $f = f_t$ (i.e. node i calls node $f(i)$). Then, the new potential at time $t+1$ is

$$\begin{aligned}
\Phi_{t+1} &= \sum_{i,j}\left(\frac{1}{2}(v_{i,j} - \frac{w_i}{n}) + \sum_{k:f(k)=i}\frac{1}{2}(v_{k,j} - \frac{w_k}{n})\right)^2 \\
&= \frac{1}{4}\sum_{i,j}(v_{i,j} - \frac{w_i}{n})^2 + \frac{1}{4}\sum_{i,j}\sum_{k:f(k)=i}(v_{k,j} - \frac{w_k}{n})^2 \\
&\quad + \frac{1}{2}\sum_{i,j}\sum_{k:f(k)=i}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) \\
&\quad + \frac{1}{2}\sum_{i,j}\sum_{k\neq k':f(k)=f(k')=i}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n}) \\
&= \frac{1}{2}\Phi_t + \frac{1}{2}\sum_{i,j,k:f(k)=i}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) \\
&\quad + \frac{1}{2}\sum_{j,k}\sum_{k'\neq k:f(k)=f(k')}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n})
\end{aligned}$$

In the last step, we used the fact that each k appears in the sum for exactly one node i. Next, we take expectations, noting that the independent and uniform choice of communication partners ensures that $P[f(k) = i] = \frac{1}{n}$ for all nodes i and k, and $P[f(k) = f(k')] = \frac{1}{n}$ whenever $k \neq k'$. Thus, we obtain that

$$\begin{aligned}
&\mathrm{E}[\Phi_{t+1} \mid \Phi_t = \phi] \\
&= \frac{1}{2}\phi + \frac{1}{2}\sum_{i,j,k}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n})\ P[f(k) = i] \\
&\quad + \frac{1}{2}\sum_{j,k}\sum_{k'\neq k}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n})P[f(k) = f(k')] \\
&= \frac{1}{2}\phi + \frac{1}{2n}\sum_{i,j,k}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) \\
&\quad + \frac{1}{2n}\sum_{j,k,k'}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n}) \\
&\quad - \frac{1}{2n}\sum_{j,k}\left(v_{k,j} - \frac{w_k}{n}\right)^2 \\
&= (\frac{1}{2} - \frac{1}{2n})\phi + \frac{1}{n}\sum_j\sum_i(v_{i,j} - \frac{w_i}{n})\sum_k(v_{k,j} - \frac{w_k}{n}) \\
&= (\frac{1}{2} - \frac{1}{2n})\phi.
\end{aligned}$$

In the last step, we used the mass conservation property, to show that the sum is 0. ■

Proof of Theorem 2.1. By taking expectations repeatedly in Lemma 2.3, and using the fact that Φ_0 is at most n, we obtain that $\mathrm{E}[\Phi_t] \leq n \cdot 2^{-t}$.

With foresight, we choose $\tau = 4\log n + \log\frac{2}{\delta}$, and an absolute error of $\hat{\varepsilon} = \varepsilon^2 \cdot \frac{\delta}{2} \cdot 2^{-2\tau}$. Then, after running for $t = \log n + \log\frac{1}{\hat{\varepsilon}} = \log n + 2\log\frac{1}{\varepsilon} + \log\frac{2}{\delta} + 2\tau$ rounds, the expectation of Φ_t is at most $\mathrm{E}[\Phi_t] \leq \hat{\varepsilon}$. Hence, by Markov's Inequality, with probability at least $1 - \frac{\delta}{2}$, the potential Φ_t is bounded by $\varepsilon^2 \cdot 2^{-2\tau}$. In particular, $|v_{t,i,j} - \frac{w_{t,i}}{n}| \leq \varepsilon \cdot 2^{-\tau}$ for all nodes i.

In order to obtain a good bound on the relative error, we still have to give a lower bound on the weights at time t. Let i be the node with largest weight at time $t_0 = t - \tau$, so that $w_{t_0,i} \geq 1$. We look at how weight diffuses from i. Consider a "message" that originates with i at time t_0, and is forwarded by all nodes that have received it. A result by Frieze and Grimmett (Theorem 5.2 in [16]) shows that with probability at least $1 - \frac{\delta}{2}$, the message reaches all nodes in time at most $4\log n + \log\frac{2}{\delta}$.

If in the "message" experiment, at any time $t' \geq t_0$, a node j receives the message originating with node i, then in

tion in mind, it does not a priori preclude the possibility of aggregating data along the edges of a tree that is fixed. This would correspond to each node sending its entire vector (with share 1) to its parent during a first phase, and parents distributing the correct aggregate evenly among their children in a second phase. The reader is encouraged to verify that our aggregation protocols then coincide with the natural way of aggregating data in a tree, and the diffusion speed is twice the height of the tree.

the Push-Vector protocol, j receives weight at least $2^{t_0-t'}$, as the weight is divided by 2 in each round. In turn, j continues to divide its weight by at most 2 in each round, and therefore, with probability at least $1-\frac{\delta}{2}$, all nodes have weight at least $2^{-\tau}$ at time $t = t_0 + \tau$. Applying a Union Bound over the potential and weight events, and dividing by the weight $w_{t,i}$ gives us that with probability at least $1-\delta$, we have $|\frac{v_{t,i,j}}{w_{t,i}} - \frac{1}{n}| \leq \varepsilon$, at time $t = O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\varepsilon})$.

Finally, to see that the same bound holds for all later times, we use a simple inductive proof. The desired inequality at time t' can be rewritten as $n \cdot v_{t',i,j} \in [(1-\varepsilon n) \cdot w_{t',i}, (1+\varepsilon n) \cdot w_{t',i}]$. Substituting these bounds into the contributions at time $t'+1$, we obtain that

$$\begin{aligned} n \cdot v_{t'+1,i,j} &= \frac{1}{2} n \cdot v_{t',i,j} + \sum_{k:f_{t'}(k)=i} \frac{1}{2} n \cdot v_{t',k,j} \\ &\in \frac{1}{2}[(1-\varepsilon n) \cdot w_{t',i} + \sum_{k:f_{t'}(k)=i} (1-\varepsilon n) \cdot w_{t',k}, \\ &\quad (1+\varepsilon n) \cdot w_{t',i} + \sum_{k:f_{t'}(k)=i} (1+\varepsilon n) \cdot w_{t',k}] \\ &= [(1-\varepsilon n) \cdot w_{t'+1,i}, (1+\varepsilon n) \cdot w_{t'+1,i}]. \end{aligned}$$

This proves the inductive step, and hence the claim holds at all times $t' \geq t$, completing the proof. ■

Under some additional technical assumptions, we can obtain similar guarantees for distributions that are not "too different" from uniform. The diffusion speed will be slowed down essentially by the largest (multiplicative) deviation from the uniform distribution. The precise statement (and more cumbersome analysis) will be given in the full version of this paper.

2.2. Impact of faults on diffusion speed

The power of gossip-based techniques lies in their implicit robustness against faults: often, there is no need to distinguish failed nodes from temporary or permanent communication disruptions; nor is any specific recovery action required. In this section, we investigate the impact of several models of failures on the diffusion speed of the Uniform Gossip mechanism.

We consider the following two failure models: random message loss and initial failures of some nodes. We assume that nodes can detect when their message has not reached its destination (for instance by means of an acknowledgment mechanism). However, they are unable to discern the reason, i.e. do not know whether a message got lost, or the destination node has failed. The Push-Sum (and Push-Vector) protocol is modified as follows: if a node detects that its message was not delivered, it sends the message to itself instead. Other than that, the protocol is not altered at all. We let μ denote the loss rate for messages, or the fraction of nodes that have failed at the beginning of the computation.

Theorem 2.4 *If $\mu < 1$ is an upper bound on the probability of message loss in each round resp. the fraction of failed nodes, then the diffusion speed T' in the presence of failures satisfies $T'(\delta, n, \varepsilon) \leq \frac{2}{(1-\mu)^2} T(\delta, n, \varepsilon)$.*

The proof of Theorem 2.4 is deferred to the full version of this paper. Note that the Theorem does not give guarantees if nodes leave the network during the computation. If we assume that they leave in an orderly fashion (after sending all of their sum and weight (or vector) to another node first), then it can be seen fairly easily that the potential Φ_t at most doubles as a result of nodes leaving, when at most half of the nodes leave in one round. In particular, at most one additional round is required for convergence. If nodes crash during the computation, then our results do not carry over. We are currently investigating the question of whether the system can recover from crash failures, and what the impact on the diffusion speed is.

2.3. Flooding

In several topologies, such as P2P or wireless radio networks, point-to-point communication may not be a reasonable assumption, while the small number of neighbors (or the physical implementation of communication) may make it feasible to send one message to all neighbors at once. If the shares assigned to each neighbor are time-independent, then we can characterize all shares by a matrix $A = (\alpha_{i,j})_{i,j}$, where the entry $\alpha_{i,j}$ denotes what fraction of its vector $\mathbf{v}_{t,i}$ node i sends to j. (In most applications, the shares $\alpha_{i,j}$ will be the same for all nodes j— however, our results hold in greater generality.) Thus, $\mathbf{e}_j^T \cdot A^t$ is exactly the vector of contributions $(v_{t,i,j})_i$ from node j at all other nodes. As this is also the probability that the Markov Chain defined by A (and starting at node j) is at states i at time t, we can leverage a large body of work on the mixing speed of Markov Chains in analyzing diffusion speeds for flooding.[2]

To make this notion precise, we recall the following definitions. π denotes the vector of stationary probabilities of the Markov Chain. For two vectors $\mathbf{a}$, $\mathbf{b}$, we define the fraction $\frac{\mathbf{a}}{\mathbf{b}}$ pointwise, i.e. its i-coordinate is $\frac{a_i}{b_i}$. Most results on mixing times for Markov Chains use either the total variation distance or the $\|\cdot\|_2$-distance (which lie within constant factors of each other). The $\|\cdot\|_{2,\pi}$-distance from the stationary probability with respect to π is $\|\frac{\mathbf{e}_j^T \cdot A^t}{\pi} - \mathbf{1}\|_{2,\pi} := \left(\sum_i (\frac{\mathbf{e}_j^T \cdot A^t \cdot \mathbf{e}_i - \pi_i}{\pi_i})^2 \cdot \pi_i\right)^{1/2}$. Here, we are interested in the $\|\cdot\|_\infty$-distance, which is $\|\frac{\mathbf{e}_j^T \cdot A^t}{\pi} - \mathbf{1}\|_\infty := \max_i |\frac{\mathbf{e}_j^T \cdot A^t \cdot \mathbf{e}_i - \pi_i}{\pi_i}|$. Whenever the

[2]We assume that the Markov Chain so defined is actually ergodic.

Markov Chain is reversible, i.e. $\pi_i \alpha_{i,j} = \pi_j \alpha_{j,i}$, for all i, j, these two distance measures can be related in the following precise sense (see Lemma 2.4.6 from [31], and its analogue for discrete time Markov Chains):

Lemma 2.5 *[31] If A defines a reversible Markov Chain, then whenever $max_j \| \frac{\mathbf{e}_j^T \cdot A^t}{\pi} - \mathbf{1} \|_{2,\pi} \leq \delta$, we have $max_j \| \frac{\mathbf{e}_j^T \cdot A^{2t}}{\pi} - \mathbf{1} \|_\infty \leq \delta^2$.*

Hence, by doubling the time for the chain to run, we obtain equally good (or usually better) bounds on the $\|\cdot\|_\infty$-norm. Now, if t is a time such that $|\frac{v_{t,i,j} - \pi_i}{\pi_i}| \leq \frac{n\varepsilon}{n\varepsilon + 2}$, then a straightforward calculation shows that $|\frac{v_{t,i,j}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n}| \leq \varepsilon$. Thus, we obtain the following

Theorem 2.6 *Let T be a function such that $max_j \| \frac{\mathbf{e}_j^T \cdot A^t}{\pi} - \mathbf{1} \|_2 \leq \epsilon$, for all $t \geq T(n, \epsilon)$. Then, $T_F(n, \varepsilon) := 2T(n, \sqrt{\frac{n\varepsilon}{2 + n\varepsilon}})$ is an upper bound on the diffusion speed for the flooding mechanism defined by A.*

Theorem 2.6 allows us to leverage a large body of literature on the convergence speed of Markov Chains and Random Walks for the analysis of our aggregate computation protocols (see for instance [1, 23, 31]). In particular, whenever the underlying network is an expander, then we obtain diffusion speed $T(n, \varepsilon) = O(\log n + \log \frac{1}{\varepsilon})$. Several Peer-to-Peer topologies explicitly generate expander graphs [22, 26], and others [30, 32, 36] build hypercube-like networks which are expected to also have good expansion. Thus, we believe that our techniques will yield quick convergence on many P2P architectures.

2.4. Practical Considerations

We present the protocols Push-Sum, Push-Random (Section 4), etc. in terms of synchronous rounds, and with a synchronized starting point. The latter is certainly unnecessary. Instead, the node at which the query was posed may simply assign a unique identifier Q, and use the underlying communication mechanism to inform all other nodes of the query. Once a node first learns about a query, it adds its own value and weight to the received values, and then participates fully in the protocol. It is fairly straightforward to see that this does not affect the behavior, and convergence will be equally fast as before once all nodes have learned of the query.

The assumption of synchronous rounds is also not truly necessary for the definition of the protocols. Instead, nodes may simply follow their own clocks in deciding when to forward a share of their values or vectors. Mass conservation is still ensured; however, the analysis of the convergence speed for Uniform Gossip or flooding needs to be altered. We conjecture that the diffusion speed of the asynchronous version matches that of the synchronous version; the analysis, however, becomes more complex.

As another practical consideration, nodes will usually want to stop processing a query after some time, when the approximation guarantee is good enough. This can be done easily if the node $\hat{v}$ posing the query Q disseminates a message stating that Q is finished. However, it also raises the interesting question of how $\hat{v}$ will be able to decide that its approximation is good enough. If the number of nodes or the network topology are known, then $\hat{v}$ has exact bounds on the quality of approximation. However, nodes will often not know the entire topology in decentralized settings. We are currently investigating techniques by which nodes can locally estimate the quality of their current approximations.

Finally, many decentralized settings, such as sensor networks, also involve frequently changing data that needs to be monitored. In our setting, a node i can simply add the amount Δ by which its value x_i changed into its own sum $s_{t,i}$ at any change, and continue disseminating this new value. This will ensure that a snapshot over the entire system will always give the correct value, although the estimates at nodes may be temporarily incorrect. However, they will eventually converge to the true average, once no more changes happen for a sufficiently long time. Thus, our protocols implement the *Eventual Consistency* paradigm [34].

3. Averages, Sums, and Aggregates

Using Theorem 2.1, it is fairly straightforward to prove the convergence of Push-Sum to the true average that we claimed in the introduction.

Theorem 3.1
1. *With probability at least $1 - \delta$, there is a time $t_0 = O(\log n + \log \frac{1}{\varepsilon} + \log \frac{1}{\delta})$, such that for all times $t \geq t_0$ and all nodes i, the relative error in the estimate of the average at node i is at most $\varepsilon \cdot \frac{\sum_j |x_j|}{|\sum_j x_j|}$ (where the relative error is $\frac{1}{|\sum_j x_j|} \cdot |\frac{s_{t,i}}{w_{t,i}} - \frac{1}{n} \cdot \sum_j x_j|$).*

   In particular, the relative error is at most ε whenever all values x_j have the same sign.

2. *The sizes of all messages sent at time t are bounded by $O(t + \max_j \text{bits}(x_j))$ bits, where $\text{bits}(x_j)$ denotes the number of bits in the binary representation of x_j.*

Proof. Theorem 2.1 guarantees that with probability at least $1 - \delta$, there is a time $t_0 = O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\varepsilon})$ such that the contributions at all times $t \geq t_0$ satisfy $\| \frac{\mathbf{v}_{t,i}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n} \cdot \mathbf{1} \|_\infty \leq \frac{\varepsilon}{n}$. At time t, the estimate of the average at node i is $\frac{\mathbf{v}_{t,i} \cdot \mathbf{x}}{w_{t,i}}$. By applying the Triangle Inequality under the sum (Hölder's Inequality), we obtain the desired bound on the relative error at node i as follows:

$$\begin{aligned}\frac{|(\frac{\mathbf{v}_{t,i}\cdot\mathbf{x}}{w_{t,i}}) - \frac{1}{n}\sum_j x_j|}{|\frac{1}{n}\sum_j x_j|} &= n\cdot\frac{|(\frac{\mathbf{v}_{t,i}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n}\cdot\mathbf{1})\cdot\mathbf{x}|}{|\sum_j x_j|}\\ &\le n\cdot\frac{\|\frac{\mathbf{v}_{t,i}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n}\cdot\mathbf{1}\|_\infty\cdot\|\mathbf{x}\|_1}{|\sum_j x_j|}\\ &\le \varepsilon\cdot\frac{\sum_j |x_j|}{|\sum_j x_j|}.\end{aligned}$$

For the second part of the theorem, notice that values are only divided by 2 in each round. Hence, adding one bit of precision in each round suffices to keep full accuracy. ■

Of course, we may substitute any other communication mechanism instead of Uniform Gossip, and will only have to adapt the statement about the speed with which the values converge to the true average to the diffusion speed of the mechanism.

If only one node (instead of all nodes) starts with weight 1, then the value computed at the nodes converges to the sum of the x_j, instead of their average, as a very similar proof to the above shows. In fact, this technique can be used to count the number of nodes in the network in a decentralized manner, if each node starts with value $x_j = 1$, but only one node with weight $w_j = 1$ (the others having weight 0).

3.1. Linear Synopses

In turn, the ability to compute sums approximately is a powerful primitive for more complex queries. In Section 4, we see how to combine it with random sampling to compute quantiles. Another application is in answering database queries that can be approximated well using *linear synopses*, i.e. functions h on multisets such that $h(S_1 \cup S_2) = h(S_1) + h(S_2)$. Following the work by Alon et al. on using sketches to estimate join sizes [3, 4], there has been a large body of work suggesting such techniques for different kinds of queries, including aggregate functions over joins [11], L_p norms [13, 15], distinct values [4, 14] and histograms [33].

If each node computes the "local synopsis" of its own data, then the synopsis of the entire data can be approximated by adding these local synopses, using the Push-Sum protocol. From the synopses, nodes can in turn compute approximate function values. It is then important to analyze how the error introduced by Push-Sum affects the error in the final outcome. A subtle difficulty arises from the fact that in many cases (e.g. sketches), the expectation of the synopsis is 0, so the actual value of the sum may be close to 0, even though the local synopses are large in magnitude. Thus, Theorem 3.1 does not give useful bounds. Instead, we want to show that even if the relative error in the approximation of the sum of the synopses is large, the effect on the function under consideration is small.

Specifically, we assume that the actual function h is approximated by $\mathrm{E}[\hat{f}]$, where $\hat{f}(h_1, \ldots, h_K)$ is computed as a polynomial from K small linear synopses of the data (which in turn are the sums of local synopses $h_k(i)$ at nodes, and are often random variables). In addition, and most crucially, we assume that $\mathrm{E}[\hat{f}]$ can be rewritten as a polynomial $\sum_r \beta_r \mathrm{E}[\prod_{k=1}^{K}\prod_i h_k(i)]^{p_{r,k,i}}$ in terms of the expectations $\mathrm{E}[h_k(i)]$ of local synopses, such that every additive term is *always non-negative*. Notice that the non-negativity requirement does not need to apply to the synopses themselves, but rather to the monomials in $\mathrm{E}[\hat{f}]$. We write $d = \max_r \sum_{k,i} p_{r,k,i}$ for the maximum degree of any term in this polynomial.

3.1.1 The Push-Synopses protocol

In the gossip protocol for synopses-based approximation, each node i locally maintains a weight $w_{t,i}$, and a vector $\mathbf{s}_{t,i}$ of K synopses $s_{t,i,k}$. The synopses are initialized to $s_{0,i,k} = h_k(i)$, and the weights to $w_{0,i} = 0$ at all nodes except a specific starting node $\hat{v}$, which has $w_{0,\hat{v}} = 1$. At time step 0, each node sends the pair $(\mathbf{s}_{0,i}, w_{0,i})$ to itself. In each time step $t \ge 1$, each node i executes the following protocol.

Algorithm 3 Protocol Push-Synopses

```
1: Let $\{(\hat{\mathbf{s}}_r, \hat{w}_r)\}$ be all pairs sent to $i$ in round $t-1$
2: Let $\mathbf{s}_{t,i} := \sum_r \hat{\mathbf{s}}_r$, $w_{t,i} := \sum_r \hat{w}_r$
3: Choose shares $\alpha_{t,i,j}$ for each $j$
4: Send $(\alpha_{t,i,j}\cdot\mathbf{s}_{t,i}, \alpha_{t,i,j}\cdot w_{t,i})$ to each $j$
5: $\hat{f}(\frac{s_{t,i,1}}{w_{t,i}}, \ldots, \frac{s_{t,i,K}}{w_{t,i}})$ is the estimate at time $t$
```

The approximation properties of the Push-Synopses protocol are summarized by the following theorem. The proof is a fairly straightforward generalization of the proof of Theorem 3.1, and deferred to the full version of this paper.

Theorem 3.2 *Let T be the convergence speed of the chosen communication mechanism in step 2. Given ε and δ, let ε' be such that $(1+\varepsilon')^d \le 1+\varepsilon$, and $t \ge T(\delta, n, \varepsilon')$.*

Then, with probability at least $1-\delta$, the relative error at all nodes i incurred by the communication layer is at most ε, at all times $t' \ge t$. Here, the relative error is $\frac{1}{\mathrm{E}[\hat{f}(h_1,\ldots,h_K)]}\cdot |\mathrm{E}[\hat{f}(\frac{s_{t,i,1}}{w_{t,i}}, \ldots, \frac{s_{t,i,K}}{w_{t,i}})] - \mathrm{E}[\hat{f}(h_1, \ldots, h_K)]|$.

Notice that ε' will be only polynomially smaller than ε, because d is a constant. For most communication mechanisms, T grows at most logarithmically in ε, so the time required to reach error bounded by ε' will only be longer by a constant factor than that to reach error ε.

3.2. Applications

Linear synopses are a common technique for computing aggregate information in database settings. We illustrate the applicability with respect to sketching techniques for join size queries, and list several other applications.

If f_u and g_u denote the frequency with which the element u appears in the relations R_f and R_g, then the join size of the two relations is $\sum_u f_u g_u$. In their work on approximating frequency moments [3, 4], Alon et al. introduced *sketches* as powerful linear synopses of such frequency tables R_f, R_g. A sketch of R_f is a random variable $X_f = \sum_{u \in U} \xi_u f_u$, where each ξ_u is a random variable with values uniformly in $\{-1, 1\}$. A straightforward calculation shows that if the ξ_u are fourwise independent, then not only is the expectation $\mathrm{E}[X_f \cdot X_g] = \sum_u f_u g_u$, but the variance is also reasonably bounded.[3] Considering X_f and X_g as synopses of the data, we see that $\mathrm{E}[X_f \cdot X_g]$ is a sum of monomials that are always non-negative, and hence the techniques introduced above apply.

In addition to join-size queries, the following synopses-based aggregation techniques satisfy the required properties above, and can hence be combined with gossip-based aggregation protocols (an elaboration on the exact form of the synopses is again deferred to the full version of this paper):

- A substantial generalization of the techniques by Alon et al. [11] to include a much larger class of aggregate queries over multi-way joins in databases.
- Approximate histogram construction using sketches [33].
- L_p-norms, by using either range-summable hash functions with limited independence [13], or p-stable distributions [19].
- Distinct Value Queries, using hash functions [14, 4].

4. Random Sampling and Quantiles

A second important task besides computing sums and averages is to find random samples and quantiles of a multiset of elements. We assume that each node i holds a multiset M_i of m_i elements, and let $M = \bigcup_i M_i$ be the union of all these multisets, writing $m = |M| = \sum_i m_i$. We give a simple protocol with small messages for sampling elements nearly uniformly at random from M, and show how to combine it with Push-Sum to compute quantiles of M in a decentralized fashion.

[3]In order to apply these results in our distributed setting (and still obtain linearity), all nodes have to use the same multipliers ξ_u. Alon et al. show how to generate fourwise independent multipliers from a random seed of length logarithmic in the size of the universe [2, 4], so it suffices to disseminate this seed to all nodes i.

4.1. Random Sampling

In order to draw a random sample from M, each node i first samples an element $q_{0,i}$ from M_i uniformly at random[4], and then sends the pair $(q_{0,i}, m_i)$ to itself. Subsequently, each node executes the following protocol Push-Random in each round.

Algorithm 4 Protocol Push-Random

```
1: Let $\{(\hat{q}_r, \hat{w}_r)\}$ be all pairs sent to $i$ in round $t-1$
2: Let $w_{t,i} := \sum_r \hat{w}_r$
3: Choose $q_{t,i}$ at random from $\{\hat{q}_r\}$ with probabilities $\frac{\hat{w}_r}{w_{t,i}}$
4: Choose shares $\alpha_{t,i,j}$ for each $j$
5: Send $(q_{t,i}, \alpha_{t,i,j} \cdot w_{t,i})$ to each $j$
6: $q_{t,i}$ is the random element at time $t$
```

The protocol only uses small messages, and at any given time, each node holds some element $q_{t,i}$. The important question is how soon this element will be close to uniformly distributed. The convergence behavior is characterized by the following Theorem:

Theorem 4.1 *Let T be the diffusion speed of the underlying communication mechanism. Then, with probability at least $1-\delta$, after $T(\delta, n, \frac{\varepsilon}{(2+\varepsilon)n})$ rounds, the element at each node i will be ε-close to uniform, i.e. each element is selected with probability between $\frac{1-\varepsilon}{m}$ and $\frac{1+\varepsilon}{m}$.*

Proof. We show by induction on the time t that $P[q_{t,i} = q_{0,j}] = \frac{v_{t,i,j} \cdot m_j}{w_{t,i}}$ (recall that $\mathbf{v}_{t,i}$ are the contributions vectors). At time 0, this is clearly true, as the ratio is 1 for $j = i$, and 0 otherwise, and node i holds its own element.

For the inductive step, consider a node i, and all the pairs it receives; let k be the node that sent $\hat{q}_k = q_{t-1,k}$, which is chosen by i with probability $\frac{\alpha_{t-1,k,i} \cdot w_{t-1,k}}{w_{t,i}}$. Using conditional probabilities over all of the k, and the induction hypothesis $P[q_{t-1,k} = q_{0,j}] = \frac{v_{t-1,k,j} \cdot m_j}{w_{t-1,k}}$, we obtain that

$$\begin{aligned} P[q_{t,i} = q_{0,j}] &= \sum_k \frac{\alpha_{t-1,k,i} \cdot w_{t-1,k}}{w_{t,i}} \cdot \frac{v_{t-1,k,j} \cdot m_j}{w_{t-1,k}} \\ &= \frac{m_j}{w_{t,i}} \cdot \sum_k \alpha_{t-1,k,i} \cdot v_{t-1,k,j} \\ &= \frac{v_{t,i,j} \cdot m_j}{w_{t,i}}. \end{aligned}$$

Given a desired quality of approximation ε, we choose $\varepsilon' \leq \frac{\varepsilon}{(2+\varepsilon) \cdot n}$, and consider a time t at which the relative error in contributions at all nodes i is less then ε',

[4]If $M_i = \emptyset$, then i initializes $q_{0,i} = \perp$ for some special symbol $\perp$. As the $\perp$ element is always associated with weight 0, it will be overridden by any true element, and hence, we can ignore this case for the rest of the analysis.

i.e. $v_{t,i,j} \in \|\mathbf{v}_{t,i}\|_1 \cdot [\frac{1}{n} - \varepsilon', \frac{1}{n} + \varepsilon']$ for all j and i. Let q be any element in M (we consider all elements distinct here), and j the node such that $q \in M_j$. Then, node i holds q if and only if $q_{0,j} = q$, and $q_{t,i} = q_{0,j}$. Since these two events are independent, the probability is $\frac{1}{m_j} \cdot \frac{v_{t,i,j} \cdot m_j}{w_{t,i}} = \frac{v_{t,i,j}}{\sum_k v_{t,i,k} \cdot m_k}$. Using the bounds from the diffusion speed both in the denominator and the numerator, we obtain that

$$\begin{aligned} P[q_{t,i} = q] &= \frac{v_{t,i,j}}{\sum_k v_{t,i,k} \cdot m_k} \\ &\in \frac{1}{m} \cdot [\frac{1/n - \varepsilon'}{1/n + \varepsilon'}, \frac{1/n + \varepsilon'}{1/n - \varepsilon'}] \\ &\subseteq [\frac{1-\varepsilon}{m}, \frac{1+\varepsilon}{m}]. \quad \blacksquare \end{aligned}$$

4.2. Quantile Computation

Here, we phrase the problem of finding quantiles as that of actually finding the ϕ-largest element, with probability at least $1 - \delta$. Our algorithm is essentially a decentralized implementation of the simple randomized "Find" algorithm [25]. It starts with the entire (multi-)set of elements, and in each round chooses a *pivot element* from among the remaining elements uniformly at random. The algorithm then counts the number of elements larger resp. smaller than the pivot, and recurses in the corresponding subinterval. A fairly straightforward analysis shows that when the random samples are uniform, and the element counts are exact, then the expected number of iterations is bounded by $O(\log m)$, and the actual number of iterations is sharply concentrated around its expectation.

In the decentralized version, given below as Algorithm 5, one node (for instance the one at which the query was posed), is considered the *leader*, and decides when to enter the next phase of the protocol. It uses the underlying communication mechanism to broadcast the information about the next phase to all other nodes. The leader maintains a candidate interval I at all times; the interval is initialized to be the entire universe $(-\infty, \infty)$, and the algorithm terminates when the interval consists of a single point.

The following theorem states that within a logarithmic number of iterations of the **while** loop, the algorithm finds the ϕ-largest element, with high probability. The proof is a relatively straightforward combination of Chernoff and Union Bounds, and deferred to the full version.

Theorem 4.2 *1. With probability at least $1 - \delta$, the Distributed-Find algorithm finds the ϕ-largest element within $O((\log m + \log \frac{1}{\delta}) \cdot T(O(\frac{1}{\log m + \log(1/\delta)}), n, O(\frac{1}{m})))$ total rounds of communication (where T is the diffusion speed of the underlying communication mechanism).*

2. In particular, using Uniform Gossip, Distributed-Find finds the ϕ-largest element within $O((\log m + \log \frac{1}{\delta}) \cdot (\log n + \log m + \log\log \frac{1}{\delta}))$ rounds of communication.

Algorithm 5 Distributed-Find

```
1: Use Push-Sum to approximate the number $m$ of elements by $\mu$, to within $1 \pm \frac{1}{2}$, with probability at least $1 - \frac{\delta}{3}$
2: Let $\hat{m} := 2\mu$, $p := 3\min(\frac{\log(4/3)}{2\log\hat{m}}, \frac{1}{8\ln(3/\delta)})$
3: while interval $I = (a, b)$ has more than one point do
4:     Use Push-Random to select a random $q$ in $I$, within $\pm\frac{1}{2\hat{m}}$ of uniform, with probability at least $1 - \frac{p}{2}$
5:     Disseminate $I$ and $q$ to all nodes
6:     Approximately count the numbers $s_1$ and $s_2$ of elements in the intervals $I_1 = (a, q), I_2 = (q, b)$, to within relative error at most $\frac{1}{3\hat{m}}$, with probability at least $1 - \frac{p}{2}$. Round $s_1$ and $s_2$ to the nearest integers
7:     Update $I$ to the sub-interval containing the $\phi$-largest element according to the counts $s_1, s_2$
8: end while
```

5. Conclusions

In this paper, we have presented a novel framework for processing many types of aggregation queries in decentralized settings. Our approach uses small messages and gossip-style local communication to provide simple and fault-tolerant protocols.

The power of the approach also comes with liabilities. In particular, when the protocols use flooding on networks with slowly mixing random walks (for instance grid-like graphs), convergence of the protocols will be slow. This suggests trying to use (decentralized) techniques to learn more about the topology, and trying to adapt the mechanism to speed up communication. We consider the question of how to judiciously use long-range connections, or how to speed up random walks, a very interesting direction for future research.

A further direction is the development of protocols for other complex types of queries. In particular, it seems promising to use our techniques for iceberg queries (i.e. finding elements with outstandingly high frequency). Also, it would be desirable to develop techniques that allow nodes to estimate the current error of approximation without knowledge of the underlying network or communication mechanism.

We are currently validating our results with practical experiments on several network topologies. For uniform gossip and several Internet-like topologies, preliminary results are very encouraging. We plan to report on these results in detail in future work.

Acknowledgements. We would like to thank Alan Demers, Sharad Goel, Jon Kleinberg, Frank McSherry, Robbert van Renesse, and Laurent Saloff-Coste for useful discussions.

References

[1] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. Book in preparation - preprint available at `http://www.stat.Berkeley.EDU/users/aldous`.

[2] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7:567–583, 1986.

[3] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *JCSS*, 64:719–747, 2002.

[4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *JCSS*, 58:137–147, 1999.

[5] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, 1975.

[6] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Message multicasting in heterogeneous networks. *SIAM J. on Computing*, 30:347–358, 2001.

[7] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Stanford University, 2003. URL: http://dbpubs.stanford.edu/pub/2003-24.

[8] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM TOCS*, 17:41–88, 1999.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 311–320, 2000.

[10] 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. 7th ACM SOSP*, pages 1–12, 1987.

[11] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proc. 2002 ACM SIGMOD*, pages 61–72, 2002.

[12] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. *Random Structures and Algorithms*, 1:447–460, 1990.

[13] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate l^1-difference algorithm for massive data streams. In *Proc. 40th IEEE FOCS*, pages 501–511, 1999.

[14] P. Flajolet and N. Martin. Probabilistic counting algorithms for data base applications. *JCSS*, 31:182–209, 1985.

[15] J. Fong and M. Strauss. An approximate l^p-difference algorithm for massive data streams. In *Proc. 17th STACS*, pages 193–204, 2000.

[16] A. Frieze and G. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discr. Appl. Mathematics*, 10:57–77, 1985.

[17] 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.

[18] S. Hedetniemi, S. Hedetniemi, and A. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1988.

[19] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proc. 41st IEEE FOCS*, pages 189–197, 2000.

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

[21] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proc. 43rd IEEE FOCS*, pages 471–480, 2002.

[22] P. Keyani, B. Larson, and M. Senthil. Peer pressure: Distributed recovery from attacks in peer-to-peer systems. In *Intl. Workshop on Peer-to-Peer computing*, pages 306–320, 2002.

[23] L. Lovász. Random walks on graphs: a survey. In D. Miklos, V. T. Sos, and T. Szonyi, editors, *Combinatorics, Paul Erdős is Eighty, Vol. 2*, pages 353–398. János Bolyai Mathematical Society, Budapest, 1996.

[24] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proc. 5th Symp. on Operating Systems Design and Implementation*, 2002.

[25] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1990.

[26] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. In *Proc. 42nd IEEE FOCS*, pages 492–499, 2001.

[27] B. Pittel. On spreading a rumor. *SIAM J. Applied Math.*, 47:213–223, 1987.

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

[29] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proc. 35th IEEE FOCS*, pages 202–213, 1994.

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

[31] L. Saloff-Coste. Lectures on finite markov chains. In *Lecture Notes in Mathematics 1665*, pages 301–408. Springer, 1997. École d'été de St. Flour 1996.

[32] 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.

[33] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *Proc. 2002 ACM SIGMOD*, pages 428–439, 2002.

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

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

[36] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, 2001.