

# Computing Frequent Elements Using Gossip

Bibudh Lahiri\* and Srikanta Tirthapura\*

Iowa State University, Ames, IA, 50011, USA  
{bibudh,snt}@iastate.edu

**Abstract.** We present algorithms for identifying frequently occurring elements in a large distributed data set using gossip. Our algorithms do not rely on any central control, or on an underlying network structure, such as a spanning tree. Instead, nodes repeatedly select a random partner and exchange data with the partner – if this process continues for a (short) period of time, the desired results are computed, with probabilistic guarantees on the accuracy. Our algorithm for frequent elements is built by layering a novel small space “sketch” of data over a gossip-based data dissemination mechanism. We prove that the algorithm converges to the approximate frequent elements with high probability, and provide bounds on the time till convergence. To our knowledge, this is the first work on computing frequent elements using gossip.

## 1 Introduction

We are increasingly faced with data-intensive decentralized systems, such as large scale peer-to-peer networks, server farms with tens of thousands of machines, and large wireless sensor networks. With such large networks comes increasing unpredictability; the networks are constantly changing, due to nodes joining and leaving, or due to node and link failures. *Gossip* is a type of communication mechanism that is ideally suited for distributed computation on such unstable, large networks. Gossip-based distributed protocols do not assume any underlying structure in the network, such as a spanning tree, so, there is no overhead of sub-network formation and maintenance. A gossip protocol proceeds in many “rounds” and in each round, a node contacts a few randomly chosen nodes in the system and exchanges information with them. The randomization inherently provides robustness, and surprisingly, often leads to fast convergence times. The use of gossip-based protocols for data dissemination and aggregation was first proposed by Demers *et al.* [1].

We focus on the problem of identifying *frequent data elements* in a network using gossip. Consider a large peer-to-peer network that is distributing content, such as news or software updates. Suppose that the nodes in the network (or the network administrators) wish to track the identities of the most frequently accessed items in the network. The relevant data for tracking this aggregate are the frequencies of accesses of different items. However, this data is distributed

---

\* Supported in part by NSF grant CNS 0520102 and by ICUBE, Iowa State University.

throughout the network – in fact, even the number of accesses to a single item may not be available at any single point in the network. Our gossip-based algorithm for frequent elements can be used to track the most frequently accessed items in a low-overhead, decentralized manner. Another application of tracking frequent items is in the detection of a distributed denial of service (DDoS) attack, where many malicious nodes may team up to simultaneously send excessive traffic towards a single victim (typically a web server), so that legitimate clients are denied service. Detecting a DDoS attack is equivalent to finding that the total number of accesses to some server has exceeded a threshold. A distributed frequent elements algorithm can help by tracking the most frequently accessed web servers in a distributed manner, and noting if these frequencies are abnormally large. With a gossip-based algorithm this computation can proceed in a totally decentralized manner.

We consider two variants of the frequent elements problem, with absolute and relative thresholds. In the absolute threshold version, the task is to identify all elements whose frequency of occurrence is at least an absolute number (threshold), which is an user-defined parameter. In the relative threshold version, the task is to identify all elements whose frequency of occurrence is more than a certain fraction of the total size of the data, where the fraction (the relative threshold) is an user-defined parameter. In a distributed dynamic network, these two problems turn out to be rather different from each other.

Our algorithms work without explicitly tabulating the frequencies of different elements at any single place in the network. Instead, the distributed data is represented by a small space “sketch” that is propagated and updated via gossip. A sketch is a space-efficient representation of the input, which is specific to the aggregate being computed, and captures the essence of the data for our purposes. The space taken by the sketch can be tuned as a function of the desired accuracy. A complication with gossip is that since it is an unstructured form of communication, it is possible for the same data item to be inserted into the sketch multiple times as the sketch propagates. Due to this, a technical requirement is that the sketch should be able to handle duplicate insertions, i.e. it should be *duplicate-insensitive*. If the gossip proceeds long enough, the sketch can be used to identify all elements whose frequency exceeds the user defined threshold. At the same time, elements whose popularity is significantly below the threshold will be omitted (again, with high probability).

To summarize our contributions, we present the first work on computing frequent elements in a distributed data set using gossip. We present randomized algorithms for both the absolute threshold and the relative threshold versions of the problem. For each algorithm, we present a rigorous analysis of the correctness, and the time till convergence. Our analysis show that gossip-based algorithms converge quickly, and can maintain frequent elements in a network with a reasonable communication overhead. We also note that similar techniques can be used in estimating the number of distinct data items in the network. Due to space limitations, details of estimating the number of distinct elements are not presented here and can be found in the full version of the paper [2].

With a gossip protocol, communication is inherently randomized, and a node can never be certain that the results on hand are correct. However, the longer the protocol runs, the closer the results get to the correct answer, and we are able to quantify the time taken till the protocol converges to the correct answer, with high probability. Gossip algorithms are suitable for applications which can tolerate such relaxed consistency guarantees. Examples include a network monitoring application, which is running in the background, maintaining statistics about frequently requested data items, or the most frequently observed data in a distributed system. In such an application, a guaranteed accurate answer may not be required, and an approximate answer may suffice.

## 1.1 Related Work

Demers *et al.* [1] were the first to provide a formal treatment of gossip protocols (or “epidemic algorithms” as they called them) for data dissemination. Kempe, Dobra and Gehrke [3] proposed algorithms for computing the sum, average, approximately uniform random sampling and quantiles using uniform gossip. Their algorithm for quantiles are based on their algorithm for the sum – they choose a random element and count the number of elements that are greater and lesser than the chosen element, and recurse on smaller data sets until the median is found. Thus their algorithms need many instances of “sum” computations to converge before the median is found. A similar approach could potentially be used to find frequent elements using gossip. In contrast, our algorithms for frequent elements are not based on repeated computation of the sum, and converge faster.

Much recent work [4,5,6] has focused on computing “separable functions” using gossip. A separable function is one that can be expressed as the sum of individual functions of the node inputs. For example, the function “count” is separable, and so is the function “sum”. However, the set of frequent elements is not a separable function. Hence, these techniques do not apply to our problem.

The problem of identifying frequent elements in data has been extensively studied [7,8,9] in the database, data streams and network monitoring communities (where frequent elements are often called “heavy-hitters”). The early work in this is due to Misra and Gries [7], who proposed a deterministic algorithm to identify frequent elements in a stream, followed by Manku and Motwani [8], who gave randomized and deterministic algorithms for tracking frequent elements in limited space. The above were algorithms for a centralized setting.

In a distributed setting, Cao and Wang [10] proposed an algorithm to find the top- $k$  elements, where they first made a lower-bound estimate for the  $k^{th}$  value, and then used the estimate as a threshold to prune away elements which should not qualify as top- $k$ . Zhao *et al.* [11] proposed a sampling-based and a counting-sketch-based scheme to identify globally frequent elements. Manjhi *et al.* [12] present an algorithm for finding frequent items on distributed streams, through a tree-based aggregation. Keralapura, Cormode and Ramamirtham [13] proposed an algorithm for continuously maintaining the frequent elements over a network of nodes. The above algorithms sometimes assume the presence of a

central node, or an underlying network structure such as a spanning tree [12,13], and hence are not applicable where the underlying network does not guarantee reliability or robustness.

## 1.2 Organization of the Paper

In Section 2, we state our model and the problem more precisely. The algorithm and analysis for the case of absolute threshold in the asynchronous time model is presented in Section 3 and the case of relative threshold is presented in Section 4. Due to space constraints, we only present sketches of the proofs. Detailed proofs can be found in the full version of the paper [2].

## 2 Model

We consider a distributed system with  $N$  nodes numbered from 1 to  $N$ . Each node  $i$  holds a single data item  $m_i$ . Without loss of generality, we assume that  $m_i \in \{1, 2, \dots, m\}$  is an integer representing an item identifier. For data item  $v \in \{1, \dots, m\}$ , the frequency of  $v$  is denoted by  $f_v$ , and is defined as the number of nodes that have data item  $v$ , i.e.  $f_v = |\{j \in [N] : m_j = v\}|$ . Note that  $f_v$  may not be available locally at any node, in fact determining  $f_v$  itself requires a distributed computation. The task is to identify those elements that have large frequencies. We note that though we describe our algorithms for the case of one item per node, they can be easily extended to the case when each node has a (multi)set of items.

We consider the scenario of *uniform gossip*, which is the basic, and most commonly used model of gossip. Whenever a node  $i$  transmits, it chooses the destination of the message to be a node selected uniformly at random from among all the current nodes in the system. The selection of the transmitting node is done by the distributed scheduler, described later in this section. We consider two variants of the problem, depending on how the thresholds are defined.

**Absolute Threshold.** The user gives an absolute frequency threshold  $k > 1$  and approximation error  $\lambda$  ( $\lambda < k$ ). An item  $v$  is considered a frequent item if  $f_v \geq k$ , and  $v$  is an infrequent item if  $f_v < k - \lambda$ . Note that with a data set of  $N$  elements there may be up to  $N/k$  frequent elements according to this definition.

**Relative Threshold.** In some cases, the user may not be interested in an absolute frequency threshold, but may only be interested in identifying items whose relative frequency exceeds a given threshold. More precisely, given a relative threshold  $\phi$  ( $0 < \phi < 1$ ), approximation error  $\psi$  ( $0 < \psi < \phi$ ), an item  $v$  is considered to be a frequent item if  $f_v \geq \phi N$ , and  $v$  is considered an infrequent item if  $f_v < (\phi - \psi)N$ . According to this definition, there may be no more than  $1/\phi$  frequent items.

In a centralized setting, when all items are being observed at the same location, the above formulations of relative and absolute thresholds are equivalent, since the number of items  $N$  is known, and any absolute threshold can be converted into a relative threshold, or vice versa. However, in a distributed setting,

a threshold for relative frequency cannot be locally converted by a node into a threshold on the absolute frequency, since the user in a large distributed system may not know the number of nodes or the number of data items in the system accurately enough. Thus, we treat these two problems separately. The lack of knowledge of the network size  $N$  does not, though, prevent the system from choosing gossip partners uniformly at random. For example, Gkantsidis *et al.* [14] show how random walks can provide a good approximation to uniform sampling for networks where the gap between the first and the second eigenvalues of the transition matrix is constant.

At the end of the gossip, the following probabilistic guarantees must hold, whether for absolute or relative thresholds. Let  $\delta$  be a user-provided bound on the error probability ( $0 < \delta < 1$ ). With probability at least  $(1 - \delta)$ , every node reports every frequent item. With probability at least  $(1 - \delta)$ , no node reports an infrequent item. In other words, the latter statement implies that the probability that an infrequent item is incorrectly reported by a node in the system is less than  $\delta$ . Note that we present randomized algorithms, where the probabilistic guarantees hold irrespective of the input.

**Time Model.** For gossip-based protocols, time is usually divided into non-overlapping rounds. We consider the *asynchronous* model, where in each round, a *single* source node, chosen uniformly at random out of all  $N$  nodes, transmits to another randomly chosen receiver. The time complexity is the number of rounds, or equivalently, the number of transmissions, since in each round there is only one transmission. We consider the synchronous model in the full version of the paper.

**Performance Metrics.** We evaluate the quality of our protocols via the following metrics: the *convergence time*, which is defined as the number of rounds of gossip till convergence, and the *communication complexity*, which is defined as the number of bytes exchanged till convergence.

### 3 Frequent Elements with an Absolute Threshold

We now present an algorithm in the asynchronous model for identifying elements whose frequency is greater than a user specified absolute threshold  $k$ . Let  $S = \{m_i : i \in [N]\}$  denote the multi-set of all input values. The goal is to output all elements  $v$  such that  $f_v \geq k$  without outputting any element  $v$  such that  $f_v < k - \lambda$ . We first describe the high level intuition.

Our algorithm is based on random sampling. The elements of  $S$  are sampled in a distributed manner, and the sampled elements are disseminated to all nodes using gossip – the cost of doing so is small, since the random sample is typically much smaller than the size of the population. The sampling also ensures that frequent elements are exchanged more often during the later gossip phase. Intuitively, suppose we sample each element from  $S$  into a set  $T$  with probability

$1/k$ . For a frequent element  $v$  with  $f_v \geq k$ , we (roughly) expect one or more copies of  $v$  to be included within  $T$ . Similarly, for an infrequent element  $u$  with  $f_u < k - \lambda$ , we expect that no copy of  $u$  will be included in  $T$ . However, some infrequent elements may get “lucky” and may be included in  $T$  and similarly, some frequent elements may not make it to  $T$ . The probabilities of these events decrease as the sample size increases.

To refine this sampling scheme, we sample with a probability that is a little larger than  $1/k$ , say  $c/k$  for some parameter  $c$ . Finally, we select those elements that occur at least  $r$  times within  $T$ , for some parameter  $r < c$  that will be decided by the analysis. It turns out that  $c$  and  $r$  will need to depend on the approximation error  $\lambda$  as well as the threshold  $k$ . The smaller  $\lambda$  is, the greater should be the sampling probability, since we need to make a more precise distinction between the frequencies of frequent and infrequent elements. In the actual algorithm, we use a sampling probability of  $\frac{12k}{\lambda^2} \ln \frac{2}{\delta}$  – note that this is  $\Omega(\frac{1}{k})$  since  $\lambda < k$  and hence  $\frac{k}{\lambda^2} > \frac{1}{k}$ .

The precise algorithm for sampling and gossip is shown in Figure 1. There are three parts to this algorithm (and all others that we describe). The first part is the *Initialization*, where each node initialized its own sketch, which is usually through drawing a random sample. The next part is the *Gossip* portion, where the nodes in the system exchange sketches with each other. The algorithm only describes what happens during each round of gossip – it is implicit that such computations repeat forever. The third part is the *Query*, where we describe how a query for frequent elements is answered using the sketch. The accuracy of the result improves as further rounds of gossip occur. Through our analysis, we give a bound on the number of rounds after which frequent elements are likely to be found at all nodes.

---

**Input:** Data item  $m_i$ , error probability  $\delta$ , frequency threshold  $k$ , approximation error  $\lambda$

**1. Initialization**

- (a) Choose  $\rho$  as a uniformly distributed random number in  $(0, 1)$ .
- (b) If  $\rho < \frac{12k}{\lambda^2} \ln \frac{2}{\delta}$  then  $S_i \leftarrow \{(i, m_i)\}$ , else  $S_i \leftarrow \Phi$  /\* null set \*/

**2. Gossip**

In each round of gossip:

- (a) If sketch  $S_j$  received from node  $j$  then  $S_i \leftarrow S_i \cup S_j$
- (b) If node  $i$  is selected to transmit, then select node  $j$  uniformly at random from  $\{1, \dots, N\}$  and send  $S_i$  to  $j$

**3. Query**

When asked for the frequent elements, report all data items which occur more than  $r = \frac{12k^2}{\lambda^2} (1 - \frac{\lambda}{2k}) \ln \frac{2}{\delta}$  times in  $S_i$  as frequent elements.

---

**Fig. 1.** Gossip algorithm at node  $i$  for finding the frequently occurring elements with an absolute threshold  $k$

### 3.1 Analysis

We now analyze the correctness and the time complexity (proof sketches only) of the algorithm in Figure 1.

**Lemma 1.** *False Negative. If  $v$  is an element with  $f_v \geq k$ , then with probability at least  $1 - \delta$ ,  $v$  is returned as a frequent element by every node after  $20N \ln N$  rounds.*

**Sketch of proof:** A false negative can occur in one of two ways. (1) Either too few copies of  $v$  were sampled during initialization or (2) The sampled copies of  $v$  were not disseminated to all nodes during the gossip. We show that the first event is unlikely by an analysis of the sampling process using Chernoff bounds. We show that the second event is also very unlikely through an analysis of the asynchronous gossip in Lemma 4.  $\square$

**Lemma 2.** *False Positive. If  $u$  is an element with  $f_u \leq k - \lambda$ , where  $k^{\frac{3}{4}} \leq \lambda < k$ , then the probability that  $u$  is returned by some node as a frequent element is no more than  $\delta$ .*

**Sketch of proof:** A false positive can occur if both the following events occur: (1)  $r$  or more copies of  $u$  were sampled initially and (2) all  $r$  copies of  $u$  reach some node in the network through gossip. We show that the first event is very unlikely, if  $f_u \leq k - \lambda$ , and hence the intersection of the events is also unlikely.  $\square$

Now that we have proved Lemmas 1 and 2, it is natural to ask what happens to an element with frequency in the range  $[k - \lambda, k)$  of length  $\lambda$ . With our algorithm, such elements could be reported as frequent items, or not. Clearly, a smaller value of  $\lambda$  implies less uncertainty, but this comes at the cost of increased sampling probability, and hence greater communication complexity of gossip. For example, with  $k = 10^8$  and  $\lambda = 5 \times 10^6$ , the approximation error with respect to  $k$  is 5%. All elements with frequency greater than  $10^8$  will be reported (w.h.p) and all elements with frequency below  $0.95 \times 10^8$  will not be reported (w.h.p., once again), and the sampling probability is approximately  $4.8 \times 10^{-5} \times \ln \frac{2}{\delta}$ . This is the fraction of input items that are gossiped through the network in finding the frequent elements in the distributed data set.

*Analysis of the Gossip.* We now shift our attention to the gossip mechanism itself. Let  $\mathcal{T}$  denote the multi-set of all items sampled during initialization  $\mathcal{T} \subseteq S$  and  $|\mathcal{T}| \leq N$ . Consider a single sampled item  $\theta \in \mathcal{T}$ . Let  $T_\theta$  be defined as the number of rounds till  $\theta$  has been disseminated to all nodes in the network.

**Lemma 3.**  $E[T_\theta] = 2N \ln N + O(N)$ .

**Sketch of proof:** Let  $\xi_t$  be the set of nodes that have  $\theta$  after  $t$  rounds. Thus  $\xi_0$  has only one node (the one that sampled  $\theta$  during the initialization step). For  $t = 1 \dots N - 1$ , let random variable  $X_t$  be the number of rounds required to increase  $|\xi|$  from  $t$  to  $t + 1$ . We can write  $T_\theta = X_1 + X_2 + \dots + X_{N-1}$ . By noting that each  $X_t$  is a geometric random variable and using linearity of expectation we can arrive at the desired result. Further details are in the full version.  $\square$

Our proof for high-probability bounds on  $T_\theta$  use the following result about a sharp concentration for the *coupon collector* problem. Suppose there are coupons of  $M$  distinct types, and one has to draw coupons (with replacement) at random until at least one coupon of each type has been collected. Initially, it is very easy to select a type not yet chosen, but as more and more types get chosen, it becomes increasingly difficult to get a coupon of a type not yet chosen. The following result can be found in standard textbooks (for example, Motwani and Raghavan [15]).

**Theorem 1 (Folklore).** *Let the random variable  $\mathcal{C}$  denote the number of trials to collect at least one coupon of each of  $M$  types. Then, for any constant  $c \in \mathcal{R}$ ,  $\lim_{M \rightarrow \infty} \Pr[\mathcal{C} > M \ln M + cM] = 1 - e^{-e^{-c}}$ .*

**Lemma 4.**  $\lim_{N \rightarrow \infty} \Pr(T_\theta > 20N \ln N) = O(\frac{1}{N^2})$

**Sketch of proof:** The dissemination of  $\theta$  by gossip can be divided into two phases. The first phase starts when  $\theta$  is at a single node and continues until it has reached  $\frac{N}{2}$  distinct nodes. The second phase starts after  $\theta$  has reached  $\frac{N}{2}$  nodes and continues until it reaches  $N$  nodes. In the first phase, in each round of gossip, it is less likely to find a source node that has  $\theta$  and at the same time, it is more likely to find a destination that does not have  $\theta$ . Once  $\theta$  has reached  $\frac{N}{2}$  nodes, the situation reverses. We analyze the number of rounds required for these two phases separately. For each phase, we bound the random variable that defines the number of rounds in the phase by a simpler random variable that can be analyzed with the help of a coupon-collector type of argument. Combining the results from the two phases yields the desired result.  $\square$

For an item  $v$ , let  $T_v$  denote the number of rounds required to disseminate all copies of  $v$  to all nodes.

**Lemma 5.**  $\lim_{N \rightarrow \infty} \Pr[T_v > 20N \ln N] = O(\frac{1}{N})$

*Proof.* The proof follows from Lemma 4 using a union bound, and the fact that there are no more than  $N$  copies of  $v$ .

Lemmas 1, 2 and 5 together lead to the following theorem about the correctness of the algorithm.

**Theorem 2.** *Suppose the distributed algorithm in Figure 1 is run for  $20N \ln N$  rounds. Then, with probability at least  $1 - \delta$ , any data item with  $k$  or more occurrences will be identified as a frequent element at every node. With probability at least  $1 - \delta$ , any data item with less than  $k - \lambda$  occurrences will not be identified as a frequent element at any node.*

**Communication Complexity.** We next analyze the communication complexity, i.e. the number of bytes transmitted during the gossip. During the algorithm, the sizes of the messages exchanged start from one item and grow as the algorithm progresses. To avoid the complexity of dealing with different message sizes,

we separately analyze the total number of bytes contributed to gossip by each sampled item, and add these contributions together. Consider any sampled item  $\theta$ . We assume that transmitting any item  $(i, m_i)$  takes a constant number of bytes. Let random variable  $\mathcal{B}$  denote the number of bytes it takes to disseminate  $\theta$  among all nodes.

**Lemma 6.**  $E[\mathcal{B}] = O(N \ln N)$

**Sketch of proof:** Let  $\xi_t$  be the set of nodes that have  $\theta$  after  $t$  rounds. In each round of gossip,  $\theta$  may or may not be transmitted. Further each time  $\theta$  is transmitted,  $|\xi|$  increases only if the destination of the message is a node which is not already in  $\xi_t$ . We analyze  $\mathcal{B}$  as the number of transmissions of  $\theta$  till  $\xi$  includes all nodes. The details of the proof, using conditional probabilities, are presented in the full version.  $\square$

We can similarly get a high probability bound on  $\mathcal{B}$  (proofs in full version).

**Lemma 7.**  $\Pr[\mathcal{B} > 3N \ln N] = O(\frac{1}{N^2})$ .

Let  $\mathcal{Y}$  denote the total number of bytes that need to be exchanged for the whole protocol until the frequent elements have been identified. By combining Lemma 7 with an estimate on the number of sampled items, we get the following result about the communication complexity of the algorithm in Figure 1.

**Theorem 3 (Communication Complexity for Absolute Threshold)**

With high probability,  $\mathcal{Y} = O(\frac{N^2 k}{\lambda^2} \ln \frac{1}{\delta} \ln N)$

## 4 Frequent Elements with Relative Threshold

Given thresholds  $\phi$  and  $\psi$ , where  $\psi < \phi$ , the goal is to identify all elements  $v$  such that  $f_v \geq \phi N$  and no element  $u$  such that  $f_u < (\phi - \psi)N$ . Unlike the case of absolute threshold, there is no fixed probability that a node can use to sample data elements locally. For the same relative frequency threshold, the absolute frequency threshold ( $\phi N$ ), as well as the approximation error ( $\psi N$ ) increases with  $N$ . Thus if  $\phi$  and  $\psi$  are kept constant and  $N$  increases, then a smaller sampling probability will suffice, because of the analysis in 3. Since we do not have prior knowledge of  $N$ , we need a more “adaptive” method of sampling where the sampling probability decreases as more elements are encountered during gossip.

To design our sketch, we use an idea similar to *min-wise independent permutations* [16]. Each data item  $m_i, i = 1 \dots N$  is assigned a weight  $w_i$ , which is a random number in the unit interval  $(0, 1)$ . The algorithm maintains a sketch  $T$  of  $(m_i, w_i)$  tuples that have the  $t$  smallest weights  $w_i$ . The value of  $t$  can be decided independent of the population size  $N$ . The intuition is that if an element  $v$  has a large relative frequency, then  $v$  must occur among the tuples with the smallest weight. Maintaining these minimum-weight elements through gossip is easy, and if we choose a large enough sketch, the likelihood of a frequent element

appearing in the sketch a sufficient number of times is very high. We identify a value  $m$  as a frequent item if there are at least  $(\phi - \frac{\psi}{2})t$  tuples in  $T$  with  $m_i = m$ ; otherwise,  $m$  is not identified as a frequent element. The algorithm for the asynchronous model is described in Figure 2. The threshold  $t$  is chosen to be  $O(\frac{1}{\psi^2} \ln(\frac{3}{\delta}))$ .

---

**Input:** Data item  $m_i$ ; error probability  $\delta$ , relative frequency threshold  $\phi$ , approximation error  $\psi < \phi$

1. **Initialization:**

(a)  $t \leftarrow \frac{128}{\psi^2} \ln(\frac{3}{\delta})$

(b) Choose  $w_i$  as a uniformly distributed random number in the real interval  $(0, 1)$ ; set  $S_i \leftarrow \{(m_i, w_i)\}$

2. **Gossip**

In each round of gossip:

(a) If sketch  $S_j$  is received from node  $j$  then

i.  $S_i \leftarrow S_i \cup S_j$

ii. If  $|S_i| > t$  then retain  $t$  elements of  $S_i$  with smallest weights

(b) If node  $i$  is selected to transmit, then select node  $j$  uniformly at random and send  $S_i$  to  $j$

3. **Query**

When queried for the frequent elements, report every value  $v$  such that at least  $(\phi - \frac{\psi}{2})t$  (value, weight) tuples exist in  $S_i$  with value equal to  $v$

---

**Fig. 2.** Gossip algorithm at node  $i$  for finding the frequently occurring elements with a relative threshold

#### 4.1 Analysis

The proofs of most of the following lemmas appear in the full version. Let  $\tau$  denote the  $t^{\text{th}}$  minimum element among the  $N$  random values  $\{w_i, i = 1 \dots N\}$ . The next lemma shows that  $\tau$  is sharply concentrated around  $\frac{t}{N}$ .

**Lemma 8.** For  $t = \frac{128}{\psi^2} \ln(\frac{3}{\delta})$ ,  $\tau$  satisfies the following properties: (1)  $\Pr[\tau < \frac{t}{N}(1 - \frac{\psi}{4})] < \frac{\delta}{3}$  and  
(2)  $\Pr[\tau > \frac{t}{N}(1 + \frac{\psi}{4})] < \frac{\delta}{3}$

We now present a bound on the dissemination time of the smallest weights. Let  $T_t$  denote the time taken for the  $t$  smallest weights to be disseminated to all nodes.

**Lemma 9.**  $\Pr[T_t > 20N \ln N] \leq O(\frac{1}{N})$ .

*Proof.* The proof follows by using the union bound along with Lemma 4.

The following lemmas provide upper bounds on the probabilities of finding a false negative and a false positive respectively, by the algorithm described in Figure 2.

**Lemma 10.** *Suppose the distributed algorithm in Figure 1 is run for  $20N \ln N$  rounds. Then, if  $v$  is a frequent element, i.e.  $f_v \geq \phi N$ , then with probability at least  $1 - \delta$ ,  $v$  is identified by every node as a frequent element.*

**Sketch of proof:** Two events need to happen for  $v$  to be recognized as a frequent element. (1) Enough copies of  $v$  must occur among the  $t$  smallest weights, and (2) The  $t$  smallest weight elements must be disseminated to all nodes via gossip. In the full proof, we show that both these events are very likely.  $\square$

**Lemma 11.** *Suppose the distributed algorithm in Figure 1 is run for  $20N \ln N$  rounds. If  $u$  is an infrequent element, i.e.  $f_u < (\phi - \psi)N$ , then, with probability at least  $1 - \delta$ ,  $u$  is not identified by any node as a frequent element.*

**Sketch of proof:** A false positive can happen if both the following events occur. (1) There are an unusually high number of copies of  $u$  among the elements with the  $\tau$  smallest weights, and (2) all these copies are disseminated to all nodes. We show that the first event is highly unlikely, and so is the probability of a false positive.  $\square$

Combining Lemmas 11, 10 and 9 we get the following theorem.

**Theorem 4.** *Suppose the distributed algorithm in Figure 2 is run for  $20N \ln N$  rounds. Then, with probability at least  $1 - \delta$ , any data item with  $\phi N$  or more occurrences will be identified as a frequent item at every node. Similarly, with probability at least  $1 - \delta$ , any data item with less than  $(\phi - \psi)N$  occurrences will not be identified as a frequent item at any node.*

Since the size of the sketch at any time during gossip is at most  $t$ , we get the following result on the communication complexity, using Lemma 9.

**Theorem 5.** *The number of bytes exchanged by the algorithm in Figure 2 till the frequent elements are identified is at most  $O(\frac{1}{\psi^2} \ln(\frac{1}{\delta})N \ln N)$ , with probability at least  $1 - O(\frac{1}{N})$ .*

## References

1. Demers, A.J., Greene, D.H., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H.E., Swinehart, D.C., Terry, D.B.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the Principles of Distributed Computing (PODC), pp. 1–12 (1987)
2. Lahiri, B., Tirthapura, S.: Computing frequent elements using gossip. Technical report, Dept. of Electrical and Computer Engineering, Iowa State University (April 2008), <http://archives.ece.iastate.edu/archive/00000415/01/gossip.pdf>
3. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS), pp. 482–491 (2003)
4. Boyd, S.P., Ghosh, A., Prabhakar, B., Shah, D.: Gossip algorithms: design, analysis and applications. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), pp. 1653–1664 (2005)

5. Boyd, S.P., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. *IEEE Transactions on Information Theory* 52(6), 2508–2530 (2006)
6. Mosk-Aoyama, D., Shah, D.: Computing separable functions via gossip. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 113–122 (2006)
7. Misra, J., Gries, D.: Finding repeated elements. *Science of Computer Programming* 2(2), 143–152 (1982)
8. Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, pp. 346–357 (2002)
9. Karp, R.M., Shenker, S., Papadimitriou, C.H.: A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.* 28, 51–55 (2003)
10. Cao, P., Wang, Z.: Efficient top-k query calculation in distributed networks. In: *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 206–215 (2004)
11. Zhao, Q., Ogihara, M., Wang, H., Xu, J.: Finding global icebergs over distributed data sets. In: *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pp. 298–307 (2006)
12. Manjhi, A., Shkapenyuk, V., Dhamdhere, K., Olston, C.: Finding (recently) frequent items in distributed data streams. In: *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pp. 767–778 (2005)
13. Keralapura, R., Cormode, G., Ramamirtham, J.: Communication-efficient distributed monitoring of thresholded counts. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 289–300 (2006)
14. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM)* (2004)
15. Motwani, R., Raghavan, P.: *Randomized Algorithms*. Cambridge University Press, Cambridge (1995)
16. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations (extended abstract). In: *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pp. 327–336 (1998)