Fully Decentralized Computation of Aggregates over Data Streams

Luca Becchetti[1], Ilaria Bordino[1], Stefano Leonardi[1], and Adi Rosen[2]

[1] Sapienza University of Rome,
{becchett, bordino, leon}@dis.uniroma1.it
[2] Laboratoire de Recherche en Informatique Université Paris-Sud and CNRS
Adi.Rosen@lri.fr

Abstract. In several emerging applications, data is collected in massive streams at several distributed points of observation. A basic and challenging task is to allow every node to monitor a neighbourhood of interest by issuing continuous aggregate queries on the streams observed in its vicinity. This class of algorithms is fully decentralized and diffusive in nature: collecting all data at few central nodes of the network is unfeasible in networks of low capability devices or in the presence of massive data sets. The main difficulty in designing diffusive algorithms is to cope with duplicate detections. These arise both from the observation of the same event at several nodes of the network and/or receipt of the same aggregated information along multiple paths of diffusion.
In this paper, we consider fully decentralized algorithms that answer *locally* continuous aggregate queries on the number of distinct events, total number of events and the second frequency moment in the scenario outlined above. The proposed algorithms use in the worst case or on realistic distributions sublinear space at every node. We also propose strategies that minimize the communication needed to update the aggregates when new events are observed. We finally present experimental analysis providing evidence for the efficiency and accuracy of the proposed strategies on realistic simulated scenarios.

Keywords: data streaming, decentralized computation.

1 Introduction

A variety of emerging network applications are based on spreading a large number of network devices, over a broad area. Examples are the continuous and distributed monitoring of IP traffic flows, in which data is collected at multiple points of observation and real-time analysis is performed on aggregated streams, or the use of sensing devices, connected in a wireless sensor network, for environmental monitoring. Sensing devices observe large amounts of events in their surroundings. These events are recorded and processed by nodes in the form of streams of data. In sensor databases we are required to answer aggregate queries over the streams of data, while in distributed monitoring applications we are interested in monitoring the events observed by the sensing devices, for instance the movement of objects or measurements of environment parameters. A number of constraints is imposed in these applications by the limited resources available at the sensing devices:

1. Communication is the most power consuming operation. Transmission of data to neighbor nodes must carefully be optimized in order to ensure a longer life to battery operated devices. A direct consequence is that large streams of raw data cannot be transmitted between nodes.

2. Sensing devices are limited in computational power. The data received from the nodes can often only be processed on the fly in a streaming fashion.

3. Sensing devices have limited storage capacity. Data can be stored, even locally, only in aggregate form.

It is a basic and challenging task to provision the network with primitives that allow every node to monitor its surrounding by issuing continuous aggregate queries on the streams of events observed by all nodes reachable within a few hops. A number of specific issues need to be addressed when querying and monitoring distributed streams of data:

Continuous queries. Differently from traditional databases, these applications are expected to answer long running queries, notifying the application whenever a new answer is found. This occurs for instance in

applications that raise an alert whenever a specific event happens, or some measurement of interest exceeds a given threshold.

Distributed streams. One first difficulty to handle when processing aggregate queries on distributed streams is to discriminate events that have been observed at multiple nodes of the network. For instance, this problem arises naturally for terrain monitoring applications, where sensing devices need to monitor objects that move across an area, or in point-to-point communication when packets are observed along all the nodes of a path. One second difficulty to handle is that nodes are inherently unreliable: they switch on and off in order to optimize power consumption, they are prone to communication faults and to power outage. In several important cases it is therefore impossible to aggregate information using hierarchical structures, e.g. trees, where faults at single nodes of the network may disrupt the whole structure. It is also impossible to rely on very few nodes for collecting the aggregates on raw data produced in the network.

Local queries. In sensor databases applications, aggregate queries may be issued by any node of the network and the answer restricted to the observations made in a neighborhood of interest. Power consumption is optimized if the nodes can decide to operate only when their observations differ substantially from other nodes in their vicinity.

Decentralized algorithms addressing the issues above are inherently diffusive in nature. The problem of detecting duplicates in aggregate information is now magnified, since a node will receive the same data as part of aggregates received through multiple paths of propagation. In this paper we'll make substantial progresses by showing how a set of continuous aggregate queries can be processed from fully decentralized algorithms on distributed streams of data.

1.1 Our Contribution

The design of algorithms for the computation of aggregates within suitable neighbourhoods of all nodes of a network entails two fundamental aspects:

1. *The algorithm used to summarize the scenario of interest.* For any node u it requires a compact summary, or *sketch* in the sequel, of the stream of events observed from u that is *composable* and *duplicate insensitive* [7, 9]: it is possible to merge the sketches of a set of nodes U to form a sketch of the union of the streams of events observed by all nodes of U. In the sequel, we consider the natural case in which U is the set of neighbors within l hops from a node u, for some $l \geq 0$.

2. *How and when information is propagated.* This in turn requires i) addressing communication costs and ii) estimating the (further) error introduced by delay in communication.

Given the massive data sets we are interested in, we impose constraints on the memory requirements for the computation and the overall amount of data exchanged to perform the task. Namely, i) the amount of memory used at every node of the network should be polylogarithmic in the number of nodes of the network and in the size of the streams. ii) The *communication complexity* measured in terms of the overall number of messages exchanged among nodes to perform the task. Below is a list of the main contributions of our work:

1. We define a very general distributed streaming scenario and we describe compact distributed data structures and a communication scheme to keep track of statistic aggregates over distributed data streams. We are able to answer at any node of the network continuous queries on the *number of distinct events*, *total number of events* and the *second frequency moment* (size of Self-join) over the streams observed within a suitably defined neighborhood of each node.

2. These algorithms require only polylogarithmic storage space at each node of the network for *number of distinct events*, and the *total number of events*. For the *second frequency moment* we show a polylogarithmic bound on the storage size at each node when the distribution of events follows a Zipf's law. Observe that in [10] no polylogarithmic upper bound is given on the amount of space needed to estimate the second frequency moment on data streams with duplicates, whereas a $\Omega(\sqrt{n})$ lower bound is provided in the worst case for algorithms based on uniform sampling.

3. We show optimized implementations and tests on large scale real and synthetic datasets and realistic network topologies. Experimental results show that the fully decentralized strategies we consider achieve in practice very good bounds on storage, computation and communication costs while providing a good approximation of the statistics of interest.

4. We characterize the trade-off between the worst case communication cost and accuracy for the sketch schemes we consider. Moreover, we experimentally evaluate the performance of a conservative strategy to trade off communication costs and accuracy, proposed in [11] for a special case of the scenario considered in this paper.

Organization of the paper. In Section 2 we introduce the model and notation we use in the rest of the paper. In Section 3 we outline our overall approach and discuss sketch schemes for fully decentralized estimation of F_0, F_1 and F_2 in the presence of duplicates. For F_2, we also present the analysis for the important case of skewed data. In Section 5 we address the fully decentralized implementation of the sketches we consider and show how to optimize the amount of communication exchanged among nodes of the network. In Section 6 we show the experimental evaluation of the algorithms.

1.2 Related work

An excellent survey of distributed streaming techniques has recently been provided in [8]. They distinguish between one-shot queries and continuous queries. Tree-based aggregation for approximately answering of one shot holistic queries use composable data synopses, as for instance CM sketches [9] for point query, range queries, quantiles, AMS sketches [2, 3] for frequency moments, FM [14] sketches for counting distinct items. Greenwald and Khanna [16] also propose decomposable synopses for computing quantiles that can be adapted to tree-based aggregation.

Alternative to tree-based aggregation is broadcasting to all neighbor vertices till the summaries reach the aggregation node of the network. These proposals need to exploit synopses that are order and duplicate insensitive as those proposed from Considine et al. [7], FM sketches [14], CM sketches [9], and duplicate-insensitive aggregation schemes [10] that propose duplicate insensitive estimation of the second frequency moment for a single aggregating point. However, even in the restricted case addressed in [10], no polylogarithmic upper bound on the needed space is given.

Decentralized computation with every node in the graph taking part in the computation and receiving the computed value has been proposed in [19, 20] for computing max, sum, avg and distinct items. This model is similar in spirit to our proposal with two main differences: a. in our setting we cannot assume direct communication between any pair of nodes in the network; b. we are interested in holistic aggregates at the local neighbor of any node of the network.

Main issue addressed so far in processing continuous queries on distributed data streams is to reduce the communication overhead between remote nodes and the aggregating point by allowing some slack in the estimation. In [5] the problem of distributing the slack between the participating sites is addressed for Top-k computation on distributed streams. In [11] several strategies of communication between the coordinating site and the distributed participants are studied for duplicate-insensitive aggregates as FM sketches and distinct sampling [15].

2 Model and Preliminaries

Model. We assume an undirected graph $G = (V, A)$, representing entities connected over a communication network. Without loss of generality we assume all communication edges bidirectional. Every node in the network observes a stream of data over time. We denote by $\mathcal{S}(u)$ the stream at node u. Each element of $\mathcal{S}(u)$ is an $(\mathsf{item}, \mathsf{attribute})$ pair (i, a), where i is a value that we assume to be in $[n] = \{0, 1, \ldots, n-1\}$, and a is an attribute from a discrete domain $\mathcal{A}$. Since all statistics we consider are order insensitive, $\mathcal{S}(u)$ may be regarded as a multiset of pairs, each pair with a multiplicity equal to the number of times it appears in the stream. Considered any stream S, we also define the *base set* $\mathcal{B}(S)$, i.e., the set of distinct pairs appearing in S. In the sequel, loosely speaking, we often use the expression *event* to mean the observation of an $(\mathsf{item}, \mathsf{attribute})$ pair at some node of the network. Given a subset $U \subset V$, we define $\mathcal{S}(U)$ in the natural way as $\cup_{u \in U} \mathcal{S}(u)$, where the multiplicity of a pair (i, a) in $\mathcal{S}(U)$ is the sum of its multiplicities in $\mathcal{S}(u), \forall u \in U$. For $l \geq 0$, $N_l(u)$ denotes the subset of G's vertices within distance l from u. In the sequel, we write $\mathcal{B}(U)$ instead of $\mathcal{B}(\mathcal{S}(U))$ for the sake of coinciseness.

Statistics. The general problem we are interested in is the following: Given $l \geq 0$, for every $u \in V$, compute some statistics over $\mathcal{B}(N_l(u))$. Depending on the choice of the attribute, we'll be able to estimate different aggregates of interest. Considered a stream S, we denote by $m_i(S)$ (or simply m_i when no

confusion arises) the number of distinct pairs (i, a) in S: $m_i = |\mathcal{B}(S)|$, for every $i \in [n]$. The p-th frequency moment of S is $F_p = \sum_{i \in S} m_i^p$. When considering $\mathcal{S}(u)$ for some vertex u or $\mathcal{S}(U)$ for some subset U of vertices, abusing notation we write $\mathbf{m}_i(u)$, $F_p(u)$, $\mathbf{m}_i(U)$ and $F_p(U)$ for $\mathbf{m}_i(\mathcal{S}(u))$, $F_p(\mathcal{S}(u))$, $\mathbf{m}_i(\mathcal{S}(U))$ and $F_p(\mathcal{S}(U))$ respectively. In the sequel, we consider F_0, the number of distinct items, F_1, the total number of distinct pairs in the stream, and F_2, the second frequency moment over the set of distinct pairs.

Communication. We make the minimal assumption that in a round of communication a node can only broadcast a message to the set of its immediate neighbours. We make no assumptions as to the topological structure of the neighbourhood of a node or the way in which communication is performed. Our model has an immediate practical counterpart in wireless networks in which communication occurs over a broadcast channel, while in other cases a round of communication may actually involve multiple transmissions along point-to-point connections.

3 Overview of the approach

Each node in the network observes a local stream over time and keeps a compact summary or *sketch* thereof, which depends on the application of interest. Consider a streaming algorithm A. We denote by Sk^A the function used by A to compute sketches and by $Sk^A(S)$ the sketch computed by A at the end of the observation of a stream S[3]. In the setting we consider, every node u locally maintains two sketches $Sk(\mathcal{S}(u))$ and $Sk(\mathcal{S}(N_l(u)))$ and performs the following actions:

i. when a pair (i, a) is observed at u, the node updates $Sk(\mathcal{S}(u))$ and $Sk(\mathcal{S}(N_l(u)))$ in a way that depends on the statistics of interest and the sketching algorithm used;
ii. if u receives a sketch $Sk(\mathcal{S}(v))$ originating from a node v (i.e., an update of $Sk(\mathcal{S}(v))$), u updates $Sk(\mathcal{S}(N_l(u)))$ to reflect the new observations done at v and forwards the update to its neighbours, *as long as* its distance from v is less than l;
iii. u continuously monitors the current estimate of the statistics of interest over $\mathcal{S}(u)$. If this value differs from the most recently propagated value significantly, the current value of $Sk(\mathcal{S}(u))$ is propagated to u's immediate neighbours.

Step ii. is achieved using messages that contain a TTL (Time To Live) field, which is decremented at each node reached by the message, as is customary in gossip-based network protocols.

Clearly, not all sketches are suitable for our setting. Following previous work, we consider sketches that are *composable* and *duplicate insensitive* [21, 7, 17]. Considered an order insensitive statistics of interest, a sketching algorithm A for its estimation is composable if, given two streams S_1 and S_2, $Sk^A(S_1 \cup S_2) = merge(Sk(S_1), Sk(S_2))$, where $merge(\cdot)$ is a suitable sketch aggregation function that depends on A and the statistics of interest [7]. A sketching algorithm is duplicate insensitive if, considered any stream S, $Sk(S) = Sk(\mathcal{B}(S))$ (where $\mathcal{B}(S)$ is regarded as a multiset). In next Section, we address the problem of designing accurate, composable and duplicate insensitive sketches for the primitives of interest in this paper.

4 Composable and duplicate insensitive sketches

4.1 Maintaining F_0 and F_1

The basic tool we consider is a *counting sketch*, i.e., a composable and duplicate insensitive counter of the number of distinct (i, a) pairs appearing in a stream S. In the sequel, we consider two approaches: the first is the original approach of Flajolet and Martin, considered in [11] and modified in [**?**]. The second is a slightly different technique proposed in [6]. Since these are well established techniques, we only give an overview to make the paper self-contained, referring the reader to [14, **?**,11, 6] for details. The use of composable and duplicate insensitive sketches has been considered previously for restricted distributed settings, especially for data aggregation at the sink of a sensor network [21, 7, 17].

[3] Note that we are in a dynamic context. Here and in the sequel, $\mathcal{S}(u)$ represents the stream observed at u until present.

FM sketches. In the sequel, we use the phrase "FM sketch" to refer to any implementation of the original counting sketch of [14]. FM sketches [14] use a simple approach in which each sketch is a vector of m entries, each entry being a bitmap of length $k = O(\log M)$, with M an upper bound on the size of the universe. In our setting, the universe is the set of possible (i, a) pairs, so that $M \leq n|\mathcal{A}|$ (in practice, $M = 2^k$, e.g., $k = 64$). Considered the s-th bitmap of the sketch. Every pair `(item, attribute)` (i, a) is hashed onto the bitmap bits using a (independently chosen) hash function $H_s(\cdot) : [n] \times \mathcal{A} \rightarrow \{0, \ldots, \log_2 M - 1\}$, such that the probability of hashing onto the h-th bit is 2^{-h}. The bit under consideration is set to 1 if it was 0. After processing the stream, let r_s denote the position of the least significant bit that is still 0 in the s-th bitmap: r_s is a good estimator for $\log_2 F_0$, the logarithm of the number of distinct pairs observed. To improve accuracy, we consider $\frac{1}{m}\sum_{s=1}^{m} r_s$ as an estimator of $\log_2 F_0$, where $m = O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$.

Bar-Yossef et al. [6]. In this approach, the sketching algorithm maps every pair (i, a) to an integer using a pairwise independent hash function $h(\cdot)$ and at any point in time it maintains the list of the L smallest distinct values observed so far, where $L = \lceil 96/\epsilon^2 \rceil$, ϵ being the required precision. If v is the L-th smallest distinct value maintained by the algorithm, LM/v is an estimator of F_0, where $M = n^3$ (see [6] for details). Precision can be increased by the standard trick of considering m independent and parallel copies of the algorithm and taking the *median* of the corresponding estimations, where again $m = O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$.

Composability. Clearly, both sketches are composable and duplicate insensitive: Considered two streams S_1 and S_2 over the same universe and their FM sketches $Sk^{FM}(S_1)$ and $Sk^{FM}(S_2)$, $Sk(S_1)\texttt{OR}Sk(S_2)$ is the sketch corresponding to $S_1 \cup S_2$. As for the sketches of [6], every such sketch is an array of m lists, each maintained according to the algorithm described above. Merging of two such sketches is simply achieved as follows: for every $s = 1, \ldots, m$, merge the s-th lists of the two sketches and keep the L smallest values of the merged list. Both approaches achieve similar bounds in terms of efficiency and precision, as stated by the following

Theorem 1 ([11, 14, 6]). *Considered a stream S of (item, attribute) pairs, it is possible to maintain an estimate $\hat{C}$ of the number C of distinct pairs in S using $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$ memory words, such that:*

$$\mathbf{P}\left[|\hat{C} - C| > \epsilon C\right] \leq \delta.$$

Any of the schemes outlined above can be used to maintain, for a stream S, an accurate and duplicate insensitive sketch for $F_0(S)$ or $F_1(S)$. This is achieved by suitably defining attributes. For F_0, we want to count the number of distinct items observed in S. This is possible by setting `a = null`. As to F_1, this is the basic problem of counting the number of distinct pairs, therefore it can be maintained with the same guarantees mentioned above.

4.2 Decentralized maintenance of F_2 with duplicates

We adapt the method proposed by Achlioptas [1] to maintaining F_2 with duplicates in a fully decentralized setting. We show that the method achieves polylogarithmic space on skewed data.

Maintaining F_2 using random projections The most effective approaches to efficiently maintain F_2 in a centralized setting are based on random projections of the frequency vectors over a space of smaller dimension. A similar approach, explicitely designed to maintain F_2 over a data stream, was proposed in [**?**] and extended in [13, 18]. Independently, a number of techniques have been proposed to achieve the more general goal of maintaining pairwise euclidean distances of a set of vectors in lower dimensional space, mostly extending or modifying a key result by Johnson and Lindenstrauss (see [12] for a relatively simple proof). A substantial simplification of the original scheme that we adopt in this work was proposed by Achlioptas [1].

Maintaining $F_2(S)$ in the absence of duplicates. In this case, $(i, a), (i, b) \in S$ implies $a \neq b$. Then, the approach described in the above paragraphs immediately applies to $\mathbf{m}(S)$ as done in [**?**]: assume $\mathbf{m}$ is the state of the frequency vector after the first t elements in S have been observed. At any time t, we maintain $\hat{\mathbf{m}} = \mathbf{mR}$ as follows: If the value of the $t+1$-th item observed is i, $\hat{\mathbf{m}}_j = \hat{\mathbf{m}}_j + \mathbf{e}_i \cdot \mathbf{R}$ for $j = 1, \ldots, d$, where $\mathbf{e}_i$ is the n-dimensional row vector whose i-th component is 1, all other components being 0. So, observation of i determines the addition or subtraction of 1 in every component of $\hat{\mathbf{m}}$.

Maintaining $F_2(S)$ in the presence of duplicates. When duplicates are present in S, we use the "tug-of-war" sketch considered in [2, 10]. In particular, we notice that $\hat{\mathbf{m}}_j(S) = I_j(S) - D_j(S)$, where $I_j(S)$ (respectively, $D_j(S)$) counts the number of *distinct* (i, a) *pairs* observed in S that determine the addition of $+1$ (addition of -1) to $\hat{\mathbf{m}}_j(S)$. Hence, for every $j = 1, \ldots, d$, we can estimate $I_j(S)$ and $D_j(S)$ using the techniques described in Section 3. More in detail, for every j, we maintain a counting sketch to estimate $I_j(S)$ (respectively $D_j(S)$). The overall tug-of-war sketch obtained this way consists of $2d$ counting sketches and it is clearly duplicate insensitive and composable. In particular, to compose two tug-of-war sketches, the component counting sketches are pairwise composed in the obvious way. Following the above paragraph on maintaining the 2-norm, if $\tilde{I}_j(S)$ (respectively, $\tilde{D}_j(S)$) denotes the estimation of $I_j(S)$ (respectively, $D_j(S)$), our estimator of $F_2(S)$ is $\frac{1}{d}\sum_{j=1}^{d}(I_j(S) - D_j(S))^2$.

Maintaining R. Maintaining $\mathbf{R}$ explicitely requires $\mathbf{O}(nd)$ space at every node, which is unfeasible. Furthermore, if we want to compute F_2 over the union stream of a subset of the vertices, it is necessary that all nodes use the same random projection. In practice, all nodes generate the entries of $\mathbf{R}$ whenever needed using the same pseudorandom generator (and thus polylogarithmic space). To generate $\mathbf{R}_{ij}$, u computes the j-th value of the random generator with initial seed i. This way, all vertices generate the same value for $\mathbf{R}_{ij}$. [4]

Analysis for skewed data The best algorithm to maintain F_2 with duplicates under general distributions requires space $O\left(\frac{1}{\epsilon^4}\sqrt{n}\log n\right)$ [10]. In the same paper, the authors prove an $\Omega(\sqrt{n})$ lower bound for algorithms based on uniform sampling and it is an open question whether this bound indeed holds in general.

In the present paper, we show that the algorithm described above provides an accurate estimation of F_2 using polylogarithmic space for skewed data, which is the case in most applications of interest. We assume that $\mathbf{m}(S)$ is distributed according to a Zipf law with parameter $\alpha > 1$, i.e.: $\mathbf{m}_i = M/i^\alpha$. The analysis for $\alpha \leq 1$ (showing decreasing accuracy as α decreases) proceeds along similar lines and will be given in the full version of the paper[5]. Under these assumptions we are able to state the following theorem:

Theorem 2. *If $\mathbf{m}(S)$ is distributed according to Zipf law with parameter $\alpha > 1$, it is possible to compute an estimation $\tilde{F}_2(S)$ such that:*

$$\mathbf{P}\left[|\tilde{F}_2(S) - F_2(S)| > 2\epsilon F_2(S)\right] \leq \delta,$$

by using an amount of $O((\frac{1}{\epsilon^4(\alpha-1)^4}\log\frac{1}{\delta})(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}))$ memory words per node.

Proof (Sketch of the proof). We consider $F_2(S)$ under the assumption that $\mathbf{m}(S)$ is distributed according to a Zipf law with parameter α. We consider the case $\alpha > 1$ in the sequel. We drop S in the rest of this subsection, and we assume without loss of generality that $\mathbf{m}_1 \geq \cdots \geq \mathbf{m}_n$:

$$\mathbf{m}_i = \frac{M}{i^\alpha},$$

where M is a positive, integer constant. It is clear that by this definition the components of $\mathbf{m}$ will be fractional. Assuming integer components does not affect the result, but it makes proofs much more involved, so we do not consider this issue here. The proof of the following simple lemma is deferred to the full paper.

Lemma 1. *If $\mathbf{m}$ follows a Zipf law with parameter $\alpha > 1$, for every $n \geq 2$:*

$$F_2 \geq \frac{M^2}{4\alpha - 2}.$$

[4] It is clear that the matrix generated this way no longer satisfies the independence assumptions of [1], but choosing the pseudorandom generators independently for every column is in practice enough and is close to the requirements of [**?**].

[5] It is clear that by this definition the components of $\mathbf{m}$ will be fractional. Assuming integer components does not affect the result, but it makes proofs much more involved.

We next characterize the error achieved by the overall algorithm. Note that this error has two sources: i) the random projection; ii) the propagation algorithm.

In particular, recall that $\hat{\mathbf{m}} = \mathbf{mR}$ and $\hat{\mathbf{m}}_j = I_j - D_j$. In fact, I_j and D_j are estimated at u from the corresponding FM sketches, so that u actually only computes two estimates $\tilde{I}_j$ and $\tilde{D}_j$ of I_j and D_j. In particular, consider realizations of I_j, D_j, $\tilde{I}_j$ and $\tilde{D}_j$ and assume that $\tilde{I}_j = (1 + \epsilon_1(j))I_j$ and $\tilde{D}_j = (1 + \epsilon_2(j))D_j$. The estimation of F_2 using the projected vector would be $\frac{1}{d}\sum_{j=1}^{d} \hat{\mathbf{m}}_j^2$. In practice, u computes a vector $\tilde{\mathbf{m}}$ that is itself an estimation of $\hat{\mathbf{m}}$. In particular, $\tilde{\mathbf{m}}_j = \tilde{I}_j - \tilde{D}_j$. This implies that our actual estimation of F_2 is $\tilde{F}_2 = \frac{1}{d}\sum_{j=1}^{d} \tilde{\mathbf{m}}_j^2$. The simple proof of the following lemma is given in the full paper for the sake of space.

Lemma 2.

$$\tilde{F}_2 = \frac{1}{d}\sum_{j=1}^{d}(I_j - D_j)^2 + E,$$

where $|E| \leq \frac{3}{d}\sum_{j=1}^{d} \max\{|\epsilon_1(j)|, |\epsilon_2(j)|\}(I_j + D_j)^2$.

In the sequel, we show that i) $\frac{1}{d}\sum_{j=1}^{d}(I_j - D_j)^2$ (i.e., the squared 2-norm of the projected vector) is a close approximation of F_2 with high probability; ii) $|E|$ can be compensated by forcing the $\epsilon_1(j)$'s and $\epsilon_2(j)$'s to be small enough. The first claim is just a restatement of Lemma 5.1 in [1].

Lemma 3 ([1]). *Let* $S = \frac{1}{||\mathbf{m}||_2^2}\sum_{j=1}^{d}(I_j - D_j)^2$. *With probability at least* $1 - \frac{\delta}{2}$ *the following holds:*

$$|S - d| \leq \epsilon d,$$

whenever $d \geq \frac{24}{3\epsilon^2 - 2\epsilon^3}\ln\frac{4}{\delta}$.

Lemma 4. *If* d *dimensions are used for the random projection, using memory* $O(\frac{d}{\epsilon^2(\alpha-1)^4}\log\frac{d}{\delta})$ *it is possible to ensure that with probability at least* $1 - \frac{\delta}{2}$ *the following holds:*

$$|E| \leq \epsilon F_2.$$

Proof. First note that, for every $j = 1, \ldots, d$:

$$I_j + D_j = \sum_{i=1}^{n}\frac{M}{i^\alpha} < M + M\int_2^{n-1}\frac{1}{i^\alpha}di < \frac{\alpha+1}{\alpha-1}M,$$

which implies $(I_j + D_j)^2 < \left(\frac{\alpha+1}{\alpha-1}\right)^2 M^2$ for every j.

Now, we represent every counter ($2d$ counters) using counting sketch (see Theorem 1) consisting of $O\left(\left(\frac{1}{\epsilon}\right)^2\left(\frac{1}{\alpha-1}\right)^4\ln\frac{d}{\delta}\right)$ bitmaps. Choosing constants suitably, Theorem 1 allows to prove that:

$$\mathbf{P}\left[|\tilde{I}_j - I_j| > \frac{\epsilon}{12\alpha - 6}\left(\frac{\alpha-1}{\alpha+1}\right)^2 I_j\right] < \frac{\delta}{4d}.$$

The same holds for $\tilde{D}_j$ and D_j. This implies:

$$\mathbf{P}\left[\exists j : \max\{|\epsilon_1(j)|, |\epsilon_2(j)|\} > \frac{\epsilon}{12\alpha - 6}\left(\frac{\alpha-1}{\alpha+1}\right)^2\right] < \frac{\delta}{2}.$$

As a result, with probability at least $1 - \frac{\delta}{2}$:

$$|E| \leq \frac{3}{d}\sum_{j=1}^{d}\max\{|\epsilon_1(j)|, |\epsilon_2(j)|\}(I_j + D_j)^2$$

$$< \frac{3}{d}\sum_{j=1}^{d}\frac{\epsilon}{12\alpha - 6}\left(\frac{\alpha-1}{\alpha+1}\right)^2(I_j + D_j)^2 < \epsilon F_2,$$

where the third inequality follows from $(I_j + D_j)^2 < \left(\frac{\alpha+1}{\alpha-1}\right)^2 M^2$ and from Lemma 1.

```
PROCESSITEM(LS, GS, i, a, l)
Require: LS, GS: SKETCH, i: item, a:
    attribute, l: distance
 1: LS = UPDATESKETCH(LS, i, a)
 2: GS = UPDATESKETCH(GS, i, a)
 3: Stat = GENSTAT(LS)
 4: if DIFF(Stat, Stat_0) > threshold then
 5:    MS = BUILDMESSAGE(LS, l)
 6:    Send MS to u's neighbours
 7:    Stat_0 = Stat
 8: end if
```

```
PROCESSMESSAGE(GS, MS)
Require: GS, SKETCH, MS: message
 1: S = SKETCH(MS)
 2: GS = MERGESKETCH(GS, S)
 3: h = TTL(MS)
 4: if h > 0 then
 5:    h = h - 1
 6:    MS = BUILDMESSAGE(S, h)
 7:    Send MS to u's neighbours
 8: else
 9:    Drop MS
10: end if
```

Fig. 1. Generic algorithm for continuous monitoring of statistics within distance l.

We can now prove the theorem.
Again, we drop S since clear from context. We apply Lemmas 3 and 4 with $d = \frac{24}{3\epsilon^2 - 2\epsilon^3} \ln \frac{4}{\delta}$. Setting

$$\hat{S} = \frac{1}{d} \sum_{j=1}^{d} (I_j - D_j)^2,$$

Lemmas 3 and 4 imply that $(|\hat{S} - F_2| \leq \epsilon F_2) \wedge (|E| < \epsilon F_2)$ with probability at least $1 - \delta$, so that we have:

$$\begin{aligned}
&\mathbf{P}\left[|\tilde{F}_2 - F_2| > 2\epsilon F_2\right] = \mathbf{P}\left[|\hat{S} + E - F_2| > 2\epsilon F_2\right] \\
&\leq \mathbf{P}\left[|\hat{S} - F_2| + |E| > 2\epsilon F_2\right] \leq \mathbf{P}\left[|\hat{S} - F_2| > \epsilon F_2\right] \\
&+\mathbf{P}[|E| > \epsilon F_2] \leq \delta.
\end{aligned}$$

The bound of Theorem 2 depends on the value of α and it becomes unbounded as $\alpha \to 1$. In fact, for $\alpha = 1$ we obtain a polylogarithmic bound independent of α and which is stronger than the one of Theorem 2 for values of α close to 1.

5 Distributed implementation and communication tradeoffs

In this section, we describe a distributed implementation of the primitives described in Section 4. Given the composability and duplicate insensitivity of the sketches we consider, the results on the accuracy immediately carry over, so we do not discuss this issue again.

5.1 Distributed implementation

Generic node behaviour (Figure 1). Upon observing a pair (i, a), node u invokes `PROCESSITEM(S, i, a, l)` to update its local and global sketches `LS` and `GS` (lines 1 and 2) and the estimate of the statistics of interest over $\mathcal{S}(u)$ (line 3). If this exceeds the last value sent[6] by more than a given threshold (line 4), a message `MS` containing `LS`'s update is built and sent, with initial `TTL = l`. Whenever u receives a message from some neighbour v, containing v's update of $Sk(\mathcal{S}(v))$, `PROCESSMESSAGE(S, MS)` extracts the sketch and updates u's global sketch (lines 1 and 2). The `TTL` is decremented and, if larger than 0, the message is forwarded to u's neighbours. Note that this generalizes the "no sharing" update scheme of [11], where they consider a star network topology with a single, central coordinator maintaining a global sketch (See Subsection 5.2). It remains to show i) how sketches are updated and merged and ii) how the threshold is defined. The former issue is briefly discussed in the next paragraphs for F_0 (and F_1) and for F_2, while the latter is addressed in Subsection 5.2.

Fully decentralized distinct counting. In this case, `LS` and `GS` in Figure 1 are counting sketches for the estimation of $F_0(u)$ and $F_0(N_l(u))$ respectively. Counting sketch update and merge operations and count

[6] More precisely, whose corresponding sketch was sent.

estimation have been succintly described in Subsection 4.1. The pseudo-code of the main routines will be described in the full paper for the sake of space.

Fully decentralized estimation of F_2. In this case, `LS` and `GS` in Figure 1 are composite sketches for the estimation of $F_2(u)$ and $F_2(N_l(u))$ respectively. Following Subsection 4.2, `LS` and `GS` each consist of $2d$ counting sketches, where d is chosen according to Lemma 3. Considering `LS` (`GS` has exactly the same structure), for every $j = 1, \ldots, d$, the j-th sketch `LS[j]` consists of two counting sketches `ILS[j]` and `DLS[j]`, to keep track of counters $I_j(\mathcal{S}(u))$ and $D_j(\mathcal{S}(u))$, as described in Subsection 4.2. Update, merge and F_2 estimation routines occur as described in Subsection 4.2. The pseudo-code will be given in the full paper for the sake of space.

5.2 Communication and Tradeoffs

In this section, we describe in more detail how continuous monitoring of statistics is performed; in particular, when and how information is propagated within the network. The general approach has been described in Section 3. In the experiments we considered the *Threshold triggered updates* approach investigated of [11] ("No sharing" policy). Carrying this policy over to our scenario, node u sends an update, i.e., its local sketch `LS`, whenever the estimate C of the statistics of interest over the local stream changes, with respect to the last propagated value C_0, i.e., whenever $C > (1 + \theta/|N_l(u)|)C_0$, for some $\theta > 0$.

We also considered an orthogonal approach in which, when a new observation occurs, only the portion of the sketch that eventually changes is propagated. This approach allows a slight improvement in the overall (worst case) number of bits transmitted and brings to messages of smaller size, which can be important in some applications, e.g., sensor networks. For the sake of space and since we used the approach of [11] in the experiments, this part will be presented in the full paper.

6 Experimental analysis

6.1 Datasets

The graph used in the experiments is a real topology of (part of) the Internet at the level of the Autonomous Systems collected by DIMES [22] in December 2008. The symmetrized version of this graph consists of 65 512 nodes and 148 364 edges, and its diameter is equal to 8.

Nodes were assigned streams of tuples $\langle i, a \rangle$, obtained from either real or synthetically generated data. Real data were extracted from a collection of HTTP requests made to the 1998 World Cup Web site, made available[7] by the Internet Traffic Archive [4] and spanning an observation period of three months (May-July 1998). These data were also used in [11]. We considered a week of data, consisting of approximately 10 million tuples. Each HTTP request record contains a number of attributes, including clientID, objectID, serverID and a timestamp. In our experiments, we did not consider serverID and we chose to focus on the triples (clientID, objectID, timestamp). The data tuples were assigned to nodes of the graph using a hash function to map clientIDs onto graph vertices. Stream tuples then consisted of (objectID, timestamp) pairs.

We generated synthetic data considering a universe of items of size $1,000,000$. We imposed m_i (number of distinct pairs with i the item) to follow a Zipf distribution with parameter $\alpha = 2.0$. Every node in the graph was assigned a stream of length uniformly distributed in $[500, 1000]$, consisting of tuples $< i, a >$, where i was chosen with probability proportional to m_i and $a \in [m_i]$. Globally, we obtained a data stream of approximately 17 million tuples.

6.2 Accuracy of the estimation

A first set of experiments only concerned the accuracy of the estimation provided by the sketches. We used counting sketches corresponding to a precision $\epsilon = 0.1$ with probability at least $1 - \delta$, where $\delta = 0.1$. For each node in the network, statistics were computed on the streams observed within neighborhoods formed by i) all nodes at distance 1 and ii) all nodes within distance 2. For every statistics of interest we considered the average of the estimates computed over i) the 100 nodes with highest degrees and ii) a set

[7] `URL: http://ita.ee.lbl.gov/html/contrib/WorldCup.html`

of 100 nodes chosen uniformly at random. For every selected node and for each statistic, we computed the average error with respect to the exact value.

In estimating F_0 (see Figure 3), we considered the two counting sketches described in Subsection 4.1, i.e, the original proposal by Flajolet and Martin [14] and a modification proposed by Bar-Yossef et al [6], denoted $BJKST$ in the sequel.

We observed that, when FM sketches are used, the counting sketch provides approximations with error lower than 5% in most cases of the number of distinct items observed by every node within its 1(2)-neighborhood. BJKST sketches in turn perform extremely well: their accuracy is comparable or slightly better than that of FM sketches. The only issue to consider is that the estimator adopted in [6] (see Subsection 4.1) is $L \cdot M/v$, with $L = 96/\epsilon^2$, that corresponds to a value of 9 600 in our case. This implies that when using BJKST sketches, the algorithm always returns an estimate no lower than this value. Since $L = 96/\epsilon^2$ is a constant for fixed ϵ, we simply keep the exact count as long as this remains below this value. We present results that use FM sketches in the rest of this section, extended results will appear in the full paper.

Accuracy for F_1 is always below 4%. For the sake of space we do not discuss these results here.

In the estimation of F_2 (see Figure 3), we used a value of d (number of dimensions of the projected space) equal to 200, which is much lower than the one requested by theoretical analysis (2567) for the values ϵ and δ we consider, which may suggest that the analysis can be improved. Complete results will be reported in the full paper. In general, as predicted by our analysis, the quality of results is very good for synthetic data, which were generated according to a very skewed distribution following a Zipf's law with parameter $\alpha = 2$. On the converse, larger errors are observed on real data that exhibit a much lower skewness than that requested by our theoretical analysis: we observed very small values (in the range $[0.6, 0.8]$) of the Zipf's parameter α for the streams at all the nodes involved in the measurements .

6.3 Communication and tradeoffs

The approach we adopt to optimize communication was described in the previous section. It basically consist of the following: every node u sends a message update for a given counter to its immediate neighbors whenever the local estimation computed for that counter exceeds the last update that was sent by more than a factor $1 + \theta/N_l(u)$.

In order to evaluate the communication costs of the developed methods, we simulated a scenario in which the nodes in the network start to process their local stream simultaneously. We computed F_0 and F_2 keeping track of the total number of message updates sent by the algorithm at ten equally spaced checkpoints, which were fixed by identifying ten intervals of uniform length within the stream of events observed at each node. We considered different values of θ ($\theta = 0.1$, $\theta = 0.2$ and $\theta = 0.4$ and $\theta = 0.8$) to vary the local threshold used by every node to decide whether send or not to send a new message update after processing a new event. For the sake of space, we omit results for F_2 and we present and discuss below accuracy/communication tradeoffs for the critical F_0 estimation primitive for distance 2 neighbourhoods. Results for distance 1 are similar and will be given in the full paper.

Figure 4 shows the results concerning the estimation of F_0 within distance $l = 2$. The general trend is clear: at the beginning, so for the first checkpoints, the number of first time observations of items grow faster and accordingly, we observe a steep increase in the number of messages sent. Conversely, their estimations stabilize after few checkpoints and the amount of exchanged messages drop dramatically. Also, since most distinct items are observed in the first part of the stream, nodes gain very early a very good accuracy in their estimations of F_0: for $\theta \in [0.1, 0.2, 0.4, 0.8]$ average error at the first checkpoint is already around $3 - 4\%$.

We also experimented with higher values of θ ($\theta \in [2, 10, 20]$), in order to trade communication with degrading accuracy too much. While $\theta = 2$ does not change the picture, for higher values of θ we observed a significant drop in the total number of message updates (by a factor in $[1.7, 2]$). Interestingly, the quality of approximation still remains very good: With $\theta = 10$, the average error gets lower than 10% as soon as the third checkpoint is reached, and it is around 7% at the end of the observation. For $\theta = 20$, the error gets lower than 15% at the third checkpoint, and it is not greater than 10% at the last checkpoint.

Memory requirements. We briefly discuss the memory requirements of the sketches considered. More details will be given in the full version of the paper. We implemented counting sketches (for F_0 and F_1) using at most 575 Bytes per sketch. This figure is compatible with commercial sensor networks platforms,

although packet sizes in these platforms are typically bounded to have smaller values[8]. In the case of F_2, memory requirements are between 145 and 224 KBytes per sensing node. While this is perfectly compatible with many applications (e.g., network traffic monitoring) it is clearly unfeasible in sensor networks.

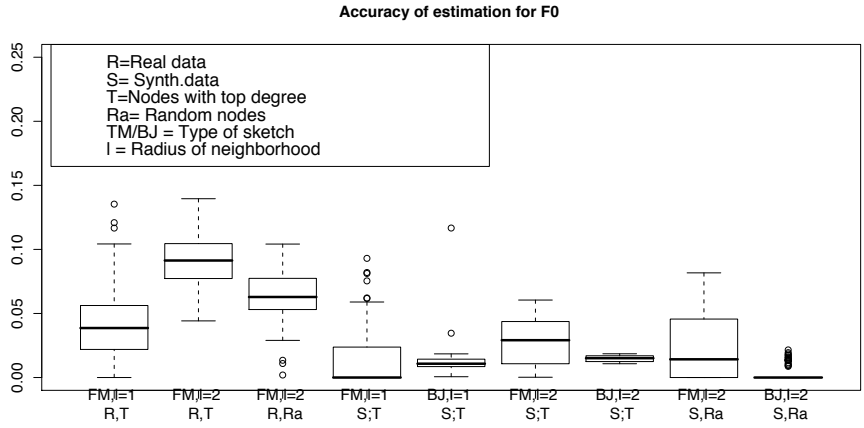

Fig. 2. Accuracy of the estimation computed for F_0

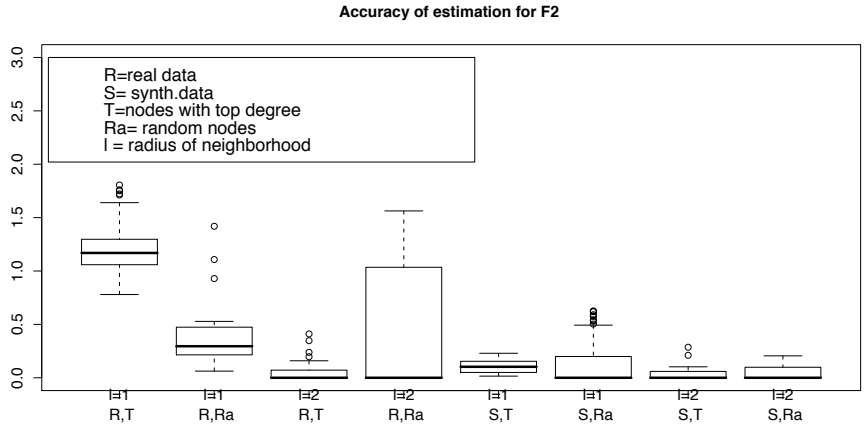

Fig. 3. Accuracy of the estimation computed for F_2

References

1. Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci*, 66(4):671–687, 2003.
2. Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002.
3. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.
4. M. Arlitt and T. Jin. 1998 world cup web site access logs, August 1998.
5. Brian Babcock and Chris Olston. Distributed top-k monitoring. In *SIGMOD Conference*, pages 28–39, 2003.

[8] This can be overcome by having a constant number of packets per update message or by reducing the size of update messages, as we show in the full paper.

6. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 1–10, Cambridge, Ma, USA, 2002. Springer-Verlag.
7. Jeffrey Considine, Feifei Li, George Kollios, and John W. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, pages 449–460, 2004.
8. Graham Cormode and Minos N. Garofalakis. Streaming in a connected world. In *VLDB*, page 1266, 2006.
9. Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.
10. Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In ACM, editor, *Proceedings of the Twenty-Fourth ACM Symposium on Principles of Database Systems*, pages 271–282. ACM Press, 2005.
11. Graham Cormode, S. Muthukrishnan, and Wei Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proceedings of the 22nd International Conference on Data Engineering*, page 57. IEEE Computer Society, 2006.
12. Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.
13. Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput*, 32(1):131–151, 2002.
14. Philippe Flajolet and Nigel G. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.
15. Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB*, pages 541–550, 2001.
16. Michael Greenwald and Sanjeev Khanna. Power-conserving computation of order-statistics over sensor networks. In *PODS*, pages 275–285, 2004.
17. Marios Hadjieleftheriou, John Byers, and George Kollios. Robust sketching and aggregation of distributed data streams. Technical Report 2005-011, CS Department, Boston University, 2005.
18. Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.
19. Srinivas R. Kashyap, Supratim Deb, K. V. M. Naidu, Rajeev Rastogi, and Anand Srinivasan. Efficient gossip-based aggregate computation. In *PODS*, pages 308–317, 2006.
20. David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491, 2003.
21. Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 250–262. ACM, 2004.
22. Yuval Shavitt and Eran Shir. Dimes: Let the internet measure itself. *CoRR*, abs/cs/0506099, 2005.

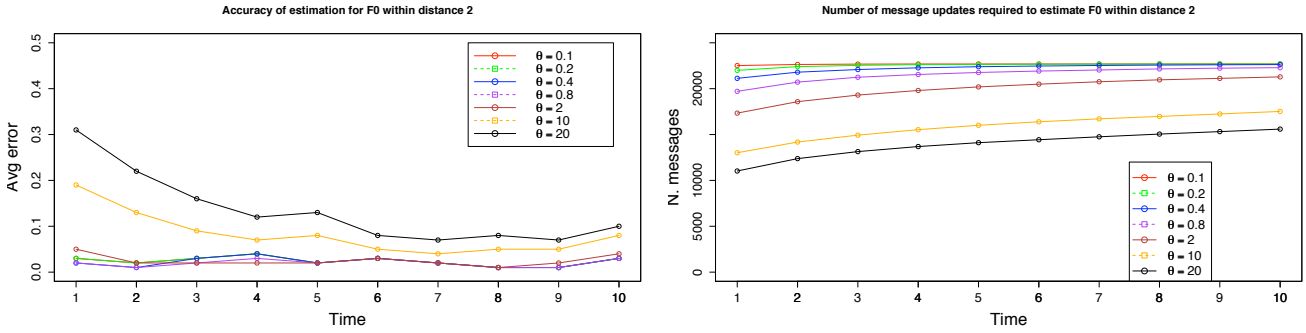

Fig. 4. Tradeoff between communication costs and accuracy for $F_0, l = 2$