# *Spectra*: Robust Estimation of Distribution Functions in Networks

Miguel Borges, Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida

HASLab / INESC TEC, Universidade do Minho
Campus de Gualtar, 4710-057 Braga, Portugal
{mborges, pcoj, cbm, psa}@di.uminho.pt

**Abstract.** Distributed aggregation allows the derivation of a given global aggregate property from many individual local values in nodes of an interconnected network system. Simple aggregates such as minima/maxima, counts, sums and averages have been thoroughly studied in the past and are important tools for distributed algorithms and network coordination. Nonetheless, this kind of aggregates may not be comprehensive enough to characterize biased data distributions or when in presence of outliers, making the case for richer estimates of the values on the network.
This work presents *Spectra*, a distributed algorithm for the estimation of distribution functions over large scale networks. The estimate is available at all nodes and the technique depicts important properties, namely: robust when exposed to high levels of message loss, fast convergence speed and fine precision in the estimate. It can also dynamically cope with changes of the sampled local property, not requiring algorithm restarts, and is highly resilient to node churn.
The proposed approach is experimentally evaluated and contrasted to a competing state of the art distribution aggregation technique.

## 1 Introduction

The ability to aggregate data is a fundamental feature in the design of scalable information systems, which allows the estimation of relevant global properties in a decentralized way in order to coordinate distributed applications, or for monitoring purposes. Usual aggregates include environment sensor data, such as temperature and humidity, and system properties, such as load and available storage.

Simple aggregates such as minima/maxima, counts, sums and averages have been thoroughly studied in the past. Nonetheless, this kind of aggregates may not be comprehensive enough to characterize biased distributions or in the presence of outliers, making the case for richer estimates of the values on the network (e.g. probability density functions, histograms, cumulative distributed functions), since statistical ordinary moments hide in many cases changes in the property that are relevant to control decisions.

The amount of scientific work is relatively scarce in what concerns more expressive aggregation metrics. A recent proposal within this domain [22] claims

to obtain estimates with a better precision than in previous approaches. It is an algorithm for the estimation of discrete cumulative distribution functions.

Despite the contribution, the proposal mentioned above is not fault tolerant and is also not sensible to the continuous variation of the sampled properties, for it demands the protocol to be restarted frequently in order to achieve quasi-continuous monitoring. Besides, the approach does not admit loss or duplication of messages.

Having this scenario as a starting point, this work presents *Spectra*, a distributed algorithm for the estimation of distribution functions over large scale networks. Its core advantages are resilience to message loss, high convergence speed and high precision of the estimate. It also supports changes of the sampled property and churn. All this is achieved without requiring the protocol to be restarted.

In detail, *Spectra* enables the estimation of the cumulative distribution function (CDF) of a given property at all nodes. This allows nodes to take advantage of having a broader view of the property on the network: they may exclude outliers or monitor particular quantiles of a property. Also, each node of the network has a local vision of the global state of the property, thus allowing them to make decisions based on local knowledge.

This work includes simulation results that support and validate the proposed approach along with a comparison with the *Adam2* [22] algorithm.

In the next section we make a short overview of the state of the art work on the context of distribution aggregation. In Section 3 we briefly state the system model used on this work. The next section presents the *Spectra* algorithm and after we show the evaluation results, contrasting with *Adam2*. Last section draws conclusions on the work and presents a few perspectives about future research directions.

## 2 Related Work

In the last decade, several distributed aggregation algorithms have been proposed to estimate the value of scalar properties (e.g. network size). Existing techniques can be divided in different classes, providing different characteristics in terms of performance (time and message load) and robustness, mainly as: hierarchy-based (or tree-based), averaging (or gossip-based), sketches and sampling approaches. A wide and comprehensive overview of the current state of the art on distributed aggregation algorithms is provided in [15].

Hierarchy-based approaches [18,20,3] rely on a convergecast process to aggregate data along a pre-established hierarchical routing structure (commonly a tree), producing the result at the root. This kind of technique is usually applied to Wireless Sensor Networks (WSN) due to its energy efficiency, despite being highly sensitive to failures. at a single point). Some algorithms [8,5,19] are found collecting samples and applying an estimation method to obtain a rough approximation of the size of a membership. This type of scheme is lightweight in terms of message load, as only a partial number of nodes might be asked

to participate in the sampling process, but is also inaccurate and produce the result at a single node. Moreover, several rounds might be required to collect a single sample, especially when sampling is performed through random walks like in [8,19], thus being slow. A more robust alternative is provided by algorithms that aggregate data through multiple paths, such as those based on the use of sketches [6,21,7,2], enabling all nodes to produce a result. These algorithms are fast (i.e. obtaining an estimate in a number of rounds close to the diameter value of the network graph), but not accurate. Another interesting alternative is provided by *averaging* techniques [16,12,4,14,1], which can reach an arbitrary accuracy, with the estimate at all nodes converging to the correct result over time.

Most of the existing approaches allow the distributed computation of aggregation functions, such as COUNT, AVERAGE, SUM, MAX/MIN, and therefore the calculation of many scalar values that can result from the combination of those functions. However, they are unable to compute more complex aggregates which provide a richer information about some property, such as the frequency distribution of an attribute. In fact, few approaches are found in the literature that allow the distributed estimation of statistical distributions [9,23,10,22], and those found exhibit robustness and accuracy issues.

Algorithms like [23] and [9] require a tree routing structure to produce an approximation of the distribution at the root, operating similarly to common tree-based aggregation techniques. In particular, each node computes a quantile summary (i.e. digest) holding the data from its sub-tree (e.g. range of values and corresponding counts) which are built in a bottom-up fashion toward the root. Like in classic tree-based approaches, a single failure may affect the aggregation process, leading to the loss of the data from a subtree.

A first gossip-based distribution estimation approach was proposed in [10], randomly exchanging and merging finite lists of bins (i.e. pairs with value and respective counter) between nodes. Initially, the list of bins at each node is set with the initial input value, and after several rounds all will produce an approximation of the distribution of values (i.e. histogram). Different merging techniques were considered by the authors, the one referred to as *equi-depth* showed to be the one with the best results (accuracy vs storage) compared to the others. The *equi-depth* method intends to minimize the counting disparity between bins. In particular, upon reception, received and local pairs are ordered and the pairs of consecutive values with the smallest combined count are merged (i.e. counts are added and the new value results from the weighted average) repeatedly until the desired number of bins is obtained. This approach allows data to reach all nodes through multiple paths (in this sense improving the robustness), but also gives rise to the occurrence of duplicates that will bias the produced estimate. This problem was acknowledged by the authors, arguing that it was better (i.e. simpler and efficient) not to try to solve it.

Adam2 [22] is a more recent gossip based approach to approximate distributions, more precisely CDF. This approach is based on the application of a classic averaging technique, namely Push-Pull Gossiping [12], and at a high abstrac-

tion level it can be simply described as the simultaneous execution of multiple instances of this protocol. In more detail, Adam2 considers a fixed list of $k$ pairs $(s_k, e_k)$, where $s_k$ is an interpolation point and $e_k$ is the fraction of nodes with a value $x$ less or equal than $s_k$. Each node $i$ that starts participating in the protocol initializes its list of pairs setting $e_k = 1$ if $x_i \leq s_k$ and $e_k = 0$ otherwise. Then, the Push-Pull Gossiping process is applied, each node randomly picking a neighbor to exchange their list of pairs and individually averaging the fractions corresponding to each interpolation point. Over time, the fractions will (be expected to) converge at all nodes to the correct value in each pair. Adam2 solves the duplication problem of the previous *equi-depth* method, considerably outperforming it according to the provided evaluation results. Nevertheless, as will be showed in Section 5, Adam2 inherits the "mass loss" problems of Push-Pull Gossiping, not converging to the correct result even in fault-free scenarios [13].

This work proposes a truly fault-tolerant and more accurate alternative, with the fractions of each interpolation point effectively converging at all nodes over time and simultaneously supporting dynamic changes.

## 3 System Model

Our model assumes the existence of a large number of distributed processes or nodes. Our goal is to estimate an accurate distribution of an attribute over the network of processes with a robust aggregation strategy.

The network of distributed nodes is modeled as a connected undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with the set $\mathcal{V}$ representing processes and the set $\mathcal{E}$ being bidirectional communication links between processes. We represent the set of adjacent nodes of node $i$ by $\mathcal{D}_i$.

The algorithm is executed synchronously as described in [17](Chapter 2). Each node executes two procedures in lockstep each round: they begin execution by generating messages to deliver to neighbors and sending them. Afterwards, nodes compute their new state as a function of its current state, the observed value and received messages from neighbors. Nodes do not have global Ids and have only to distinguish the members of the set of neighbors.

Message loss are taken into consideration and modeled as follows: per round, each sent message can be dropped according to a predefined uniform random probability. In terms of dynamic changes (input values and churn), it is assumed that they occur at the beginning of each round (i.e. before the message generation procedure). Departing nodes are chosen uniformly at random, and it is assumed that they do not return to the network (i.e. leave forever). Arriving nodes connect to random points of the network (according to the considered topology) and establish a number of links matching the network properties (i.e. degree).

## 4 Spectra – Robust Distribution Estimation

In this section we describe a novel distributed algorithm, *Spectra*, to estimate the distribution of a global attribute, more specifically its Cumulative Distribution

Function (CDF). A CDF can be approximated by a mapping from points to the frequencies of the values that are less than or equal to each point. More precisely, considering $n$ nodes and an input value $x_i$ at each node $i$, the CDF of $x$ can be approximated by a set of $k$ pairs $(s, e)$, where $s$ is an interpolation point and $e$ is the fraction of values less or equal than $s$ (i.e., $e = |\{x_i \mid x_i \leq s\}| / n$).

Considering each pair $(s, e)$ in the CDF, it is possible to estimate $e$ through an averaging algorithm as follows: setting 1 as the input value of each node $i$ that satisfies the condition $x_i \leq s$ and 0 otherwise, the average of these input values will be the fraction of nodes that fulfill the previous condition, i.e. it will be $e$. This means that $e$ can be computed as result of the execution of a distributed averaging algorithm.

The main idea of the proposed algorithm is the combination of this observation with the adaptation of a robust distributed averaging algorithm, Flow Updating [14] (FU), to work with vectors instead of scalars, one component of the vector for each point of the CDF. Simultaneously, whilst the algorithm is converging, a distributed computation of the global minimum and maximum of the values is performed to determine the interval in which the $k$ points of the CDF are estimated.

This new algorithm is referred to as *Spectra*, and its core is based on the application of FU to estimate a CDF. The computation performed at each node $i$ is detailed by Algorithm 1. The algorithm adapts FU averaging to use vectors instead of scalars. Namely, the flows $F_i$ map for each neighbor a vector of flows (one for each point in the CDF); $\boldsymbol{v_i}$ is a vector which contains the contribution of the node according to the input value $x_i$, being used as the input to the FU algorithm; and the estimation function yields a vector of estimates, the $k$ points of the CDF.

The algorithm does not assume knowledge of the global minimum and maximum values. Instead, each node keeps a local knowledge of the minimum of maximum known so far in the interpolation interval state variable ($I_i$). The interval is sent in messages to neighbors, which merge the received intervals. After $d$ rounds, where $d$ is the network diameter, each node $i$ will have the the global minimum and maximum values in the $I_i$ variable.

The present algorithm computes an equi-width approximation of the CDF at $k$ equi-distant points in the interval between the global minimum and maximum (although other variants are possible). We use a notation where an interval $I = (l, u)$ is indexed, i.e., $I(j)$, with $j$ from 0 to $k - 1$, resulting in the $k$ equi-distant points of interest from the lower to the upper value (line 29). In this paper we assume that the number of points, $k$, is fixed and known to all nodes. It is possible to relax this assumption and derive a system where $k$ can be adapted in execution time.

Before global minimum and maximum convergence, the vectors calculated at each node or in different rounds refer to different points. At each iteration, as a new interval is computed by merging intervals in messages, the algorithm needs to transform both the received vectors as well as the vectors from the previous iteration, so that they are meaningful for the new (and potentially different) set

Algorithm 1: Spectra: Algorithm to estimate CDF in distributed networks.

1 **inputs:**
2 $x_i$, value to aggregate
3 $\mathcal{D}_i$, set of neighbors
4 $k$, number of interpolation points
5 **state variables:**
6 flows: initially, $F_i = \{\}$ `/* mapping from neighbors to flow vectors */`
7 base frequency vector: initially, $\boldsymbol{v_i} = [1 \mid 0 \leq j < k]$
8 interpolation interval: initially, $I_i = (x_i, x_i)$
9 **message-generation function:**
10 $\text{msg}_i(F_i, \boldsymbol{v_i}, I_i, j) = (i, I_i, \boldsymbol{f}, \text{est}(\boldsymbol{v_i}, F_i))$;
11 **with** $\boldsymbol{f} = \begin{cases} F_i(j) & \text{if } (j, \_) \in F_i \\ \boldsymbol{0} & \text{otherwise} \end{cases}$
12 **state-transition function:**
13 $\text{trans}_i(F_i, \boldsymbol{v_i}, I_i, M_i) = (F_i', \boldsymbol{v_i}', I_i')$
14 **with**
15 $I_i' = \text{merge}(I_i \cup \{I \mid (\_, I, \_, \_) \in M_i\})$
16 $\boldsymbol{v_i'} = [\text{if } x_i \leq I_i'(j) \text{ then 1 else } 0 \mid 0 \leq j < k]$
17 $F = \{j \mapsto -\text{transform}(\boldsymbol{f}, I, I_i') \mid j \in \mathcal{D}_i \wedge (j, I, \boldsymbol{f}, \_) \in M_i\} \cup$
18 $\quad \{j \mapsto \text{transform}(\boldsymbol{f}, I_i, I_i') \mid j \in \mathcal{D}_i \wedge (j, \_, \_, \_) \notin M_i \wedge (j, \boldsymbol{f}) \in F_i\}$
19 $E = \{i \mapsto \text{est}(\boldsymbol{v_i'}, F)\} \cup$
20 $\quad \{j \mapsto \text{transform}(\boldsymbol{e}, I, I_i') \mid j \in \mathcal{D}_i \wedge (j, I, \_, \boldsymbol{e}) \in M_i\} \cup$
21 $\quad \{j \mapsto \text{transform}(\text{est}(\boldsymbol{v_i}, F_i), I_i, I_i') \mid j \in \mathcal{D}_i \wedge (j, \_, \_, \_) \notin M_i\}$
22 $\boldsymbol{a} = (\sum\{\boldsymbol{e} \mid (\_, \boldsymbol{e}) \in E\})/|E|$
23 $F_i' = \{j \mapsto \boldsymbol{f} + \boldsymbol{a} - E(j) \mid (j, \boldsymbol{f}) \in F\}$
24 **estimation function:**
25 $\text{est}(\boldsymbol{v}, F) = \boldsymbol{v} - \sum\{\boldsymbol{f} \mid (\_, \boldsymbol{f}) \in F\}$
26 **interval merging:**
27 $\text{merge}(S) = (\min(\{l \mid (l, \_) \in S\}), \max(\{u \mid (\_, u) \in S\}))$
28 **interval interpolation:**
29 $(l, u)(j) = l + j \times (u - l)/(k - 1)$
30 **vector transformation function:**
31 $\text{transform}(\boldsymbol{u}, I, I') = [\boldsymbol{u}(\max(\{0\} \cup \{l \mid 0 \leq l < k \wedge I(l) \leq I'(j)\})) \mid 0 \leq j < k]$

of $k$ points. For that all vectors involved in the execution of the algorithm are transformed from their old to the new interval through the vector transformation function (line 31). This function implements a simple heuristic to obtain the new vector, using the value corresponding to the largest point not greater than the new point (or the first in the vector if no such point exists). Also, the vector of input values to FU, $\boldsymbol{v_i}$, is calculated at each round according to the new interval (line 16).

At each round, in the message generation function (lines 9–11), a single type of message is sent, containing the self id $i$, its interpolation interval $I_i$, the flows vector $\boldsymbol{f}$ for each current neighbor $j$. Sent flows are set according to the current state and otherwise (initially or when a node is added) to 0.

The state-transition function (lines 12–23) takes state (i.e., flows $F_i$, the node's base frequency vector $\boldsymbol{v_i}$, interpolation interval $I_i$) and the set of messages $M_i$ received by the node and returns a new state (i.e., flows $F_i'$, base frequency vector $\boldsymbol{v_i'}$ and interpolation interval $I_i'$). It computes the new interpolation interval setting the lower bound with the minimum of the received minima and does the upper bound with the maximum (line 27). Base frequency vector $\boldsymbol{v_i'}$ is computed from the new interpolation interval $I_i'$ and the initial value $x_i$. Then, the averaging steps are executed according to FU taking care to transform the involved vectors in order to apply the averaging process to matching interpolation points. These steps result in the creation of the new flows.

The self-adapting nature of the core averaging algorithm, Flow-Updating, on *Spectra* enables it to cope with the dynamic adjustment of all involved vectors. In particular, *Spectra* supports dynamic network changes (i.e., nodes arriving/leaving), simply by adding/removing the flow data associated to neighbors. Moreover, it is also able to seamlessly adapt to changes of the input value $x_i$ – in this case simply by recomputing the vector $\boldsymbol{v_i}$. This is sufficient to allow *Spectra* to operate in settings where the global maximum and minimum do not change.

If the extreme values change due to dynamism, especially if the maximum decreases and the minimum increases, the algorithm as it is will not produce wrong results, but over an overly wide interval. To tighten the interval to the interesting range between the new minimum and maximum additional modifications must to be made. This dynamic adjustment of the global extreme values (i.e., maximum and minimum) is left for future work.

At each node $i$, the estimated CDF at the $k$ equi-distant points in the interval $I_i$ is obtained by the estimation function (i.e., $\text{est}(\boldsymbol{v_i}, F_i)$). Over time, the estimated frequency associated to each point converges to the correct value. This is confirmed by the results obtained from evaluation (see Section 5).

## 5 Evaluation

The results presented in this work have been obtained using a custom made simulator that implements the model defined in section 3.

We used two error metrics to quantify the fitness of the estimate to the underlying distribution. They are computed at every round.

The basis of these metrics is the Kolmogorov–Smirnov statistic, that calculates the maximum label-wise difference between the estimate and the distribution, as presented in Equation 1. The metric is computed at every round $r$, for each node $n$. For every label $l$, the measure is given by the difference between the cumulative value of the real distribution and the cumulative value of the estimated distribution on node $n$ at round $r$.

$$KS_r^n = \max_{l \in Labels} \mid P(X \leq l)_r - \widehat{P(X \leq l)}_r^n \mid \quad (1)$$

We use global metrics for the whole network: one to reflect the worst node (see Equation 2) and another to reflect the average error (see Equation 3). Both equations are computed every round $r$.

$$KS_{maxr} = \max_{n \in N}(KS_r^n) \quad (2)$$

$$KS_{\mu r} = \frac{1}{\mid N \mid} \sum_{n \in N} KS_r^n \quad (3)$$

### 5.1 Comparison with an existing approach

To the best of our knowledge, this work is pioneer in the estimation of statistical distributions with fault tolerant and robustness properties.

In *Adam2* [22], whose protocol is based on the *Push-Pull gossiping* averaging algorithm [11], presents a few drawbacks stemming from the fact that it behaves poorly under message loss and also because the simulator used on the above-cited work (PeerSim) does not emulate synchronous message exchange correctly, assuming atomic state changes - this behavior is, from our point of view, unrealistic when considering real systems.

*Adam2* partitions the range of the monitored property in a set of interpolation points. Nodes start the algorithm with a pre-known minimum and maximum and with equally spaced interpolation points. Probabilistically, new instances are created every few rounds. These new instances re-compute those points based on the previous instance's points set, using re-sampling heuristics that aim to minimize interpolation errors, i.e., it aims to concentrate interpolation points in the areas where frequency counts are more prevalent.

In order to compare our approach with the strategy used on *Adam2*, we assume that both minimum and maximum are previously known to all nodes and the sampling points are evenly distributed between minimum and maximum. These assumptions do not invalidate the usefulness of the re-sampling heuristics presented, but help us compare the performance of both approaches in a common frame of reference. Also, these heuristics are also applicable to *Spectra* in a scenario with multiple instances, but that falls out of the scope of the present work.

We've simulated the following scenarios using a 1000 nodes random network with an average connectivity of 3, unless stated otherwise. The underlying initial values follow a Gaussian distribution $rNorm(10, 2)$, mean 10 and variance 2. Results were averaged from 30 trials for each scenario.

Figure 1 presents a graph with the average Kolmogorov-Smirnov distance to the real distribution on the nodes (as per 3). It shows the performance of both algorithms. One can observe that *Adam2* converges asymptotically to a non-zero value with a continuous offset error while *Spectra* converges indefinitely to

zero. Also, the convergence speed is notoriously higher in *Spectra*, with orders of magnitude smaller error.

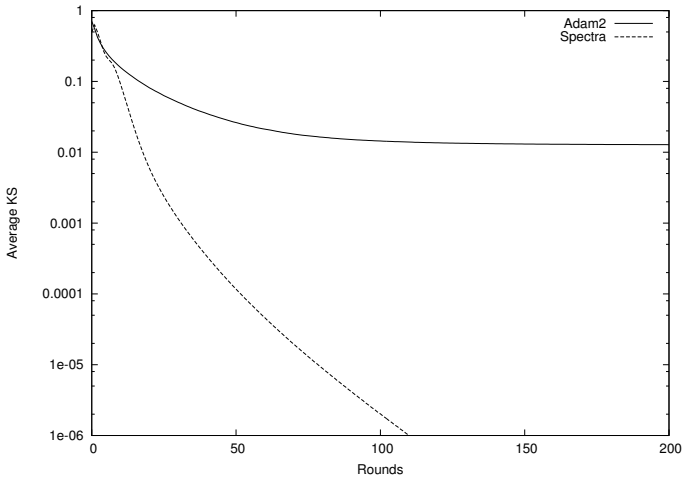


**Fig. 1.** Average KS error rate on a 1000 nodes random network comparing *Spectra* and *Adam2*

### 5.2 Fault tolerance

In this scenario we have simulated message loss rates of 5%, 10% and 20% in each round. Results are presented in 2.

Regarding *Spectra*, we notice a slower overall convergence rate with 5% message loss when compared to the other loss reates. This indicates that the algorithm is not only resilient to message loss but also that with a message loss rate of up to 20%, the convergence rate improves. This result is coherent with results obtained in [14].

This emergent behavior is in a way contradictory with the intuition that convergence performance should degrade with message loss increase. Simulation results suggest the opposite. This behavior may be understood if we look at message loss as momentary changes in the network topology (a lost message is equivalent to the extinction of an edge during a round). This effect is justified by the fact that the number of cycles in the network topology tend to deteriorate the convergence performance in the underlying averaging algorithm [14].

We have also applied *Adam2* to the same rates of message loss. Loss of messages seem to introduce a systematic offset error.

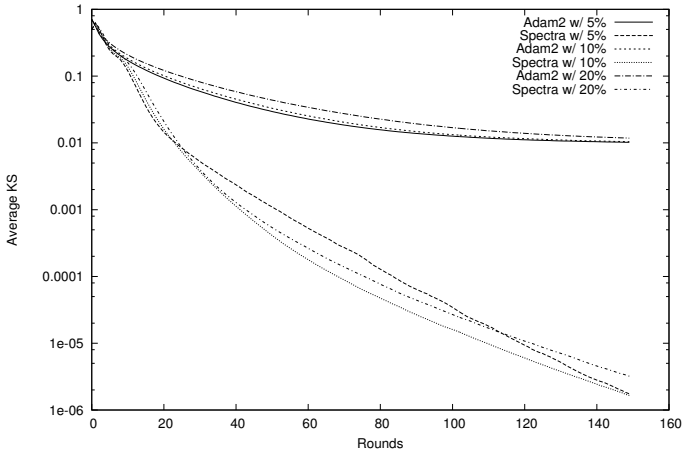


**Fig. 2.** Average KS error rate on a 1000 nodes random network comparing message loss effect on *Spectra* and *Adam2*.

### 5.3 Dynamic adaptation to changes

In order to evaluate the algorithm's behavior under dynamics in the sampled value, we have conducted a simulation that introduces a disturbance on 20% of the nodes chosen uniformly random at round 75. The disturbance increased sampled values in 10%, with care not to change the minimum nor maximum of the network. The issues concerning changes in minimum and maximum and the way it affects the estimate will be addressed in future work.

The obtained results illustrate the adaptive nature of the proposed algorithm (see 3). Without need to restart the protocol, the error in the estimate increases at the moment of the disturbance and quickly converges to error rates similar to pre-disturbance values. This behavior stems from the averaging protocol used underneath and to the way each node preserves its own sampled value and relative position. Iteratively, as rounds progress, all nodes adjust their estimates taking into account the perceived value changes and the consequent flow adjustments.

In order to test the resilience of *Spectra* to churn, we have submitted the algorithm to the departure and arrival of new nodes. In particular, it starts with a network of 1000 nodes and at round 50, the number of nodes starts to linearly increase (a 1% increase per round), up to 1250 nodes at round 75. Then, after 50 rounds of stability, nodes randomly leave the network at the same rate until it reaches again 1000 nodes. Node departure is equivalent to nodes crashing, as they leave silently without notifying any neighbors. In order to prevent network partitioning, the average node degree has been increased to 7 (i.e. $\approx ln(1000)$), following what has been done in [14]. Data presented in Figure 4 depicts the average KS (as per Equation 3) and maximum KS error (as per Equation 2) for the whole procedure. It also depicts a profile with the number of nodes that

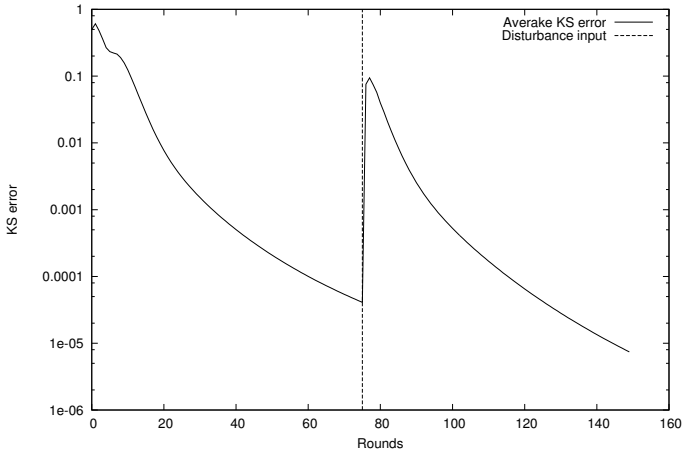


**Fig. 3.** Average KS error on a 1000 nodes random network running *Spectra* with disturbance on initial values.

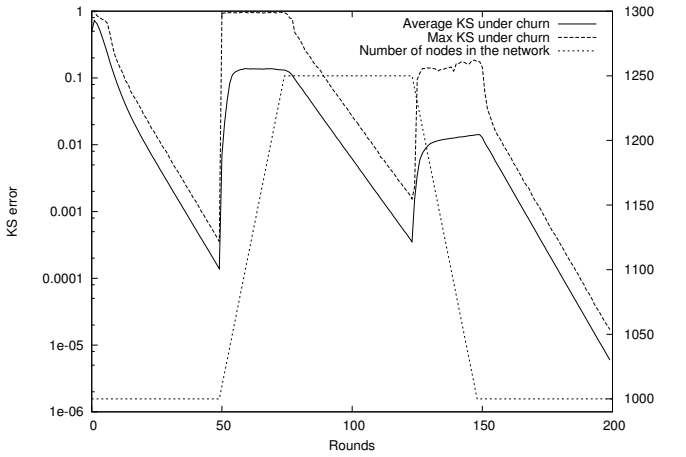


**Fig. 4.** Average KS error, maximum KS error and node count on a random network running *Spectra* while subjected to churn.

constitute the network throughout the rounds. The arrival of nodes introduces new values to the distribution. The estimation has to be adjusted and thus the surge in the error levels. As soon as the node number stabilizes, the error levels decrease. Node departure introduces a similar effect. These results show the algorithm's adaptability to high churn rates and how quickly it converges to near zero error. The estimates are computed uninterruptedly, without any

need to restart the algorithm - this property makes it highly adaptable and fault tolerant.

## 6 Conclusions and Future Work

We've presented a distributed algorithm that computes the estimate of cumulative distributed functions over a large scale network. The proposed algorithm, *Spectra*, overcomes the problems that previous approaches exhibited. Our solution converges to the correct distribution, even when facing high levels of message loss and churn in the network membership and topology. It also allows dynamic adaptation to changes in the monitored values (and their distribution), and avoids the need to re-start the algorithm and loose progress.

All the nodes have access to a high precision estimate of the CDF, and can infer the associated distribution function. This data, being richer than more simple statistics (e.g. average) allows a precise characterization of the target network property and permits more accurate control decisions in the presence of outliers and skewed distributions.

As future work we intend to evolve the technique in order to allow for variations in the minimum and maximum of the target property, since currently we only adapt to growing maxima and decreasing minima. Another improvement is in allowing an adaptive growth in the number of sampled intervals, $k$, that is fixed at present. Finally we plan to address strategies for consistent placement of the sample points, that will be no longer uniform across the min-max range, as this will permit increased sampling in areas where the changes in the property are more expressive, and further increase precision.

## References


1. Almeida, P.S., Baquero, C., Farach-Colton, M., Jesus, P., Mosteiro, M.A.: Fault-Tolerant Aggregation: Flow Update Meets Mass Distribution. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) 15th International Conference On Principles Of Distributed Systems (OPODIS). pp. 513–527. Lecture Notes in Computer Science, Springer Berlin/Heidelberg (2011)
2. Baquero, C., Almeida, P.S., Menezes, R., Jesus, P.: Extrema Propagation: Fast Distributed Estimation of Sums and Network Sizes. IEEE Transactions on Parallel and Distributed Systems (PrePrints) (2012)
3. Birk, Y., Keidar, I., Liss, L., Schuster, A., Wolff, R.: Veracity Radius: Capturing the Locality of Distributed Computations. In: 25th annual ACM symposium on Principles of Distributed Computing (PODC). pp. 102–111. ACM (2006)
4. Chen, J.Y., Pandurangan, G., Xu, D.: Robust Computation of Aggregates in Wireless Sensor Networks: Distributed Randomized Algorithms and Analysis. IEEE Transactions on Parallel and Distributed Systems 17(9), 987–1000 (2006)
5. Cheng, S., Li, J., Ren, Q., Yu, L.: Bernoulli Sampling Based ($\varepsilon$, $\delta$)-Approximate Aggregation in Large-Scale Sensor Networks. In: 29th IEEE conference on Information communications (INFOCOM). pp. 1–9 (2010)

6. Considine, J., Li, F., Kollios, G., Byers, J.: Approximate aggregation techniques for sensor databases. In: 20th International Conference on Data Engineering (ICDE). pp. 449–460 (2004)
7. Fan, Y.C., Chen, A.L.: Efficient and Robust Schemes for Sensor Data Aggregation Based on Linear Counting. IEEE Transactions on Parallel and Distributed Systems 21(11), 1675–1691 (2010)
8. Ganesh, A.J., Kermarrec, A.M., Le Merrer, E., Massoulié, L.: Peer counting and sampling in overlay networks based on random walks. Distributed Computing 20(4), 267–278 (2007)
9. Greenwald, M., Khanna, S.: Power-Conserving Computation of Order-Statistics over Sensor Networks. In: 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles Of Database Systems (PODS). pp. 275–285. ACM (2004)
10. Haridasan, M., van Renesse, R.: Gossip-based Distribution Estimation in Peer-to-Peer Networks. In: 7th International conference on Peer-To-Peer Systems (IPTPS). USENIX Association (2008)
11. Jelasity, M., Montresor, A.: Epidemic-style proactive aggregation in large overlay networks. In: In Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04. pp. 102–109. IEEE Computer Society (2004)
12. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems 23(3), 219–252 (2005)
13. Jesus, P., Baquero, C., Almeida, P.S.: Dependability in Aggregation by Averaging. In: Simpósio de Informática (INForum) (2009)
14. Jesus, P., Baquero, C., Almeida, P.S.: Fault-Tolerant Aggregation for Dynamic Networks. In: 29th IEEE Symposium on Reliable Distributed Systems. pp. 37–43 (2010)
15. Jesus, P., Baquero, C., Almeida, P.S.: A Survey of Distributed Data Aggregation Algorithms. Tech. Rep. CoRR abs/1110.0725, HASLab / INESC TEC, Universidade do Minho (2011), http://arxiv.org/abs/1110.0725
16. Kempe, D., Dobra, A., Gehrke, J.: Gossip-Based Computation of Aggregate Information. In: 44th Annual IEEE Symposium on Foundations of Computer Science. pp. 482–491 (2003)
17. Lynch, N.A.: Distributed algorithms. pp. 17–23. Morgan Kaufmann Publishers Inc. (1996)
18. Madden, S., Franklin, M., Hellerstein, J., Hong, W.: TAG: a Tiny AGgregation service for ad-hoc sensor networks. ACM SIGOPS Operating Systems Review 36(SI), 131–146 (2002)
19. Mane, S., Mopuru, S., Mehra, K., Srivastava, J.: Network Size Estimation In A Peer-to-Peer Network. Tech. rep., Department of Computer Science, University of Minnesota (2005)
20. Motegi, S., Yoshihara, K., Horiuchi, H.: DAG based In-Network Aggregation for Sensor Network Monitoring. In: International Symposium on Applications and the Internet (SAINT). pp. 292–299. IEEE Computer Society (2006)
21. Nath, S., Gibbons, P., Seshan, S., Anderson, Z.: Synopsis diffusion for robust aggregation in sensor networks. In: 2nd International Conference on Embedded Networked Sensor Systems (SenSys). pp. 250–262. ACM (2004)
22. Sacha, J., Napper, J., Stratan, C., Pierre, G.: Adam2: Reliable distribution estimation in decentralised environments. In: Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on. pp. 697 –707 (june 2010)
23. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: new aggregation techniques for sensor networks. In: 2nd International Conference on Embedded Networked Sensor Systems (SenSys). pp. 239–249 (2004)