Extrema Propagation: Fast Distributed Estimation of Sums and Network Sizes

Carlos Baquero Paulo Sérgio Almeida Raquel Menezes

DI/CCTC and DMCT, Universidade do Minho

Abstract

Aggregation of data values plays an important role on distributed computations, in particular over peer-to-peer and sensor networks, as it can provide a summary of some global system property and direct the actions of self-adaptive distributed algorithms. Examples include using estimates of the network size to dimension distributed hash tables or estimates of the average system load to direct load-balancing.

Distributed aggregation using non-idempotent functions, like sums, is not trivial as it is not easy to prevent a given value from being accounted for multiple times; this is specially the case if no centralized algorithms or global identifiers can be used.

This paper introduces a novel technique, Extrema Propagation, for distributed estimation of the sum of positive real numbers. It is more expressive than previous approaches as it encompasses summing naturals and counting. As a special important case we show how it can be applied to network size estimation.

The technique relies on the exchange of duplicate insensitive messages and can be applied in flood and/or epidemic settings, where multi-path routing occurs; it is tolerant of message loss; it is fast, as the number of message exchange steps can be made just slightly above the theoretical minimum; and it is fully distributed, with no single point of failure and the result produced at every node.

1 Introduction

Aggregation is recognized as an important building block for distributed applications in peer-to-peer or sensor network infrastructures [18, 11, 12]. Aggregating data values can provide a summary of some global system property and play an important role in directing the actions of self-adaptive distributed algorithms.

Examples can be found when using estimates of the network size to direct the dimensioning of distributed hash table structures [17], when setting a quorum for voting algorithms [1], when estimates of the average system load are needed to direct local load-balancing decisions or when an estimate of the total disk space in the network is required in a P2P sharing system.

Distributed computation of aggregation functions in a network is not trivial. Unlike aggregation in a tree [14, 13], where each value is guaranteed to contribute only once, in a graph it is not easy to prevent a given value from being accounted for multiple times; this is specially the case if no centralized algorithms or global identifiers can be used. Thus, calculating general non-idempotent functions (e.g. COUNT, SUM, AVG) is problematic and we are restricted to idempotent functions that are duplicate insensitive (e.g. MIN, MAX) [15, 2]. Aggregation functions that can be made duplicate insensitive have the advantage of being usable under multi-path routing.

This paper introduces a novel technique, Extrema Propagation, for distributed estimation of the sum of positive real numbers. It is a probabilistic technique that exchanges duplicate insensitive messages and thus can be applied in flood and/or epidemic settings, where multi-path routing occurs, and is tolerant of message loss.

This technique has some important properties: the precision is controlled by message size, independently of network size; it is fast: the number of message exchange steps can be made just slightly above the theoretical minimum; it is fully distributed, with no single point of failure, and the result is produced at every node. As a special important case (and for presentation purposes) we show how this technique can be applied to network size estimation.

2 Network Size Estimation

In order to simplify the description we concentrate on a specific counting problem: *How many nodes are present in a given network?* Moreover, we aim for a distributed assessment of such estimate and to have it available at every node after a short number of message exchange steps.

Our assumptions are: (1) Each node can communicate with a set of neighbour nodes; (2) Each node has access to a random number generator. We also make use of some assumptions that, although not necessary for this class of algorithms, simplify the presentation and analysis: (a) messages are not lost or corrupted; (b) the network is static and completely connected; (c) connections are bidirectional (the graph is undirected). Message loss is addressed informally in a later section.

2.1 Minimum Number of Steps Towards Estimation

Our technique avoids the construction of a tree, and works directly on an unstructured network where each node only needs to know its neighbours. It should be noticed that tree construction over an unstructured network would require a number of message exchange steps proportional to D, the network diameter (between $D/2$ and D, depending on the node chosen for the tree root). Subsequent aggregation along the tree would again require an identical number of message exchanges steps. Moreover, such procedure would not tolerate link failures and the calculated result would be available at just a single node (the tree root, a single point of failure); dissemination to other nodes would require further message exchange steps. Therefore, making the result available at every node would take at least $3D/2$ steps in the best case. A tree based algorithm cannot reach the theoretical lower bound for the number of steps, which can trivially be shown to be D.

Proposition 1. *Let D be the network diameter: the maximum length of the shortest paths between two nodes in the graph. Obtaining at every node an estimate of the number of nodes needs at least D message exchange steps.*

Proof. Consider two nodes, a and b, such that the shortest path between them has length D. In order to have in a an estimate of the network size that takes b into account, one needs at least D message exchange steps. □

Thus, the fastest estimations cannot be done in less than D steps (to be available at every node; it can be made earlier at some nodes, and the average number of rounds across the network can be less than D). Our technique is close to optimal in the number of steps, as it terminates in some small steps above the theoretical minimum. As we will see, this extra steps are needed just in the termination detection, as the estimate is already calculated before, but cannot be reported as available.

2.2 Synopsis of the Estimation Technique

Our approach to estimation is based on finding an idempotent message structure that allows the counting of nodes. One trivial approach would be the use of one unique identifier per node (an additional assumption) and a protocol that collects the set of all identifiers, aggregating by set union. Such protocol would estimate in D steps, but creates messages that are linear with the network size.

Our technique avoids the need for unique identifiers and aggregate sizes which depend on network size [16]. It is based on idempotent operations on numbers, more specifically the minimum function, and the use of statistical inference.

The insight to our approach is the following: if we generate a random real number in each node using a known probability distribution (e.g. Gaussian or exponential), and aggregate across all nodes using the minimum function, the resulting value has a new distribution which depends on the number of nodes.

The basic idea is then to generate a vector of random numbers at each node, aggregate each component across the network using the pointwise minimum, and then use the resulting vector as a sample from which to infer the number of nodes (by a maximum likelihood estimator).

We will show that if a vector of K numbers is generated per node, it is possible to provide an estimate $\widehat{N}$ of the network size N with a standard deviation of $N/\sqrt{K-2}$. This means that the relative accuracy can be chosen independently of the network size, and is determined by K.

If we want to enforce a maximum relative error $r = |\widehat{N} - N|/N$ with a confidence of 95% we need to make $K = 2 + \left(\frac{1.96}{r}\right)^2$. For example, for an error $r = 10\%$, one needs to make $K = 387$.

The focus of our technique is not accuracy but speed: we do not aim for very low errors (e.g. 1% would lead to large messages) but for a fast computation of an useful approximation that can serve as input to some other algorithm (in some cases even 10% may be more than enough, only the order of magnitude may be needed).

2.3 Basic Extrema Propagation

The basic algorithm that every node runs is shown in Algorithm 1. Each node maintains a vector x of K numbers, initialized using function $rExp(1)$, which returns a random number with an exponential distribution of rate parameter 1.

Algorithm 1 Basic Extrema Propagation

```
  const $K$
  var $n, x[1..K]$
Upon: Init
  $n \leftarrow neighbours(self)$
  for all $i \in 1..K$ do $x[i] \leftarrow rExp(1)$
  Send $x$ to every $p \in n$
Upon: Receive $m_1..m_j$ from all $p \in n$
  for all $l \in 1..j$ do
    $x \leftarrow pointwisemin(x, m_l)$
  end for
  Send $x$ to every $p \in n$
Upon: Query
  return $\hat{N}(x)$
```

The algorithm consists of a series of rounds towards convergence. In each round every node sends a message containing vector x to its neighbours, collects the corresponding messages from its neighbours and computes the pointwise minimum of x and all corresponding vectors received, updating x with the result.

Each node uses function $\hat{N}(x)$, which takes as parameter the vector of K aggregated minimums, and returns an estimation of the number of participants (network size). In this first version we do not deal with termination and assume that a node can be queried at any time, possibly before convergence is reached, i.e. before we have collected the pointwise minimum of every vector in the network. Termination is addressed below.

3 Estimation Function

We first introduce the maximum likelihood estimator $\widehat{N}_F$ used to estimate the unknown parameter N. We then proceed with the theoretical study of its main properties, namely bias and variance. The likelihood function is obtained from the extreme value theory, which is a branch of statistics dealing with the extreme deviations from the median of probability distributions. Next results deal with deviations imposed by the minimum function, but similar results can be easily derived for the maximum.

Proposition 2. *Let $F_{min}(x) = 1 - (1 - F(x))^N$ be the limiting distribution for the minimum of a large collection $X_1, ..., X_N$ of random observations from the same arbitrary distribution $F(x)$ [7].*

Given a vector of K minimums $x[1], ..., x[K]$, which are observed values from $F_{min}(x)$ distribution, then the maximum likelihood estimator for the unknown parameter N is

$$\widehat{N}_F = -\frac{K}{\sum_{i=1}^{K} \log\{1 - F(x[i])\}}. \qquad (1)$$

Proof. The limiting density for the minimum is $f_{min}(x) = \frac{d}{dx} F_{min}(x) = N f(x)(1 - F(x))^{N-1}$, where $f(x) = \frac{d}{dx} F(x)$. According to the likelihood method, we wish to maximize the function $L(N) = \prod_{i=1}^{K} f_{min}(x[i])$, or equivalently, to maximize $\log L(N)$ where $\log L(N) = K \log N +$

$\sum_{i=1}^{K} \log f(x[i]) + (N-1) \sum_{i=1}^{K} \log\{1-F(x[i])\}$. From $\frac{d}{dN} \log L(N) = 0$ one concludes that

$$N = -\frac{K}{\sum_{i=1}^{K} \log\{1 - F(x[i])\}}.$$

□

We now concentrate on the special case of using the exponential distribution for $F(x)$ as it will lead to a simple estimator. We will also derive a unbiased estimator for this distribution. (The generic estimator $\hat{N}_F$ above is not necessarily unbiased.) We denote the exponential distribution with rate 1 by $Exp(1)$.

Now, $F(x) = 1 - e^{-x}$, $x \geq 0$ and the corresponding estimator for N becomes

$$\widehat{N}_{Exp} = \frac{K}{\sum_{i=1}^{K} x[i]}.$$

Moreover, $F_{min}(x) = 1 - e^{-Nx}$, $x \geq 0$, is an exponential distribution with rate N, denoted by $Exp(N)$.

In order to correct the bias in $\widehat{N}_{Exp}$ there is a need for an auxiliary lemma.

Lemma 3. *If $X_1, \ldots, X_k$ are independent random variables (r.v.'s) from distribution $Exp(N)$, then*

a) $\sum_{i=1}^{K} X_i$ is a r.v. from a gamma distribution with shape and scale parameters equal to K and N, respectively.

b) Furthermore, the next expectation and variance hold:

$$\mathrm{E}\left[\frac{1}{\sum_{i=1}^{K} X_i}\right] = \frac{N}{K-1}$$

and

$$\mathrm{Var}\left[\frac{1}{\sum_{i=1}^{K} X_i}\right] = \frac{N^2}{(K-1)(K-2)} - \frac{N^2}{(K-1)^2}.$$

Proof. The proof for a) is straightforward from the classic theory of Mathematical Statistics (see e.g. [9]). The proof for b) considers the well known results $\mathrm{E}[g(X)] = \int g(x)f(x)dx$, where $f(x)$ is the density function of r.v. X, and $\mathrm{Var}[g(X)] = \mathrm{E}[g(X)^2] - \mathrm{E}[g(X)]^2$. □

It is now possible to introduce an unbiased estimator for N.

Proposition 4. *The estimator given by*

$$\widehat{N} = \frac{K-1}{K}\, \widehat{N}_{Exp} = \frac{K-1}{\sum_{i=1}^{K} x[i]} \tag{2}$$

is unbiased.

Proof. We need to prove that the expectation $\mathrm{E}[\widehat{N}]$ is equal to N. Let X_i be the r.v. related to the observed value $x[i]$. First, by Lemma 3, one has

$$\mathrm{E}[\widehat{N}_{Exp}] = \mathrm{E}\left[\frac{K}{\sum_{i=1}^{K} X_i}\right] = K\, \frac{N}{K-1}$$

and

$$\mathrm{E}[\widehat{N}] = \mathrm{E}\left[\frac{K-1}{K}\, \widehat{N}_{Exp}\right] = N.$$

□

Proposition 5. *Variance of $\widehat{N}$ is given by*

$$Var[\widehat{N}] = \frac{N^2}{K-2}.$$

Proof. This proof is again straightforward from the application of Lemma 3

$$\mathrm{Var}[\widehat{N}] = (K-1)^2\, \mathrm{Var}\left[\frac{1}{\sum_{i=1}^{K} X_i}\right] = \frac{N^2}{K-2}.$$

□

We now generalize this result so that one can estimate a sum of positive reals. This new estimator can be applied to a broad class of aggregations that can be expressed by operations on sums, e.g. AVG. Here the variance is determined by the magnitude of the sum that is to be estimated.

Proposition 6. *For $1 \leq i \leq N$, let X_i be independent r.v.'s from distribution $Exp(\lambda_i)$ with $\lambda_i > 0$, and $\mathrm{minimum}(X_1, \ldots, X_N)$ a new r.v. from distribution $Exp(\sum_{i=1}^{N} \lambda_i)$. Given a set of K minimums $x[1], \ldots, x[K]$, which are observed values from $Exp(\sum_{i=1}^{N} \lambda_i)$, then an unbiased estimator for $Sum = \sum_{i=1}^{N} \lambda_i$ is*

$$\widehat{Sum} = \frac{K-1}{\sum_{i=1}^{N} x[i]}$$

with

$$Var[\widehat{Sum}] = \frac{Sum^2}{K-2}.$$

Proof. The proof is straight forward from the proofs of Propositions 4 and 5, renaming N to Sum. $\square$

For presentation purposes, the remaining sections will concentrate on the practical properties and application of the estimator for network size, $\hat{N}$. Nevertheless, most of the analysis is also applicable to the more generic $\widehat{Sum}$ estimator.

4 Binary Encoding

In some application contexts, e.g. mobile ad-hoc networks and sensor networks, message size has an important practical impact both in speed and energy consumption.

Although precision is dictated by the choice of K, there are some relevant design decisions in the floating point encoding of the numbers in the vector. It is intuitive to see that, when aiming for a precision of only a few percent, storing each value naively as a float or double would probably be using a much higher precision than needed. Therefore we tried encoding values with less precision.

After numerically studying several combinations of bit allocations in a binary mantissa and exponent encoding we have concluded that it is appropriate to store only the exponent. Moreover, looking at values that occur in a exponential distribution, and the way that they contribute to the sum in the estimator, even though there can be more than 20 binary orders of magnitudes in the values that occur, a range of only 9 values in the exponent contributes to 99.9% of the result.

Table 1 shows the relative cumulative contribution of values from higher to lower exponents occurring in a exponential distribution. The exponents shown, from 3 to -5 would be the ones contributing almost exclusively to the sum, for $N = 1$ (1 node network). The distribution of minimums for a N node network is also exponential, but with the range of meaningful values scaled by $1/N$. As N is unknown, we must use a range of exponents that is $9 + \log_2(N)$. This leads to using 5 bits for storing the exponent, to account for possibly large networks: 5 bits gives a range of 32 for the exponent; this means networks up to $2^{32-9} = 2^{23}$ nodes. (Using 4 bits would only allow up to $2^{16-9} = 2^7 = 128$ nodes.)

exponent	contribution (%)
3	0.350
2	10.26
1	42.64
0	74.99
-1	91.54
-2	97.53
-3	99.33
-4	99.82
-5	99.95

Table 1: Relative cumulative contribution.

A given real value v in vector x is encoded by the integer $\text{floor}(\log_2 v)$, and when reconstructed becomes $\underline{v} = 2^{\text{floor}(\log_2 v)}$. Likewise, the base 2 discretization of vector x is denoted by $\underline{x}$.

Although $\hat{N}(x)$ was proved to be unbiased, the coarser grain discretization due to encoding introduces a bias in $\hat{N}(\underline{x})$. This bias can be corrected as it is possible to calculate a scale factor $s(K)$ such that $E[\hat{N}(x)] \approx E[s(K)\hat{N}(\underline{x})]$.

We considered the possibility of choosing higher bases for encoding, with the intent of reducing the number of bits needed to encode the same range. We can observe that in order to reduce one bit we need to square the base. Thus, if we encode in 5 bits for base 2, we can encode in 4 for base 4, and in 3 bits for base 16. However the bias correcting scale factor for other bases $b > 2$ shows a non negligible dependence on N, with a periodic oscillation on $\log_b N$. Although some additional corrections can be devised, the total impact on accuracy does not compensate the space savings on encoding bits, since K would need to increase as well.

Calculation of the base 2 scale parameter $s(K)$, was performed numerically and is depicted in Table 2. This value shows a slight dependence on K. This is due to a small change in the shape of the distribution of $\hat{N}$ for small values of K, since the r.v. $\hat{N}$ follows a Gamma distribution with shape parameter K.

Since K is known and configured in the protocol, and the relative periodic oscillations on N are less than 0.001 for base 2, one simply needs to pick the appropriate scale factor for the used K. In short, under binary encoding the estimator for N

K	*sample*	$s(K)$	$sd[s(K)]$
10000	100	0.7212	0.0007
1000	1000	0.7212	0.0008
100	10000	0.7208	0.0008
10	100000	0.7161	0.0008

Table 2: Scale factor $s(K)$ and respective standard deviation. 5 bits, base=2. Using 50 points and *sample* repetitions per point.

K	*sample*	TRE	ORE
10000	10	0.0100	0.0098
1000	100	0.0316	0.0328
100	1000	0.1010	0.1047
10	10000	0.3535	0.3651

Table 3: Theoretical and observed relative errors. 5 bits, base=2. Using 200 points from $N = 1$ to $N = 2^{20}$ and *sample* repetitions per point.

becomes:

$$s(K)\frac{K-1}{\sum_{i=1}^{K} \underline{x[i]}},$$

where $s(K)$ is the scale parameter for a given K as depicted in Table 2.

From the results in Section 3 we can define a metric that indicates the relative error of the estimation. The metric is named TRE (*Theoretical Relative Error*) and is defined as follows:

$$TRE = \frac{\sqrt{Var[\hat{N}]}}{N} = \frac{1}{\sqrt{K-2}}.$$

This metric indicates how the estimation deviates from N as a proportion of N.

In order to numerically measure the quality of the estimator after encoded and scale corrected, we define the following metric, named ORE (*Observed Relative Error*)

$$ORE = \frac{\sqrt{\frac{\sum_{i=1}^{J}\left(\hat{N}_i - N\right)^2}{J}}}{N},$$

where $\hat{N}_i$, for $i = 1..J$, is a set of observations of the estimate of a given N. Both metrics are defined in terms of the MSE (*Mean Square Error*), since *Relative Error* can be seen as $\frac{\sqrt{MSE}}{N}$.

In Table 3 we consider, for each chosen value for $K \in \{10, 100, 1000, 10000\}$, a set of 200 values of N ranging from $N = 1$ to $N = 2^{20} \approx 10^6$. The table shows how in each case the values for TRE and ORE compare. We can conclude that for practical purposes the observed values agree with the theoretical values.

5 Termination Detection

Until now we have not addressed termination. In the basic algorithm, nodes can be queried at any time, before we have taken into account the vectors from every node in the network. If we query it too soon, messages from distant nodes will not have yet contributed and the estimate will be a number smaller and unrelated to the network size.

As the algorithm collects vectors from all neighbours, at each new round the algorithm takes into account vectors from all nodes one hop further than in the previous round; i.e. the "visibility radius" increases at each round. In the worst case, for nodes in the periphery of the network, after D rounds, where D is the network diameter, all vectors will have contributed. The problem is that, in general, we will not know the network diameter in advance.

The intuition for the termination is as follows: at each round we collect information from some new nodes (all nodes that entered the expanded visibility radius); as each of these nodes contributes with K random numbers, the probability that no new minimum is obtained at any index in the vector (i.e. that "no news" has arrived) is small; moreover the probability that such "absence of news" occurs T times in a row is close to zero even for a small T (smaller T for larger K).

The termination of the algorithm is based precisely on the detection of T (for some configurable T) consecutive rounds where no component changed in the vector stored locally. When that happens the algorithm assumes that all nodes have contributed and the result can be reported. The algorithm including termination detection is shown in Algorithm 2.

To find out which values of T are appropriate, we have simulated runs of the algorithm, each time until x at each node converges to the global pointwise minimum. In each run we have kept, at each node, the maximum number of consecutive rounds with no update in x before the value converges; we have also kept the number of rounds needed for conver-

Algorithm 2 With termination detection

```
  const $K, T$
  var $n, x[1..K]$
  var $oldx[1..K], nonews, converged$
Upon: Init
  $nonews \leftarrow 0$
  $converged \leftarrow False$
  $n \leftarrow neighbours(self)$
  for all $i \in 1..K$ do $x[i] \leftarrow rExp(1)$
  Send $x$ to every $p \in n$
Upon: Receive $m_1..m_j$ from all $p \in n$
  $oldx \leftarrow x$
  for all $l \in 1..j$ do
    $x \leftarrow pointwisemin(x, m_l)$
  end for
  if $oldx \neq x$ then
    $nonews \leftarrow 0$
  else
    $nonews \leftarrow nonews + 1$
  end if
  if $nonews \geq T$ then
    $converged \leftarrow True$
  end if
  Send $x$ to every $p \in n$
Upon: Query & $converged = True$
  return $\hat{N}(x)$
```

gence.

We have simulated runs of the algorithm for several networks with different topologies and sizes, using different values for K. We have used three network topologies: “2d” – proximity based networks (e.g. sensor networks) where nodes are spread geographically and each node communicates with others at some distance; “rand” – where each node is linked with others at random; and “attach” – which, like the web, follows a “preferential attachment” where some nodes are much more popular than others. “2d” networks have a large diameter and “attach” a small diameter, with “rand” in between.

Table 4 shows, for different K (10, 100 and 1000), and for the different topologies, each with different N (100, 1000 and 10000 nodes): the average across all nodes (AR) and the maximum (MR) number of rounds for convergence, and the maximum number of consecutive rounds with no news (NN) that was observed, while running 10 times the algorithm for each combination of K, topology and N. The figures regard the version of the algorithm that stores numbers as a 5 bits exponent in base 2; using floating point numbers gave similar results.

The table shows that for “rand” and “attach” networks, the observed NN is quite small; for these networks it will suffice choosing: for $K = 1000$, T equal to 3 or 4; for $K = 10$ or $K = 100$, T equal to 4 or 5. These values are conservative towards ensuring that each node will have converged; in fact we have observed that, for these topologies, over 90% of the nodes get x updated every round until convergence, and over 99% of the nodes see at most one consecutive round with no news.

For “2d” networks, NN varies more considerably with network size and K. More specifically, we have observed a relatively linear dependence between NN and the network diameter. The diameter, for 2d networks grows typically as the square root of the number of nodes and can reach some considerable size. $K = 10$ is not appropriate for 2d networks, as it would be difficult to choose a value for T. Even if we could make a choice of a large enough T, the extra T rounds would be a considerable overhead (at least about 50%) over the average number of rounds necessary for the x vector to converge; this overhead decreases to about 20% for $K = 100$ and about 10% for $K = 1000$.

K = 10

Top.	N	AR	MR	NN
2d	100	8.2	14	4
2d	1000	28.3	47	15
2d	10000	82.0	132	46
attach	100	2.8	5	2
attach	1000	3.8	7	2
attach	10000	4.6	8	2
rand	100	4.7	8	3
rand	1000	6.3	11	3
rand	10000	8.1	14	4

K = 100

Top.	N	AR	MR	NN
2d	100	9.5	14	2
2d	1000	31.5	47	5
2d	10000	95.4	139	18
attach	100	3.2	5	1
attach	1000	4.3	8	2
attach	10000	5.1	9	2
rand	100	5.5	8	2
rand	1000	7.2	11	2
rand	10000	9.2	15	4

K = 1000

Top.	N	AR	MR	NN
2d	100	9.8	14	1
2d	1000	33.4	47	2
2d	10000	101.7	140	10
attach	100	3.3	5	1
attach	1000	4.8	8	1
attach	10000	5.6	9	2
rand	100	5.7	8	1
rand	1000	7.9	12	2
rand	10000	10.0	15	3

Table 4: Average and maximum rounds for convergence and maximum consecutive rounds with no news

It should be pointed out that very large "2d" networks are unrealistic [8], and that these would lead to large diameters (already around 140 hops in the 10000 node network used). Assuming that larger than 10000 nodes "2d" networks will not occur, we can choose $T = 19$ for $K = 100$ and $T = 11$ for $K = 1000$ (or $T = 6$ and $T = 3$ respectively for up to 1000 nodes networks).

6 Message Loss and Slow Links

A strong point in our estimation technique is that it is suitable to address scenarios where message loss can occur. Contrary to techniques such as [11], that cannot afford to loose messages, in ours the knowledge in each message is made obsolete by subsequent ones: if a message from A to B containing vector x is lost, a subsequent message will have content y, where $y \leq x$ (in pointwise order).

This means that our algorithm can be easily modified to deal with message loss. The algorithms presented send a message to all neighbours and wait from messages from all neighbours. This means that a single message loss will deadlock the entire system. Some simple modifications to deal with the problem are possible:

- A possibility is the use of a timeout. Normally the algorithm would wait for messages from all neighbours, but if more than some time elapsed, it would proceed using the messages received so far.

- Another possibility is to design the algorithm to cope with the failure of F messages, for small F like 1 or 2, and make it wait from messages from all neighbours minus F.

The second variation is interesting in another point: it would make the algorithm robust to slow links. Waiting from all minus e.g. 1 neighbour means that if the last message would take much more time to arrive it would not slow down the starting of the next round. The vector in these late messages could be accounted for (in the subsequent round) in computing x, so that we do not ignore a node whose messages are consistently the last one to arrive. The possible increase in the number of rounds would be balanced by faster rounds.

In these variations each message would be tagged with a round number, to distinguish messages from the current round from older rounds. A combination of these possibilities would be to wait for all neighbours until a timeout and then wait for all still missing minus F.

A careful analysis of these variants, including the impact of F on convergence detection and algorithm absolute running time due to faster rounds is beyond the scope of this paper.

7 Related Work

In [10] a network size estimation technique is constructed by establishing successor relations among nodes. Each node that enters the network needs to be set as successor to an existing node selected uniformly at random. Under these conditions it is possible to provide coarse estimations of the network size, ranging from $N/2$ to N^2.

Two network size estimation algorithms are presented on [16], in both cases the authors assume that all nodes have unique identifiers and estimates are calculated at an initiator node. The first algorithm *Hops Sampling Protocol* is a gossip based technique requiring a membership list chosen uniformly at random and the capability to establish connections between arbitrary nodes.

The second approach, *Interval Density*, has much lighter requirements and thus can be compared with our solution. In this case, nodes have randomized ids mapped into the interval $[0, 1]$. The initiator collects the node ids that fall in a given subregion of size I in the interval and estimates by multiplying the number of received ids by $1/I$. The weakness is that it is difficult to set an adequate I since N is not known and that the transmitted data is always a fraction of N. For both algorithms the achieved relative accuracy after optimizations was 5%.

The use of idempotent messages for duplicate insensitive aggregations in sensor networks was presented in [4, 3, 15]. These papers make use of a sketching technique developed by Flajolet and Martin in [6] and recently enhanced in [5]. The technique, referred to as FM sketches, was developed to estimate the number of distinct elements in a multiset.

Our approach, building on extreme value statistics, operates in the real domain and can estimate sums of positive reals. FM sketches, builds on the use of hash functions and bitmaps and is a discrete technique than can estimate sums of positive integers. It follows that FM sketches are less general.

Although intrinsically different the two techniques have important similarities. If K is the number of units dedicated to the estimation, both estimate with a relative standard error of roughly $O(1/\sqrt{K})$.

When considering the effect of binary encoding, we observe that in [3, 4] the authors use the non enhanced FM sketches and thus would only be able to encode in 5 bits a network size up to 2^5. For practical uses they would need at least 16 bits per unit. Considering the enhanced version of FM sketches in [5], one could expect in 5 bits to be able to count up to 2^{32} while we are limited to about 2^{23}. However it is not clear how this version would adequate to estimations of both small and large values of N since the technique was developed for large cardinalities.

A different trade-off in network size estimation can be found in the technique described in [11]. The approach starts by setting a value v to 0 in all nodes and to 1 in a given node. Then the protocol selects pairs of nodes and averages the values in the nodes. When all values converge each node has an estimate of N in $\hat{N} = 1/v$. Message state can be very small since one needs to encode a single real with high precision. Convergence requires a number of message exchange steps much larger than the network diameter and some atomicity and isolation concerns must be taken into account. The protocol is very slow, with this factor having a greater impact on network topologies with a large diameter. However, this averaging approach can be well suited for high precision estimates in small diameter networks.

8 Conclusions

We have introduced Extrema Propagation, a new approach to distributed aggregation, based on the use of the statistical theory of extreme values. The resulting unbiased estimators for exponential distributions lead to very simple algorithms and efficient implementations. Being able to estimate sums of positive reals, we are more expressive than previous approaches: our technique encompasses summing naturals and counting, constituting a new building block for the construction of aggregate functions.

The technique is fast: all nodes have correct estimates after, at most, a number of communication steps equal to the network diameter, and in this sense we operate at the theoretical minimum. The approach to termination detection makes the estimate available after a short additional number of communication steps.

Finally, Extrema Propagation possesses an assortment of interesting properties: it is fully distributed with no single point of failure and with result produced at every node, it does not require system-wide identifiers and it is suitable to tolerate message loss.

9 Acknowledgments

We would like to thank Luís Machado and Bruno Cecílio de Sousa for their helpful comments on the extreme value estimation.

References

[1] Ittai Abraham and Dahlia Malkhi. Probabilistic quorums for dynamic systems. In Faith E. Fich, editor, *DISC*, volume 2848 of *Lecture Notes in Computer Science*, pages 60–74. Springer, 2003.

[2] Carlos Baquero and Francisco Moura. Using structural characteristics for autonomous operation. *Operating Systems Review*, 33(4):90–96, 1999.

[3] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. Estimating aggregates on a peer-to-peer network. Technical Report TR-2003-24, Stanford University, 2003.

[4] Jeffrey Considine, Feifei Li, George Kollios, and John W. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, pages 449–460. IEEE Computer Society, 2004.

[5] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities (extended abstract). In Giuseppe Di Battista and Uri Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*, pages 605–617. Springer, 2003.

[6] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[7] E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958.

[8] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[9] R. V. Hogg and A. F. Craig. *Introduction to Mathematical Statistics*. Prentice-Hall, Upper Saddle River, New Jersey, 5th edition, 1995.

[10] Keren Horowitz and Dahlia Malkhi. Estimating network size from local information. *Inf. Process. Lett.*, 88(5):237–243, 2003.

[11] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[12] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491. IEEE Computer Society, 2003.

[13] Ji Li, Karen R. Sollins, and Dah-Yoh Lim. Implementing aggregation and broadcast over distributed hash tables. *Computer Communication Review*, 35(1):81–92, 2004.

[14] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[15] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor

networks. In John A. Stankovic, Anish Arora, and Ramesh Govindan, editors, *SenSys*, pages 250–262. ACM, 2004.

[16] D. Psaltoulis, D. Kostoulas, I. Gupta, K. Birman, and A. Demers. Practical algorithms for size estimation in large and dynamic groups. Technical report, University of Illinois, Urbana-Champaign, 2004.

[17] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[18] Robbert van Renesse. The importance of aggregation. In André Schiper, Alexander A. Shvartsman, Hakim Weatherspoon, and Ben Y. Zhao, editors, *Future Directions in Distributed Computing*, volume 2584 of *Lecture Notes in Computer Science*, pages 87–92. Springer, 2003.