Distributed Clustering for Robust Aggregation in Large Networks

Ittay Eyal Idit Keidar Raphael Rom
Department of Electrical Engineering, The Technion — Israel Institute of Technology

Abstract

We present a scalable protocol for robust data aggregation in a large, error-prone network. The protocol aggregates the multidimensional distribution of any number of data samples (sensor reads) and removes data errors using constant size synopses, by clustering *samples and detecting* outliers*. Initial simulations show that the protocol achieves robustness to both crashes and data errors.*

1 Introduction

In years to come, we can expect to see sensor networks with thousands of light-weight nodes monitoring conditions like seismic activity or temperature [1, 14]. In addition, large-scale networked services are now being increasingly deployed in computation clouds and Internet-based overlay networks. Such networks need constant monitoring in order to detect failures and other anomalous situations, e.g., poor load balance.

The sizes of these networks, together with processing and bandwidth limitations, prohibit a centralized solution in which the monitored data is accumulated at a single location. Instead, there is a need for more succinct data *aggregation*. When the monitored data is a scalar value, such as the average temperature, it can indeed be summarized succinctly. The standard solution is to form a spanning tree, where the leaves obtain samples (data reads) and the data is aggregated to the root [10, 12]. Each node maintains a *synopsis* of the data — in this example, the estimated mean and the number of samples in its subtree. The synopsis in each internal node is created by *merging* the synopses of its children. Unfortunately, this solution is extremely sensitive to topology changes and node failures; maintaining a spanning tree in a dynamic environment is costly, and moreover, topology changes may lead to problems such as double counting and data loss [8].

To overcome these problems, gossip solutions that calculate means and sums have been suggested [9, 11]. As with a spanning tree, each node holds a synopsis of samples. But unlike the spanning tree solution, nodes exchange information with random neighbors, and continuously improve the accuracy of their synopses. This approach is robust to crashes and topology changes.

Nevertheless, there is another significant source of errors that may arise in aggregation and is not handled by these solutions: sensors may produce incorrect samples due to hardware malfunctions, software bugs, or sensing errors (e.g., an animal sitting on a temperature sensor). Such errors may cause substantial changes in the aggregation result. For example, a single buggy temperature read of $1000^{\circ}\mathrm{C}$ may outweigh thousands of correct ones. In a system with a large number of sensors, the probability of such errors cannot be neglected.

A second limitation of most existing synopsis-based solutions is that they are restricted to scalar values such as means and sums. Yet certain problems, like detecting anomalies, require learning a more detailed picture of the data distribution. For example, a security seismic penetration detection system has to discover where there are irregular shakes, such as a moving truck, but ignore a shake common to all sensors like a mild earthquake. Another example is a DDoS attack on machines in a grid computer network like PlanetLab, where load probes may show that one third of the computers are working at full load and the rest are idle. Knowing the average load of 33% is useless. Instead, we would like to learn that there are two *clusters* of load values, with averages of 1% and 99%. Moreover, if the overloaded nodes are running the same (exploited) operating system, we would wish to learn this as well.

Recently, algorithms that estimate the *distribution* of samples have been proposed [6, 13]. However, these algorithms are limited to one-dimensional data, and thus cannot correlate high CPU load with other factors such as operating system version. Similarly, in sensor networks, the geographical distribution of the information is of essence, and single-dimensional data cannot convey the location of a fire or a penetration.

In this paper, we suggest an approach to overcome both of these limitations. We present a protocol that aggregates the multidimensional distribution of the samples by *clustering* them. This enables both the removal of data errors and the aggregation of elaborate data. The key to data error robustness is *outlier detection*, i.e., finding out which samples do not belong to the common data distribution. Outliers are not necessarily isolated; for example, a source of noise may

interfere with the readings of multiple sensors. We formally define the model and the problem in Section 2.

The challenge is to identify these outliers in a distributed manner, where each node has partial information. It seems that we are facing a catch-22 situation [7] — in order to identify an outlier, a node needs a synopsis of the good samples (excluding outliers), but in order to calculate this synopsis, it must know which samples are outliers and ignore them. No member of the system has enough information to resolve this.

Section 3 presents our new protocol, which uses constant size synopses to represent any number of samples (Some details are deferred to Appendix A). The key to overcoming the catch-22 is the observation that the more samples are available, the easier it becomes to recognize the good distribution and separate the outliers. At first, each node has only a few samples and is unable to identify outliers, but the synopsis also loses only a small amount of information. In later stages, when a node has enough information to identify outliers, it performs a more aggressive compression (clustering) of the bulk of the data, while keeping isolated outliers intact. Synopses are propagated using gossip, thus achieving both crash robustness and scalability. After the protocol has converged, each node has a synopsis that (inaccurately) describes the distribution of all the samples collected throughout the entire system. From this synopsis, nodes may either filter out the outliers, or analyze their data, depending on the application.

In Section 4 we present initial and promising simulation results of our protocol. We show its success in distributively identifying outliers, using very small synopses, as well as its robustness. We plan to conduct further simulations to understand the protocol's behavior in many different input scenarios and topologies, and to further analyze its properties; we discuss our conclusions and future directions in Section 5.

Related Work

Kempe et al. [9] introduce an approach for computing aggregates such as sums and means using gossip, and prove that it converges in logarithmic time. They further show how to compute quantiles by invoking multiple rounds of this protocol. However, they do not deal with data errors, outlier detection, or clustering. Our protocol generalizes their basic approach so as to compute non-scalar aggregates such as clusters and outliers, (which cannot be derived from quantiles), using a single round.

Nath et al. [12] present a generic framework for robust aggregation using synopses. Synopses are aggregated using some merge function, which is order- and duplicate-insensitive, and hence allows for unstructured diffusion patterns such as those that arise in gossip. Their merge function is based on [5], and is also used in [11]. However, these works only consider scalar aggregation and do not deal with data errors. We generalize this approach by exhibiting a merge function for aggregating clusters.

Haridasan and van Renesse [6] estimate one dimensional distributions in sensor networks by estimating histograms. Our protocol achieves competitive accuracy, as demonstrated in Section 4.2. Sacha et al. [13] provide more accurate results by using iterations to improve the histograms, achieving a better estimation. Our protocol uses only a single iteration to provide good accuracy. Unlike both of these works, we provide multidimensional distribution estimation and show that we can detect outliers and remove data errors.

Jain et al. [8] provide an approach to quantify errors in scalar aggregated data caused by topology changes and crash failures, but not data errors.

Branch et al. [2] identify outliers by their distances from their neighbors, in a data stream collected by a small number of nodes. This protocol detects isolated outliers, but neither detects outlier clusters nor aggregates the data of non-outlying samples. The protocol performs multiple rounds before converging, and was only evaluated with 53 nodes, hence its scalability is unclear. We intend to compare this protocol to ours, in large networks, for the special case of detecting isolated outliers.

Clustering has been extensively studied for centrally available data sets, e.g., databases. To improve performance, parallelization is sometimes used. Parallel clustering differs from distributed clustering in that all the data is available to all threads, and communication is cheap.

2 Problem and Model

A set $v_1, \ldots v_n$ of nodes is connected by a set of edges. Each node i obtains a multiset S_i of samples from $\mathbb{R}^d$ for some d. The multiset S of all the samples obtained is the union of two multisets — a multiset G of *good* samples, created by a distribution $f_G(x)$, and a multiset B of *bad* samples, created arbitrarily.

The nodes strive to compute some target function of the good distribution and of the bad samples, $h(f_G, B)$. Nodes have limited bandwidth and are unable to accumulate all of S. Instead, they diffuse synopses; a node holding a synopsis *syn* calculates *distillSyn(syn)*, which is an estimate of $h(f_G, B)$.

Since $h()$ may be any function, the protocol strives to create a synopsis that estimates both f_G and B as accurately as possible. The quality of the synopsis therefore consists of: (1) the accuracy of the estimated f_G, according to some distance metric; and (2) the accuracy of the estimated B. Since samples in B are, in general, indistinguishable from ones created by f_G, we replace B in (2) by *outliers*, defined as follows:

Definition 1 (Outliers) *Given a multiset of samples S, a distribution f_G, and some threshold f_{min}, the* outliers *are*

the samples in S with probability density smaller than f_{min}, according to f_G.

Note that it is possible, though by definition not probable, that samples in G, created by the good distribution, would be defined as outliers. It is also possible that samples from B would not be defined as outliers.

3 The Protocol

We present a general aggregation framework based on ideas from [12, 9] in Section 3.1. We instantiate it with a cluster aggregation protocol in Section 3.2, and describe a clustering based on *Gaussian mixtures* in Section 3.3.

3.1 Synopsis Aggregation Framework

To use our generic synopsis aggregation framework, one specifies four functions — *makeSyn, halfSyn, mergeSyn*, and *distillSyn*. Each *synopsis* encapsulates both some summary of the samples it describes and a mass.

Initially, a node i obtains samples S_i and calls *makeSyn(S_i)* to summarize them in a synopsis syn with mass $|S_i|$. The function *halfSyn(syn)* returns a synopsis with the same summary as *syn*, but with half the mass. (Though mass division may be generalized, for clarity of presentation, we restrict ourselves to halving).

Synopses are disseminated using gossip. Algorithm 1 describes a gossip step. In each step, a node divides its synopsis *syn* into two halves, keeping one and sending the other to a random neighbor. When receiving synopses, a node calls *mergeSyn* to merge them with its own, producing a new synopsis whose mass is the sum of the masses of the merged ones.

Algorithm 1 Gossip step

half $\leftarrow$ *halfSyn*(*syn*)
send *half* to a random neighbor
$\{$*neighborSyn*$_i\}$ $\leftarrow$ all synopses received
syn $\leftarrow$ *mergeSyn*(*half*, $\{$*neighborSyn*$_i\}$)

The synopses in this framework maintain a conservation of mass, i.e., the total mass in the system is invariant and equals the number of samples taken. Moreover, each sample's mass is conserved, divided among the nodes. At each node, the mass of each sample converges to $\frac{1}{|S|}$ of the total mass in the node's synopsis.

After any step, a node can produce an estimate of the function $h(f_G, B)$ by calling *distillSyn(syn)*.

3.2 Clusters Synopsis

We instantiate the above framework with synopses that represent the distribution of samples as clusters. A single sample $s \in \mathbb{R}^d$ is 1 unit of mass at s. A set of samples or parts thereof may be grouped into a cluster, whose mass is the sum of the masses grouped into it, and whose mean is at the center of mass.

A parameter k bounds the number of clusters in a synopsis; it must be larger than the number of isolated outliers that ought to be detected. If k is sufficiently larger than the number of samples each node obtains, then in early stages, all clusters are singletons. Once a node has learned about more than k samples, it can infer which are isolated enough to be considered outliers. These continue to be maintained as singletons, while other samples are clustered. Thus, a synopsis can be roughly divided into:

1. a set of singletons representing outliers; and
2. a set of clusters, which provide a compressed description of the rest of the samples.

The framework's functions are as follows:

makeSyn(S_i) creates $|S_i|$ clusters, each containing one sample, and if $|S_i|$ exceeds k, the result is merged using *mergeSyn* (below).

halfSyn$(\boldsymbol{syn})$ halves the masses of all the clusters in *syn*.

mergeSyn$(\boldsymbol{syn}_1, \cdots, \boldsymbol{syn}_m)$ creates a new set of up to k clusters from the clusters in $syn_1, \cdots, syn_m$. This step employs lossy compression.

The protocol diffuses partial cluster masses among the nodes. These masses originate from the samples. The protocol converges to a point where each sample has an equal portion of its mass at each node [9], and each node has an estimate of the distribution whose inaccuracy is the product of the lossy compression of *mergeSyn*.

3.3 Gaussian Mixture Aggregation

We now present a clustering where the clusters are Gaussians and the synopsis is a *Gaussian mixture*:

Definition 2 (Gaussian Mixture) *A* Gaussian Mixture (GM) *is a multivariate distribution represented as a weighted sum of Gaussians $\{G_i\}_{i=1}^k$, each described as a tuple $G_i = \langle m_i, \underline{\mu_i}, \underline{\underline{\sigma_i}} \rangle$ such that:*

- $m_i \in \mathbb{R}$ *is the mass of Gaussian i*
- $\underline{\mu_i} \in \mathbb{R}^d$ *is the mean of Gaussian i.*
- $\underline{\underline{\sigma_i}} \in \mathbb{R}^{d \times d}$ *is the covariance matrix of Gaussian i.*

Initially, *makeSyn* creates a singular Gaussian for each sample. This Gaussian has a weight of 1, its mean is the sample's value and its covariance matrix is $\varepsilon \cdot I_d$, where ε is very small and I_d is the unit matrix of dimension d. The function *halfSyn* halves the masses of all the clusters:

$$\textit{halfSyn}(\{\langle m_i, \mu_i, \sigma_i \rangle\}_{i=1}^k) = \{\langle \frac{1}{2} m_i, \mu_i, \sigma_i \rangle\}_{i=1}^k$$

The function *mergeSyn* takes a set of Gaussian mixtures and produces a k-GM, which is a Gaussian mixture with up to k clusters. Ideally, *mergeSyn* should be performed by means of *MAP* (Maximum A Posteriori) i.e., finding the k-GM for which the superposition of the original mixtures has the maximal likelihood. Since this problem is NP-hard, we approximate it by implementing the *Expectation Maximization* (EM) algorithm [3]. More details are given in Appendix A.

The function *distillSyn* estimates B as the outliers, i.e., low-weight clusters, and f_G as the GM of the remaining clusters. It can then calculate any function of the estimated (f_G, B).

4 Simulation Results

We present initial simulation results of our protocol. All simulations use a fully connected topology. Each node obtains 1 sample. The probability density threshold f_{min} is set to 5×10^{-5}. Like previous works [4, 6], we measure progress in rounds, where in each round each node performs one gossip step. Since gossip partners are chosen uniformly at random, the number of nodes does not significantly impact the accuracy of the results or the convergence time. Our simulations all use 1,000 nodes.

4.1 Robustness

Data robustness We first evaluate our protocol's ability to overcome data errors. We use 950 samples from the standard normal distribution i.e., with a mean $(0, 0)$ and a unit covariance matrix I. Additional 50 samples are outliers distributed normally with covariance matrix $0.1 \cdot I$ and mean $(0, \Delta)$, with Δ ranging between 0 and 25. The samples' distribution is illustrated in Figure 1a. For each value of Δ, the results are shown after the protocol has reached convergence. The target function $h(f_G, B)$ is *mean*(f_G). We use $k = 2$, so that each node has at most 2 clusters at any given time — hopefully one for good samples and one for outliers. It turns out that the mass of single samples is often divided between these clusters.

The results are shown in Figure 1b. The dotted line shows the average ratio of mass belonging to outliers yet incorrectly assigned to the good cluster. The other two lines show the error in calculating the mean, where error is the average over all nodes of the distance between the estimated mean and the true mean $(0, 0)$. The solid line shows the result of our algorithm, which removes outliers, while the dashed line shows the result of regular aggregation, which does not.

We see that when the outlier cluster is close to the main one, the number of misses is large — the proximity of the clusters makes their separation difficult. However, due to the small distance, this mistake hardly influences the estimation. As the outliers' mean moves further from the true mean, the identification of outliers becomes accurate and their influence is nullified.

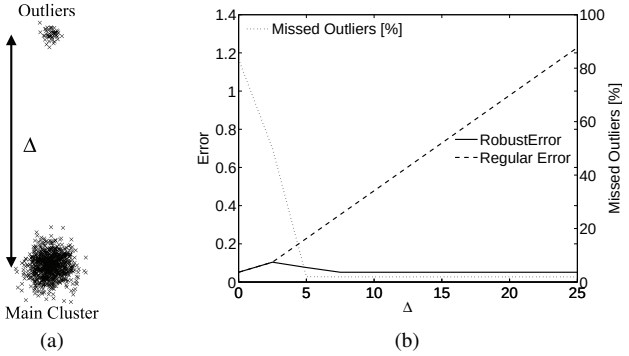


Figure 1: Effect of the separation of outliers on the calculation of the mean: A 1000 samples are distributed in two clusters (Figure 1a). As the outlier cluster moves away from the good cluster, the regular aggregation error grows linearly. However, once the distance is large enough, our protocol can remove the outliers, and results in an accurate estimation of the mean.

Note that even for large Δ's there is always a certain amount of outliers which are missed. These are samples from the good distribution, relatively close to the main cluster, yet with probability density lower than f_{min}. The protocol considers these to be good samples, though according to Definition 1 they are not. Additionally, around $\Delta = 5$ the miss rate is dropped to its minimum, yet the robust error does not. This is due to the fact that bad samples are located close enough to the good mean so that their probability density is higher than f_{min}. The protocol mistakes those to belong to f_G and allows them to influence the mean. That being said, for all Δ's the error remains small, confirming the conventional wisdom that "clustering is either easy or not interesting".

Crash robustness We next examine how crash failures impact the results obtained by our protocol. Figure 2 shows that the outlier removal mechanism is indifferent to crashes of nodes. The source data is similar to the one above, with $\Delta = 10$. After each round, each node crashes with probability 0.05. We show the average node estimation error of the mean in each round. As we have seen above, our protocol achieves a lower error then the regular one.

4.2 1D Distribution Aggregation

We begin the presentation of distribution aggregation by studying the case of one dimensional classical distributions.

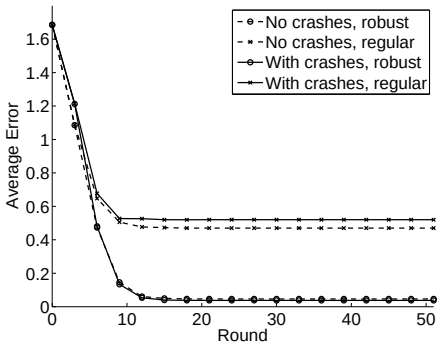


Figure 2: Effect of crashes on the accuracy of the mean.

We compare our results to those obtained in [6], for the data distributions used therein: uniform, exponential ($\lambda = 1.5$), Pareto ($k = 5, x_m = 1$) and bimodal (composed of two normal distributions with parameters $\mu_1 = 5, \sigma_1 = 1$ and $\mu_2 = 8, \sigma_2 = 0.5$). As they used 50 bin histograms, we use 50 clusters. We measure the accuracy of the estimation according to the Kolmogorov-Smirnov (K-S) test — the largest difference between the correct and the estimated cumulative distribution functions over all nodes. Table 1 compares the K-S test results our protocol achieves ("Gaussian Clusters" column) to the ones achieved in [6] ("Histogram" column). Both algorithms produce good test results in the area of 0.07. Figure 3 shows the estimated and true distributions our protocol aggregates for the uniform case.

Table 1: K-S Test for One Dimensional Distributions

Distribution	Histogram[6]	Gaussian Clusters
Uniform	0.064	0.029
Exponential	0.069	0.072
Pareto	0.067	0.095
Bimodal	0.077	0.057

4.3 Rich Data Aggregation

Preliminary simulations show promising results of the aggregation of multidimensional distributions. Figure 4 demonstrates one such simulation. Samples are created according to a distribution f_G and are aggregated using our protocol with $k = 7$. This distribution might describe temperature readings on a fence by the woods, whose right side is close to a fire outbreak. Each sample is comprised of the sensor's location x and the recorded temperature y.

Figure 4a depicts the original distribution containing 3 Gaussian clusters in the two dimensional space. The ellipses are equidensity contours of normal distributions. Figure 4b shows the samples generated and Figure 4c shows the

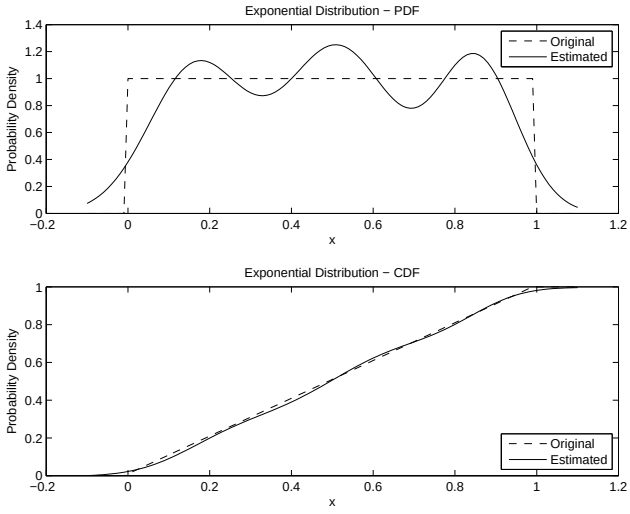
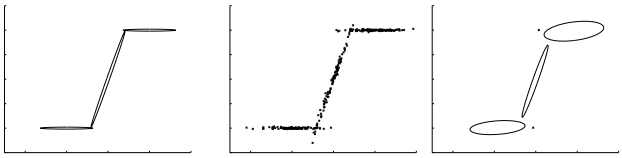


Figure 3: PDF and CDF of estimated distribution vs. original distribution.

estimated f_G after aggregation. Five clusters were created, of which two are of isolated samples.

We intend to perform further simulations and quantify the accuracy of the result in different scenarios.



(a) Distribution f_G (b) Samples (c) Result

Figure 4: Aggregation of rich data using $k = 7$. 'Result' is the synopsis at an arbitrary node after convergence.

5 Conclusion and Future Directions

We presented a new scalable and crash robust protocol for multidimensional distribution aggregation in large networks. The protocol removes outliers, achieving robustness to data errors. By using clusters, the complete distribution of samples can be aggregated (inaccurately), with very low bandwidth.

While our initial results are promising, many issues are yet to be explored. Most notably, we would like to investigate the convergence properties of our algorithm: What output it converges to and how fast, for various distributions and k's. In particular, we would like to experiment with topologies that typically arise in sensor networks, e.g., unit-disk graphs, and with real life data traces of environmental samples such as precipitation or temperature. In these contexts, we would also like to compare our protocol to others, e.g., [2].

Beyond additional empirical tests, it would be valuable to formally prove that the protocol always eventually stabilizes, and moreover, to prove an upper bound on the resulting error and the rate in which this error diminishes.

Finally, our protocol currently performs one-shot aggregation, and does not adapt to sensor readings that change on-the-fly, i.e., data streams. It would be interesting to extend the protocol to work with changing inputs.

A GM Synopsis Merger

We define the operation *mergeSyn* that merges a set of Gaussians mixtures producing a k-GM, a Gaussian mixture with up to k clusters.

We first define the notion of a Gaussian merger.

Definition 3 (Gaussian Merger) *Consider two Gaussians defined by their masses, means and covariance matrices $G_a = \langle m_a, \mu_a, \sigma_a \rangle$ and $G_b = \langle m_b, \mu_b, \sigma_b \rangle$. Their* merger*, $G_m = \langle m_m, \mu_m, \sigma_m \rangle$ denoted $G_a \circ G_b$ is [15]:*

$$\begin{aligned} m_m =& m_a + m_b \\ \mu_m =& \frac{m_a}{w_m}\mu_a + \frac{m_b}{m_m}\mu_b \\ V_m =& \frac{m_a}{m_m}V_a + \frac{m_b}{m_m}V_b + \\ &+ \frac{m_a}{m_m}\frac{m_b}{m_m} \cdot (\mu_a - \mu_b) \cdot (\mu_a - \mu_b)^T \end{aligned}$$

It is possible to show that the Gaussian merger operation is both commutative and associative, so we define the merge operation of a multiset of Gaussian clusters as follows:

$$\text{merge}(\{G_i\}_{i=1}^n) = G_1 \circ G_2 \circ \cdots \circ G_n$$

Our goal is to cluster a Gaussian mixture GM_{old}, which is a union of *GM*s from a set of synopses, to a k-GM.

We use the following notation:

- GM_{old}: The original Gaussian mixture, a set of weighted Gaussians.
- GM_{new}: The new set of up to k Gaussians defined by their parameters.
- V: The d dimensional space in which the distributions are defined.
- $f_X(v)$: The probability density at point v of distribution X. If X is a mass distribution such as a Gaussian mixture, it is normalized s.t. it constitutes a PDF.

We define likelihood, as used for the *MAP* estimation:

Definition 4 (likelihood) *The* likelihood *that the samples concisely described by GM_{old} are the result of the probability distribution described by GM_{new} is:*

$$L = \sum_{c \in GM_{new}} \sum_{g \in GM_{old}} \left(\int_{v \in V} m_c f_c(v) \cdot m_g f_g(v) dv \right)$$

The $mergeSyn$ operation employs the *Expectation Maximization* algorithm [3] to approximate *MAP* — group and merge the given clusters to a k-GM for which the original clustering has the maximal likelihood.

References

[1] G. Asada, M. Dong, T. Lin, F. Newberg, G. Pottie, W. Kaiser, and H. Marcy. Wireless integrated network sensors: Low power systems on a chip. In *ESSCIRC*, pages 9–16, sep 1998.

[2] J. W. Branch, B. K. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *ICDCS*, 2006.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Stat. Soc.*, 39(1):1–38, 1977.

[4] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. In *DSN*, pages 443–452, 2001.

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

[6] M. Haridasan and R. van Renesse. Gossip-based distribution estimation in peer-to-peer networks. In *International Workshop on Peer-to-Peer Systems (IPTPS 08)*, February 2008.

[7] J. Heller. *Catch-22*. Simon & Schuster, 1961.

[8] N. Jain, P. Mahajan, D. Kit, P. Yalagandula, M. Dahlin, and Y. Zhang. Network imprecision: A new consistency metric for scalable monitoring. In *OSDI*, pages 87–102, 2008.

[9] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491, 2003.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[11] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *PODC*, 2006.

[12] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys*, pages 250–262, 2004.

[13] J. Sacha, J. Napper, C. Stratan, and G. Pierre. Reliable distribution estimation in decentralised environments. *Submitted for Publication*, 2009.

[14] B. Warneke, M. Last, B. Liebowitz, and K. Pister. Smart dust: communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, Jan 2001.

[15] W. Xu, J. Duchateau, K. Demuynck, and I. Dologlou. A new approach to merging gaussian densities in large vocabulary continuous speech recognition. In *IEEE Benelux Signal Processing Symposium*, 1998.