Robust Aggregation in Sensor Networks

George Kollios, John Byers, Jeffrey Considine, Marios Hadjieleftheriou, and Feifei Li
Computer Science Dept., Boston University
{gkollios, byers, jconsidi, marioh, lifeifei}@cs.bu.edu

Abstract

In the emerging area of sensor-based systems, a significant challenge is to develop scalable, fault-tolerant methods to extract useful information from the data the sensors collect. An approach to this data management problem is the use of sensor "database" systems, which allow users to perform aggregation queries on the readings of a sensor network. Due to power and range constraints, centralized approaches are generally impractical, so most systems use in-network aggregation to reduce network traffic. However, these aggregation strategies become bandwidth-intensive when combined with the fault-tolerant, multi-path routing methods often used in these environments. In order to avoid this expense, we investigate the use of approximate *in-network aggregation using small sketches and we survey robust and scalable methods for computing duplicate-sensitive aggregates.*

1 Introduction

As computation-enabled devices shrink in scale and proliferate in quantity, a relatively recent research direction has emerged to contemplate future applications of these devices and services to support them. A canonical example of such a device is a *sensor mote*, a device with measurement, communication, and computation capabilities, powered by a small battery [21]. Individually, these motes have limited capabilities, but when a large number of them are networked together into a *sensor network*, they become much more capable. Indeed, large-scale sensor networks are now being applied experimentally in a wide variety of areas — some sample applications include environmental monitoring, surveillance, and traffic monitoring.

In a typical sensor network, each sensor produces a stream of sensory observations across one or more sensing modalities. But for many applications and sensing modalities, such as reporting temperature readings, it is unnecessary for each sensor to report its entire data stream in full fidelity. Moreover, in a resource-constrained sensor network environment, each message transmission is a significant, energy-expending operation. For this reason, and because individual readings may be noisy or unavailable, it is natural to use data aggregation to summarize information collected by sensors. As a reflection of this, a database approach to managing data collected on sensor networks has been advocated [24, 29], with particular attention paid to efficient query processing for aggregation queries [24, 29, 31].

In the TAG system [24], users connect to the sensor network using a workstation or base station directly connected to a sensor designated as the sink. Aggregate queries over the sensor data are formulated using a simple SQL-like language, then distributed across the network. Aggregate results are sent back to the workstation over a spanning tree, with each sensor combining its own data with results received from its children. If there are no failures, this in-network aggregation technique is both effective and energy-efficient for distributive

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

and algebraic aggregates [19] such as MIN, MAX, COUNT and AVG. However, as shown in [7], this technique is much less effective in sensor network scenarios with moderate node and link failure rates. Node failure is inevitable when inexpensive, faulty components are placed in a variety of uncontrolled or even hostile environments. Similarly, link failures and packet losses are common across wireless channels because of environmental interference, packet collisions, and low signal-to-noise ratios [31]. For example, [31] reports on experiments in which more than 10% of the links suffered average loss rate greater than 50%.

When a spanning tree approach is used for aggregate queries, as in TAG, a single failure results in an entire subtree of values being lost. If this failure is close to the sink, the change in the resulting aggregate can be significant. Retransmission-based approaches are expensive in this environment, so solutions based upon multi-path routing were proposed in [24]. For aggregates such as MIN and MAX which are monotonic and exemplary, this provides a fault-tolerant solution. But for duplicate-sensitive aggregates such as COUNT or AVG that give incorrect results when the same value is counted multiple times, existing methods are not satisfactory.

In this paper, we survey robust and scalable methods for computing duplicate-sensitive aggregates across faulty sensor networks. Guaranteeing exact solutions in the face of losses is generally impractical, so we focus on approximate methods that are robust to both link and node failures, but are inexact. For example, we describe the application of well-known sketches to handle SUM and COUNT aggregates [14, 4] to this problem The methods we present combine duplicate insensitive sketches with multi-path routing techniques to produce accurate approximations with low communication and computation overhead.

2 Related Work

In-network Aggregate Query Processing: A simple approach to evaluate an aggregation query is to reliably route all sensed values to the base station and compute the aggregate there. Although this approach is simple, the number of messages and the power consumption can be large. A better approach is to leverage the computational power of the sensor devices and compute aggregates in-network. Aggregates that can be computed in-network include all decomposable functions [24]. Using decomposable functions, the value of the aggregate function can be computed for disjoint subsets, and these values can be used to compute the aggregate of the whole using an appropriate merging function. Our discussion is based on the Tiny Aggregation (TAG) framework used in TinyDB [24]. However, similar approaches are used to compute aggregates in other systems [29, 31, 22].

In TAG, the in-network query evaluation has two phases, the *distribution* phase and the *collection* phase. During the distribution phase, the query is flooded in the network and the nodes are organized into an *aggregation tree*. The base station broadcasting the query is the *root* of the tree. The query message has a counter that is incremented with each retransmission and counts the hop distance from the root. In this way, each node is assigned to a specific level equal to the node's hop distance from the root. Also, each sensor chooses one of its neighbors with a smaller hop distance from the root to be its parent in the aggregation tree.

During the collection phase, each leaf node produces a single tuple and forwards this tuple to its parent. The non-leaf nodes receive the tuples of their children and combine these values. Then, they submit the new partial results to their own parents. This process runs continuously and after h steps, where h is the height of the aggregation tree, the total result will arrive at the root. In order to conserve energy, sensor nodes sleep as much as possible during each step where the processor and radio are idle. When a timer expires or an external event occurs, the device wakes up and starts the processing and communication phases. At this point, it receives the messages from its children and then submits the new value(s) to its parent. After that, if no more processing is needed for that step, it enters again into the sleeping mode [25].

Best-Effort Routing in Sensor Networks: Recent years have seen significant work on best-effort routing in sensor and other wireless networks. Due to high loss rates and power constraints, a common approach is to use dispersity multi-path routing, where more than one copy of a packet is sent to the destination over different paths. For example, directed diffusion [22] uses a flood to discover short paths which sensors would use to send back responses. Various positive and negative reinforcement mechanisms are used to improve path quality. Braided diffusion [15] builds on directed diffusion to use a set of intertwined paths for increased resilience. A slightly different approach is used by GRAB [30], where paths are not explicitly chosen in advance, but the width of the upstream broadcast is controlled. The techniques we describe are meant to complement and leverage any of these routing techniques [7]. We note that combining these methods with duplicate-insensitive in-network aggregation will allow some of the overhead of these techniques to be amortized and shared amongst data items from many different sensors.

Counting Sketches: A counting sketch for the purpose of quickly estimating the number of distinct items on a stream (the COUNT aggregate) in one pass while using only a small amount of space, was introduced by Flajolet and Martin (FM) in [14]. Since then, there has been much work developing and generalizing counting sketches (e.g., [1, 2, 8, 13, 16, 18, 5, 9]). The original FM sketches are particularly well-suited to sensor network applications, since they are very concise and accurate in practice. We describe them in more detail in Section 3, and show how to extend these sketches for computing SUM aggregates.

The Count-Min sketch, a counting sketch for computing the frequency of elements on a stream (per element or for ranges of elements), was introduced by Cormode and Muthukrishnan [9]. We show how the Count-Min sketch can be made duplicate-insensitive for exploiting multi-path routing in sensor networks. Even though other frequency counting techniques have been proposed in the past [5, 26, 17, 12, 11], the Count-Min sketch is robust, small in size, and provides error guarantees. A detailed analysis of the Count-Min sketch appears in Section 3.

3 Sketch Theory

One of the core ideas behind our work is that duplicate-insensitive sketches will allow us to leverage the redundancy typically associated with multi-path routing. We now present some of the theory behind such sketches and extend it to handle more interesting aggregates. First, we present details of the FM sketches of [14] along with necessary parts of the theory behind them. Then, we generalize these sketches to handle summations, and show that they have the same accuracy as FM sketches. Finally, we present the Count-Min sketch of [9] and show how it can be combined with FM sketches to produce a duplicate-insensitive frequency counting sketch that can provide frequency esimates for both a single element and ranges of elements on a stream.

3.1 FM Sketches

Given a multi-set of items $M = \{x_1, x_2, x_3, \dots\}$, the *distinct counting* problem is to compute $n \equiv |\text{distinct}(M)|$. The FM sketch of M, denoted $S(M)$, is a bitmap of length k. The entries of $S(M)$, denoted $S(M)[0, \dots, k-1]$, are initialized to zero and are set to one using a "random" binary hash function h applied to the elements of M. Given $x \in M$ and an integer i, then $h(x, i) = 1$ with probability 0.5 and $h(x, i) = 0$ otherwise (with the same probability). Formally,

$$S(M)[i] \equiv 1 \text{ iff } \exists x \in M \text{ s.t. } \min\{j \mid h(x, j) = 1\} = i.$$

By this definition, each item x is capable of setting a single bit in $S(M)$ to one – the minimum i for which $h(x, i) = 1$. This gives a simple serial implementation which is very fast in practice and requires two invocations of h per item on average. It has been shown that a single element can be inserted into an FM sketch in $O(1)$ expected time. We now describe some interesting properties of FM sketches observed in [14].

Property 1: The FM sketch of the union of two multi-sets is the bit-wise OR of their FM sketches. That is, $S(M_1 \cup M_2)[i] = (S(M_1)[i] \vee S(M_2)[i])$.

Property 2: $S(M)$ is determined only by the distinct items of M. Duplication and ordering do not affect $S(M)$.

Property 1 allows each sensor to compute a sketch of locally held items and send the small sketch for aggregation elsewhere. Since aggregation via union operations is cheap, it may be performed in the network without significant computational burden. Property 2 allows the use of multi-path routing of the sketches for robustness without affecting the accuracy of the estimates. We expand upon these ideas in Section 4.

The next lemma provides key insight into the behavior of FM sketches and will be the basis of efficient implementations of summation sketches later.

Lemma 3: For $i < \log_2 n - 2\log_2\log_2 n$, $S(M)[i] = 1$ with probability $1 - O(ne^{-\log_2^2 n})$. For $i \geq \frac{3}{2}\log_2 n + \delta$, with $\delta \geq 0$, $S(M)[i] = 0$ with probability $1 - O\left(\frac{2^{-\delta}}{\sqrt{n}}\right)$.

The lemma implies that given an FM sketch of n distinct items, one expects an initial prefix of all ones and a suffix of all zeros, while only the setting of the bits around $S(M)[\log_2 n]$ exhibit much variation. This gives a bound on the number of bits k required for $S(M)$ in general: $k = \frac{3}{2}\log_2 n$ bits suffice to represent $S(M)$ with high probability. It also suggests that just considering the length of the prefix of all ones in this sketch can produce an estimate of n. Formally, let $R_n \equiv \min\{i \mid S(M)[i] = 0\}$ when $S(M)$ is an FM sketch of n distinct items. That is, R_n is a random variable marking the location of the first zero in $S(M)$. In [14], the following relationship between R_n and n is proven: $\mathbf{E}[R_n] = \log_2(\varphi n) \pm 10^{-5} + o(1)$. Thus, after ignoring small terms, R_n can be used as an unbiased estimator of $\log_2 \varphi n$. The authors also prove that the variance of $R_n$is slightly more than one, which is a concern, as it implies that estimates of n will often be off by a factor of two or more in either direction. However, standard methods for reducing the variance exist, including those discussed in detail in [7]. Furthermore, using results from [16], FM sketches can give an (ϵ, δ)-approximation method for estimating the distinct items of a multi-set using $O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta})\log(n))$ space.

3.2 Summation FM Sketches

In order to make the application of FM sketches practical for in-network query processing in sensor networks, approximate counting sketches can be generalized to handle summations. Let M be a multi-set of the form $\{x_1, x_2, x_3, \ldots\}$ where $x_i = \langle k_i, c_i \rangle$ and c_i is a non-negative integer. The *distinct summation* problem is to calculate: $n \equiv \sum_{\text{distinct}(\langle k_i, c_i \rangle \in M)} c_i$.

For small values of c_i, it is practical to just count c_i different items based upon k_i and c_i, e.g., by considering the stream of *sub-items*: $\langle k_i, c_i, 1 \rangle, \ldots, \langle k_i, c_i, c_i \rangle$. Since this is merely c_i invocations of the counting insertion routine, the analysis for FM sketches applies, and the running time of an insertion is $O(c_i)$ expected time. But for large values of c_i, this approach is impractical, leading us to consider more scalable alternatives. The basic idea is to set the bits in the summation *as if* we had performed c_i successive insertions into an FM sketch, but without actually performing them. Such an emulation of c_i insertions can be done by sampling uniformly at random (repeatably for a given $\langle k_i, c_i \rangle$) from the distribution of sketches produced when c_i keys are chosen uniformly at random and inserted into an empty sketch.

A key observation behind our faster sampling procedure is that most of the simulated insertions attempt to set the same low-order bits, and repeated attempts at setting those bits have no further effect. We observe that if a prefix of the first δ bits in the sketch is already set to 1, then the probability of an inserted sub-item modifying a bit (i.e., setting a zero bit to one) is at most $2^{-\delta}$. By sampling the Bernoulli distribution with the correct value of δ corresponding to the current state of the sketch, we can quickly emulate whether the insertion of a sub-item will change the sketch. If not, we do nothing else, otherwise we determine (again in O(1) expected time), precisely which bit is set. Also, by exploiting the memorylessness of the sampling procedure, we can quickly simulate "skipping ahead" to identify sub-items which actually change the sketch. With this approach, and using fast lookup tables to implement Walker's alias method for sampling an arbitrary probability density function in O(1) time and small space [28], insertion can be achieved in $O(\log c_i)$ expected time. Full details appear in [6].

3.3 Duplicate-Insensitive Count-Min Sketches

The CM sketch is a data structure that can be used to to estimate the frequency of individual elements or ranges of elements on a stream [9]. Let a multi-set $S = \{s_1, \ldots, s_N\}$ denote the stream. The CM structure consists of a $k \times m$ matrix CM of counters, and a set of k hash functions $h_i(\cdot) : \{1 \cdots |\mathrm{distinct}(S)|\} \rightarrow \{1 \cdots m\}$. An insertion to the sketch evaluates all $h_i(\cdot)$ on every $s \in S$ and increases by one the counters with indices $CM[i, h_i(s)]$. In total, exactly k counters are associated with every element. If the hash functions are independent, then with high probability no two elements will hash to exactly the same set of counters. After the sketch is constructed, the estimated frequency of element s is given by $\widehat{f}(s) = \min_i (CM[i, h_i(s)]), 1 \leq i \leq k$. Essentially the counter with the smallest frequency estimate is the one that is the least affected by hash function collisions from other elements.

It can be easily shown that the CM sketch of the union of two multi-sets is the simple matrix addition of their individual CM sketches. Ideally, we would like the CM sketch to be duplicate-insensitive in terms of the union operation. Note that double-counting of CM sketches in a combined sketch, can yield unbounded errors in the estimated frequencies, subject to the total number of duplicate additions that occurred. Duplicate-insensitivity can be achieved by replacing every counter in the CM matrix with one Summation FM sketch, and taking advantage of the union property of FM sketches. Essentially, the Summation FM sketches are used to estimate the original magnitude of each counter. For this new construction, an element frequency can be estimated, in the worst case, as the minimum estimate of any of its associated FM sketches. More details can be found in [20].

4 Approximate Estimation of Duplicate-Sensitive Aggregates

We now discuss how to use duplicate-insensitive sketches to build a robust, loss-resilient framework for aggregation. The framework leverages two main observations. First, the wireless communication of sensor networks gives the ability to broadcast a single message to multiple neighbors simultaneously. Second, the duplicate-insensitive sketches discussed in Section 3 allow a sensor to combine all of its received sketches into a single message to be sent. Given proper synchronization, this will allow us to robustly aggregate data with each sensor sending just one broadcast.

We adapt the basic communication model of TAG [24] for continuous queries (one-shot queries simply terminate earlier). Given a new continuous query, the computation proceeds in two phases. In the first phase, the query is distributed across the sensor network, often using some form of flooding. During this phase, each node also computes its level (i.e., its hop distance from the root), and notes the level values of its immediate neighbors. The second phase is divided into a series of *epochs* specified by the query. The specified aggregate will be computed once for each epoch.

At the beginning of each epoch, each node constructs a sketch of its local values for the aggregate. The epoch is then sub-divided into a series of rounds, one for each level, starting with the highest level (farthest from the root). In each round, the nodes at the corresponding level broadcast their sketches, and the nodes at the next level receive these sketches and combine them with their sketches in progress. In the last round, the root receives the sketches of its neighbors, and combines them to produce the final aggregate. Duplicate-insensitivity of the sketching methods prevents double-counting from occurring. A similar communication model for computing aggregates, the rings overlay, was proposed by Nath et al. [27].

The tight synchronization described so far is not actually necessary. Our methods can also be applied using gossip-style communication - the main advantage of synchronization and rounds is that better scheduling is possible and power consumption can be reduced. However, if a node receives no acknowledgments of its broadcast, it may be reasonable in practice to retransmit. More generally, loosening the synchronization increases the robustness of the final aggregate as paths taking more hops are used to route around failures. (Similar ideas appear in the WILDFIRE protocol proposed by Bawa et al. [3].) This increased robustness comes at the cost of power consumption, since nodes broadcast and receive more often (due to values arriving later than expected) and increased time (and variability) to compute the final aggregate. As mentioned earlier, this general principle allows us to make use of any best-effort routing protocol (e.g., [22, 15]), with the main performance metric of interest being the delivery ratio. Other approaches are based on detecting frequent message losses and changes in network conditions, and adapting the topology hoping to reduce the loss rate [27]. With this protocol a node is allowed to jump to different levels of the tree, based on various heuristics according to the number of times that nodes from higher level have successfully included its sketch in the last few epochs, and the number of transmissions that the node can overhear from neighboring levels.

Alternatively, it may be possible to use CM sketches to approximate a consecutive sequence of values that each sensor generates, and then perform the computation only at the end of the sequence. In that case, each sensor computes a CM sketch of its last T values and then forwards this sketch to the upper levels. The advantage of this approach is the reduction in the number of messages (by a factor of T). However, at the same time the error in the approximation will increase. Another interesting direction is to combine sketch-based with tree-based methods in a *hybrid* approach. Furthermore, it will be interesting to combine robust sketch-based methods with model-based approaches proposed recently [10, 23].

5 Conclusions

We have presented new methods for approximately computing duplicate-sensitive aggregates across distributed datasets. Our immediate motivation comes from sensor networks, where energy consumption is a primary concern, faults occur frequently, and exact answers are not required or expected. An elegant building block which enables our techniques are the duplicate-insensitive sketches of Flajolet and Martin, which give us considerable freedom in our choices of how best to route data and where to compute partial aggregates. In particular, use of this duplicate-insensitive data structure allowed us to make use of dispersity routing methods to provide fault tolerance that would introduce inappropriate errors otherwise.

References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[2] Z. Bar-Yossef, T.S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. *Proc. of RANDOM*, 2002.

[3] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The Price of Validity in Dynamic Networks. *Proc. of ACM SIGMOD*, pp. 515–526, 2004.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. of the ACM (CACM)*, 13(7):422–426, 1970.

[5] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[6] J. Considine, Schedule-oblivious data management. Ph.D. Thesis, Boston University, December 2004.

[7] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. *Proc. of the International Conference on Data Engineering (ICDE)*, March 2004.

[8] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *Proc. of the VLDB*, 2002.

[9] G. Cormode and S. Muthukrishnan. Improved data stream summaries: The count-min sketch and its applications. *Journal of Algorithms*, 2004.

[10] A. Deshpande, C. Guestrin, S. Madden, J.M. Hellerstein, and W. Hong. Model-Driven Data Acquisition in Sensor Networks. *Proc. of the VLDB*, pp. 588–599, 2004.

[11] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313, 2003.

[12] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *ACM SIGCOMM Computer Communication Review*, 28(4):254–265, 1998.

[13] P. Flajolet. On adaptive sampling. *COMPUTG: Computing*, 43, 1990.

[14] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31, 1985.

[15] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *ACM Mobile Computing and Communications Review*, 5(4), 2001.

[16] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. *Proc. of ACM SIGMOD*, 2003.

[17] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. *Proc. of ACM SIGMOD*, pp. 331–342, 1998.

[18] P.B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. *Proc. of ACM SPAA*, pp. 281–291, 2001.

[19] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[20] M. Hadjieleftheriou, J.W. Byers, and G. Kollios. Robust Sketching and Aggregation of Distributed Data Streams. *BU Technical Report*, March 2005.

[21] M. Horton, D. Culler, K. Pister, J. Hill, R. Szewczyk, and A. Woo. Mica, the commercialization of microsensor motes. *Sensors*, 19(4):40–48, April 2002.

[22] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *Proc. of MobiCOM*, 2000.

[23] Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. *Proc. of IEEE ICDE*, 2005.

[24] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. *Proc. of OSDI*, 2002.

[25] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. *Proc. of ACM SIGMOD*, 2003.

[26] G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. *Proc. of VLDB*, pp. 346–357, 2002.

[27] S. Nath, P.B. Gibbons, S. Seshan, and Z. Anderson. Synopsis Diffusion for Robust Aggregation in Sensor Networks. *Proc. of ACM SenSys*, pp. 250–262, 2004.

[28] A.J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):253–256, 1977.

[29] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record 31(3)*, 2002.

[30] F. Ye, G. Zhong, S. Lu, and L. Zhang. GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks. *ACM Wireless Networks (WINET)*, 11(2), 2005.

[31] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. *Proc. of SNPA*, 2003.