Synopsis Diffusion for Robust Aggregation in Sensor Networks

SUMAN NATH
Microsoft Research
PHILLIP B. GIBBONS
Intel Research Pittsburgh
SRINIVASAN SESHAN
Carnegie Mellon University
ZACHARY ANDERSON
University of California, Berkeley

Previous approaches for computing duplicate-sensitive aggregates in wireless sensor networks have used a tree topology, in order to conserve energy and to avoid double-counting sensor readings. However, a tree topology is not robust against node and communication failures, which are common in sensor networks. In this paper, we present *synopsis diffusion*, a general framework for achieving significantly more accurate and reliable answers by combining energy-efficient multi-path routing schemes with techniques that avoid double-counting. Synopsis diffusion avoids double-counting through the use of *order- and duplicate-insensitive (ODI) synopses* that compactly summarize intermediate results during in-network aggregation. We provide a surprisingly simple test that makes it easy to check the correctness of an ODI synopsis. We show that the properties of ODI synopses and synopsis diffusion create *implicit* acknowledgments of packet delivery. Such acknowledgments enable energy-efficient adaptation of message routes to dynamic message loss conditions, even in the presence of asymmetric links. Finally, we illustrate using extensive simulations the significant robustness, accuracy, and energy-efficiency improvements of synopsis diffusion over previous approaches.

Categories and Subject Descriptors: C.2.4 [**Computer Communication Networks**]: Distributed Systems; C.3 [**Special-Purpose and Application-Based Systems**]: Embedded Systems

General Terms: Algorithms, Performance, Reliability, Theory

Additional Key Words and Phrases: Sensor Networks, Synopsis Diffusion, Query Processing

A preliminary version of this paper appeared in the *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, November 2004. Work done while the first author was a PhD student at CMU and an intern at Intel Research Pittsburgh, and the fourth author was an undergraduate at CMU. Authors' addresses: Suman Nath, Microsoft Research, One Microsoft Way, Redmond, WA 98052, sumann@microsoft.com; Phillip B. Gibbons, Intel Research Pittsburgh, 4720 Forbes Avenue, Suite 410, Pittsburgh, PA 15213, phillip.b.gibbons@intel.com; Srinivasan Seshan, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, srini@cs.cmu.edu; Zachary Anderson, Computer Science Division, University of California, Soda Hall, Berkeley, CA 94720, zra@cs.berkeley.edu.
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 2007 ACM 1529-3785/2007/0700-0001 $5.00

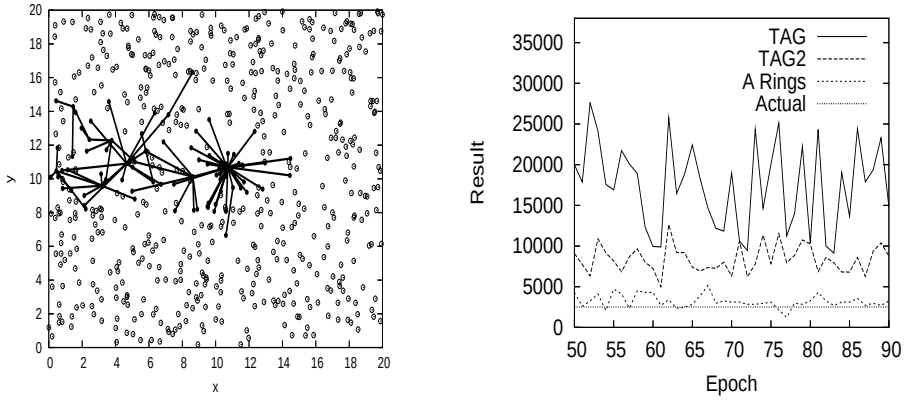
1. INTRODUCTION

In a large wireless sensor network, aggregation queries often assume greater importance than individual sensor readings. Previous studies [Madden et al. 2002; Zhao et al. 2003] have shown that computing aggregates *in-network*, *i.e.*, combining partial results at the intermediate nodes during message routing, significantly reduces the amount of communication and hence the energy consumed. A popular approach used by sensor database systems [Madden et al. 2003; Yao and Gehrke 2003] is to construct a spanning tree in the network, rooted at the querying node, and then perform in-network aggregation along the tree. Partial results propagate level-by-level up the tree in distinct epochs, with each node listening for messages from all its children before sending a new partial result to its parent. The tree topology in this aggregation scheme ensures that (i) each node sends only one message in computing an aggregate result and (ii) the reading from each sensor is accounted for only once in this result.

However, aggregating along a tree is very susceptible to node and transmission failures, which are common in wireless sensor networks [Madden et al. 2002; Zhao and Govindan 2003; Zhao et al. 2003]. This is because there is only a single path in the tree from a sensor reading to the querying node. Moreover, messages are typically sent using energy-efficient but unreliable communication, in order to conserve energy. Thus, each node or transmission failure loses an entire subtree of readings. As a result, a large fraction of the readings are typically unaccounted for in a tree-based scheme, causing significant errors in query answers [Considine et al. 2004; Madden et al. 2002; Zhao et al. 2003]. Figure 1 shows a typical example when using a tree-based scheme (TAG [Madden et al. 2002]), for a query computing the average of all the sensor readings at each epoch. In a typical epoch (Figure 1(a)), only 89 of the 600 readings reach the querying node, and over a range of epochs, the computed average exceeds the actual average by a factor of 4–11 (see the TAG curve in Figure 1(b)).

1.1 Existing Robust Aggregation Techniques

Two classes of techniques have been proposed to make the in-network aggregation process more robust. The first class uses reliable communication protocols [Wan et al. 2004; Stann and Heidemann 2003]. These approaches incur significant energy and latency overhead, as shown by Reliable Directed Diffusion [Stann and Heidemann 2003]. The second class of techniques uses topologies that are more robust than a tree topology. Examples of such techniques include gossip-based aggregation [Boyd et al. 2005; Chen and Pandurangan 2005; Dimakis et al. 2006; Gupta et al. 2001; Kempe et al. 2003] and Tiny Aggregation over a DAG [Madden et al. 2002]. In the latter, each node with accumulated value v sends v/k to each of its k parents in a directed acyclic graph (DAG) topology. For aggregates such as Count or Sum, this reduces from v to v/k the error resulting from a single packet loss, but the overall aggregation error remains high. This is demonstrated in Figure 1(b), which shows that both the tree (TAG) and the DAG (TAG2, two parents) versions consistently overestimate the actual average value. Moreover, the high variance of the computed aggregate suggests that simply scaling the measured value up or down will not solve the problem. To avoid this problem, gossip-based aggregation [Boyd



(a) Nodes accounted for in TAG (b) Avg computed by different schemes

Fig. 1. (a) Random placement of 600 sensors in a 20×20 unit area and their simulated activity with a realistic wireless communication model (details in Section 6). The lines show the paths taken by the sensor readings that reached the querying node at the center, at the completion of a typical epoch. All other readings were lost due to communication failures. (b) The average value computed with different aggregation schemes, for each of 40 epochs. Each sensor has a value inversely proportional to the square of its distance from the querying node at the center, emulating the intensity readings of a radiation source at the center.

et al. 2005; Chen and Pandurangan 2005; Dimakis et al. 2006; Kempe et al. 2003], in which nodes repeatedly send a fraction of their accumulated value to a random other node, requires a reliable communication protocol. However, as noted above, reliable communication incurs significant energy overhead. [Gupta et al. 2001] provides a gossip-based aggregation algorithm that does not require reliable communication, but it requires expensive mechanisms such as explicit maintenance of a balanced tree and (multi-hop) communication between random pairs of nodes. Finally, there are many highly robust gossip-based protocols [Jenkins et al. 2001; Karp et al. 2000; Vogels et al. 2003] that do not require reliable communication, but they are designed primarily for information dissemination and can not be directly used for in-network aggregation.

The fundamental problem with these schemes is that aggregation and the required routing topology are *tightly coupled*. As a result, it is not possible to use arbitrarily robust routing, like multi-path routing, to mask node and transmission failures. While desirable for its robustness, multi-path routing creates the problem of *message duplication*: individual readings and partial sums sent along multiple paths cause a large fraction of the readings to be accounted for *multiple* times in the query answer. For example, if a partial sum were sent along four paths (to improve the likelihood that at least one path succeeds), and three of them happen to succeed, that partial sum would contribute to the total sum three times instead of once.

1.2 Synopsis Diffusion

In this paper, we present *synopsis diffusion*, a general framework for combining multi-path routing schemes with clever algorithms to avoid double-counting. By

	Reliable communication	*Unreliable communication*
Tree topology	**Robust**, Not energy-efficient E.g., Reliable Directed Diffusion [Stann and Heidemann 2003]	**Energy-efficient**, Not robust E.g., TAG [Madden et al. 2002], Directed Diffusion [Intanagonwiwat et al. 2000]
More robust topology	**Robust**, Not energy-efficient E.g., Gossip [Kempe et al. 2003]	**Energy-efficient** & **Robust** E.g., Synopsis diffusion (this paper)

Table I. Classification of aggregation schemes based on whether they use a robust topology and whether they use energy-efficient unreliable communication.

decoupling aggregation from message routing, synopsis diffusion enables the use of arbitrary multi-path routing. Thus, for example, the level of redundancy in message routing, as a trade-off with energy consumption, can be adapted to sensor network conditions. As a result, highly accurate and reliable answers can be obtained, all the while consuming roughly the same energy as (inaccurate, unreliable) tree-based schemes. Moreover, the aggregation process remains relatively independent of the underlying dynamics of message routing, so that the aggregation code can be written without worrying about such dynamics.

Table I summarizes how synopsis diffusion and different existing approaches fit within the general design space.

Synopsis diffusion achieves its decoupling of aggregation and routing through the use of *order- and duplicate-insensitive (ODI) synopses*. To the best of our knowledge, this is the first paper to formally define and study this important class of synopses. Previous and concurrent works [Alon et al. 1999; Bawa et al. 2004; Considine et al. 2004; Flajolet and Martin 1985; Palmer et al. 2002; Tao et al. 2004] consider only isolated examples of such synopses. ODI synopses are small-size digests of the partial results received at a node such that any particular sensor reading is accounted for only once. In other words, the synopsis at a node is the same regardless of (1) the order in which readings or partial results are received, and (2) the number of times a given reading from a given sensor arrives at the node (either directly or indirectly via partial results). While developing ODI synopses for aggregates such as Max and Min is trivial, ODI synopses for duplicate-sensitive aggregates (e.g., Sum, Count, Avg, Median, Uniform sample) are more challenging to devise.

This paper establishes a formal foundation for synopsis diffusion and demonstrates its implications to sensor network aggregation. It makes the following contributions:

—*A Novel Aggregation Framework.* We introduce and formalize the synopsis diffusion framework and the ODI synopses. Moreover, we present simple properties that characterize ODI synopses, and show how these properties can be used to ease the design of (provably correct) synopsis diffusion algorithms.

—*Better Aggregation Topologies.* We show how ODI synopses enable energy-saving communication strategies such as (1) exploiting the wireless broadcast communication medium by having any and all listeners take advantage of any message they hear, (2) eliminating acknowledgment messages because ODI synopses en-

able implicit acknowledgments, and (3) quickly accounting for changes in network connectivity. By exploiting these techniques, we show how to construct an adaptive aggregation topology (*Adaptive Rings*) that is as energy efficient as—but much more robust than—a tree topology. Its significant accuracy improvement is demonstrated in Figure 1(b) by the A.Rings curve.

—*Example Aggregates.* We present a number of aggregates that can be accurately estimated using ODI synopses. These include count, sum, average, min, max, moment statistics, quantiles, range aggregates, frequent items, and uniform samples.

—*Performance Evaluation.* We present an extensive performance study on a realistic simulator (the TAG system simulator) demonstrating the significant robustness, accuracy, and energy-efficiency improvements achieved by synopsis diffusion.

Synopsis diffusion has recently been implemented on top of the sensornet protocol (SP) [Polastre et al. 2005] and evaluated on several hardware platforms including *mica2* and *telos*.

Concurrent with our work, Considine *et al.* [Considine et al. 2004] independently proposed using duplicate-insensitive sketches for robust aggregation in sensor networks and demonstrated the advantages of a broadcast-based multi-path routing topology over previous tree-based approaches. However, as we will discuss in Section 7, they primarily focused on energy-efficient computation of the Sum aggregate, and did not address the other contributions listed above.

The remainder of the paper is organized as follows. Section 2 presents the basic synopsis diffusion approach. Section 3 presents our formal framework and theorems for ODI synopses. Section 4 presents ODI synopses for additional aggregates. Section 5 describes our Adaptive Rings routing scheme. Section 6 describes our experimental results and various trade-offs that synopsis diffusion enables. Section 7 describes related work, and conclusions appear in Section 8. Some additional details appear in the appendix.

2. SYNOPSIS DIFFUSION

In this section, we describe *synopsis diffusion*, a novel in-network aggregation framework that enables robust, highly-accurate estimations of duplicate-sensitive aggregates. The basic approach is to use best effort, multi-path routing schemes (e.g., [Ganesan et al. 2001]) together with duplicate-insensitive in-network aggregation schemes. This section describes the general framework and, to illustrate the framework's use, presents examples of both a routing scheme (called *Rings*) and an aggregation scheme (for the Count aggregate). Although the description is based on adapting the TAG communication model and continuous query scheme [Madden et al. 2002], it is not dependent on the particular model or scheme.

Synopsis diffusion performs in-network aggregation. The partial result at a node is represented as a synopsis [Babcock et al. 2002; Gibbons and Matias 1999], a small digest (*e.g.,* histogram, bit-vectors, sample, etc.) of the data. The aggregate computation is defined by three functions on the synopses:

—**Synopsis Generation:** A synopsis generation function $SG(\cdot)$ takes a sensor reading (including its metadata) and generates a synopsis representing that data.

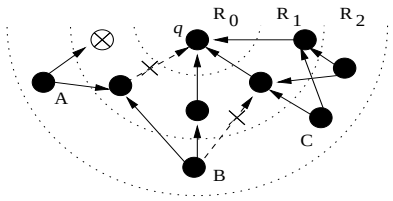


Fig. 2. Synopsis diffusion over the Rings topology. Crossed arrows and circles represent failed links and nodes.

—**Synopsis Fusion:** A synopsis fusion function $SF(\cdot, \cdot)$ takes two synopses and generates a new synopsis.
—**Synopsis Evaluation:** A synopsis evaluation function $SE(\cdot)$ translates a synopsis into the final answer.

The exact details of the functions $SG()$, $SF()$, and $SE()$ depend on the particular aggregate query to be answered. An example is given at the end of this section; additional examples are presented in Section 4.

A synopsis diffusion algorithm consists of two phases: a *distribution phase* in which the aggregate query is flooded through the network and an aggregation topology is constructed, and an *aggregation phase* where the aggregate values are continually routed toward the querying node. Within the aggregation phase, each node periodically uses the function $SG()$ to convert sensor data to a local synopsis and the function $SF()$ to merge two synopses to create a new local synopsis. For example, whenever a node receives a synopsis from a neighbor, it may update its local synopsis by applying $SF()$ to its current local synopsis and the received synopsis. Finally, the querying node uses the function $SE()$ to translate its local synopsis to the final answer. The continuous query defines the desired period between successive answers, as well as the overall duration of the query [Madden et al. 2003; Yao and Gehrke 2003]. One-time queries can also be supported as a special, simplified case.

An important metric when discussing the quality of query answers in the presence of failures is the fraction of sensor nodes contributing to the final answer, called the *percent contributing*. With synopsis diffusion, a sensor node contributes to the final answer if there is at least one failure-free "propagation path" from it to the querying node. A *propagation path* is a hop-by-hop sequence of successfully transmitted messages from the sensor node to the querying node. Note that it does not require that the sensor's reading actually be transmitted in the message, because with in-network aggregation, the reading will typically be folded into a partial result at each node on the path.

Although the synopsis diffusion framework is independent of the underlying topology, to make it more concrete, we describe next an example overlay topology, called *Rings*, which organizes the nodes into a set of rings around the querying node.

2.1 Synopsis Diffusion on a Rings Overlay

During the query distribution phase, nodes form a set of rings around the querying node q as follows: q is in ring R_0, and a node is in ring R_i if it receives the

query first from a node in ring R_{i-1} (thus a node is in ring R_i if it is i hops away from q). The subsequent query aggregation period is divided into *epochs* and one aggregate answer is provided at each epoch. As in [Madden et al. 2002], we assume that nodes in different rings are loosely time synchronized and are allotted specific time intervals when they should be awake to receive synopses from other nodes. The duration of the allotted time is determined a priori based on the density of deployment (so that even if the sensors perform carrier sensing, all the sensors get enough time to transmit their messages once). For simplicity, we assume that this allotted time is the same for all nodes in the deployment. This time can be determined by first identifying the maximum local density of the deployment (i.e., the maximum number of nodes whose transmissions interfere with each other) and then finding, via experiments, the total amount of time the nodes require to transmit at least one message each.

We now describe the query aggregation phase in greater detail, using the example Rings topology in Figure 2 for illustration. In this example, node q is in R_0, there are five nodes in R_1 (including one node that fails during the aggregation phase), and there are four nodes in R_2. At the beginning of each epoch, each node in the outermost ring (R_2 in the figure) generates its local synopsis $s = SG(r)$, where r is the sensor reading relevant to the query answer, and *broadcasts* it. A node in ring R_i wakes up at its allotted time, generates its local synopsis $s := SG(\cdot)$, and receives synopses from all nodes within transmission range in ring R_{i+1}[1]. Upon receiving a synopsis s', it updates its local synopsis as $s := SF(s, s')$. At the end of its allotted time the node broadcasts its updated synopsis s. Thus, the fused synopses propagate level-by-level toward the querying node q, which at the end of the epoch returns $SE(s)$ as the answer to the aggregate query.

Figure 2 shows that even though there are link and node failures, nodes B and C have at least one failure-free propagation path to the querying node q. Thus, their sensed values are accounted for in the answer produced this epoch. In contrast, all of the propagation paths from node A failed, so its value is not accounted for.

Because the underlying wireless communication is broadcast, each node transmits exactly once; therefore, *Rings generates the same optimal number of messages as tree-based approaches* (e.g., [Madden et al. 2002; 2003; Madden et al. 2002; Zhao et al. 2003]). However, because synopses propagate from the sensor nodes to the querying node along multiple paths, *Rings is much more robust.* (This added robustness is quantified in Section 6.)

2.2 Duplicate-Sensitive Aggregates

With synopsis diffusion, aggregation can be done over arbitrary message routing topologies. *The main challenge of a synopsis diffusion algorithm is to support duplicate-sensitive aggregates correctly for all possible multi-path propagation schemes.* As we will show in Section 3, to achieve this, we require the target aggregate function (*e.g.*, Count) to be mapped to a set of *order-* and *duplicate-insensitive* (ODI) synopsis generation and fusion functions. Intuitively, such a set of functions

[1]Note that there is no one-to-one (or even static) relationship between the nodes in ring R_i and those in ring R_{i+1} — a node in ring R_i fuses all the synopses it overhears from the nodes in ring R_{i+1}.

ensure that a partial result at a node u is determined by the set of readings from sensor nodes with propagation paths to u, independent of the overlap in these paths and any overlap with redundant paths. No matter in what combination the fusion functions are applied, the result is the same. Thus, a sensor reading is accounted for (exactly once) in the aggregate if there is a propagation path from the sensor node to the querying node, and it is never accounted for more than once. We illustrate such functions using the following algorithm for Count.

Count. This algorithm counts the approximate total number of live sensor nodes in the network. (It can be readily adapted to other counting problems.) Note that the standard in-network approach for Count, where each node sums its children's accumulated counts and sends the sum to its parent(s), will not work with arbitrary topologies—the same value may be counted more than once if the topology is not a tree. The approximation algorithm we present here is adapted from Flajolet and Martin's algorithm (FM) [Flajolet and Martin 1985] for counting distinct elements in a multi-set. It is a well-known algorithm for duplicate-insensitive approximate Count [Bawa et al. 2004; Considine et al. 2004; Przydatek et al. 2003]. The algorithm uses the following *coin tossing experiment CT(x)*: toss a fair coin until either the first heads occurs or x coin tosses have occurred with no heads, and return the number of coin tosses. For example, possible outcomes of $CT(3)$ are 1 (when the first toss is heads), 2 (first toss tails, second toss heads), or 3 (first two tosses tails); these occur with probability $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{4}$, respectively. Note that $CT()$ simulates the behavior of the exponential hash function that is used in FM:

$$\text{for } i = 1, \ldots, x-1 : CT(x) = i \text{ with probability } 2^{-i} \tag{1}$$

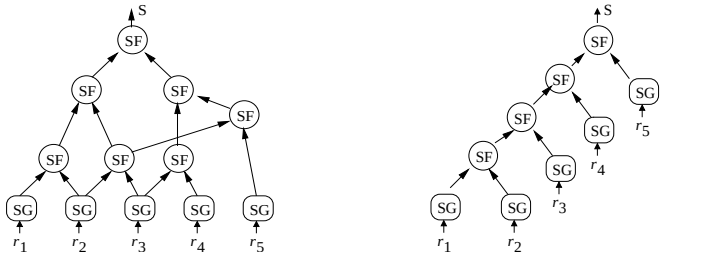
In the Appendix 9.2, we will show how $CT()$ can be efficiently implemented in resource-constrained sensors.

The different components of the synopsis diffusion algorithm for Count are as follows.

—*Synopsis:* The synopsis is a bit vector of length $k > \log(n)$, where n is an upper bound on the number of sensor nodes in the network.[2]
—$SG()$*:* Output a bit vector s of length k with only the $CT(k)$'th bit set.
—$SF(s, s')$*:* Output the bit-wise Boolean OR of the bit vectors s and s'.
—$SE(s)$*:* If i is the index of the lowest-order bit in s that is still 0, output $2^{i-1}/0.77351$ [Flajolet and Martin 1985].

If all the live sensor nodes have at least one failure-free propagation path, then the final bit vector s to which $SE()$ is applied will indicate precisely which bit positions have been set by at least one node. Intuitively, the number of live sensor nodes, N, is proportional to 2^{i-1} because by (1) the probability of N nodes all failing to set the i'th bit is $(1 - 2^{-i})^N$. This is approximately $1/e < 0.37$ when $N = 2^i$, and even smaller for larger N. The accuracy of the algorithm can be improved by having each synopsis maintain multiple independent bit-vectors and then taking the average of the indices within $SE()$ [Flajolet and Martin 1985].

[2]The upper bound can be approximated by the total number of sensor nodes deployed initially, or by the size of the sensor-id space.



(a) Aggregation DAG (b) Canonical left-deep tree

Fig. 3. Equivalent graphs under ODI-correctness

In Section 3, we will prove formally the order- and duplicate-insensitivity of this algorithm and that the approximation error guarantees of [Flajolet and Martin 1985] hold for the algorithm.

Additional examples in Section 4 demonstrate that synopsis diffusion can be used for very differing aggregates, if suitable ODI synopses can be found.

3. FORMAL FRAMEWORK, THEOREMS, AND IMPLICATIONS

In this section, we present the first formal foundation for duplicate-insensitive aggregation. We define a synopsis diffusion algorithm to be "ODI-correct" if and only if its $SG()$ and $SF()$ functions are order- and duplicate-insensitive. Intuitively, these two properties ensure that the final result is independent of the underlying routing topology—the computed aggregate is the same irrespective of the order in which the sensor readings are combined and the number of times they are included during the multi-path routing. We formalize these two requirements later in this section. We begin with the following definitions.

3.1 Definitions

A *sensor reading* r is a tuple consisting of both one or more sensor measurements and any meta-data associated with the measurements (*e.g.*, timestamp, sensor id, and location). A *synopsis computation* is a finite collection of sensor readings, $SG()$ functions applied to the sensor readings, and $SF()$ functions applied to pairs of synopses generated by $SG()$ or $SF()$ functions, resulting in a synopsis s. We can represent a synopsis computation by its *aggregation DAG*, as shown in Figure 3(a). There is a node for each of the different instantiations of the functions $SG()$ (the leaf nodes in the DAG) and $SF()$ (the non-leaf nodes). There is an edge $e : f_1 \rightarrow f_2$ if and only if the output of the function f_1 is an input to the function f_2. Thus, all internal nodes have two incoming edges and 0 or more outgoing edges.

We define a *synopsis label* function, $SL()$, for a synopsis s inductively from its aggregation DAG, as follows. There are two cases for $SL(s)$, depending on whether the synopsis s results from an application of $SF()$ or an application of $SG()$:

$$SL(s) = \begin{cases} SL(s_1) \uplus SL(s_2) & \text{if } s = SF(s_1, s_2) \\ \{r\} & \text{if } s = SG(r) \end{cases}$$

The operator $\uplus$ takes two multi-sets and returns the multi-set consisting of all the elements in both multi-sets, including any duplicates. For example, $\{a, b, c, c\} \uplus$

$\{b, c, d\} = \{a, b, b, c, c, c, d\}$. $SL()$ is a multi-set consisting of all the sensor readings contributing to s, possibly with duplicates. It is determined by the sensor readings and the applications of $SG()$ and $SF()$—it is independent of the particulars of $SG()$ and $SF()$. Note that a synopsis label is a virtual concept, used only for reasoning about the correctness of $SG()$ and $SF()$ functions: $SL()$ is *not* executed by the sensor network.

The notion of what constitutes a "duplicate" may vary from query to query, *e.g.*, a query computing the number of sensors with temperature above 50°F considers two readings from the same sensor as duplicates, whereas a query for the number of distinct temperature readings considers any two readings with the same temperature as duplicates. For a given query q, we define a *projection operator*

$$\Pi_q : \textit{multi-set of sensor readings} \mapsto \textit{ordered set of values}$$

that converts a multi-set of sensor readings (tuples) to its corresponding ordered set of subtuples (called "values") by selecting some set of the attributes in a tuple (the same set for all tuples), discarding all other attributes from each tuple, and then removing any duplicates in the resulting multi-set of subtuples. The set of selected attributes must be such that two readings are considered duplicates for the query q if and only if their values are the same. For example, for a query computing the number of distinct temperature readings, the value for a sensor reading is its temperature measurement. For a query computing the average temperature, the value of a sensor reading is its (temperature measurement, sensor id) pair. The set of values is ordered according to an arbitrary total order on the value domain.

3.2 ODI-Correctness

We now define what it means to be order- and duplicate-insensitive. Let $\mathcal{R}$ be the universe of valid sensor readings. Consider a $SG()$ function, a $SF()$ function, and a projection operator Π_q; these define a universe, $\mathcal{S}$, of valid synopses over the readings in $\mathcal{R}$. We assume that $SF()$ is a deterministic function of its inputs. The formal definition of the properties we seek is:

—A synopsis diffusion algorithm is **ODI-correct** if $SF()$ and $SG()$ are *order- and duplicate-insensitive* functions, *i.e.*, they satisfy: For all synopsis computations resulting in some synopsis $s \in \mathcal{S}$: $s = SG^*(V)$, where $V = \Pi_q(SL(s)) = \{v_1, \ldots, v_k\}$ and $SG^*()$ is defined inductively as

$$SG^*(V) = \begin{cases} SF(SG^*(V - \{v_k\}), SG(r_k)) & \text{if } |V| = k > 1 \\ SG(r_1) & \text{if } |V| = 1 \end{cases}$$

where $\Pi_q(\{r_i\}) = \{v_i\}$.[3]

Figure 3 helps illustrate ODI-correctness. Corresponding to an aggregation DAG (Figure 3(a)), ODI-correctness defines a canonical left-deep tree (Figure 3(b)). The leaf nodes are the functions $SG()$ on readings that yield *distinct* values under Π_q (in this simple example, $\Pi_q(\{r_1\}) < \cdots < \Pi_q(\{r_5\})$), and the non-leaf nodes are the functions $SF()$. A synopsis diffusion algorithm is ODI-correct if for *any* aggregation

[3]If there are multiple distinct $r_i \in SL(s)$ such that $\Pi_q(\{r_i\}) = \{v_i\}$, the same s must be computed regardless of which r_i is selected.

DAG, the resulting synopsis is *identical* to the synopsis s produced by the canonical left-deep tree.

More simply, regardless of how $SG()$ and $SF()$ are applied (*i.e.*, regardless of the redundancy arising from multi-path routing), the resulting synopsis is the same as when each distinct value is accounted for only once in s. We use a left-deep tree for our canonical representation because it lends itself to an important connection with traditional data streams (as discussed in Section 3.3).

3.2.1 *A Simple Test for ODI-Correctness.* The definition of ODI-correctness captures the overall goal of order- and duplicate-insensitivity. However, it is not immediately useful for designing synopsis diffusion algorithms because verifying correctness using this definition would entail considering the *unbounded* number of ways that $SG()$ and $SF()$ can be applied to a set of sensor readings and comparing each against the synopsis produced by the canonical tree.

Thus, a key contribution of this paper is in deriving the following simple test for ODI-correctness. There are four properties to check to complete the test.

—Property P1: $SG()$ **preserves duplicates:** $\forall r_1, r_2 \in \mathcal{R} : \Pi_q(\{r_1\}) = \Pi_q(\{r_2\})$ implies $SG(r_1) = SG(r_2)$. That is, if two readings are considered duplicates (by Π_q) then the same synopsis is generated.
—Property P2: $SF()$ **is commutative:** $\forall s_1, s_2 \in \mathcal{S} : SF(s_1, s_2) = SF(s_2, s_1)$.
—Property P3: $SF()$ **is associative:** $\forall s_1, s_2, s_3 \in \mathcal{S} : SF(s_1, SF(s_2, s_3)) = SF(SF(s_1, s_2), s_3)$.
—Property P4: $SF()$ **is same-synopsis idempotent:** $\forall s \in \mathcal{S} : SF(s, s) = s$.

While the first three properties are perhaps intuitive, note that the fourth property is much weaker than the duplicate-insensitivity property required for ODI-correctness. In particular, property P4 refers only to what happens when $SF()$ is applied to the *exact same synopsis* for both its arguments. It says nothing about what happens when $SF()$ is applied to differing arguments that come from overlapping sets of sensor readings.[4]

Given the simplicity of properties P1–P4, it is surprising that they characterize ODI-correctness. The next theorem shows that indeed this is the case.

THEOREM 1. *Properties P1–P4 are necessary and sufficient properties for ODI-correctness.*

The proof is given in Appendix 9.1.

We illustrate how these properties can be used to prove the ODI-correctness of a synopsis diffusion algorithm by revisiting the Count algorithm that estimates the number of sensor nodes in the network.

CLAIM 1. *The Count algorithm in Section 2.2 is ODI-correct.*

[4]For example, consider the $SF()$ function that takes two numbers x and y and returns their average. This satisfies property P4, because the average of x and x equals x. However, the function cannot be used to compute a duplicate-insensitive average of all the sensor readings. For example, if the readings are 2, 4, and 36, we have SF(SF(2,4),SF(2,36)) = 11 but SF(SF(2,36),SF(4,36)) = 19.5 (and the exact average is 14).

Proof. Consider a projection operator Π_q that maps a set of sensor readings to the corresponding sensor ids. In the Count algorithm, $SF(s, s')$ is the Boolean OR of the bit vectors s and s'. Since Boolean OR is commutative and associative, so is $SF()$. Next, observe that $\Pi_q(\{r_1\}) = \Pi_q(\{r_2\})$ if and only if r_1 and r_2 have the same sensor id and hence are the same reading. Thus $SG(r_1) = SG(r_2)$.[5] Finally, $SF(s, s)$ is the Boolean OR of the bit vector s with itself, which equals s. Therefore, properties P1–P4 hold, so by Theorem 1, the algorithm is ODI-correct. □

Note that the $SE()$ function did not factor into the considerations of ODI-correctness. ODI-correctness only shows that $SE()$ will see the same synopsis as the left-deep tree. The accuracy of the approximate answer, on the other hand, depends on the accuracy of applying $SE()$ to this synopsis. Clever algorithms are still required to get provably good approximations, although the task has been simplified to being able to show (1) the ODI-correctness of $SG()$ and $SF()$, and (2) the accuracy of $SE()$ when applied to synopses from left-deep trees.

3.2.2 *Algebraic Structure and Implications.* We begin with the following corollary of Theorem 1.

Corollary 1. *Consider an ODI-correct synopsis diffusion algorithm with functions $SG()$ and $SF()$. The set $\mathcal{S}$ of synopsis generated by $SG()$ together with the binary function $SF()$ forms a* semi-lattice *structure.*

A semi-lattice [Davey and Priestley 2002] is an algebraic structure with the property that for every two elements in the structure there is an element that is their least upper bound. The function $SF()$ is essentially the *join operator* in lattice terminology; and therefore,

$$\text{if } z = SF(x, y) \text{ then } SF(x, z) = z \text{ and } SF(y, z) = z \tag{2}$$

An example of a semi-lattice is the fixed size bit-vectors used in the Count algorithm with the Boolean OR function. The top of the lattice is the all 1's bit-vector, the bottom is the all 0's bit-vector, and for any two bit-vectors x and y, if x OR $y = z$, then x OR $z = z$ and y OR $z = z$. Corollary 1 follows immediately from Theorem 1 because it is well known that a commutative, associative, idempotent binary function on a set forms a semi-lattice [Davey and Priestley 2002].

Implications. The semi-lattice structure of ODI synopses and the $SF()$ function has an attractive practical implication in the context of ad hoc wireless sensor networks. In such networks, the underlying routing topology needs to be continuously adapted to cope with unpredictable node and communication failures. Using explicit acknowledgments for this purpose wastes considerable energy. A common solution in ad hoc wireless networks is to use implicit acknowledgments [Johnson and Maltz 1996] to monitor communication failures. Each node u sending to u' snoops the subsequent broadcast from u' to see if u's message was indeed forwarded (and, therefore, was previously received) by u'. However, no known approaches could support implicit acknowledgments as part of in-network aggregation. Consider, for

[5] We assume here that SG is applied only once to a sensor reading. The case where SG can be redundantly applied can be (provably) handled by using the exponential hash function of FM, instead of the simpler CT-based generation.

example, computing the Sum with the TAG protocol. If u sends the value x to u', and later overhears u' transmitting some value $z \geq x$, there can be two possibilities: either u' has heard from u and has included x in z, or u' has not heard from u[6] and z is the sum of the values u' heard from its other children. Thus, u has no way of determining whether transmission through u' is reliable.

The use of ODI synopses provides an *implicit acknowledgment* mechanism and avoids the effect of this crucial problem. By (2) above, if a node u transmits the synopsis x and later overhears some parent node u' transmitting a synopsis z such that $SF(x, z) = z$, it can infer that its synopsis has been *effectively* included into the synopsis z of that parent.[7] Otherwise, it can infer that its message to that parent has been lost. Thus, overhearing a synopsis $z = SF(x, z)$ acts as an implicit acknowledgment for the node u. On inferring message loss, a sensor can retransmit its message or adapt the topology accordingly (*e.g.,* switch its parent in a Tree topology or change its level in a Rings topology).

3.3 Error Bounds of Approximate Answers

Using synopses may provide only an approximate answer to certain queries. In fact, there are two distinct sources of errors in the final answers computed by a synopsis diffusion algorithm A. The first one is the *communication error*, which is defined as the fraction of sensor readings not accounted for in A's answer in a given epoch (*i.e.*, 1 minus the *percent contributing*). This error is introduced by the underlying routing scheme; it occurs when some of the sensors have no failure-free propagation paths to the querying node. The second source of error is the *approximation error*, which is defined as the relative error of the answer computed by A with respect to the answer computed by a corresponding exact algorithm using all the readings accounted for in A's final answer. This error is introduced by the $SG()$, $SF()$, and $SE()$ functions.

We argue that with a reasonably dense deployment (e.g., each node having 2-3 neighbors towards the querying node) and a sufficiently robust routing scheme, the communication error can be made negligible. We illustrate this using a simple analysis. Suppose the underlying multi-path routing constructs a DAG G rooted at the querying node. We consider a regular DAG of height h where each node at level $i, 1 \leq i \leq h$, has k neighbors at level $(i-1)$ to transmit its synopses toward the querying node. For simplicity of the analysis, assume that level i has d^i nodes, where d is some constant. Also assume for this analysis that message losses occur independently at random with probability p. Then the number of sensor readings N that can reach the querying node is given by $N \geq \sum_{i=0}^{h}(1-p^k)^i d^i = \frac{d^{h+1}(1-p^k)^{h+1}-1}{d(1-p^k)-1}$. Thus, the overall communication error is upper bounded by approximately $1-(1-p^k)^h$. To make it more concrete, assume that $p = 0.1, h = 10$. Then, with $k = 1$ (*i.e.*, a tree topology), the error is around 0.65,

[6]Because wireless communication can be asymmetric, u may hear from u' even if u' does not hear from u.

[7]We say it is *effectively* included because the condition $SF(x, z) = z$ does not precisely imply that the transmission from u has been received by u'. Rather, it implies that even if the transmission were lost, the loss had no effect on the synopsis transmitted by u' (because it happened to have been compensated by the synopses from other children of u').

while it is less than 0.1 and 0.01 for $k = 2$ and $k = 3$, respectively. Hence, by increasing the number of neighbors to transmit synopses toward the querying node (*i.e.,* increasing the redundancy of the underlying message routing), through denser sensor deployment if necessary, the communication error can be made insignificant.

Thus, with a robust routing topology, the main source of error in the result computed by a synopsis diffusion algorithm is the approximation error. Next, we summarize a generic framework to analyze this approximation error.

Traditionally, the error properties of approximation algorithms are analyzed in a *centralized model* where the algorithms are applied at a central place (*e.g.,* the querying node) where all the values are first collected. For example, data stream algorithms [Babcock et al. 2002] use this model. However, synopsis diffusion presents a *distributed model* where the $SG()$ and $SF()$ functions are applied in the distributed set of sensors. The following theorem shows the equivalence of these two models for an ODI-correct synopsis diffusion algorithm.

THEOREM 2. *The answer computed by an ODI-correct synopsis diffusion algorithm is the same as that computed by first collecting the values that can reach the querying node through at least one failure-free propagation path and then applying the* $SG(), SF()$, *and* $SE()$ *functions on them.*

Proof. (sketch) Consider an arbitrary instance of synopsis diffusion aggregation. By ODI-correctness, the corresponding aggregation DAG (*e.g.*, Figure 3(a)) can be reduced to a canonical left-deep tree (*e.g.*, Figure 3(b)). This left-deep tree can be viewed as processing a data stream of sensor readings at a centralized place: to each new stream value, we first apply SG and then apply SF with the current stream synopsis. □

Hence, the final result computed by a synopsis diffusion algorithm has the following semantics: (1) the final answer includes all the values that can reach the querying node through at least one failure-free propagation path, and (2) the result is the same as that found by applying the function SE on the output of a centralized data stream algorithm using SG and SF as indicated above.

Theorem 2 shows that *any approximation error guarantees provided for the well-studied centralized data stream scenario immediately apply to a synopsis diffusion algorithm, as long as the data stream synopsis is ODI-correct.* Thus, we can effectively leverage existing data stream error analysis, as illustrated in the following claim, which is an immediate corollary of Theorem 2.

CLAIM 2. *The Count algorithm in Section 2.2 has the same approximation error guarantees as Flajolet-Martin's (FM) distinct count algorithm [Flajolet and Martin 1985].*

The precise error guarantees depend on the number of independent bit-vectors used per synopsis. See [Considine et al. 2004; Flajolet and Martin 1985; Gibbons 2007] for further details.

4. EXAMPLE AGGREGATES WITH ODI-CORRECT ALGORITHMS

Many aggregates have ODI-correct synopsis diffusion algorithms, as shown in Table II. Maximum and Minimum are trivial. Count was discussed in Section 2.2.

Aggregate	Notes
Maximum, Minimum	trivial
Count, Count Distinct	see §2.2
Sum, Average, Standard deviation, Second moment	see §4.1
Uniform sample	see §4.2
Mean, kth Statistical moments	see §4.2
Medium, Quantiles	see §4.2
Frequent items	see §4.3
Range aggregates, Inner product queries	see §4.4

Table II. Summary of ODI-correct algorithms discussed in this paper.

Count Distinct can be done using a trivial adaption of Flajolet and Martin's algorithm (FM) [Flajolet and Martin 1985], along the lines of the Count algorithm. This section presents new ODI-correct synopsis diffusion algorithms for some additional important aggregates, as listed in the table.

4.1 Sum

An approximate sum of (nonnegative integer) sensor readings can be computed using a simple generalization of the Approximate Count algorithm: If the sensor node v has the value val_v to contribute to the final answer, it pretends to be a collection of val_v distinct nodes. Specifically, for $SG()$ each node v outputs a bit vector of length k' (where k' is sufficiently large to hold the maximum sum) with the following bits set: for each of val_v times, perform $CT(k')$ and set the returned bit. $SF()$ and $SE()$ are the same as in the Approximate Count algorithm.

However, running $CT()$ for val_v times, as in the above algorithm, may consume a large amount of energy when val_v is large. Instead, we give below an alternative algorithm that avoids this overhead. This algorithm is adapted from a variant of FM that instead of returning $2^{i-1}/0.77351$, where i is the index of the lowest-order 0-bit, returns 2^j, where j is the index of the *highest-order 1-bit* [Alon et al. 1999]. Because the algorithm keeps track of only the maximum bit set, the synopsis can be smaller.

—**Synopsis:** Assume that the values we wish to add are integers in the range $[0..X]$. Because the sum can be bounded by nX, where n is an upper bound on the number of nodes in the network, the synopsis is an integer in the range $[1..\log(nX)]$, i.e., its size is $\log\log(nX)$ bits.
—$SG()$**:** For node v, select a random number x_v in $[0, 1]$ and output $\lceil -\log_2(1 - x_v^{1/val_v})\rceil$.
—$SF(s, s')$**:** Output $\max(s, s')$.
—$SE(s)$**:** Output 2^{s-1}.

The intuition behind the $SG()$ function is as follows. The goal is to mimic the process where for each of val_v times, $CT(k')$ is done and the returned bit is set. The probability that the i'th bit will be the maximum bit set after m trials ($m = val_v$ in this case) *equals* the probability that all m trials return the i'th bit or less *minus* the probability that all m trials return the $(i-1)$th bit or less, i.e., $(1 - 2^{-i})^m - (1 - 2^{-(i-1)})^m$. To select a maximum bit set according to this probability distribution, $SG()$ selects a random number x and finds the smallest

integer $i \geq 1$ such that $x \leq (1 - 2^{-i})^m$. Solving for i, we seek the smallest integer i such that $i \geq -\log_2(1 - x^{1/m})$, namely, $\lceil -\log_2(1 - x^{1/m}) \rceil$.

As with approximate Count, the variance of the approximation can be decreased by maintaining multiple independent synopses and having $SE()$ output $2^{\bar{s}}$, where $\bar{s}$ is the average of the indices of the highest-order 1-bits [Alon et al. 1999].

Concurrent with our work, Considine *et al.* [Considine et al. 2004] developed an energy-efficient variant of the approximate Sum algorithm based on the original FM. Note that the FM algorithm they use is somewhat more accurate (i.e., has lower variance) in practice than the [Alon et al. 1999] variant we use. Intuitively, this is because the lowest-order 0-bit (as in FM) is "supported" by all the lower order bits being set, whereas the highest-order 1-bit (as in [Alon et al. 1999]) can be the result of a single random outlier. Thus, their algorithm is somewhat more accurate than ours for the same amount of energy, and so we use their algorithm in our Approximate Sum experiments in Section 6.

Aggregates computed from Sum. Average, Standard Deviation, and Second Moment can be computed by applying the Sum algorithm (and the Count algorithm) over suitably defined values [Considine et al. 2004]. For example, Average can be computed by applying both the Sum and the Count algorithms, and then dividing.

4.2 Uniform Sample of Sensor Readings

Suppose each node u has a value val_u. Our goal is to compute a uniform sample of a given size K of the values occurring in all the nodes in the network. Because of message loss, however, one cannot guarantee a uniform sample of all the nodes. Instead, our algorithm outputs *a uniform sample of all the contributing nodes*, *i.e.*, of all the nodes with failure-free propagation paths (regardless of whether they are selected for the sample). Note that the set of contributing nodes represents a very high percentage of the nodes, given reasonably dense deployments. Recall from Section 3.3 that we expect more than 99% of the nodes to contribute if each node has just two neighbors towards the querying node, and the percentage increases rapidly with additional neighbors. Therefore, as demonstrated by the experiments in Section 6.9, even when considering all the nodes (not just the contributing nodes), our algorithm outputs a reasonable approximation of a uniform sample. The components of our algorithm are as follows:

—*Synopsis:* A sample of size K of ⟨value, random number, sensor id⟩ tuples. (Initially, it will have fewer than K tuples, until there are at least K nodes contributing to the synopsis.)
—$SG()$*:* At node u, output the tuple $\langle val_u, r_u, id_u \rangle$, where id_u is the sensor id for node u, and r_u is a uniform random number within the range $[0, 1]$.
—$SF(s, s')$*:* From all the tuples in $s \cup s'$, output the K tuples $\langle val_i, r_i, id_i \rangle$ with the K largest r_i values. If there are less than K tuples in $s \cup s'$, output them all.
—$SE(s)$*:* Output the set of values val_i in s.

Because the $SG()$ function labels each value with a uniform random number and thus places it in a random position in the global ordering of all the values in the network, selecting the K largest positions results in a uniform sample of the values

from contributing nodes. The (duplicate-removing) union operation in SF ensures that the synopsis accounts for a given node's value at most once.[8]

Note that traditional sampling procedures [Kostic et al. 2003] are ill-suited for multi-path routing because they are duplicate-sensitive. This extra duplication results in samples that are far from uniform (even when considering only the contributing nodes).

Aggregates computed from uniform samples. Many useful holistic aggregates, for which there are no efficient and exact in-network aggregation algorithms, can be approximated from a uniform sample computed using the previous algorithm. For example, given the sensor values $val_1, val_2, \ldots, val_n$, the k-th Statistical Moment $\mu_k = \frac{1}{n}\sum_{i=1}^{n} {val_i}^k$ (*e.g.*, μ_1 is the Mean) and the k-th percentile/quantile value for $0 < k < 100$ (e.g., $k = 50$ is the Median) can be approximated with ϵ additive error[9] and with probability $1-\delta$ by using a sample of size $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ [Bar-Yossef et al. 2001]. Given the sample S of size $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$, the k-th percentile of the sensor values is estimated by selecting the k-th percentile value in S. Thus, our random sampling algorithm provides an efficient way to estimate these holistic aggregates.

4.3 Frequent Items

The goal of this algorithm is to return all items occurring at least T times, for a given threshold T, and their counts (both approximated). It uses the $CT()$ function from the Count algorithm described in Section 2.

—*Synopsis:* A set of (value, weight) pairs, where the values are unique and the weights are at least $\log(T)$.
—*SG():* At node u, compute $CT(k)$ where $k > \log(n)$ and n is an upper bound on the total number of items; call this the "weight" of val_u. If the weight is at least $\log(T)$, output $(val_u, weight)$. Otherwise, output the empty set.
—*SF*(s, s')*:* For each distinct value v in $s \cup s'$, discard all but the pair (v, weight) with maximum weight for that value. Then output the remaining pairs.
—*SE*(s)*:* For each (value, weight) pair in s, output (value, 2^{weight}) as a frequent value and its approximate count.

Essentially, the algorithm determines the frequency of an item by running an ODI-correct Count algorithm for each value. The counting is done using an Alon *et al.* variant [Alon et al. 1999] of Flajolet-Martin's algorithm (FM). In particular, we estimate the number of distinct occurrences of a value by keeping track of the highest outcome of $CT(k)$ for that value, and then estimating the number of such distinct occurrences by taking 2 to the power of that highest outcome. It follows from Equation (1) in Section 2 that a value occurring at least T times is expected to have at least one of its calls to $CT()$ return at least $\log(T)$. Thus, the algorithm only tracks weights that are at least $\log(T)$. Note that because global popularity is determined by the *maximum* weight per value (as opposed to, say, by *summing* up

[8]Note that in practice, id_u need not be included in the synopsis, because equality in the random id r_u can effectively detect duplicates. Also, the range need not be $[0, 1]$, e.g., an integer range suffices as long as the probability of random number collisions is small.
[9]For k-th percentile aggregates, the error is with respect to the rank of the value not its magnitude.

estimated local popularities), there are no issues with globally popular items being overlooked because they are not locally popular.

To reduce the number of false positives and false negatives, multiple independent instances of the $SG()$ can be run at each node. The synopsis consists of one set of pairs for each instance, $SF()$ is applied to each instance independently, and $SE()$ outputs values that appear in at least half the sets.

Note that the size of the synopsis increases with the number of frequent items, and hence is nonincreasing in T. Thus, it is necessary to select T sufficiently large in order to maintain a small synopsis.

4.4 Count-Min Sketch Generation

Cormode et al. [Cormode and Muthukrishnan 2005] present a sublinear space data structure, called a *Count-Min Sketch*, for summarizing data streams. Let I be the set of possible item labels. The sketch is a two-dimensional count array A, with each row r having a random pairwise independent hash function h_r that maps elements in I uniformly to columns in A. On arrival of an item $i \in I$ with count c, for each row j of A, $A[j, h_j(i)]$ is incremented by c. Cormode et al. show that this sketch can be used to estimate many aggregates with good time and space complexity, such as point queries (what is the sum of the counts of all items with a given label i?), range queries, inner product queries, finding quantiles, frequent items, etc.

Although the Count-Min Sketch has been proposed in the context of a single stream, it can be extended to be used in the synopsis diffusion framework by replacing the duplicate-sensitive counter of each array cell with an ODI Sum synopsis from Section 4.1. As our ODI Sum synopsis handles only the sum of nonnegative integers, we require the counts associated with each node to be nonnegative integers. As in [Cormode and Muthukrishnan 2005], the goal is to produce an estimate within ϵN additive error with probability at least $1 - \delta$, where N is the sum of all the item counts and ϵ and δ are arbitrarily chosen target values between 0 and 1.

—*Synopsis:* A $w \times d$ two-dimensional array of Sum synopses, where $w = O(\frac{1}{\epsilon})$ and $d = O(\log(1/\delta))$.
—*SG():* At node u, initialize a $w \times d$ two-dimensional array A_u to all zeros. Let $label_u$ and val_u be the label and count associated with the node u. Let $y_u = SG_{Sum}()$, where SG_{Sum} is the generation function for Sum synopses (which uses val_u). For each row r of A_u, store y_u in $A_u[r, h_r(label_u)]$. (We assume that each node has the same set of hash functions $h_1(), \ldots, h_w()$.)
—*SF(s, s'):* For each cell (x, y), output $SF_{Sum}(s_{x,y}, s'_{x,y})$, where SF_{Sum} is the fusion function for Sum synopses and $s_{x,y}$ is the cell (x, y) of the synopsis s.
—*SE(s):* For each cell (x, y), output $SE_{Sum}(s_{x,y})$, where SE_{Sum} is the evaluation function for Sum synopses.

The $SG()$ function outputs an array A_u for node u that corresponds to the state of the original Count-Min Sketch after the arrival of a single item with label $label_u$ and count val_u, except that the count val_u in the original sketch is replaced by the Sum synopsis y_u. The $SF()$ function mimics the accumulating counts of the Count-Min Sketch, again with duplicate-sensitive counters replaced by ODI Sum synopses. The $SE()$ function converts the Sum synopses in each cell to estimated sums; call this two-dimensional count array A. Any of the specific estimation

algorithms in [Cormode and Muthukrishnan 2005] can then be applied to A. For example, an estimate for the sum of the counts of all items with a given label i can be computed by taking the minimum of $A[r, h_r(i)]$ over all rows r. See [Cormode and Muthukrishnan 2005] for further details on the various estimation procedures.

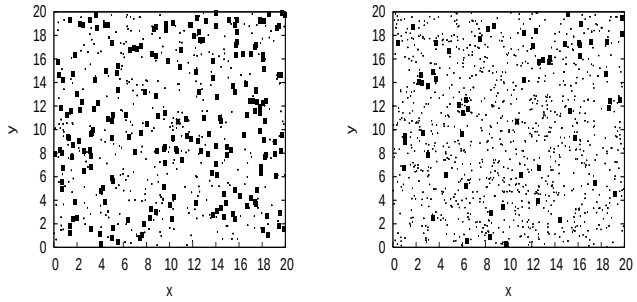
5. ADAPTING THE TOPOLOGY

As mentioned in Section 3.2, the implicit acknowledgments provided by ODI synopses can readily be exploited to infer when to retransmit a synopsis or to adapt the routing topology. Using retransmissions expends energy and it delays query responses because each level in a topology may need to wait for possible retransmissions before preceding. Thus, we will focus on *adapting* the topology when message loss is frequent, hoping that it will reduce loss rate in the long run. In this section, we show how to modify the Rings topology described in Section 2.1 to construct a more robust topology, which we call *Adaptive Rings*. The basic idea is to let each node decide whether it has sufficiently good connectivity with its current parents, and if not (*e.g.,* due to node failures or long term changes in link loss rates), assign itself to a different ring so that it gets a new set of parents. To be assigned to a new ring i, a node simply starts waking up during the time slot assigned for the i'th ring in the epoch. This will automatically give it children in ring $i+1$ and parents in ring $i-1$; no coordination is required to change rings (more details in [Nath 2005]).

The Adaptive Rings topology decides when and how to adapt the ring assignments of the nodes as follows. A node x in the ring i uses implicit acknowledgements to keep track of n_{i-1}, the number of times the transmissions from any node in ring $i-1$ has effectively included x's synopses in the last k (an application-defined parameter) epochs. When n_{i-1} is below some threshold, x tries to assign itself to a new ring. To do that, it computes n_j, the number of times it overhears the transmissions of any nodes in a nearby ring j for the last k epochs. Since nodes in different rings transmit at different time slots of an epoch, x can compute n_j by listening during the appropriate time slot. The node x in ring i then uses the following heuristics:

(1) Assign itself to ring $i+1$ with probability p if (i) $n_i > n_{i-1}$, and (ii) $n_{i+1} > n_{i-1}$ and $n_{i+2} > n_i$.
(2) Assign itself to ring $i-1$ with probability p if (i) $n_{i-2} > n_{i-1}$ and (ii) $n_{i-1} > n_{i+1}$ and $n_{i-2} > n_i$.

Intuitively, the heuristics try to assign x to a ring so that it can have a good number of parent nodes from the neighboring ring to forward its synopses toward the base station at ring 0. For example, consider the first heuristic above. Condition (i) ensures that x will now have parents with better connectivity after switching rings, and condition (ii) hints that higher rings have smaller loss rates than lower rings and hence switching to a higher ring is probably good. Although, condition (ii) makes the switching decision conservative, our experience shows that it is effective in avoiding repeated switching between rings. The probabilistic nature of the heuristics avoids synchronous ring transition of the nodes and provides better stability of the topology. In our evaluation in Section 6, we use $k = 10$ and $p = 0.5$.



(a) Rings (b) Adaptive Rings

Fig. 4. Random placement of 600 sensors in a 20 ft $\times 20$ ft area and their simulated activity with a realistic communication model. The querying node is at the center of the deployment area. The small dots (solid squares) indicate the nodes accounted for (not accounted for, respectively) in the answer computed in a typical epoch.

ODI synopses play two key roles in this adaptation. First, implicit acknowledgments are used to maintain n_{i-1}, providing link quality estimates without the overheads of acknowledgment messages. Second, nodes can change rings at any time with minimal or no concern that values will be lost or double-counted.

We make another change to increase the robustness of Adaptive Rings. Because the nodes in ring 1 have only one node (the querying node) receiving their transmissions, ring 1's transmissions are more susceptible to transmission losses. In other words, ring 1 can not use multi-path routing. To cope with this, we suggest (1) using multiple querying nodes (in ring 0) who form a mesh and combine the aggregated value at the end of each epoch, or (2) making nodes in ring 1 transmit multiple times if the implicit acknowledgment from the querying node (which broadcasts the final synopsis at the end of each epoch) implies that it has not received a synopsis. The latter approach, although slightly more power consuming, uses the traditional model of having a single querying node; we use this approach in our evaluation (where each node in ring 1 transmits twice).

Figure 4 shows the effectiveness of the adaptation with a snapshot (from the querying node's point of view) of a single epoch. It graphically shows that the percent contributing with Rings (Figure 4(a)), which is already significantly higher than in a tree-based scheme, can be further improved by having ring 1 nodes transmit twice (Figure 4(b)). (The effectiveness of Adaptive Rings' heuristic for topology adaptation is highlighted in Section 6.6.)

6. EVALUATION

In this section, we evaluate our synopsis diffusion scheme and compare it with existing schemes through simulation. We present the accuracy of a few synopsis diffusion algorithms running over the Adaptive Rings scheme and show the sensitivity of Adaptive Rings to different network parameters (*e.g.,* loss rate, node failures, node density, node mobility).

Polastre *et al.* [Polastre et al. 2005] have recently implemented our synopsis diffusion scheme on top of the sensornet protocol (SP) and evaluated it on several platforms including *mica2* and *telos*. Their evaluation shows that synopsis diffu-

sion, when implemented on SP, has very small code size and memory footprint. Moreover, its flexibility in routing makes it easy to piggy-back synopses with other (non-aggregation) messages, providing further energy savings. These results complement the results we present here that study the energy consumption and accuracy of various aggregates.

6.1 Methodology

Topology. To evaluate the performance of synopsis diffusion and different aggregation topologies, we implemented the algorithms within the TAG simulator used in [Madden et al. 2002]. In our simulations unless otherwise noted, we collect a `sum` aggregate on a deployment of 600 sensors placed randomly in a 20 ft $\times$ 20 ft area. The querying node is at the center of the deployment area. Unless otherwise stated, sensors report their node-ids, which are assigned sequentially from 1 to 600, as their sensor readings.

Aggregation Schemes. We simulate five different aggregation schemes: TAG (TAG's standard tree-based approach), TAG2 (the TAG approach with value-splitting among two parents)[10], Gossip (gossip-based aggregation described in [Kempe et al. 2003]), Rings (the synopsis diffusion (SD) algorithm over the Rings topology), Adaptive Rings (SD over the scheme described in Section 5, called `A.Rings` in the graphs) and Flood. Flood uses SD over a flat topology—at the beginning of each epoch, each node broadcasts its synopsis to all of its neighbors, and at the end of each epoch, each node updates its own synopsis by applying $SF()$ on the synopses received from its neighbors. To ensure that all nodes contribute to the synopsis at the querying node, Flood runs for $D + 1$ epochs, where D is the maximum distance of any node of the network from the querying node.

In each simulation, we collect results over 500 epochs – we collect a single aggregate value each epoch. We begin data collection only after the underlying aggregation topologies for both synopsis diffusion and TAG are stable.

Message size. We use 48-byte messages, as used by the TinyDB system. Each `sum` synopsis bit-vector uses 32 bits. However, in transmitting multiple bit-vectors, we reduce the size of the synopsis by interleaving the bit-vectors and applying run-length encoding [Palmer et al. 2002]. In our experiments for computing `sum`, we use twenty 32-bit synopses that when compressed take around 14 bytes on average. Two sets of `sum` synopses (or one set of `average` synopses that computes both the `sum` and the `count`) fit in a single TinyDB packet along with headers and extra room to handle the variation in the compression ratio.

Transmission model. The TAG simulator supports a realistic transmission loss model based on measurements of the wireless network interfaces in the Berkeley MICA motes. This loss model, described in [Madden et al. 2002], assigns loss probability of links based on the distance between the transmitter and receiver as follows: the loss probabilities are 0.05, 0.24, 0.4, 0.57, 0.92, and 0.983 within the range 1, 2, 3, 4, 5, and 6 ft respectively, and 1.0 outside the range of 6 ft.[11] Note

[10] We do not consider TAG-k where each node splits its value to $k > 2$ parents, since our experimental results, described later, show that TAG-k has the same average error as TAG. This is also formally shown in [Madden et al. 2002].

[11] Such a high loss rate is common in practice [Zhao and Govindan 2003; Zhao et al. 2003].

Scheme	% nodes	Error(Uniform)	Error(Skewed)	Error(Gaussian)
TAG	$< 15\%$	0.87	0.99	0.94
TAG2	N/A	0.85	0.98	0.92
Gossip	N/A	0.91	0.99	0.93
Rings	65%	0.33	0.19	0.21
Adapt. Rings	95%	0.15	0.16	0.15
Flood	$\approx 100\%$	0.13	0.13	0.13

Table III. Comparison of aggregation schemes

that these are message level loss probabilities; the simulator does not model bit-level loss probability as TOSSIM [Levis et al. 2003] does. Like many real systems, we do not assume link level retransmission or any reliable communication mechanism.

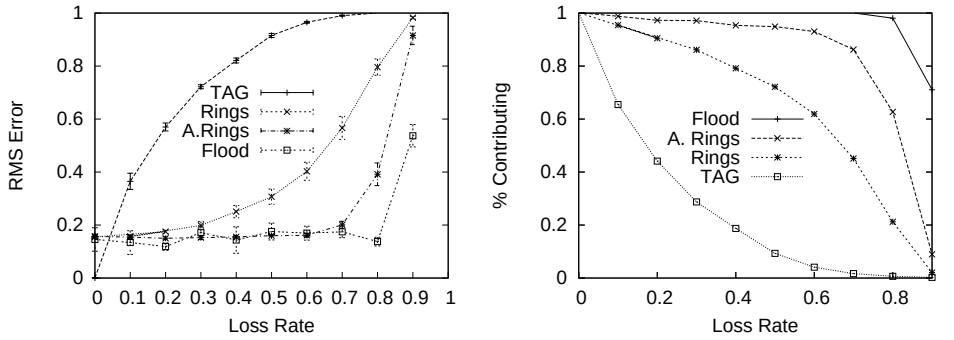
Accuracy. To quantify the performance of the schemes, we use the relative root mean square (RMS) error—defined as $\frac{1}{V}\sqrt{\sum_{t=1}^{T}(V_t - V)^2/T}$, where V is the actual value and V_t is the aggregate computed at time t. The closer this value is to zero the closer the aggregate is to the actual value.

Power consumption. There are two main sources of power consumption on the sensor hardware: computation and communication. To enable our code to execute on actual sensor hardware, we have implemented the synopsis diffusion algorithm for computing `sum` and some other aggregates within the TinyOS and the TinyDB environment. By analyzing the binary code compiled by TinyOS and using the data-sheet of the mote hardware [Atmel AVR Microcontroller Datasheet 2004], we found that our code uses at most a few hundred additional CPU cycles in comparison to the TAG implementation. This difference was insignificant in both the overall power budget as well as in the relative communication power consumption of the different schemes.[12] Therefore, we choose to simply use the network communication power consumption to compare the performance of different schemes. We model the communication power consumption according to the real measurement numbers reported in [Madden et al. 2003].

6.2 Comparison of Aggregation Schemes

Table III shows how different schemes perform in computing `sum` with a random node placement and the realistic network loss model described above. We consider three different distributions of data reported by sensors and the last three columns of the table show the average RMS errors of the computed aggregates for these different distributions. Column 3 considers a scenario where each sensor reports its node-id as its value. Thus, each value between 1 and 600 occurs exactly once, in a randomly placed node. Column 4 considers a scenario where each sensor reports a value inversely proportional to the square of its distance from the querying node at the center, emulating the intensity readings of a radiation source at the center (as in Figure 1). Finally, in Column 5, sensor data are distributed according to a Gaussian distribution with mean 600 and standard deviation 200. At a high level, the table shows that both TAG and TAG2 incur large RMS error because

[12] Measurements [Madden et al. 2003] indicate that 1 bit of transmission (or reception) is equivalent to approximately 1000 cycles of computation.



(a) RMS error (b) % values included

Fig. 5. Impact of packet loss on aggregation schemes

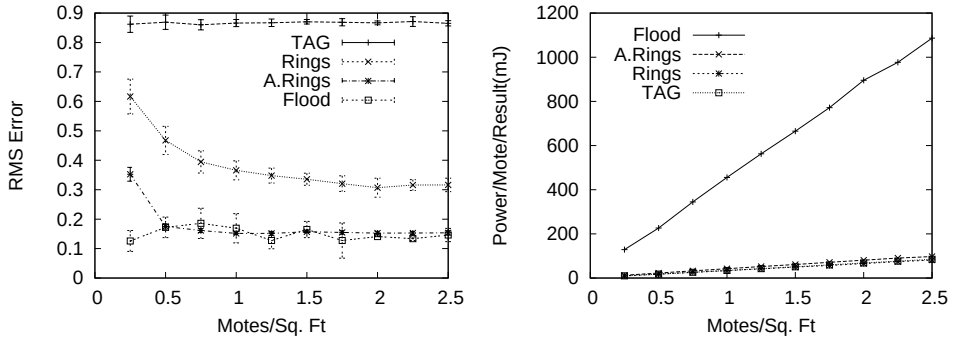
only a small fraction of the nodes report to the querying node.[13] RINGS, which is as energy efficient as TAG and TAG2, is much more robust than these two. Note that GOSSIP performs poorly too. This happens because under unreliable communication GOSSIP loses its "mass" and results in values smaller than actuals. It also shows that the performance of ADAPTIVE RINGS is significantly better than RINGS and is very close to FLOOD under this realistic setup. Note that the errors in FLOOD come from only the approximation algorithm.

The effectiveness of synopsis diffusion, as shown above, comes from its *replicating* aggregate information via broadcast, which makes it extremely robust under communication failures. With synopsis diffusion, a node sends the *same* information to multiple neighbors and it is sufficient if only one of the neighbors receive the information. In contrast, with techniques like TAG2 and GOSSIP, a node *distributes* its aggregate information and send partial information through multiple neighbors. Even when communication fails with only one of the neighbors, partial information sent to it is lost, resulting in inaccurate answer.

Note that the poor performance of TAG and GOSSIP comes due to unreliable communication. As pointed out in Section 3.3, there are two sources of error: communication error (reducing the percent contributing) and approximation error (from using synopses). Because TAG and GOSSIP do not incur this latter source of error, they can be more accurate than ADAPTIVE RINGS whenever the communication error is very small (e.g., due to reliable communication) or lost messages have little impact (e.g., when computing Average over non-skewed data). However, such scenarios are not the common case.

Because both TAG and TAG2 provide similar average RMS errors, we report only the performance of TAG in the rest of the experiments. We also omit GOSSIP because of its high error under unreliable communication. Note that some gossip-based aggregation protocols [Gupta et al. 2001] are more robust to communication failures, but they require additional mechanisms (as discussed in Section 1.1) and therefore do not present fair comparison points with Synopsis Diffusion.

[13]This is consistent with the theoretical and experimental results reported in [Madden et al. 2002]



(a) RMS error (b) Power consumption

Fig. 6. Impact of sensor density on aggregation schemes

6.3 Effect of Communication Losses

In this set of experiments, we use a simpler loss model in which each packet is dropped with a fixed probability. Figure 5(a) shows the impact of changing this loss probability on the accuracy of the different schemes. For each loss rate, the plot shows the average RMS error as well as the 95% confidence interval across all the trials. Even with loss rates as low as 10%, the RMS error for TAG is 0.36, whereas the RMS errors for Rings, Adaptive Rings, and Flood are only around 0.15. More importantly, Adaptive Rings perform as well as Flood even when the loss rate is as high as 60%.[14] We also note that the performance of TAG degrades much more quickly with increasing loss rate than any of the synopsis diffusion approaches. From Figure 5(b), we can see that this degradation is directly related to the fact that the readings of fewer and fewer nodes are incorporated into the reported aggregate. In addition, we can see that the impact of excluding sensor nodes dominates the impact of any approximation errors.

6.4 Effect of Deployment Densities

The density of sensors in a deployed region influences the loss rates observed as well as the topology used to aggregate the sensor readings. To evaluate the impact of sensor density, we vary the number of sensors within a fixed deployment region. We employ the realistic packet loss model described earlier.

Figure 6(a) shows the impact of changes in density on the accuracy of TAG, Rings, Adaptive Rings and Flood. As the network becomes sparser, the aggregation schemes are forced to use longer, more error-prone links. This has little impact on Flood, which has a high degree of redundancy in its data collection. Rings and Adaptive Rings, having limited redundancy compared to Flood, perform worse with very low sensor density. However, in reasonably dense networks, Adaptive Rings performs as well as Flood due to the large amount of redundancy it can take advantage of. Sparse networks surprisingly also have little impact on TAG. TAG prefers to construct short trees because deep trees combined with

[14] At high loss rate, Flood fails to provide 100% contributing nodes because the flood runs for only a limited number of epochs.

packet losses result in very poor performance. As a result, the average parent-child link distance does not change significantly with density. This results in a similar percentage of sensors readings being omitted from the aggregate and, therefore, similar error performance regardless of density.

The added redundancy of FLOOD and ADAPTIVE RINGS comes at a cost in terms of overhead. Figure 6(b) plots the impact of density on communication power consumption (the breakdown of the power consumed to transmit and receive messages can be found in [Nath 2005]). Because the nodes in TAG and RINGS remain awake for receiving messages for roughly the same amount of time [Madden et al. 2002], and roughly the same number of transmissions occur in both schemes, the nodes' network interfaces in both schemes receive approximately the same number of messages. Thus, both TAG and RINGS have the optimal overhead for transmission power. ADAPTIVE RINGS consumes slightly more transmission energy due to the use of redundant transmissions in ring 1 (see Section 5) and the reception of the implicit acknowledgment. Note that, however, the RINGS and ADAPTIVE RINGS approach force each node to process all of the received packets, in contrast to a TAG node processing a smaller subset of these message per epoch. Fortunately, the cost of processing a message is far less than receiving the message. Finally, as expected, FLOOD has the highest overhead for transmission and reception among the schemes.

In addition to density, the rough *shape* of a sensor deployment can affect the performance of the different aggregation schemes. We have also performed experiments evaluating the impact of deployment shape. Specifically, we varied the width and height of the rectangular deployment area while keeping the size and the number of sensors constant. Our results show that while the performance of TAG degrades as the diameter of the network increases (*i.e.,* the height of the tree increases), the performance of RINGS degrades only slightly. The details can be found in [Nath 2005].

6.5 Effect of Asymmetric Links

Asymmetric links are common in real wireless sensor networks and cause significant problems for topology creation. The problems arise from the fact that if node u_1 hears from node u_2, it may choose node u_2 to be its parent. However, with asymmetric links, there is no guarantee that node u_2 hears messages from node u_1. To see the impact of this factor, we model asymmetric links in our simulation based on realistic measurements [Zhao and Govindan 2003]. With such links, the accuracy of TAG drops by 15% and Rings by 10%. The implicit acknowledgments of synopsis diffusion help avoid this problem by identifying asymmetric links. As a result, the performance of Adaptive Rings degrades only slightly ($< 3\%$) with such links.

6.6 Effect of Correlated Node Failures

Figure 7 shows the effectiveness of ADAPTIVE RINGS using a scenario where at time $t = 300$, we disable all the sensors within a 6 *ft* $\times$ 8 *ft* rectangular region of the 20 *ft* $\times$ 20 *ft* deployment area, which causes a loss of 13% of the total sensors. To separate out the effects of two key components, nodes in ring 1 transmitting twice and all nodes adapting their rings to cope with the network dynamics, we

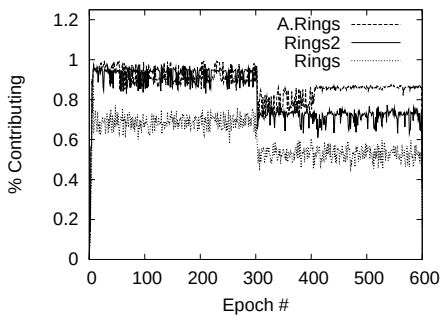


Fig. 7. Correlated node failures

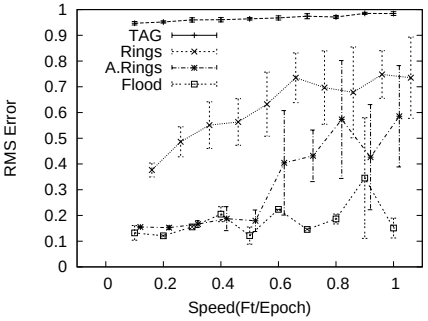


Fig. 8. Node mobility

compare Adaptive Rings with a scheme called Rings2. Rings2 is basically the Rings scheme with the nodes in ring 1 sending their synopses twice (*i.e.*, Adaptive Rings without the topology adaptation).

As the graph shows, Adaptive Rings performs better (with higher % nodes and lower variance) than the other schemes even when there is no drastic network dynamics (i.e., $t < 300$). Rings2 performs better than Rings showing the effectiveness of having the nodes in ring 1 send twice. Immediately after $t = 300$, all the schemes suffer because the dead sensors break all the paths to the querying node from a significant portion of the live sensors. However, Adaptive Rings gradually adapts its routing around the dead sensors and, thus, lets almost all the live sensors communicate again with the querying node. In contrast, in Rings2, 12% of the nodes who could contribute to the computed aggregate before $t = 300$ fail to do so after $t = 300$. The convergence time of Adaptive Rings after $t = 300$ depends on the parameters of the adaptation heuristic. This result shows the contributions of both the ring adaptation and ring 1's retransmissions to the robustness of the Adaptive Rings scheme.

We have observed a similar result in scenarios where a large number of *randomly chosen* sensors fail within a short period of time (details are in [Nath 2005]).

6.7 Effect of Mobile Sensors

Sensors may be mobile for a number of reasons. They may be deployed on mobile objects (*e.g., Robots*), or they may be moved passively by the environment (*e.g.,* by wind or water currents). Mobility can cause a number of challenges, including: 1) the same sensor transmitting its readings from multiple locations (creating duplicate messages), and 2) sensor movement changing the connectivity of the network. Due to synopsis diffusion's resilience to losses, duplicate messages and connectivity changes, it is able to handle mobility much more easily than approaches like TAG.

Figure 8 shows the impact of mobility, depicting RMS error as a function of sensor velocity. For a given experiment with velocity x feet/epoch, each sensor picks a random direction of motion at each epoch and moves x feet in that direction. Nodes check for possible adaptation on every 4th epoch.

Because TAG relies on the continued existence of the links that form the aggregation tree, it must repair the aggregation tree whenever sensor mobility removes one of these key links. In TAG, whenever a node is disconnected from its parent,

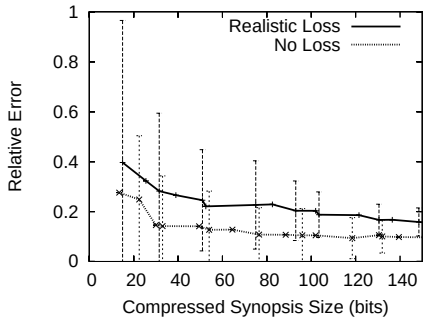


Fig. 9. Synopsis size

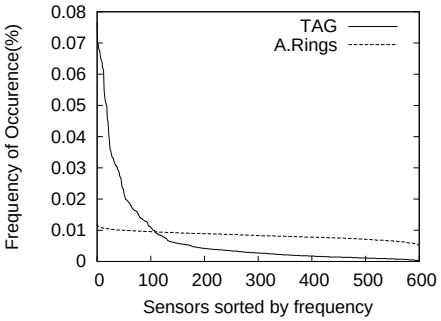


Fig. 10. Uniform sample

it connects to the next node that it hears from. In addition, to prevent loops, the disconnected node also disconnects from all its children. This essentially forces the entire disconnected subtree to be recreated. As a result, TAG's performance degrades with higher rates of mobility, as seen in Figure 8.

The resilience of synopsis diffusion to connectivity changes depends closely on the type of propagation used. For example, FLOOD uses no history of past connectivity to collect results. As a result, changes in connectivity should have little effect on the behavior of the system or its performance. Note that Figure 8 does indicate some performance degradation. We suspect that this is a result of the diameter of the network changing as a result of mobility – preventing the flood from completing. ADAPTIVE RINGS matches FLOOD at low speeds, but its adaption is not able to keep up at high speeds. Nevertheless, in all cases it outperforms RINGS.

6.8 Effect of Synopsis Size

Synopsis diffusion provides the opportunity to select a desired approximation accuracy based on the affordable energy overhead (as determined by the message size). For example, in the approximate `sum` algorithm a larger synopsis enables additional independent bit-vectors to be used, reducing the approximation error.

To see how the relative error of synopsis diffusion changes with the size of the synopsis, we increase the number of bit-vectors in the `sum` synopsis (and hence the total number of bits in the compressed synopsis). Figure 9 shows the average of the relative errors of the final answer for the realistic loss rate and for no loss rate. The x-axis of the graph shows the number of bits of the compressed bit-vectors (we increase the number of bit-vectors by four and report the length of the compressed synopsis, thus the use of 20 bit-vectors in our other simulations corresponds to the use of around 100 bits). The graph also shows the 95% confidence interval of the computed answers. For clarity, such intervals are shown only for every other point in the plot. The graph shows that both the average approximation error and the confidence interval can be decreased significantly by using more bits (*i.e.,* more bit-vectors) in the synopsis.

6.9 Beyond Sum

Uniform Sample. Figure 10 compares the sampling algorithm described in Section 4 running over ADAPTIVE RINGS with an existing random sampling algorithm

known as RanSub [Kostic et al. 2003] running over TAG. The algorithms compute a sample of size 5, and the graph shows the histograms of the node ids included in 10,000 samples. Note that RanSub must be run over a tree topology because its synopsis is not ODI. Moreover, both RanSub and our sampling algorithm provide a uniform sample when there is no message loss. However, with a realistic loss model, RanSub with TAG provides a distribution far from uniform. Synopsis diffusion algorithm, using ADAPTIVE RINGS, approximates a uniform distribution much better than RanSub.

Top-k. We have also simulated the synopsis diffusion algorithm to find the 5 most frequent values in the network, where the value of a sensor is the integer part of its distance from the querying node (this creates a slightly skewed distribution of the popularity of the data). We use 10 synopses from which $SE()$ estimates the 5 most popular items. We quantify the accuracy of our estimation $\{x_1, \ldots, x_k\}$ by using the metric *relative rank-error* (RRE) $= \frac{1}{k}\sum_{i=1}^{k}(|i - r_i|)$, where r_i is the actual rank of x_i in the descending order of frequency of all the unique items. With the realistic loss model and a random placement of the sensors, our algorithm provides very small (≈ 0.6) relative rank-error.

6.10 Discussion

Our results have quantified a number of advantages that synopsis diffusion provides over tree-based aggregation schemes. First, we have shown how synopsis diffusion reduces answer errors in lossy environments. Second, we have shown how synopsis diffusion helps address the challenges imposed by node failures. Finally, we have shown that synopsis diffusion can achieve these gains without a significant increase in power consumption.

While our measurements have shown that synopsis diffusion is preferable to tree-based approaches, they may not have made the choice of aggregation topology as clear. Our comparisons show that the ADAPTIVE RINGS topology, made possible by implicit acknowledgments, incurs approximately the same overhead as the RINGS topology while providing much better accuracy/robustness. ADAPTIVE RINGS is especially superior in the face of mobility and node failures. The trade-offs between ADAPTIVE RINGS and FLOOD are more subtle. ADAPTIVE RINGS collects about 90% of the sensor readings in most reasonable settings while FLOOD collects 100%. However, in practice, one might deploy extra sensors to compensate for the lost readings and to decrease their number. The significantly lower power consumption of ADAPTIVE RINGS would significantly reduce the frequency of sensors replacement. In situations where deployments are short-lived, every sensor reading is critical, sensors are sparsely deployed, or network conditions fluctuate dramatically, FLOOD may be an appropriate choice. Otherwise, ADAPTIVE RINGS provides a much better set of trade-offs.

7. RELATED WORK

We discuss related work in two different areas. First we discuss general in-network aggregation and robustness techniques. Then we place synopsis diffusion in the context of previous streaming scenarios.

7.1 In-network Aggregation and Robustness

The early approaches for in-network aggregation, including Cougar [Bonnet et al. 2001], directed diffusion [Intanagonwiwat et al. 2000], and TAG [Madden et al. 2002; 2003; Madden et al. 2002], use a tree topology with unreliable communication, and hence are not robust against node and link failures. Note that the SG and SF functions of a synopsis diffusion algorithm may be implemented as filters within these approaches.

We now describe several techniques to make the aggregation process more robust.

Reliable Communication. One approach to robust aggregation is to use reliable communication such as RMST (Reliable Multi-segment Transport) [Stann and Heidemann 2003] and PSFQ (Pump Slowly Fetch Quickly) [Wan et al. 2004]). These protocols can mask transient message losses, making individual links of the aggregation topology more reliable. For example, RMST has been used over directed diffusion for guaranteed delivery of data, even under high loss rates. However, reliable communication protocols use extra messages (*e.g.*, acknowledgements, retransmissions) that have high overhead in terms of energy consumption, channel utilization, and latency [Stann and Heidemann 2003].

Robust Topology. Another approach to robust aggregation is to use an aggregation topology more robust than a tree. Gossip-based aggregation [Boyd et al. 2005; Chen and Pandurangan 2005; Dimakis et al. 2006; Kempe et al. 2003; Gupta et al. 2001] also uses a robust aggregation topology. In the approach proposed by Kempe et al., and in its variants [Boyd et al. 2005; Chen and Pandurangan 2005; Dimakis et al. 2006], each node starts with an initial value, its *mass*, depending on the target aggregate function. The whole aggregation process is loosely synchronized. In the beginning of each round, a node transfers a fraction (*e.g.*, half) of its current mass to a randomly chosen node in the network. At the end of the round, each node combines its current mass with the mass it receives from other nodes in that round. In this way, the total mass in the network is always conserved. It can be shown that, after $O(\log(n))$ rounds, the mass of each node in the network converges to the result of the target aggregate function. Although highly robust, this algorithm has some drawbacks when used in sensor networks. First, to ensure mass conservation, it requires reliable communication; at least, the sender needs to know whether the message has been successfully delivered. Second, it needs $O(n \log(n))$ messages, each of which is to a random other node in the network and hence often needs to be relayed through multiple nodes to reach its destination. Finally, techniques are known to compute only a small number of aggregates (*e.g.*, Sum, Average, and Count); computing more complex aggregates is still an open issue.

A special case of the above mass conservation principle is the *value-splitting* technique used in TAG with the goal of improving robustness [Madden et al. 2002]. The idea is to use a directed acyclic graph (DAG) instead of a tree, and have each node with accumulated value v send v/k to each of its k parents. For aggregates such as Count or Sum, this reduces the error resulting from a single message loss from v to v/k, but the aggregation error remains high (recall Figure 1 and Table III).

Note that there are many extremely robust gossip-based protocols [Jenkins et al. 2001; Karp et al. 2000; Vogels et al. 2003] that do not require reliable communication, but they are designed primarily for information dissemination and can not

be directly used for in-network aggregation. [Gupta et al. 2001] provides a gossip-based aggregation algorithm that does not require reliable communication, but it requires expensive mechanisms such as explicit maintenance of a balanced tree, coarse synchronization (for synchronously switching the nodes between the distinct phases of the algorithm), and (possibly multi-hop) communication between random pairs of nodes.

Model-based Aggregation. Model-based aggregation [Deshpande et al. 2004; Mukhopadhyay et al. 2004] uses temporal correlation of data to correct errors due to transient losses in sensor networks. It uses a model of temporal variation to predict future data. The next observed sensor reading is compared to the predicted value to assess the likelihood that the observed reading is erroneous. If the observed value is assessed to be erroneous, the predicted value is reported instead of the observed value. The model is adjusted over time based on the observed values. The success of this technique depends on the feasibility of building a good model of the data. In many cases where representative sensor data are not available to build an initial model, or when sensors are supposed to report rare events, such techniques are unlikely to work. Deshpande *et al.* proposed techniques to model sensor data to improve the energy-efficiency of data acquisition [Deshpande et al. 2004]. This class of techniques are orthogonal to other previously described techniques, and can be used on top of them to further improve robustness.

Duplicate-insensitive Synopses. Concurrent to our work, Bawa *et al.* [Bawa et al. 2004] and Considine *et al.* [Considine et al. 2004] independently proposed duplicate-insensitive approaches for estimating certain aggregates. In the context of peer-to-peer networks, Bawa *et al.* studied the semantics of aggregates computed while the topology is changing, and present algorithms to achieve a given target semantics. Considine *et al.* is the most closely related work to ours. As mentioned in Section 1, they independently proposed using duplicate-insensitive sketches for robust aggregation in sensor networks and demonstrated the advantages of the Rings topology over previous tree-based approaches. A key result in their paper is an order- and duplicate-insensitive Sum algorithm. Because this algorithm offers superior accuracy over the one we developed (Section 4), we have used it in our experimental study in Section 6. Our work extends [Considine et al. 2004] in a number of important ways: (1) we present the first formal definition of duplicate-insensitive synopses; (2) we prove powerful theorems characterizing ODI synopses and their error guarantees—their paper has no analogous result; (3) we present solutions for a wider range of aggregates; (4) we consider techniques for adaptive rings that reduce message loss; and (5) our simulation results use a more realistic communication loss model, and consider scenarios not addressed in their paper such as correlated node failures.

7.2 Query Processing over Data Streams

The use of synopses in synopsis diffusion is related to that in data stream algorithms. There has been a flurry of recent work in the data stream community devising clever synopses to answer aggregate queries on data streams (see [Babcock et al. 2002; Muthukrishnan 2003] for surveys, and [Cormode et al. 2005; Zhang et al. 2005] for some more recent work). There the goals are to estimate an aggregate with one

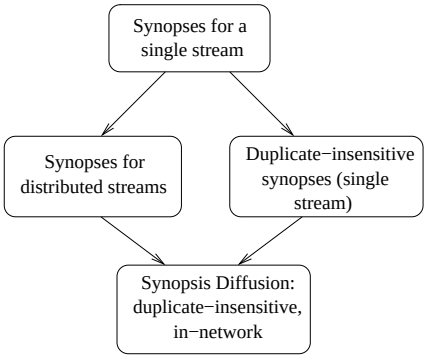


Fig. 11. Hierarchy among synopsis problems. Each edge is directed from an easier problem to a harder problem.

pass through the data and only limited memory. Thus at any point in scanning the data stream, the limited memory data stream synopsis is a small digest suitable for producing a highly-accurate estimate of the stream to that point. The traditional data streams model is (i) centralized , *i.e.*, the synopsis is generated at a single place, and (ii) order-, but not duplicate-, insensitive. Thus, the model is not adequate for the ODI synopses required for synopsis diffusion.

The same synopsis used in traditional data streams can sometimes be used for synopsis diffusion. However, synopsis diffusion introduces two complications beyond traditional data streams. First, the data is not presented as a sequential stream to a single party. Instead, the data is spread among multiple parties and the aggregation must occur in-network. Specifically, the Synopsis Fusion function merges two synopses, not just a current synopsis with a next stream value. More related then is work on distributed streams algorithms [Gibbons and Tirthapura 2001; 2002]. In the distributed streams model, there are multiple parties, each observing a stream and having limited memory, and the goal is to estimate aggregates over the union of these streams by exchanging synopses at query time. This requires the merging of multiple synopses (ala Synopsis Fusion). Second, synopsis diffusion requires duplicate-insensitive synopses. None of the prior work on sequential or distributed data streams was concerned with duplicate sensitivity. (The exception is for aggregates that are by definition duplicate-insensitive, such as Count Distinct.) Only recently, Tao *et al.* [Tao et al. 2004] have used duplicate-insensitive counting in mobile environments.

Figure 11 diagrams a path for developing new synopsis diffusion algorithms, where each edge is directed from an easier problem to a harder problem.

8. CONCLUSIONS

In this paper, we presented *synopsis diffusion*, a general framework for designing energy-efficient, highly-accurate in-network aggregation schemes for wireless sensor networks. Synopsis diffusion enables aggregation algorithms and message routing to be optimized independently, through its use of order- and duplicate-insensitive (ODI) synopses. Our paper is the first to define and study this important class of synopses; previous work only considered isolated examples of such synopses.

We proved the powerful and somewhat surprising result that four easy-to-check properties on the synopsis generation and fusion functions characterize ODI synopses. We showed how many aggregates can be computed in-network using ODI synopses and how ODI synopses provide implicit acknowledgments for network transmissions. Light-weight monitoring of transmissions using such acknowledgments can be exploited to create an adaptive, energy-efficient aggregation topology such as Adaptive Rings. Finally, we provided an extensive performance study on a realistic simulator demonstrating the significant robustness, accuracy, and energy-efficiency improvements achieved by using an ODI-synopsis based approach running on Adaptive Rings.

Our ongoing efforts on synopsis diffusion include understanding the trade-offs between synopsis diffusion and existing tree-based schemes (e.g., approximation errors, message sizes) and developing hybrid aggregation schemes that combine the advantages of the two schemes. Our initial results include a hybrid topology called Tributary-Delta [Manjhi et al. 2005], which runs both synopsis diffusion and tree-based schemes simultaneously in different regions of the network, depending on current loss conditions. We are also developing ODI synopses for additional aggregates. For example, in [Manjhi et al. 2005], we proposed ODI synopsis for computing the frequent items in a network. Finally, our future plans include implementing and evaluating synopsis diffusion on real sensor deployments.

Acknowledgment. We thank John Byers, Amol Deshpande, Joe Hellerstein, Wei Hong, Brad Karp, and Sam Madden for discussions on this research. We also thank the anonymous referees for their helpful suggestions. This research was supported in part by NSF Grant No. ANI-0092678 and funding from Intel Research.

REFERENCES

Alon, N., Matias, Y., and Szegedy, M. 1999. The space complexity of approximating the frequency moments. *J. of Computer and System Sciences 58*, 137–147.

Atmel AVR Microcontroller Datasheet. 2004. http://www.atmel.com/dyn/resources/prod_documents/2467s.pdf.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. 2002. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*. ACM Press, New York, NY, USA, 1–16.

Bar-Yossef, Z., Kumar, R., and Sivakumar, D. 2001. Sampling algorithms: lower bounds and applications. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*. ACM Press, New York, NY, USA, 266–275.

Bawa, M., Gionis, A., Garcia-Molina, H., and Motwani, R. 2004. The price of validity in dynamic networks. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM Press, New York, NY, USA, 515–526.

Bonnet, P., Gehrke, J. E., and Seshadri, P. 2001. Towards sensor database systems. In *IEEE Mobile Data Management*. Springer, Berlin, Germany.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. 2005. Gossip algorithms: Design, analysis, and applications. In *Proceedings of the IEEE INFOCOM*. IEEE Computer Society, Washington, DC, USA.

Chen, J.-Y. and Pandurangan, D. X. G. 2005. Robust aggregates computation in wireless sensor networks: Distributed randomized algorithms and analysis. In *4th International Symposium on Information Processing in Sensor Networks (IPSN)*.

Considine, J., Li, F., Kollios, G., and Byers, J. 2004. Approximate aggregation techniques for sensor databases. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Washington, DC, USA, 449.

Cormode, G., Garofalakis, M., Muthukrishnan, S., and Rastogi, R. 2005. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *Proceedings of the ACM SIGMOD international conference on management of data*. ACM Press, New York, NY, USA.

Cormode, G. and Muthukrishnan, S. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms 55,* 1 (April), 58–75.

Davey, B. A. and Priestley, H. A. 2002. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge.

Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J. M., and Hong, W. 2004. Model-driven data acquisition in sensor networks. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, St. Louis, MO, USA.

Dimakis, A. G., Sarwate, A. D., and Wainwright, M. 2006. Geographic gossip : Efficient aggregation for sensor networks. In *5th International Symposium on Information Processing in Sensor Networks (IPSN)*. Nashville, TN.

Flajolet, P. and Martin, G. N. 1985. Probabilistic counting algorithms for database applications. *Journal of Computer and System Sciences 31*, 182–209.

Ganesan, D., Govindan, R., Shenker, S., and Estrin, D. 2001. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review (M2CR) 5,* 4, 11–25.

Gibbons, P. B. 2007. Distinct-values estimation over data streams. In *Data Stream Management: Processing High-Speed Data Streams*, M. Garofalakis, J. Gehrke, and R. Rastogi, Eds. Springer, New York, NY, USA.

Gibbons, P. B. and Matias, Y. 1999. Synopsis data structures for massive data sets. In *External memory algorithms*. American Mathematical Society, Boston, MA, USA, 39–70.

Gibbons, P. B. and Tirthapura, S. 2001. Estimating simple functions on the union of data streams. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures (SPAA)*. ACM Press, New York, NY, USA, 281–291.

Gibbons, P. B. and Tirthapura, S. 2002. Distributed streams algorithms for sliding windows. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures (SPAA)*. ACM Press, New York, NY, USA, 63–72.

Gupta, I., van Renesse, R., and Birman, K. P. 2001. Scalable fault-tolerant aggregation in large process groups. In *Proceedings of the 2001 International Conference on Dependable Systems and Networks (DSN)*. IEEE Computer Society, Washington, DC, USA, 433–442.

Intanagonwiwat, C., Govindan, R., and Estrin, D. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*. ACM Press, New York, NY, USA, 56–67.

Jenkins, K., Hopkinson, K., and Birman, K. 2001. A gossip protocol for subgroup multicast. In *Proceedings of the 21st International Conference on Distributed Computing Systems*. IEEE Computer Society, Washington, DC, USA.

Johnson, D. B. and Maltz, D. A. 1996. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, Imielinski and Korth, Eds. Vol. 353. Kluwer Academic Publishers, Norwell, MA, USA.

Karp, R., Schindelhauer, C., Shenker, S., and Vocking, B. 2000. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Washington, DC, USA, 565.

Kempe, D., Dobra, A., and Gehrke, J. 2003. Gossip-based computation of aggregate information. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Washington, DC, USA.

Kostic, D., Rodriguez, A., Albrecht, J. R., Bhirud, A., and Vahdat, A. 2003. Using random subsets to build scalable network services. In *USITS*. USENIX, Berkeley, CA, USA.

Levis, P., Lee, N., Welsh, M., and Culler, D. 2003. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 126–137.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2002. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th symposium on Operating systems design and implementation (OSDI)*. ACM Press, New York, NY, USA, 131–146.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2003. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM Press, New York, NY, USA, 491–502.

Madden, S., Szewczyk, R., Franklin, M. J., and Culler, D. 2002. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*. IEEE Computer Society, Washington, DC, USA, 49.

Manjhi, A., Nath, S., and Gibbons, P. B. 2005. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *Proceedings of the ACM SIGMOD international conference on management of data*. ACM Press, New York, NY, USA.

Mukhopadhyay, S., Panigrahi, D., and Dey, S. 2004. Model based error correction for wireless sensor networks. In *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*. IEEE Computer Society, Santa Clara, CA, USA.

Muthukrishnan, S. 2003. Data streams: Algorithms and applications. Tech. rep., Rutgers University, Piscataway, NJ.

Nath, S. 2005. Exploiting redundancy for robust sensing. Ph.D. thesis, Carnegie Mellon University. Tech. rep. CMU-CS-05-166.

Palmer, C. R., Gibbons, P. B., and Faloutsos, C. 2002. ANF: A fast and scalable tool for data mining in massive graphs. In *ACM KDD*. ACM Press, New York, NY, USA.

Polastre, J., Hui, J., Levis, P., Zhao, J., Culler, D., Shenker, S., and Stoica, I. 2005. A unifying link abstraction for wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 76–89.

Przydatek, B., Song, D., and Perrig, A. 2003. SIA: Secure information aggregation in sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*. ACM Press, New York, NY, USA.

Stann, F. and Heidemann, J. 2003. RMST: Reliable data transport in sensor networks. In *Proceedings of 1st IEEE International Workshop on Sensor Net Protocols and Applications (SNPA)*. IEEE Computer Society, Washington, DC, USA.

Tao, Y., Kollios, G., Considine, J., Li, F., and Papadias, D. 2004. Spatio-temporal aggregation using sketches. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Washington, DC, USA, 214.

Vogels, W., van Renesse, R., and Birman, K. 2003. The power of epidemics: robust communication for large-scale distributed systems. *SIGCOMM Comput. Commun. Rev. 33,* 1, 131–135.

Wan, C.-Y., Campbell, A. T., and Krishnamurthy, L. 2004. Reliable transport for sensor networks. In *Wireless Sensor Networks*. Springer, New York, NY, USA, 153–182.

Watson, E. J. 1962. *Mathematics of Computation*. Vol. 16. American Mathematical Society, Providence, RI, USA, 368–369.

Yao, Y. and Gehrke, J. 2003. Query processing in sensor networks. In *Proceedings of the First Conference on Innovative Data Systems Research (CIDR)*. Morgan Kaufmann, St. Louis, MO, USA.

Zhang, R., Koudas, N., Ooi, B. C., and Srivastava, D. 2005. Multiple aggregations over data streams. In *Proceedings of the ACM SIGMOD international conference on management of data*. ACM Press, New York, NY, USA.

Zhao, J. and Govindan, R. 2003. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*. ACM Press, New York, NY, USA.

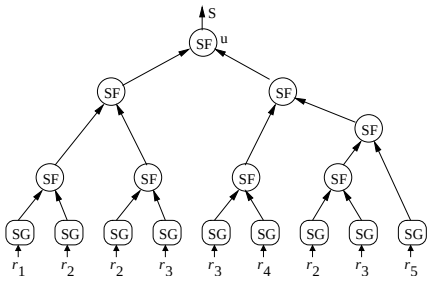


Fig. 12. Graph G_1 used in the proof

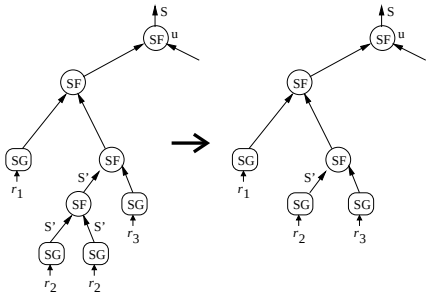


Fig. 13. Merging two duplicate leaves. Only the left half of the tree is shown. Property P3 is used to transform G_1 into the tree on the left. Properties P1 and P4 ensure that the same synopsis s' is computed where shown. Thus, replacing the three nodes with one, as on the right, leaves the synopses computed by the rest of the tree unchanged.

Zhao, J., Govindan, R., and Estrin, D. 2003. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications*. IEEE Computer Society, Washington, DC, USA.

9. APPENDIX

9.1 Proof of Theorem 1

We now present the proof that Properties P1–P4 are necessary and sufficient for ODI-correctness.

P1–P4 are sufficient. Consider an arbitrary execution of synopsis diffusion, producing a synopsis s. Let G be the aggregation DAG corresponding to this execution, and let u be the node in G that outputs s. We will use the DAG in Figure 3(a) as a running example. In this proof, we will perform a series of transformations to G that, by properties P1–P4, will not change the output of u, and yet will result in the canonical left-deep tree in the definition of ODI-correctness (*i.e.,* Figure 3(b) in our running example).

First, let G_1 be the tree rooted at u corresponding to G, resulting from replacing each node in G with outdegree $k > 1$ with k nodes of outdegree 1, replicating the entire subgraph under the original node for each of the k nodes. See Figure 12. This may create many duplicate SF and SG nodes. Also, any node in G without a path to u is discarded (it did not affect the computation of s). G_1 corresponds to

a valid execution because SF is deterministic (so applying it in independent nodes results in the same output, given the same inputs), and likewise $SG(r_i) = SG(r_i)$ is a special case of property P1. Note that there is exactly one leaf in G_1 for each tuple in the synopsis label $SL(s)$.

Second, by properties P2 and P3, we can reorganize G_1 into an equivalent tree G_2 where the leaves of G_2 are sorted by $\Pi_q(\{r\})$ values: leaf $SG(r_i)$ precedes leaf $SG(r_j)$ if and only if $\Pi_q(\{r_i\}) \leq \Pi_q(\{r_j\})$.

Third, for each pair of adjacent leaves $SG(r_i), SG(r_j)$ such that $\Pi_q(\{r_i\}) = \Pi_q(\{r_j\})$, we can reorganize G_2 (by applying P2 and P3) such that they are the two inputs to an SF node. By property P1, both inputs are the same synopsis s', so by property P4, this SF node outputs s'. Replace the three nodes (the SF node and its two leaf children) with either the leaf node $SG(r_i)$ or the leaf node $SG(r_j)$. See Figure 13. Note that either choice of leaf node will produce the same output s'. Repeat until all adjacent leaf nodes are such that $\Pi_q(\{r_i\}) < \Pi_q(\{r_j\})$. Call this G_3. Note that there is exactly one leaf in G_3 for each value in $\Pi_q(SL(s))$.

Finally, reorganize the tree G_3 using P2 and P3 into a left-deep tree G_4 (Figure 3(b)); this is precisely the canonical binary tree. In particular, there is exactly one leaf node in G_4 for each value in $V = \Pi_q(SL(s))$, and the left-deep tree corresponds to the definition of $SG^*(V)$. Because performing the SG and SF functions as indicated by G_4 produces the original output s (*i.e.*, the transformations have not changed the output), the algorithm is ODI-correct.

P1–P4 are necessary. First we observe that by the definition of ODI-correctness, if two synopsis computations for a query q have the same synopsis label, they result in the same synopsis s, namely, $s = SG^*(\Pi_q(SL(s)))$. Now consider arbitrary s_1, s_2 and s_3 in $\mathcal{S}$, and a corresponding aggregation DAG for each. The synopsis label for $SF(s_1, s_2)$ is the same as the synopsis label for $SF(s_2, s_1)$, because the $\uplus$ operator is commutative. Similarly, the synopsis label for $SF(s_1, SF(s_2, s_3))$ is the same as for $SF(SF(s_1, s_2), s_3)$, because the $\uplus$ operator is associative. Thus, properties P2 and P3 follow from the observation. Furthermore, let SL_1 and SL_2 be synopsis labels for two synopsis computations for a query q. Our second observation is that by the definition of ODI-correctness, if SL_2 differs from SL_1 only in that readings may occur more times in SL_2 than in SL_1, the two computations result in the same synopsis. Now consider an arbitrary $s \in \mathcal{S}$ and a corresponding aggregation DAG. The synopsis label for $SF(s, s)$ differs from the synopsis label for s only in that readings occur twice as often. Thus, property P4 follows from our second observation.

Finally, consider an arbitrary r_i and r_j in $\mathcal{R}$ such that $\Pi_q(\{r_i\}) = \Pi_q(\{r_j\}) = \{v_1\}$ for some value v_1. Let $s = SF(SG(r_i), SG(r_j))$. By the definition of ODI-correctness, $s = SG^*(\Pi_q(\{r_i, r_j\})) = SG^*(\{v_1\})$, which equals both $SG(r_i)$ and $SG(r_j)$. Thus, property P1 follows.

9.2 Efficient Generation of Random Bits

Several of the algorithms described in Sections 2.2 and 4 require generating random bits, in particular, the experiment $CT()$ requires tossing a fair coin. We now describe an efficient technique for generating such bits on resource-constrained sensors nodes.

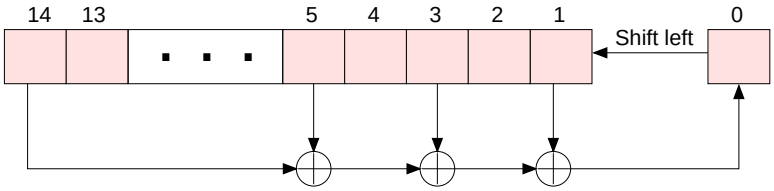


Fig. 14. Generating random bits, based on the primitive polynomial (14, 5, 3, 1, 0), by using a shift register and exclusive-or gates.

```
static unsigned int seed=nodeId;    /* Variable to hold 15 bits.
                                        Can be anything except 0.  */
unsigned int randomBit() {
   unsigned int newBit
   newBit = ((seed >> 13) & 1)
            ^ ((seed >> 4) & 1)     /* Note:^ is exclusive-or */
            ^ ((seed >> 2) & 1)
            ^ (seed & 1);
   seed = (seed << 1) | newBit;
   return newBit;
}
```

Fig. 15. A software implementation (in the C language) to generate random bits based on the primitive polynomial (14, 5, 3, 1, 0).

The solution is based on "primitive polynomials modulo 2," a special class of polynomials with coefficients 0 or 1 [Watson 1962]. Such a polynomial of degree n defines a recurrence relation for obtaining a new random bit from the n preceding bits. The recurrence relation is guaranteed to produce a sequence of maximal length, *i.e.*, cycle through all possible sequences of n bits (except all zeros) before it repeats. Therefore one can seed the sequence with any initial bit pattern (except all zeros), and get $2^n - 1$ random bits before the sequence repeats. For example, consider the primitive polynomial $x^{14} + x^5 + x^3 + x + 1$ (compactly represented by the nonzero powers of x: (14, 5, 3, 1, 0)). Let the bits be numbered from 1 (most recently generated) through n (generated n steps ago), and denoted $a_1, a_2, \ldots, a_n$. Then, the new bit a_0 can be computed by the recurrence formula: $a_0 = a_{14} \oplus a_5 \oplus a_3 \oplus a_1$, where $\oplus$ is the exclusive-or operation. The process can be repeated to generate $(2^{14} - 1)$ random bits before repeating the same sequence. For more examples of such polynomials, see [Watson 1962].

The above algorithm can be efficiently implemented in software and hardware, even in resource-constrained sensor nodes. Figure 14 shows a hardware implementation of the algorithm based on the primitive polynomial (14, 5, 3, 1, 0). It uses a shift register: the contents of selected bits are combined by exclusive-or operations, and the result is shifted in from the right. Figure 15 shows a software implementation of the same function; note that all it uses are bit-wise and, or, xor, and shift operations.