Synopsis Diffusion for Robust Aggregation in Sensor Networks

Suman Nath[†,*] Phillip B. Gibbons[*] Srinivasan Seshan[†] Zachary R. Anderson[†]

[*]Intel Research Pittsburgh [†]Carnegie Mellon University
Pittsburgh, PA 15213, USA
{sknath, srini, zra}@cs.cmu.edu, phillip.b.gibbons@intel.com

ABSTRACT

Previous approaches for computing duplicate-sensitive aggregates in sensor networks (*e.g.*, in TAG) 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. We show that this property can, in turn, enable the system to adapt message routing 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

Keywords

Sensor Networks, Synopsis Diffusion, Query Processing

1. INTRODUCTION

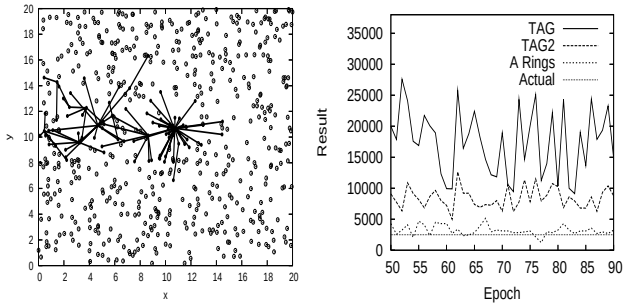
In a large sensor network, aggregation queries often assume greater importance than individual sensor readings. Previous studies [19, 29] have shown that computing aggregates *in-network*, *i.e.*, combining partial results at the intermediate nodes during message routing, significantly reduces the amount of communication and, hence, the energy consumed. A popular approach used by sensor database systems [20, 27] 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 awaiting messages from all its children before sending a new partial result to its parent.

However, aggregating along a tree is very susceptible to node and transmission failures, which are common in sensor networks [19, 28, 29]. Because each of these failures loses an entire subtree of readings, a large fraction of the readings are typically unaccounted for in a spanning tree based system (Figure 1(a)). This introduces significant error in the query answer [6, 19, 29]. Efforts to reduce losses by retransmitting packets waste significant energy and delay query responses [19]. An improvement proposed for TAG [19] was to use a DAG instead of a tree, with each node with accumulated value v sending v/k to each of its k parents. 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 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.

The fundamental stumbling block is that in these common solutions, aggregation and the required routing topology are *tightly coupled*, and, therefore, it is not possible to use arbitrarily robust routing, like multi-path routing. Multi-path routing often results in message duplication, which would cause an overcounting of a large fraction of the readings. For example, if an individual reading or a partial sum is sent along four paths (to improve the likelihood that at least one path succeeds), and three of them happen to succeed, that value or partial sum will contribute to the total sum three times instead of once.

In this paper, we present *synopsis diffusion*, a general framework for combining multi-path routing schemes with clever algorithms to avoid double-counting. By decoupling aggregation from message routing, synopsis diffusion enables the use of arbitrary multi-path routing schemes and

SenSys'04, November 3–5, 2004, Baltimore, Maryland, USA.



(a) Nodes counted in TAG (b) Computing Avg with TAG

Figure 1: (a) Random placement of the sensors in a 20×20 grid and their activity with a realistic communication model during a typical epoch. The lines show the paths taken by sensor readings that have reached the querying node at the center. (b) The average value computed with different aggregation schemes by the querying node at different 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.

allows the level of redundancy in message routing (as a trade-off with energy consumption) to be adapted to sensor network conditions. As a result, highly accurate and reliable answers can be obtained using roughly the same energy consumption as with tree-based schemes.

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 [1, 3, 5, 6, 8, 22, 26] 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 at the node, 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 enable 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 computed using ODI synopses. For example, we provide provably accurate answers to Median, a *holistic aggregate* [13] that was considered not amenable to in-network aggregation [19].
- *Performance Evaluation.* We present an extensive performance study on a realistic simulator (the TAG system simulator) demonstrating the significant robustness and accuracy improvements achieved by synopsis diffusion for roughly the same power consumption as tree-based approaches.

Concurrent with our work, Considine *et al.* [6] 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, 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.

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., [9]) 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 [19], 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 [3, 10], 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.

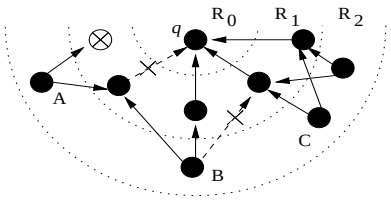


Figure 2: Synopsis diffusion over the Rings topology. Crossed arrows and circles represent failed links and nodes.

- **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 [20, 27]. 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 [19], 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).

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., [19, 20, 21, 29]). 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 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

[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}.

illustrate such functions using the following algorithm for Count.

Count. This algorithm counts the approximate total number of 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) [8] for counting distinct elements in a multi-set. It is a well-known algorithm for duplicate-insensitive approximate Count [5, 6, 25]. 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. Note that $CT()$ simulates the behavior of the exponential hash function that is used in FM:

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

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 $2^{i-1}/0.77351$ [8].

In Section 3, we prove that this algorithm is order- and duplicate-insensitive and that the approximation error guarantees of [8] hold for the algorithm. Intuitively, the number of sensor nodes is proportional to 2^{i-1} because if no node sets the i'th bit, then by (1) there are probably less than 2^i nodes. 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 $SE()$ [8].

Considine *et al.* [6] recently showed how to extend this algorithm to the Sum aggregate, in an energy-efficient manner, by devising a suitable ODI synopsis. We will use their Sum algorithm in our Sum experiments in Section 6. 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 by defining, in the next section, some of the key terms used in our formal framework.

[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.

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). Because of the meta-data, sensor readings are assumed to be unique (*e.g.*, there is only one reading corresponding to a given sensor id and timestamp).

We define a *synopsis label* function $SL()$, which computes the label of a synopsis. The *label* of a synopsis s is defined as the set consisting of all sensor readings contributing to s. More formally, $SL()$ is defined inductively, 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 $SG()$:

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

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, $\{a, b, c, c\} \uplus \{b, c, d\} = \{a, b, b, c, c, c, d\}$. Note that $SL()$ is determined by the sensor readings and the applications of $SG()$ and $SF()$—it is independent of the particulars of $SG()$ and $SF()$. Note also 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.

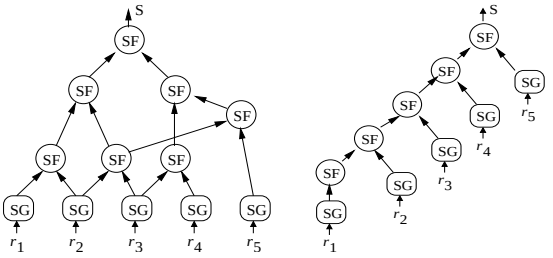
The notion of what constitutes a "duplicate" may vary from query to query, *e.g.*, a query computing the number of sensors with temperature above 50°F considers two readings from the same sensor as duplicates, whereas a query for the number of distinct temperature readings considers any two readings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

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

that converts a multi-set of sensor readings (tuples) to its corresponding 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.

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 syn-



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

Figure 3: Equivalent graphs under ODI-correctness

opses 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: $\forall s \in \mathcal{S} : s = SG^*(V)$, 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 $\Pi_q(\{r_i\}) = v_i$.

Figure 3 helps illustrate ODI-correctness. We can represent the SG() and SF() functions performed to compute a single aggregation result using an *aggregation DAG*, as shown in Figure 3(a). There is a node for each of the different instantiations of the functions $SG()$ (which form the leaf nodes) and $SF()$ (which form the non-leaf nodes). There is an edge $e : f_1 \rightarrow f_2$ iff 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. Corresponding to the aggregation DAG, ODI-correctness defines a canonical left-deep tree (Figure 3(b)). The leaf nodes are the functions $SG()$ on the *distinct* values (in this example, all readings result in distinct values), and the non-leaf nodes are the functions $SF()$. A synopsis diffusion algorithm is ODI-correct if for *any* aggregation 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 chose 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

We believe that 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 very important 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:** $\forall s \in \mathcal{S} : SF(s, s) = s$.

Note that property P4 is much weaker than the duplicate-insensitivity property required for ODI-correctness. It only refers 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.[3]

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 the appendix.

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.*

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 $SG(r_1) = SG(r_2)$.[4] 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]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).

[4]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.

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 [7] 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,

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

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 [7].

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 [16] 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 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[5] 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.[6] 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).

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

[6]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').

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 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, 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 [3] 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 (Figure 3(a)) can be reduced to a canonical left-deep tree (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.

CLAIM 2. *The Count algorithm in Section 2.2 has the same approximation error guarantees as Flajolet-Martin's (FM) distinct count algorithm [8].*

Proof: Follows from Theorem 2.

4. ADDITIONAL EXAMPLES

In this section, we present examples of ODI-correct synopsis diffusion algorithms to show the generality of the framework. Because of page limitations, we only sketch the results.

Previous results. Maximum and Minimum are trivial. Count and Sum were discussed in Section 2. Average, Standard Deviation, and Second Moment can be computed by applying the Sum algorithm over suitably defined values [6]. Count Distinct can be done using a trivial adaption of Flajolet and Martin's algorithm (FM) [8].

In the remainder of the section, we present new ODI-correct synopsis diffusion algorithms for some additional important aggregates.

Uniform sample of sensor readings. Suppose each node u has a value val_u. This algorithm computes a uniform sample of a given size K of the values occurring in all the nodes in the network. Note that traditional sampling procedures [17] would not produce a uniform sample in the presence of multi-path routing because they are duplicate-sensitive. However, our ODI synopses produce a uniform sample of the contributing nodes (*i.e.*, the nodes with failure-free propagation paths, regardless of whether they are selected for the sample). The components of the algorithm are as follows:

- *Synopsis:* A sample of size K of 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 (val_u, r_u, id_u), 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 (val_i, r_i, id_i) 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.[7]

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 $x_1, x_2, \ldots, x_n$, the k-th Statistical Moment $\mu_k = \frac{1}{n}\sum_{i=1}^{n} x_i^k$ (*e.g.*, μ_1 is the Mean) and the k-th percentile value for $0 < k < 100$ (e.g., $k = 50$ is the Median) can be approximated with ϵ additive error[8] and with probability $1 - \delta$ by using a sample of size $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ [4]. Thus, our random sampling algorithm provides an efficient way to estimate these holistic aggregates.

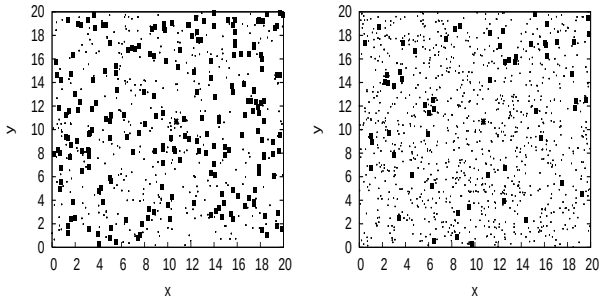
Most Popular Items. The goal of this algorithm is to return the K values that occur the most frequently among all the sensor readings. (When $K = 1$, this is the Mode.) It uses the $CT()$ function used in the Count algorithm described in Section 2.

- *Synopsis:* A set of the K most popular items (estimated).
- $SG()$*:* At node u, output the (value, weight) pair $(val_u, CT(k))$, where $k > log(n)$ and n is an upper bound on the total number of items.
- $SF(s, s')$*:* 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 K pairs with maximum weight. If there are less than K pairs, output them all.
- $SE(s)$*:* Output the set of values in s.

Essentially, the algorithm determines the frequency of an item by running an ODI-correct Count algorithm for each value. The counting is done using the Alon *et al.* variant [1] of Flajolet-Martin's algorithm (FM), which estimates the number of distinct values by keeping track of the highest-order bit that is set to 1. Maintaining the Count for *every* item would result in very large synopses being exchanged. Instead, our algorithm keeps track of the items generating the highest-order bits (and thus, probabilistically, occurring the most frequently in the network). To reduce the number of false positives and false negatives, each synopsis can contain multiple independent sets of popular items and the function $SE()$ can choose the K items that appear

[7]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.

[8]For k-th percentile aggregates, the error is with respect to the rank of the value not its magnitude.



(a) Rings (b) Adaptive Rings

Figure 4: Random placement of the sensors in a 20×20 grid and their activity with a realistic communication model during a typical epoch. The querying node is at the center of the grid. The small dots (solid squares) indicate the nodes accounted for (not accounted for, respectively) in the answer computed in that epoch.

in the greatest number of sets. Details are omitted due to page limitations.

The algorithm can be adapted to approximately answer the iceberg query to find all items occurring above a certain threshold T as follows: $SF()$ retains all the values with $weight \geq log(T)$.

5. ADAPTING THE TOPOLOGY

As mentioned in Section 3.2.2, 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 if 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 Adaptive Rings topology copes with changes in network conditions (*e.g.*, the addition or deletion of sensors, long term changes in link loss rate) by adapting the ring assignments of the nodes, as follows. A node u in ring i keeps track of n_{i-1}, the number of times the transmissions from any node in ring $i-1$ have effectively included u's synopses in the last w epochs.[9] When n_{i-1} is small, u tries to assign itself to a new ring. To do that, it keeps track of n_{i-2}, the number of times its synopses have been included by any node in ring $i-2$ in the last w epochs, and n_j, the number of times it overhears the transmissions of nodes in ring j in the last w epochs, for $j = i, i+1, i+2$. Node u then uses the following heuristic: (1) If $n_{i-1} < n_{i+1}$ and $n_{i-1} < n_i < n_{i+2}$, it assigns itself to ring $i+1$ with probability p, and (2) If $n_{i+1} < n_{i-1}$ and $n_{i+1} < n_i < n_{i-2}$, it assigns itself to ring $i-1$ with probability p. Intuitively, the heuristic tries to assign u to a ring so that it can have a good number of nodes from the neighboring ring to forward its synopses toward the querying node at ring 0. The probabilistic nature of the heuristic avoids synchronous ring transition of the nodes and provides better stability of the topology. In our evaluation in Section 6, we use $w = 10$ and $p = 0.5$.

ODI synopses play two key roles in this adaptation. First, implicit acknowledgment provides an estimation of the quality of the existing links through n_{i-1} and second, it ensures that double counting a value during the adaptation does not hurt.

We make another change to increase the robustness of Adaptive Rings. Because the nodes in ring 1 have only one node receiving the transmission (the querying node), ring 1's transmissions are more susceptible to random transmission losses. 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 significantly higher than with a tree topology, can be significantly improved by having ring 1 nodes transmit twice (Figure 4(b)). (The effectiveness of Adaptive Rings' topology adaptation heuristic is highlighted in Section 6.6.)

6. EVALUATION

In this section, we evaluate our synopsis diffusion scheme and compare it with existing schemes. 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).

6.1 Methodology

Topology. To evaluate the performance of synopsis diffusion and different aggregation topologies, we implement the algorithms within the TAG simulator used in [19]. 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* grid. The querying node is at the center of the grid. 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), 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 ap-

[9] As a trade-off between adaptivity and energy consumption, the frequency of listening for implicit acknowledgments can be reduced from every epoch to every several epochs.

plying $SF()$ 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 [24]. 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 the wireless network interfaces in the Berkeley MICA motes. This realistic loss model, described in [19], 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.[10] Note that these are message level loss probabilities; the simulator does not model bit-level loss probability as TOSSIM [18] does. We do not assume any link level retransmission.

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 [2], 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.[11] 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 [20].

6.2 Realistic Loss Experiments

[10]Such a high loss rate is common in practice [28, 29].

[11]Measurements [20] indicate that 1 bit of transmission (or reception) is equivalent to approximately 1000 cycles of computation.

Scheme	% nodes	Error(Uniform)	Error(Skewed)
TAG	$< 15\%$	0.87	0.99
TAG2	N/A	0.85	0.98
RINGS	65%	0.33	0.19
ADAPT. RINGS	95%	0.15	0.16
FLOOD	$\approx 100\%$	0.13	0.13

Figure 5: Comparison of aggregation schemes

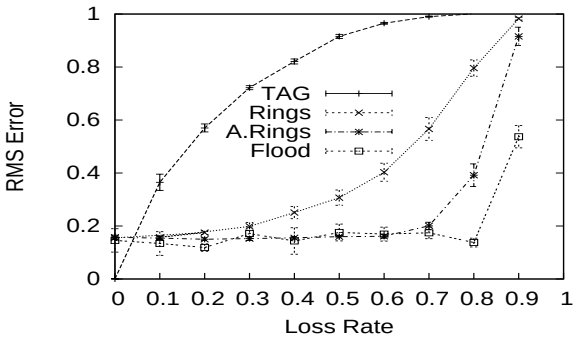


Figure 6: Impact of packet loss (RMS Error)

Figure 5 shows how different schemes perform in computing `sum` with a random node placement and a realistic network loss model. The last two columns show the average RMS errors of the computed aggregates when the sensor data is uniform (column 3) and when is it skewed (column 4) as in Figure 1. At a high level, it shows that both TAG and TAG2 incur large RMS error because only a small fraction of the nodes report to the querying node. Since both TAG and TAG2 provide similar average RMS errors, we report only the performance of TAG in the rest of the experiments. RINGS, which is as energy efficient as TAG and TAG2, is much more robust than these two. 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.

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 6 shows the impact of changing this loss probability on the accuracy of the different schemes. 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%.[12] 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 7, 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.

[12]At high loss rate, FLOOD fails to provide 100% contributing nodes since we allow the flood to run for a limited number of epochs.

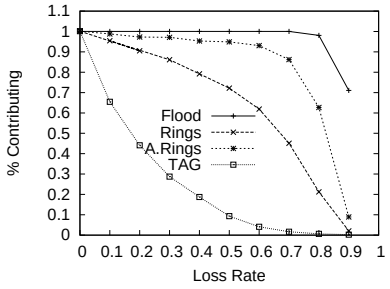


Figure 7: Impact of packet loss (% values included)

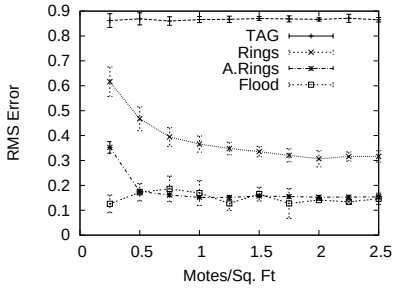


Figure 8: Impact of sensor density (RMS Error)

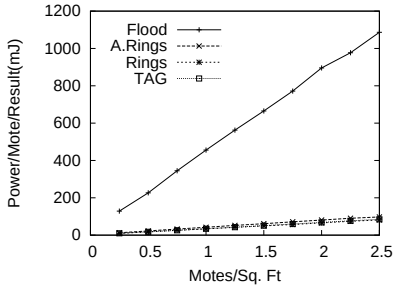


Figure 9: The impact of sensor density on power consumption

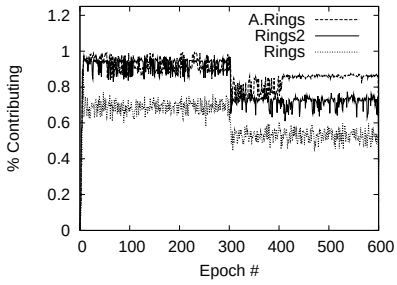


Figure 10: Correlated node failures

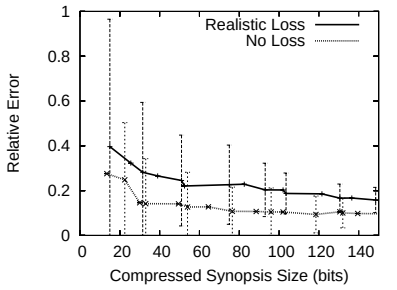


Figure 11: Synopsis size

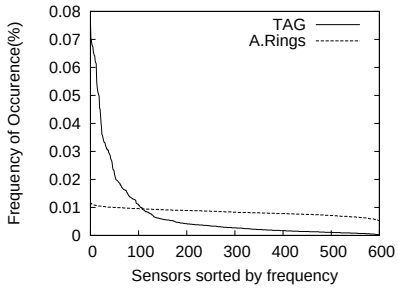


Figure 12: Uniform sample

6.4 Effect of Deployment Densities

The placement of sensors can influence the loss rates observed as well as the topology used to aggregate sensor readings. Here, we consider two different variations in the distribution of sensors: the density of sensors and the shape of the sensor deployment region.

To evaluate the impact of sensor density, we vary the number of total sensors while keeping the region (size and shape) in which the sensors are deployed constant. In addition, we employ the realistic packet loss model described earlier.

Figure 8 shows the impact of changes in density on the accuracy of TAG, RINGS, ADAPTIVE RINGS and FLOOD. As the sensor network becomes more sparse, 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 since deep trees combined with 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 9 plots the impact of density on our overhead metric, communication power consumption (the breakdown of the power consumed to transmit and receive messages can be found in [23]). Because the nodes in TAG and RINGS remain awake for receiving messages for roughly the same amount of time [19], 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 of the schemes.

In addition to density, the rough shape of a sensor deployment can also affect the performance of the different aggregation schemes. To evaluate this effect, 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 [23].

6.5 Effect of Asymmetric Links

Asymmetric links are common in real 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 [28]. 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 10 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* grid, 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 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 [23]).

6.7 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 11 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 with no loss rate. 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.

6.8 Beyond Sum

Uniform Sample. Figure 12 compares the sampling algorithm described in Section 4 running over ADAPTIVE RINGS with an existing random sampling algorithm known as RanSub [17] 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, while the synopsis diffusion algorithm, using ADAPTIVE RINGS, closely approximates a uniform distribution.

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.

6.9 Discussion

Our results have quantified a number of advantages that synopsis diffusion provides over previous tree-based and ring-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 correlated 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 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 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

Computing aggregates in sensor networks has been studied in a number of recent papers [19, 20, 21, 29]. The proposed approaches use a tree or DAG topology (with value splitting), and hence are not robust against node and link failures.

To achieve robustness, Zhao *et al.* [29] focus on dynamically maintaining a relatively robust aggregation tree. The approach is orthogonal to ours and requires each node to maintain statistics on link quality (to choose a stable parent). For duplicate-insensitive aggregates, they propose a technique called *digest diffusion*, based on flooding.

Directed diffusion [15] provides a scalable and robust paradigm for communication in sensor networks, focusing on robustly moving specific pieces of information from one place in the network to another. Techniques are presented for establishing and maintaining the diffusion gradients that are used to guide data collection. Note that $SG()$ and $SF()$ of a synopsis diffusion algorithm may be implemented as filters within directed diffusion.

Gupta *et al.* [14] propose a gossip-based fault-tolerant approach for computing aggregates over large process groups. However, the solution is not energy-efficient, relies on eventual convergence, and some of the assumptions (*e.g.,* all the processes are synchronized with different phases of the algorithm) are impractical for real sensor deployments.

Bawa *et al.* [5] have independently proposed duplicate-insensitive approaches for estimating certain aggregates in peer-to-peer networks. However, the work mainly focuses on the different semantics of the computed aggregates and the required topology and algorithms to achieve that. They do not address the formal requirements of the algorithms; they use a peer-to-peer network for evaluation and do not consider many of the sensor-relevant issues addressed in this paper.

There has been a flurry of recent work in the data stream community devising clever synopses to answer aggregate queries on data streams (see [3, 22] for surveys). This work has not focused on the ODI synopses required for synopsis diffusion. Note that 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, our synopsis fusion function merges two synopses, not just a current synopsis with a next stream value. More closely related is work on distributed streams algorithms [11, 12], which also requires the merging of multiple synopses. Second, previous data streams work has not focused on duplicate insensitivity. (Although it has focused on aggregates that are by definition duplicate-insensitive, such as `Count Distinct`.) One exception is a recent paper by Tao *et al.* [26] that uses duplicate-insensitive counting in mobile environments.

SIA [25] uses similar algorithms (*e.g.,* a variant of FM for counting, sampling for computing holistic aggregates) as ours. However, it focuses mainly on the security aspects of the algorithms. It assumes aggregation over a tree, and does not address the duplicate-insensitive properties of the algorithms.

Considine *et al.* [6] 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. Our work extends their work in a number of important ways: we present the first formal definition of duplicate-insensitive synopses; we prove powerful theorems characterizing ODI synopses and their error guarantees—their paper has no analogous result; we present solutions for a wider range of aggregates; we consider techniques for adaptive rings that reduce message loss; and our simulation results use a more realistic communication loss model, and consider scenarios not addressed in their paper such as correlated node failures.

8. CONCLUSIONS

In this paper, we present *synopsis diffusion*, a general framework for designing energy-efficient, highly-accurate in-network aggregation schemes for 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 prove the powerful and somewhat surprising result that four easy-to-check properties on the synopsis generation and fusion functions characterize ODI synopses. We give a number of examples of aggregates that can be computed in-network using ODI synopses. We have shown how ODI synopses can provide the implicit acknowledgments for network transmissions. In addition, we show that the light-weight monitoring of transmissions using these acknowledgments can be exploited to create an energy efficient and adaptive aggregation topology. Finally, we provide an extensive performance study on a realistic simulator demonstrating the significant robustness, accuracy, and energy-efficiency improvements achieved by an ODI-synopsis based approach running over our adaptive aggregation topology.

Our ongoing efforts on synopsis diffusion include implementing and evaluating it by using real sensor deployments, studying the effects of sensor mobility on synopsis diffusion over Adaptive Rings (see [23] for preliminary results), and developing ODI synopses for additional aggregates.

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

9. REFERENCES

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

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

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *ACM PODS*, 2002.

[4] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *ACM STOC*, 2001.

[5] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The price of validity in dynamic networks. In *ACM SIGMOD*, 2004.

[6] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *IEEE ICDE*, 2004.

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

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

[9] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review (M2CR)*, 1(2), 2002.

[10] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. *DIMACS: Series in Discrete Math. and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization*, 1999.

[11] P. B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *ACM SPAA*, 2001.

[12] P. B. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *ACM SPAA*, 2002.

[13] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *IEEE ICDE*, 1996.

[14] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *IEEE Dependable Systems and Networks*, 2001.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MobiCom*, 2000.

[16] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

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

[18] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *ACM SenSys*, 2003.

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

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

[21] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *IEEE WMCSA*, 2002.

[22] S. Muthukrishnan. Data streams: Algorithms and applications. Technical report, Rutgers University, Piscataway, NJ, 2003.

[23] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. Technical Report IRP-TR-04-13, Intel Research Pittsburgh, PA, April 2004.

[24] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. ANF: A fast and scalable tool for data mining in massive graphs. In *ACM KDD*, 2002.

[25] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys*, 2003.

[26] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. Spatio-temporal aggegration using sketches. In *IEEE ICDE*, 2004.

[27] Y. Yao and J. Gehrke. Query processing in sensor networks. In *CIDR*, 2003.

[28] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *ACM SenSys*, 2003.

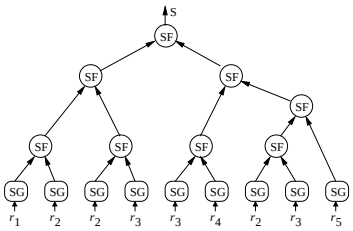


Figure 13: Graph used in the proof

[29] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *IEEE SPNA*, 2003.

Appendix

Proof of Theorem 1. (sketch) Consider an arbitrary execution of synopsis diffusion, producing a synopsis s. Let G be the aggregation DAG corresponding to this execution (Figure 3(a)), and let u be the node in G that outputs s. 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 (Figure 3(b)).

First, let G_1 (Figure 13) 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. This may create many duplicate SF and SG nodes. Also, any node in G without a path to x 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 $SG(r) = SG(r)$ 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 $SG(r_j)$ 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 one of the leaf nodes (say the left one). 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 $SG^*(V)$. Since 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.

It is not difficult to show that each of the properties is necessary by considering aggregation DAGs with at most four sensor readings (omitted due to page limitations). For example, if property P4 were not true, then $SF(SG(r_1), SG(r_1))$ would not produce the synopsis that the canonical tree $SG(r_1)$ does. □