Synopsis Diffusion for Robust Aggregation in Sensor Networks

Suman Nath and Phillip B. Gibbons

IRP-TR-03-08
August 2003

Intel Research

Synopsis Diffusion for Robust Aggregation in Sensor Networks

Suman Nath[†,*] Phillip B. Gibbons[*]
[*]*Intel Research Pittsburgh* [†]*Carnegie Mellon University*

Abstract

Aggregating sensor readings within the network is an essential technique for conserving energy in sensor networks. Previous work proposes aggregating along a tree overlay topology in order to conserve energy. However, a tree overlay is very fragile, and the high rate of node and link failures in sensor networks often results in a large fraction of readings being unaccounted for in the aggregate. Value splitting on multi-path overlays, as proposed in TAG, reduces the variance in the error, but still results in significant errors. Previous approaches are fragile, fundamentally, *because they tightly couple aggregate computation and message routing.* In this paper, we propose a family of aggregation techniques, called *synopsis diffusion*, that decouples the two, enabling aggregation algorithms and message routing to be optimized independently. As a result, the level of redundancy in message routing (as a trade-off with energy consumption) can be adapted to both expected and encountered network conditions. We present a number of concrete examples of synopsis diffusion algorithms, including a broadcast-based instantiation of synopsis diffusion that is as energy efficient as a tree, but dramatically more robust.

1 Introduction

In a large-scale network of wireless sensors, aggregate queries assume greater importance than individual sensor readings because of energy and communication constraints. Previous studies [11, 15] have shown that computing aggregates *in-network*, *i.e.*, combining partial results as they meet at nodes during message routing, significantly reduces the amount of communication and hence the energy consumed. A popular approach 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. Because any scheme requires $N-1$ messages to account for N sensor readings, and a tree uses only $N-1$ messages, the tree-based approach uses an optimal number of messages, *in the absence of failures*.

However, when a sensor network is deployed in a harsh environment, node and link failures are the norm [11, 15]. New nodes may join the network and existing nodes may leave, as a result of hardware failures, resource depletion, duty cycling (sleeping) to conserve energy, deployment of additional sensors, *etc.* Moreover, packets may frequently and persistently be lost because of noise, obstacles moving through the environment, or message collision. In short, the population and topology of the network are often highly dynamic and unpredictable.

As a result, approaches to in-network aggregation along a tree suffer from a significant error in the query answer [15], because even a single link failure results in a fraction of the readings being absent from the aggregate computation. There are several approaches proposed in the literature for addressing this problem. First, one can retransmit lost packets, but retransmitting unicast packets fails to exploit the broadcast medium and does not solve the problem for node failures or more persistent link failures. Second, one can construct a new spanning tree that avoids the problem nodes and links, but tree maintenance and repair protocols cost packets (and delay the query answer). Finally, as proposed for TAG [11], one can use a DAG instead of a tree: each node with accumulated value v sends v/k to each of its k parents. For aggregates such as Count or Sum, this reduces the error for a link failure from v to v/k, but the expected aggregation error remains as bad as for the tree [11].

Moreover, for the class of *holistic aggregates* [9] such as Median, no previous approach has successfully used in-network aggregation (even for tree routing), because "no useful partial aggregation can be performed" [11].

Synopsis Diffusion. In this paper, we propose *synopsis diffusion*, a framework well-suited to energy-efficient aggregation in dynamic and unpredictable network topologies. Synopsis diffusion is based on three key ideas:

- **Decouple aggregate computation from message routing.** With synopsis diffusion, the *message propagation scheme* (*i.e.*, *which* nodes send and receive messages *when*) can be decided independently of the aggregation. The goal for a message propagation scheme is to maximize the number of distinct sensor nodes with propagation paths to the querying node,[1] while mini-

[1] A *propagation path* is a hop-by-hop sequence of successfully transmit-

mizing the number of messages sent (*i.e.*, the energy consumption). There is a continuum to this trade-off, and synopsis diffusion provides the flexibility to use any point on this continuum, to incorporate any desired level of redundant propagation of readings and partial results (for protection against possible failures), and to be fully adaptive to the node and link dynamics actually encountered during routing.

- **Use duplicate-insensitive synopses to summarize partial results.** Multi-path routing [5] is essential for achieving robustness. With synopsis diffusion, each node sends its *complete partial result* along the multiple paths, thereby avoiding the problems with value splitting, but introducing the concern that readings will be double-counted. Duplicate-insensitive 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 the number of times a given reading from a given sensor arrives at the node (either directly or indirectly via partial results).
- **Exploit the wireless broadcast medium.** Aggregation on a tree superimposes a point-to-point message exchange on an inherently broadcast medium; synopsis diffusion enables any and all listeners to take advantage of the messages they hear. Unlike value splitting, the sender need not know the exact number of receivers. We show that synopsis diffusion using a broadcast-based message propagation scheme (a RINGS topology, as described in Section 2) is as energy efficient as tree-based aggregation, but dramatically more robust.

Previous sensor network research has shown that for aggregates that are inherently duplicate-insensitive (*e.g.*, Max and Min), robust in-network aggregation using broadcast is both low-energy and highly accurate [15]. However, most aggregates are duplicate-sensitive (e.g., Sum, Count, Average, Median), and the challenge is to devise duplicate-insensitive synopses for these duplicate-sensitive aggregates. To this end, we develop a formal definition of duplicate-insensitive synopses. This definition captures the overall goal of duplicate-insensitivity, but it is not immediately useful for designing synopses. Thus we derive simple properties that provably imply the more general definition, and show how these can be used to design (provably correct) synopses.

An important concern in using duplicate-insensitive synopses is that they often introduce approximation errors in the aggregate being computed. For example, previous approaches for computing a Count aggregate accumulate the count along a tree; in the absence of failures, each partial result is an exact count for its subtree. In contrast, a duplicate-insensitive Count synopsis (an example is given in Section 2) introduces an approximation error even in the absence of failures. However, we show that with realistic packet loss rates (5–30%), the errors from missing values in the tree-based approach are far greater than the approximation errors we introduce. Moreover, whereas previous approaches were unsuccessful in using in-network aggregation for holistic aggregates such as Median (as discussed above), synopsis diffusion, by allowing for approximation, enables in-network holistic aggregation even with multi-path routing. Finally, one can typically derive bounds on the approximation error, and select from a continuum of trade-offs between approximation guarantees and message size. Surprisingly, we show these error bounds are often independent of the message propagation scheme.

We believe that synopsis diffusion is a powerful framework for the design of sensor network aggregation schemes. Instead of tightly coupling in-network aggregation and message routing, synopsis diffusion permits independent explorations of these areas. In the remainder of this paper, we present our early progress on synopsis diffusion. Section 2 describes the basic synopsis diffusion approach. Section 3 describes duplicate-insensitive synopses for several holistic aggregates. Section 4 elaborates on some of the trade-offs synopsis diffusion enables. Various extensions are discussed in Section 5, followed by related work in Section 6 and conclusions in Section 7.

2 Synopsis Diffusion

Synopsis diffusion performs in-network aggregation. The partial result at a node is represented as a synopsis [2, 6], a small digest (*e.g.,* histogram, bit-vectors, sample, etc.) of the data. Synopses, being small by definition, keep the messages small and make it possible to piggyback them on the existing ad hoc networking protocols (if there are any). The aggregate computation is defined by three functions on the synopses:

Synopsis Generation. A synopsis generation function $SG(\cdot)$ takes a sensor reading and generates a synopsis representing that data.

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 to the final answer.

The exact details of the functions $SG()$, $SF()$, and $SE()$ depend on the particular aggregate query to be answered. Examples are given in Section 2.2 and Section 3.

A synopsis diffusion algorithm consists of two phases: a *distribution phase* in which the aggregate query is flooded

ted messages from the sensor node to the querying node. 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.

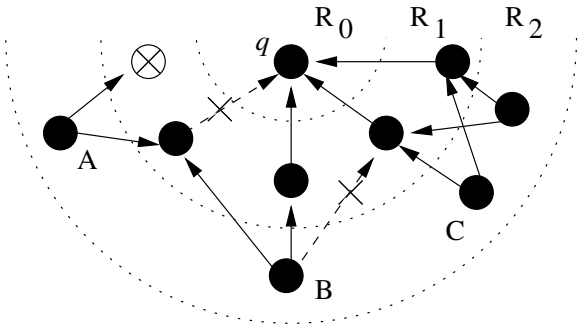

Figure 1: Synopsis diffusion over the Rings topology. Crossed arrows and circles represent failed links and nodes.

through the network and the target overlay topology is constructed,[2] and a *collection phase* where the aggregate values are continually routed towards the querying node. Within the collection phase, each node periodically uses the function $SG()$ to convert local 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. Both one-time and continuous queries can be supported.

To make the framework more concrete, we describe next synopsis diffusion on a RINGS overlay topology; this topology organizes the nodes in 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 [11], nodes in different rings are loosely time synchronized and allotted specific time intervals when they should be awake to receive synopses from other nodes.

We now describe the query aggregation in greater detail, using the example Rings topology in Figure 1 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 $s = SG(v)$, where v is the value to be included in 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 its neighbors in ring R_{i+1}[3]. 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 towards the querying node q, which at the end of the epoch returns $SE(s)$ as the answer to the aggregate query.

Figure 1 shows that even though there were link and node failures, node B has one failure-free propagation path to the querying node q. Thus its sensed value(s) are accounted for in the answer produced this epoch. With synopsis diffusion, as long as there is at least one failure-free propagation path from a node, its sensed value(s) are accounted for in the answer. Node C has *two* failure-free propagation paths to q; as we discuss below, the $SG()$ and $SF()$ functions must ensure that the node's sensed value(s) are not double-counted. On the other hand, *all* of the propagation paths from node A fail; hence its sensed value(s) are *not* included in the answer. With robust topologies such as Rings, there are typically many propagation paths from each node, so the probability of having all of them fail—resulting in a missed value—is small.

Because the underlying wireless communication is broadcast, *synopsis diffusion on* RINGS *generates the same optimal number of messages as the previous tree-based approach*.[4] However, because synopses propagate from the sensor nodes to the querying node along multiple paths, *it is much more robust*. (This added robustness is quantified in Section 4.)

2.2 Duplicate-Sensitive Aggregates

With synopsis diffusion, the aggregate computation is orthogonal to the message propagation scheme. *The main challenge of a synopsis diffusion algorithm is to support duplicate-sensitive aggregates correctly for all possible multi-path propagation schemes.* To achieve this, we map the target aggregate function (*e.g.*, Count) to a set of *duplicate-insensitive*, *commutative* and *associative* synopsis generation and fusion functions. Intuitively, such a set of functions ensures that a partial result at a node v is determined by the set of readings from sensor nodes with propagation paths to v, 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 in the aggregate if there is a propagation path from the sensor node to the querying node. We illustrate such functions using a few examples, and then provide formal definitions.

[2] This topology may be adapted at any time as network conditions change.

[3] Note that there is no one-to-one 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}.

[4] Specifically, for a sensor network of N nodes, the aggregation phase uses $N-1$ messages to produce an answer.

Approximate Count. This algorithm counts the total number of sensor nodes in the network. The algorithm is adapted from Flajolet and Martin's algorithm (FM) [4] for counting distinct elements in a multi-set. It 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 used in FM: for $i < x$, $CT(x) = i$ with probability 2^{-i}. The different components of the synopsis diffusion algorithm for approximate 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 [5].
- $SG()$**:** Output a bit vector s of length k with only the $CT(k)$'th bit set.
- $SF(s,s')$**:** Output the 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 $2^{i-1}/0.77351$ [4].

The function $SE()$ is based on the following intuition. Consider the final synopsis s to which $SE()$ is applied, and the set, V, of sensor nodes accounted for in s (*i.e.*, V is the set of nodes with failure-free propagation paths to the querying node). From the experiment $CT()$, the probability that the j'th bit of s is set by a given node in V is 2^{-j}. If no node sets the j'th bit, then there are probably less than 2^j nodes in V, and if we have at least 2^j nodes in V, then we expect the jth bit to be set. Thus the number of sensor nodes in V is proportional to 2^{i-1}, where i is the index of the lowest-order bit that is not set by any node in V. Thus $SE(s)$ is proportional to the number of sensor nodes in V. We refer the reader to [4] for a derivation of the constant of proportionality (0.77351) and for a proof that this estimate has good error bounds.

For more accurate approximations (decreased variance), each synopsis can maintain multiple independent bit-vectors and the function $SE()$ can use the average of the indices of the lowest-order 0-bits [4].

Intuitively, the $SG()$ and $SF()$ functions are duplicate-insensitive because $SG()$ maps an individual reading to a particular bit being set, and once set, the bit is never unset ($SF()$ takes a bitwise OR). Hence, a synopsis is the same whether a given reading is accounted for one or many times. We prove this formally in Section 2.3.

Approximate Sum. An approximate Sum of (nonnegative integer) sensor readings can be computed using a simple generalization of the above algorithm for Count: 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* [1]. Because the algorithm keeps track of only the maximum bit set, the synopsis can be smaller. The algorithm assumes that a node can quickly generate a random number between 0 and 1.

- **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 $\lceil -\log_2(1-x_v^{-val_v})\rceil$.
- $SF(s,s')$**:** Output $\max(s,s')$.
- $SE(s)$**:** Output 2^s.

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 ith 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 ith 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 $i \geq -\log_2(1-x^{-m})$, namely, $\lceil -\log_2(1-x^{-m})\rceil$.

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 the indices of the highest-order 1-bits [1].

2.3 Formal Definition of Duplicate Insensitiveness

Synopsis diffusion relies on having $SG()$ and $SF()$ functions that are commutative, associative, and duplicate-insensitive. In this section, we formalize these requirements. We begin with the following 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

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

to be unique (*e.g.*, there is only one reading corresponding to a given sensor id and timestamp).

The *synopsis label* function $SL()$ computes the label of a synopsis. The *label* of a synopsis s is defined as the set consisting of all sensor readings included into s. More formally, $SL()$ is defined inductively, as follows. If s is a synopsis,

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

The operator $\uplus$ takes two multi-sets and returns the multi-set consisting of all the elements in both multi-sets, including any duplicates. *E.g.*, $\{a,b,c,c\} \uplus \{b,c,d\} = \{a,b,b,c,c,c,d\}$. 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 a pair 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 sensor nodes, the value for a sensor reading is its sensor id, because the goal is to count the number of distinct sensor ids. 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.

Formal Properties. We now consider the three properties: commutativity, associativity, and duplicate-insensitivity. 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 definitions of the properties we seek are:

- Property P1: $SF()$ **is commutative:**

  $$\forall s_1, s_2 \in \mathcal{S} : SF(s_1, s_2) = SF(s_2, s_1)$$

- Property P2: $SF()$ **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 P3: $SF()$ **and** $SG()$ **are duplicate-insensitive:**

  $$\forall s \in \mathcal{S} : s = SG^*(\Pi_q(SL(s))),$$

  where

  $$SG^*(V) \quad = \quad \begin{cases} SF(SG^*(V - \{v_k\}), SG(v_k)) & \text{if } |V| = k > 1 \\ SG(v_1) & \text{if } |V| = 1 \end{cases}$$

  In other words, $SF()$ and $SG()$ are duplicate-insensitive if, regardless of how they are applied (*i.e.*, regardless of the redundancy arising from multi-path routing), the resulting synopsis s is the same as what arises from applying SG to each *distinct* value and then merging these synopses into one synopsis using a binary tree of SF nodes. Or more simply, the result is the same as when each distinct value is accounted for only once in s.

We say that a synopsis diffusion algorithm is *correct* if it satisfies properties P1, P2, and P3.

Key Properties. We believe that Property P3 captures the overall goal of duplicate-insensitivity. However, it is not immediate useful for designing synopses because verifying correctness of a synopsis would entail considering the *unbounded* number of ways that $SG()$ and $SF()$ can be applied to a set of sensor readings.

Thus an important contribution of this paper is in deriving the following two simple properties, and showing that they imply Property 3.

- Property P3a:

  $$\forall r_1, r_2 \in \mathcal{R} : \Pi_q(\{r_1\}) = \Pi_q(\{r_2\}) \text{ implies } SG(r_1) = SG(r_2)$$

  That is, if two readings are considered duplicates (by Π_q) then the same synopsis is generated.

- Property P3b:

  $$\forall s \in \mathcal{S} : SF(s, s) = s$$

  That is, given the same synopsis for both its arguments, SF simply returns that synopsis.

Given the simplicity of properties P3a and P3b, it is surprising that they imply P3. The next theorem shows that indeed this is the case.

Theorem 1 *Let $SF()$, $SG()$ and Π_q be such that properties P1, P2, P3a, and P3b hold. Then property P3 holds as well.*

The proof is given in the appendix.

We illustrate how these properties can be used to prove the correctness of a synopsis diffusion algorithm by revisiting the Approximate Count algorithm for counting the number of sensor nodes in the network.

Claim 1 *The Approximate Count algorithm in Section 2.2 is correct.*

Proof. Consider a projection operator Π_q that maps a set of sensor readings to the corresponding sensor ids. In the Approximate 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 $SG(r_1) = SG(r_2)$.[6] Finally, $SF(s,s)$ is the Boolean OR of the bit vector s with itself, which equals s. Hence since properties P1, P2, P3a, and P3b hold, then so does property P3. Thus the algorithm is correct, and the claim is proved.

Similarly, it is not difficult to prove the correctness of each of the other synopsis diffusion algorithms presented in this paper.

2.4 Approximation vs. Lack of Robustness

As discussed in the introduction and illustrated in the previous examples, duplicate-insensitive synopses often introduce approximation errors in the aggregate. In this section, we argue that when node and link failures are prevalent, this approximation is preferable to the inaccuracy in the result computed by previous tree-based or DAG-based approaches.

First, the answer provided by a synopsis diffusion algorithm is semantically richer than that returned by a tree-based approach. With a robust aggregation topology in place, the error in the final answer is primarily attributed to the approximation algorithms in use – which typically provide well-defined probabilistic error bounds. In contrast, with a non-robust topology (*e.g.,* tree), the error in the final answer is attributed to the dynamicity of the network, and there may not be any effective strategies for quantifying this error for the given aggregate.

Second, when node and link failures are prevalent, synopsis diffusion produces more accurate answers than previous tree-based or DAG-based approaches. Figure 2 shows the simulation results of computing the Sum of the values sensed by 2401 sensors placed in a 49×49 grid. The sensed values are uniformly random within the range $[1,100]$. The querying node is at the center of the grid. In this simulation, we only assume link failures, and use the loss model of a real testbed as described in [15] (the message loss rate is around 30%). We simulate three aggregation schemes: TREE, the standard tree-based approach, DAG, the TAG approach using two parents, and SD, the synopsis diffusion algorithm over the Rings topology. The figure shows how the aggregate reported to the base station changes over time. The TREE and DAG schemes consistently underestimate the

[6] 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.

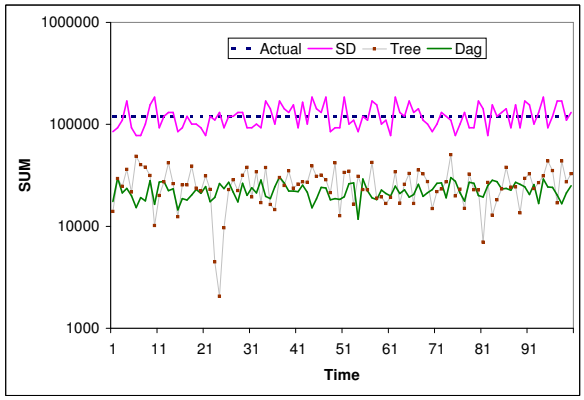

Figure 2: Comparison of different aggregation schemes

aggregate (because with each link failure, some fraction of the Sum is lost), whereas the SD scheme oscillates around the actual sum (which means the base station can maintain a running average over the last few aggregates to get even better estimates). The relative root mean square error (RMS)—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—for the TREE and DAG schemes are around 0.8, whereas the RMS error is less than 0.3 for the SD scheme. Even with as low as 5% message loss, the RMS errors for TREE and DAG are around 0.6, whereas the RMS error for SD is around 0.25. This shows the usefulness of the synopsis diffusion algorithms over other approaches.

Finally, synopsis diffusion enables in-network aggregation for holistic aggregates, as discussed in the next section.

3 Holistic Aggregates

Computing exact answers to holistic aggregates (*e.g.,* Count Distinct, Median) requires collecting at the base station all the data to be aggregated. Synopsis diffusion provides an energy-efficient alternative that provides approximate answers using in-network aggregation. In this section, we give three examples.

Approximate Count Distinct. This algorithm counts the distinct elements in the network (again) using an adaptation of Flajolet and Martin's algorithm. Suppose the elements to count are drawn from the set V. It uses a random hash function $H_e : V \mapsto [0,r], r > \log(|V|)$, with an exponential distribution. The distribution ensures that half the elements in V are hashed to 0, a quarter to 1, one-eighth to 2, *etc*.

- **Synopsis:** The synopsis consists of a bit-vector of length $(r+1)$.
- $SG()$**:** Output a bit vector s of length r with only the $H_e(x)$'th bit set, where x is the value.

- $SF(s,s')$ **and** $SE(s)$**:** Same as the Approximate Count algorithm.

Most Popular Items. The goal of this algorithm is to return the K values that occur the most frequently among all the sensor readings.

- **Synopsis:** A set of the K most popular items (estimated).
- $SG()$**:** At node u, output the (value, weight) pair $(val_u, CT())$.
- $SF(s,s')$**:** 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 all of them.
- $SE(s)$**:** Output the set of values in s.

Sampling-Based Estimation. There are a number of aggregates, holistic or otherwise, that can be well-estimated from a uniform sample of the sensor readings. For example, Median can be approximated by taking the median of the sample. Unfortunately, traditional sampling procedures would not produce a uniform sample in the presence of multi-path routing because they are duplicate-sensitive. However, the algorithm we give next does produce a uniform sample.

Suppose each node u has a value val_u, and our goal is to output a uniform sample of a given size K of the values occurring in all the nodes in the network. Let $H : values \mapsto [0..l]$, for some constant $l > \#values$, be a uniform random hash function known by all the nodes (it can be sent as part of the query distribution). The components of the algorithm are as follows:

- **Synopsis:** A sample of size K.
- $SG()$**:** At node u, output the (value, id) pair (val_u, id_u), where id_u is the sensor id for node u.
- $SF(s,s')$**:** Hash each of the values in $s \cup s'$ (note that the union will remove precisely any duplicate readings from the same sensor) and output the (value, id) pairs for the K values with maximum hash values. If there are less than K pairs in $s \cup s'$, output all of them.
- $SE(s)$**:** Output the set of values in s.

Because each value is equally likely to be hashed to one of the top K hashed values (assuming a truly random hash function), a uniform sample is obtained.

4 Discussion of Trade-Offs

In this section we discuss the fundamental trade-offs between robustness and energy-efficiency and between approximation error and energy-efficiency. We also show how synopsis diffusion provides the flexibility to choose any points in the continua of these trade-offs.

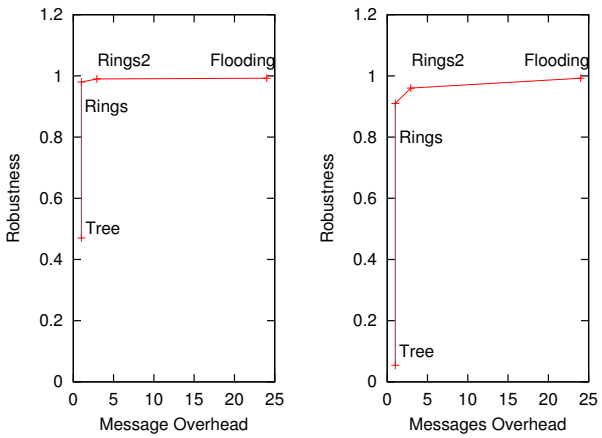

Figure 3: Trade-off between robustness and energy overhead of different topologies (Loss rate 5% and 30% respectively)

4.1 Robustness vs. Energy-Efficiency

To quantify the trade-off between robustness and energy-efficiency, we define the following two metrics. The *message overhead* of a propagation scheme (without considering any failures) is defined to be the ratio of the total number of messages generated before values from all the nodes in the network can reach the querying node *to* the optimal number of messages required for achieving the same. The *robustness* of a propagation scheme under a given message loss rate is defined to be the fraction of the nodes included in the final answer. We measure these metrics through simulation.

Figure 3 shows the trade-off between robustness and message overhead of level-by-level propagation schemes based on different topologies, under two different message loss rates. Using the simulation setup described in Section 2, we study four different topologies: TREE, RINGS (the scheme SD in Section 2), RINGS2 (a variant of RINGS where the transmission from a node in ring R_i is received by both its immediate and 2-hop neighbors in R_{i-1}), and FLOODING. As expected, TREE and RINGS have the optimal overhead. RINGS2 has slightly more overhead than RINGS because a message from a node needs to be transmitted to its 2-hop neighbors by intermediate nodes. FLOODING has the highest overhead. As the graphs show, the robustness of the tree topology is very poor, because a single link failure for the duration of an epoch will exclude a subtree from the final answer. Surprisingly, RINGS provides very high (> 0.9) robustness even when the average number of next-ring neighbors of a node is small (between 2 and 3 in the experiment).

Synopsis diffusion enables the flexibility to use any topology within the continuum presented in Figure 3.[7] Moreover, a propagation scheme may change the aggregation topology dynamically depending on the dynamicity of the network. Different topologies can be used in different regions

[7] Other continua are possible too.

of the network, in reaction to correlated failures. The overheads of maintaining and repairing specific topologies can be avoided. Unlike static aggregation schemes, low-quality links, asymmetric links, and new nodes can be opportunistically exploited.

Note that there is a fundamental difference between the routing *redundancy* used by synopsis diffusion and the routing *dispersion* used in TAG. With TAG's value splitting, increasing the number of parents for a node *increases* the probability of undercounting, because all paths have to reach the root in order for data not to be lost. With synopsis diffusion, increasing the number of parents *decreases* the probability of undercounting, because only one path has to reach the root.

4.2 Accuracy vs. Energy-Efficiency

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 in Section 2, 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 did the following experiment with the approximate Sum algorithm. The experiment computes an approximate Sum of the values (within the range $[1, 100]$) sensed by 2401 sensors. We use bit vector of length 20. The size of the synopsis is reduced by interleaving the bit-vectors (if more than one are used) and applying run-length encoding [14]. To isolate the synopsis approximation error from any errors resulting from lost readings, the experiment is run on a fault-free 49×49 grid. Figure 4 shows the result. The average and 95% confidence interval for each point in the graph have been computed over 100 trials. The graph shows that both the relative error and the 95% confidence interval improves as the total size of the synopsis increases, and with a synopsis of several hundred bits, synopsis diffusion provides around 10% relative error.

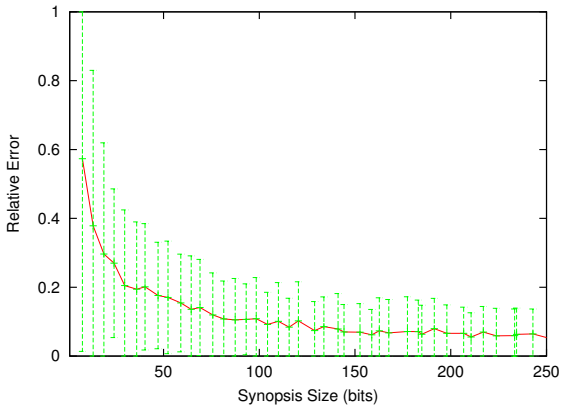

Figure 4: Trade-off between the relative error of the approximate Sum algorithm and the size of synopses used

5 Extensions

In this section we discuss several extensions to the methodology and techniques discussed thus far.

5.1 Other Aggregates

In general, it may be helpful for developing duplicate-insensitive synopses to look to the data streams literature [2]. There the goals are to estimate an aggregate with one 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 aggregates applied to the stream to

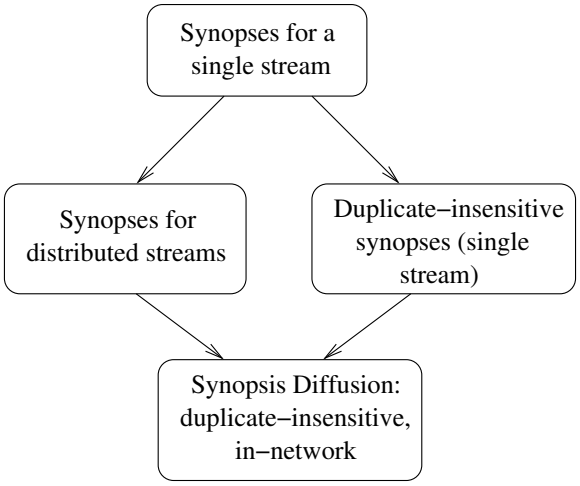

Figure 5: Hierarchy among synopsis problems

that point. The same synopsis 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 [7, 8]. 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 previous work on sequential or distributed data streams focused on duplicate sensitivity. (The exception is for aggregates that are by definition duplicate-insensitive, such as Count Distinct.) Figure 5 diagrams a path for developing new synopsis diffusion algorithms, where each edge is directed from an easier problem to a harder problem.

5.2 Error Bounds of the Approximate Answers

Using synopses may provide only an approximate answer to the query[8]. We here briefly present a generic framework to analyze the error bounds of a synopsis diffusion algorithm.

For all the algorithms we have presented so far, the functions $SG()$ and $SF()$ are duplicate-insensitive, associative, and commutative. The left-deep tree in the definition of $SG^*()$ in property P3 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. Thus the equivalence established in Theorem 1 implies the following semantics of the aggregate answer computed by a synopsis diffusion algorithm: (1) The final answer includes all the values that can reach the querying node through at least one 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. Thus any approximation error guarantees provided for the centralized data stream scenario immediately apply to the synopsis diffusion algorithm (as long as properties P1, P2, P3a, and P3b hold).

5.3 Outliers

We are currently working towards incorporating new in-network outlier detection techniques with the synopsis diffusion algorithms such that faulty readings of the sensors are automatically detected and reported, or excluded from the final aggregate answers. We take advantage of the fact that in many cases, such as when collecting a temperature readings of a building, the underlying sensor readings are continuous in time or space, or both.

To this end, we are exploring two directions. First, each node remembers the values it has heard from the nodes in its neighborhood in recent past. Before generating a local synopsis, a node performs a few statistical tests to determine if the local value is a possible outlier. If the new value passes the outlier detection, a local synopsis is generated as usual; otherwise the value is discarded.

The second approach we are exploring is to divide the network into small regions and include uniform sample of values from each region. A node can use the sample from its own region to detect whether its local value is an outlier in which case no synopsis is generated for that outlier value.

5.4 Supporting Mobile Sensors

Since the functions $SG()$ and $SF()$ are duplicate-insensitive, associative, and commutative, a mobile sensor can correctly contribute its data to the final aggregate answer by broadcasting its local synopsis (computed at the beginning of the epoch) periodically or when it moves.

6 Related Work

Computing aggregates in sensor networks has been studied in a number of recent papers [11, 12, 13, 15]. 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, Jhao *et al.* [15] 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 (in order to choose a stable parent). Moreover, it increases the communication overhead by including statistics about neighbors in the transmitted messages. Finally, it does not address node failures. For duplicate-insensitive aggregates, they propose a novel technique called *digest diffusion*, based on flooding.

Gupta *et al.* [10] propose a gossip-based fault-tolerant approach for computing aggregates over large process groups. However, the solution is not energy-efficient 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.* [3] have independently proposed duplicate-insensitive approaches for estimating certain aggregates in peer-to-peer networks. However, the work neither formally addresses a general framework that is suitable for sensor networks nor considers many of the sensor-relevant issues addressed in this paper.

[8] Some of the synopsis diffusion algorithms (*e.g.,* sampling algorithm described in section 3, finding maximum/minimum) provide exact answers.

Finally, there has been a flurry of recent work in the data stream community devising clever synopses to answer aggregate queries on data streams (see [2]). As discussed in Section 5, the work has not focused on the duplicate-insensitive synopses required for synopsis diffusion.

7 Conclusions

In this paper, we proposed *synopsis diffusion*, a general framework for designing in-network aggregation schemes for sensor networks. Unlike previous approaches, the aggregate computation is decoupled from the message propagation scheme, enabling independent optimization of each. Our preliminary simulations highlight the advantages of synopsis diffusion for maximizing answer quality for a given energy consumption.

We are currently working towards quantifying the robustness of our approach for both static and mobile sensors, using more extensive simulation and real deployment in a mote network.

Acknowledgment. We thank Joe Hellerstein, Sam Madden, Amol Deshpande, Wei Hong, Brad Karp and Srini Seshan for discussions on this research.

References

[1] ALON, N., MATIAS, Y., AND SZEGEDY, M. The space complexity of approximating the frequency moments. *J. of Computer and System Sciences 58* (1999), 137–147.

[2] BABCOCK, B., BABU, S., DATAR, M., MOTWANI, R., AND WIDOM, J. Models and issues in data stream systems. In *21st ACM Symposium on Principles of Database Systems (PODS 2002)* (2002).

[3] BAWA, M., GARCIA-MOLINA, H., GIONIS, A., AND MOTWANI, R. Estimating aggregates on a peer-to-peer network. http://www-db.stanford.edu/ bawa/Pub/aggregates.ps.

[4] FLAJOLET, P., AND MARTIN, G. N. Probabilistic counting algorithms for database applications. *Journal of Computer and System Sciences 31* (1985), 182–209.

[5] GANESAN, D., GOVINDAN, R., SHENKER, S., AND ESTRIN, D. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review (M2CR) 1*, 2 (2002).

[6] GIBBONS, P. B., AND MATIAS, Y. Synopsis data structures for massive data sets. *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization* (1999).

[7] GIBBONS, P. B., AND TIRTHAPURA, S. Estimating simple functions on the union of data streams. In *SPAA* (2001).

[8] GIBBONS, P. B., AND TIRTHAPURA, S. Distributed streams algorithms for sliding windows. In *SPAA* (2002).

[9] GRAY, J., BOSWORTH, A., LAYMAN, A., AND PIRAHESH, H. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *ICDE* (1996).

[10] GUPTA, I., VAN RENESSE, R., AND BIRMAN, K. Scalable fault-tolerant aggregation in large process groups. In *Proc. Conf. on Dependable Systems and Networks* (2001).

[11] MADDEN, S., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. Tag: A tiny aggregation service for ad hoc sensor networks. In *OSDI* (2002).

[12] MADDEN, S., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. The design of an acquisational query processor for sensor networks. In *SIGMOD* (2003).

[13] MADDEN, S., SZEWCZYK, R., FRANKLIN, M. J., AND CULLER, D. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of Fourth IEEE Workshop on Mobile Computing Systems and Applications* (2002).

[14] PALMER, C. R., GIBBONS, P. B., AND FALOUTSOS, C. ANF: A fast and scalable tool for data mining in massive graphs. In *Eighth ACM SIGKDD Intl. Conf. on Knowledge and Data Mining* (2002).

[15] ZHAO, J., GOVINDAN, R., AND ESTRIN, D. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications* (2003).

A Appendix

In this appendix, we present the proof of Theorem 1.

Proof. (sketch)

Consider an arbitrary execution of synopsis diffusion, producing a synopsis s. Let G be a directed acyclic graph corresponding to this execution: the leaves are $SG(r)$ nodes for some sensor reading r, each non-leaf node is an SF node,

and there is an edge from node u to node v if the output of node u is an input for node v. Thus all internal nodes have two incoming edges and 0 or more outgoing edges. Let x be the node in G that outputs s.

In this proof, we will perform a series of transformations to G that, by properties P1, P2, P3a, and P3b, will not change the output of x, and yet will result in the binary tree described in property P3.

First, let G_1 be the tree rooted at x 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 affect the computation of s). This is 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 P3a. Note that there is exactly one leaf in G_1 for each tuple in the synopsis label $SL(s)$.

Second, by properties P1 and P2, 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 P1 and P2) such that they are the two inputs to an SF node. By property P3a, both inputs are the same synopsis s', so by property P3b, 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 P1 and P2 into a left-deep tree G_4; this is precisely the binary tree described in property P3. 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 transofmations have not changed the output), property P3 holds. This completes the proof of Theorem 1.