Suppression and Failures in Sensor Networks: A Bayesian Approach*

Adam Silberstein [1] Gavino Puggioni [2] Alan Gelfand [3] Kamesh Munagala [4] Jun Yang [5]
[1,4,5]Department of Computer Science and [2,3]Institute of Statistics and Decision Sciences, Duke University

ABSTRACT

Sensor networks allow continuous data collection on unprecedented scales. The primary limiting factor of such networks is energy, of which communication is the dominant consumer. The default strategy of nodes continually reporting their data to the root results in too much messaging. *Suppression* stands to greatly alleviate this problem. The simplest such scheme is *temporal suppression*, in which a node transmits its reading only when it has changed beyond some ϵ since last transmitted. In the absence of a report, the root can infer that the value remains within $\pm\epsilon$; hence, it is still able to derive the history of readings produced at the node.

The critical weakness of suppression is message failure, to which sensor networks are particularly vulnerable. Failure creates ambiguity: a non-report may either be a suppression or a failure. Inferring the correct values for missing data and learning the parameters of the underlying process model become quite challenging. We propose a novel solution, *BaySail*, that incorporates the knowledge of the suppression scheme and application-level redundancy in Bayesian inference. We investigate several redundancy schemes and evaluate them in terms of in-network transmission costs and out-of-network inference efficacy, and the trade-off between these. Our experimental evaluation shows application-level redundancy outperforms retransmissions and basic sampling in both cost and accuracy of inference. The BaySail framework shows suppression schemes are generally effective for data collection, despite the presence of failures.

1 Introduction

The emerging technology of wireless sensor networks is poised to make data collection practical on unprecedented scales. In areas where continuous collection was previously cumbersome or impossible, nodes are deployed, take sensor measurements at regular intervals, form an ad hoc network using radio communication, and deliver messages to a root (base station) node, from which the user retrieves the data. At the application level, the simplest way to implement data collection is to instruct each node to transmit its readings to the root as it acquires them. The definitive limitation of sensor networks, however, is their limited battery life, and radio transmission is the dominant energy consumer. The continuous reporting strategy will cause nodes to deplete their batteries and die relatively quickly. This is especially true for nodes nearest the root, which must relay readings on behalf of all other nodes. This problem becomes more severe as the network scales. Our goal is to find an energy-efficient way to support data collection applications.

Besides the obvious solution of lowering the rate of periodic sampling, *suppression* [5, 27] is a promising alternative approach towards reducing communication. Nodes monitor conditions in-network and only transmit to the root when those conditions are triggered. These conditions intuitively capture how "interesting" each new reading is. For example, consider simple *value-based temporal suppression*: A node only transmits its current reading to the root if it differs from its last transmitted value by more than some threshold, ϵ. In the absence of a report, the root assumes the node's value is within $\pm\epsilon$ of its last received report. If the readings fluctuate little over time, we would expect a high rate of suppression and low rate of communication; meanwhile, the root is still able to bound the reading for every timestep to within $\pm\epsilon$, without continuous updates from the node.

In many scenarios, such as monitoring environmental data (e.g., soil moisture, temperature, light, etc.), node measurements change infrequently from timestep to timestep. Thus, even value-based temporal suppression, which only scratches the surface of the scope of suppression schemes, is likely to greatly reduce communication. Moreover, measurements may change not only slowly, but often predictably. We foresee a tremendous amount of work on building sophisticated suppression schemes that analyze and rely on the expected behavior of deployments, and need only report deviations from that, resulting in a high suppression rate. Before moving in that direction, however, we must address the major problem of failure, which afflicts all suppression schemes.

Coping with Failure We just stated in the absence of a report from a node, the root assumes its value is unchanged. We can safely do so only if we can guarantee all reports generated by the node reach the root. Unfortunately, this is not the case. Sensor networks are very prone to message failure. Failure is particularly problematic in conjunction with suppression because combined they create a state of ambiguity at the root. Is a non-report the result of a suppression or a failure? Whereas in the absence of failure we could safely bound the node's value within ϵ of its last report, interpretation of missing reports now becomes quite challenging. To illustrate the challenges and some intuition, we use a simple example.

Example 1. *Consider the following sequence received by the root from node u: y_t, $\perp_{t+1}$, y_{t+2}, where $\perp$ denotes a non-report. If we know a model governing how the series $\{y_t\}$ evolves over time,*

*This work is supported by the NSF CAREER and DDDAS programs (under awards IIS-0238386 and CNS-0540347), and by an IBM Faculty Award.

VLDB '07, September 23-28, 2007, Vienna, Austria.

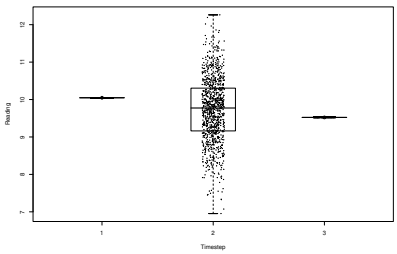

Figure 1: Missing value.

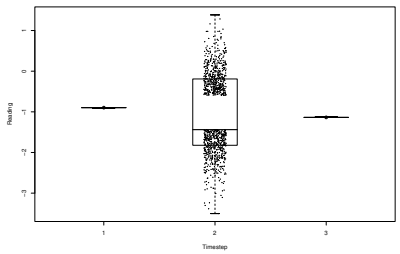

Figure 2: Missing value is failure.

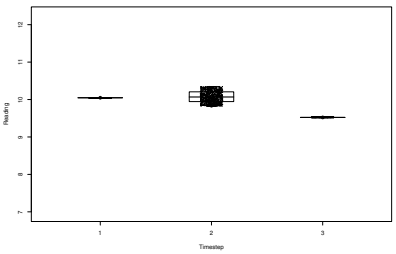

Figure 3: Missing value is suppression.

such as an auto-regressive(1) model, we can set y_{t+1} in an informed way using the values received, y_t and y_{t+2}. Figure 1 depicts a sample of possible assignments between the two known endpoint values. Each dot represents an assignment. The box spans the 25th to 75th percentiles of sample values, with the horizontal bar denoting the median.

Still, we can do better using knowledge about the suppression scheme in addition to the model. If, for example, $y_{t+2} - y_t \leq \epsilon$, a failure must have occurred on delivery of $\perp_{t+1}$. Had $\perp_{t+1}$ instead been suppressed, u would have compared y_{t+2} to y_t and suppressed y_{t+2}. This observation provides a constraint on possible settings for the missing value and leads to more informative guesses. Figure 2 again depicts a sample of assignments, but with very clear boundaries on values $\perp_{t+1}$ cannot have. Specifically, because $\perp_{t+1}$ is a failure, $|\perp_{t+1} - y_t| > \epsilon$ and $|\perp_{t+1} - y_{t+2}| > \epsilon$. This example also illustrates a subtle point. u is not aware of whether its delivery of $\perp_{t+1}$ fails. Therefore, its decision on whether to transmit y_{t+2} is based on comparison with $\perp_{t+1}$, rather than y_t, as the root might hope.

Now suppose $y_{t+2} - y_t > \epsilon$. In this case, we cannot determine whether $\perp_{t+1}$ is a suppression or failure. Suppose, however, using techniques we will discuss shortly, we determine it is a suppression. Figure 3 gives the corresponding sample assignments. In this case, $|\perp_{t+1} - y_t| \leq \epsilon$, while no constraint exists between $\perp_{t+1}$ and y_{t+2}.

Although this is a simple example aimed at reconstructing a single missing value sandwiched between two reported ones, it illustrates the progression of information gained in knowing a model of node behavior, the suppression scheme, and the cause of missing values. In reality, e.g., in environmental monitoring applications that use the observational data to learn about stochastic ecological processes, we would not even know the process parameters exactly; the goal is to combine our prior knowledge, the knowledge of the suppression scheme, and the observational data in a principled way to infer the model parameters as well as the missing data.

Our Approach In this paper, we propose a framework, *BaySail*, to cope with the presence of suppressions and failures, combining both in-network suppression and out-of-network data analysis. BaySail abbreviates *BAYesian analysis of Suppression and fAILures*. We use this framework to support estimating the missing values as precisely as possible, as well as estimating the process parameters for the model generating those values. BaySail is built on the following key ideas:

- Data analysis is Bayesian, allowing principled use of the received values and priors of the model parameters. This approach also provides posterior distributions of the missing values and model parameters, which allow better and less misleading interpretations of these than single-point estimates.
- Non-received data is not treated as generically missing. We instead incorporate knowledge of the suppression scheme into inference, thus effectively accounting for *why* they are missing.
- To help reduce uncertainty and improve inference, we enhance the suppression scheme with redundancy. Because the goal of suppression is to remove redundancy from reporting, however, we seem now to be rendering suppression irrelevant. This is not the case; by first removing the naturally occurring redundancy that comes with continuous reporting (to only receive "interesting" readings), we can then add redundancy back in a controlled manner, tuning it as we see fit (e.g., to achieve a particular level of accuracy in inference).

The above list outlines our vision for BaySail. It also raises a number of challenges, which we tackle in this paper. First, BaySail requires novel, non-standard application of Bayesian analysis in order to exploit the knowledge of the suppression scheme and the extra information provided by the redundancy scheme. We show that we can capture them as constraints on the posterior distribution of data and incorporate them into inference based on Gibbs sampling [15]. We demonstrate that, by incorporating such information, BaySail can provide estimates with much greater accuracy and less uncertainty than the basic Bayesian approach that simply treats all non-received data as missing.

Second, the choice of the redundancy scheme exposes many interesting trade-offs. What are the costs and benefits of redundancy? What type of redundant information helps most? Should we introduce redundancy at the application-level or let the communication layer automatically inject redundancy? We show that not only does the *degree* of redundancy affect the quality of inference, the *type* of redundancy also plays a critical role. We find application-level redundancy to be easier to control, more flexible, and generally more effective than built-in redundancy provided by the communication layer in the form of retransmissions. Moreover, we show that different types of redundant information can have dramatic impact on the efficiency of inference. Certain types of redundancy (e.g., message counters attached to each transmission) turn out to be computationally more expensive to utilize in inference, while others (e.g., a list of recent timesteps when transmissions were attempted) are easier to utilize. We consider this new interesting aspect of redundancy design for a number of schemes.

A third challenge is ensuring the practicality of BaySail. Because of difficulties in engineering reliable sensor nodes and networks, our philosophy is to make programming nodes as simple as possible. We argue for simple yet effective suppression and redundancy schemes in-network. Our implementation in TinyOS [2] on Mica2 motes [7] validates the simplicity of our design. On the other hand, Bayesian inference can be complex, but it takes place entirely out-of-network at the base station. In choosing the appropriate suppression and redundancy schemes, we carefully balance the implementation complexity on nodes, efficiency of out-of-network inference, energy costs, and the quality of inference. Through experiments, we demonstrate that BaySail provides better and more informative results than previous approaches. Its in-network implementation is simple, and its out-of-network inference is reasonably

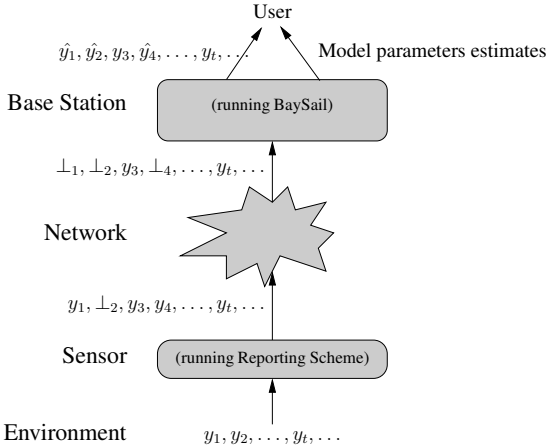

Figure 4: System overview.

efficient. Hence, it provides a practical and principled solution to the thorny problem of disambiguating suppression and failure in sensor networks.

2 Overview of BaySail

BaySail targets two types of users. "***Statistical modelers***" are the type of users that we interact with in our own project of deploying an environmental sensor network in Duke Forest. Our ultimate goal is to build ecological models of the networked area. The modeler wants "all the data," but only for purposes of fitting and comparing models that help to explain the data. They are ultimately interested in learning the parameters controlling the model, as well as the uncertainty associated with an incomplete raw dataset. "***Vineyard owners***" refer to more casual users of data, e.g., someone who uses Crossbow's SmartDust sensor networks [8] to monitor the growth and environmental conditions in a vineyard. We do not expect such users to have a background in statistics; they would prefer to view raw data and scan for aberrations. It is still useful to quantify the uncertainty in the data presented to them, however, because different confidence levels may call for very different actions. Furthermore, even though the users are not interested in the model parameters per se, a well-trained model can still work under the hood to help improve system efficiency (e.g., by fine-tuning suppression).

Our overall system is illustrated in Figure 4. The sensor measures raw data $y_1, y_2, \ldots, y_t$ from the environment. The sensor runs a *reporting scheme* that dictates what data are transmitted toward the base station and when. In general, a reporting scheme may include both a *suppression scheme*, which seeks to lower communication cost by suppressing "uninteresting" transmission, and a *redundancy scheme*, which injects redundant information in transmission to help cope with potential failures. In this figure, we show the sequence of readings transmitted out from the sensor, after the suppression scheme has been applied (we omit redundancy in this figure); $\perp$ denotes a non-reported value. Here, the sensor chooses not to transmit y_2. We discuss the reporting schemes in detail in Section 3.

These transmissions go through the network, where they are vulnerable to failure. In Figure 4, y_1 and y_4 are lost, denoted $\perp_1$ and $\perp_4$. Notice the base station interprets $\perp_1$, $\perp_2$ and $\perp_4$ as missing values, and cannot distinguished suppressions from failures.

The base station runs the BaySail data analysis procedure, discussed in detail in Section 4. Data analysis considers a model of the process being monitored, which the statistical modelers wish to fit. We might have some prior knowledge of the model parameters, but in general they are assumed to be unknown. Example 2 below describes one model of particular relevance to our sensor network. The data analysis procedure utilizes the observations (including both readings and any redundant information received by the base station), as well as the knowledge of what reporting scheme is running in the network. The result of the analysis provides estimates of both the model parameters and the missing data, in the form of a joint probability distribution that fully captures the uncertainty within and dependencies among the estimates.

As we will see in Section 4, we represent this distribution by a collection of samples from it; each sample corresponds to a potential realization of the model parameters and missing readings. It is straightforward to support expectation and quantile queries using these samples, e.g.: What is the expected value of this model parameter? What is the probability that these two missing data readings are both below a given value?

Example 2. *For the data collected from an environmental sensor network, we are naturally drawn to the class of dynamic spatial models. Recall the basic linear dynamic model [29]:*

$$\begin{aligned} \textit{Observation equation:} \quad & \mathbf{y}_t = A\mathbf{x}_t + \boldsymbol{\epsilon}_t; \\ \textit{Transition equation:} \quad & \mathbf{x}_t = B\mathbf{x}_{t-1} + \boldsymbol{\eta}_t. \end{aligned}$$

If the vectors are $n \times 1$, associated with spatial locations, say $s_1, s_2, ..., s_n$, we have a dynamic spatial model. See [14] for more details and schemes for fitting of such models within a Bayesian framework. In particular, $\boldsymbol{\epsilon}_t$ represents observation errors and $\boldsymbol{\eta}_t$ represents independent innovations of a colored noise process. At any t, $\mathbf{y}_t = (\mathbf{v}_t, \mathbf{w}_t)$, where $\mathbf{v}_t$ represents the components of $\mathbf{y}_t$ actually received by the root, and $\mathbf{w}_t$ represents the components that are missing (due to either suppression or failure). We seek to learn about all of the parameters in this dynamic model as well as $\mathbf{w}_t$ and $\mathbf{x}_t$, which have not been directly observed by the root.

To more clearly reveal the nature of the computational and inferential issues, we focus on the following considerably simplified model of soil moisture in the subsequent sections. Let $y_{s,t}$ denote the measure at location s at time t. The spatio-temporal model can be written as:

$$y_{s,t} = c_t + \phi y_{s,t-1} + \varepsilon_{s,t}. \tag{1}$$

Here, c_t is known time series of precipitation amounts for the region, which is observable directly at the root. The autocorrelation coefficient $|\phi| < 1$ controls how fast the moisture escapes the soil. The covariance function for the above model, shown below, takes into account both the temporal and the spatial correlation:

$$\mathrm{Cov}(y_{s,t}, y_{s',t'}) = \sigma^2 \frac{\phi^{|t-t'|}}{1-\phi^2} \exp(-\tau||s-s'||), \tag{2}$$

where $\sigma^2\phi^{|t-t'|}/(1-\phi^2)$ is the usual autocovariance function for the AR(1) process, and $\exp(-\tau||s-s'||)$ is the exponential spatial correlation function, with $||s-s'||$ denoting the distance between two locations s and s'. We want to estimate the process parameters ϕ, σ^2, and τ as well as the missing $y_{s,t}$'s.

Comparison with Previous Approaches Note that BaySail seeks to infer missing readings and model parameters *simultaneously*, and the result of inference are *distributions* (represented as samples) that fully capture uncertainty and dependency in data and model parameters. This approach differs from most previous approaches to data cleaning proposed in the database community (e.g., HiFi [13]), which are *staged*: The raw reading stream would be first subject to a data cleaning stage that produces a *single-point estimate* for each missing reading; then, the sanitized stream can be used further by the application to, for example, train models.

BaySail offers several fundamental advantages over these previous approaches. First, single-point estimates give no indication of uncertainty and may indeed be misleading (e.g., for a bimodal distribution, a single-point estimate based on the mean may in fact be a very unlikely value). On the other hand, BaySail is able to provide full distributional information, which avoids these problems.

Second, single-point estimates do not help inference of model parameters, and may in fact hurt if the ensuing model-fitting stage uses them as its input. For example, linear interpolation for a random walk provides reasonable guesses for the missing data, but will bias parameter estimation to a walk with some drift and zero variance. Furthermore, in a staged approach, if sanitized data no longer carries uncertainty, it is impossible for subsequent data analysis to assess information loss and quantify uncertainty in the model parameters obtained from such data, which are of particular concern to statistical modelers.

Finally, we argue that even if the sensor network application targets only "vineyard owner" types of users, there is a clear benefit to incorporating models in reconstructing missing data. Naive interpolation methods (e.g., linear interpolation between two received readings, or stepwise constant interpolation that assumes readings remain unchanged until the next received one) are attractive because they are computationally very efficient. However, as we show in experiments (Section 5), these methods can lead to arbitrarily wrong estimates and cannot provide any measure of confidence. We note that the model-based approach toward data reconstruction has also been advocated by *BBQ* [11]. In contrast to our work, which treats model parameters as unknowns, BBQ assumes the model parameters are known, and does not seek to fit the model using observations. Also, data acquisition in BBQ is pull-based rather than push-based, so the challenge of incorporating suppression into inference does not arise. In our approach, the decision of whether to transmit a reading is independent of the model whose parameters we are trying to learn; this feature makes our approach less vulnerable to inadequate models than BBQ.

3 Reporting Schemes

There is a large design space for push-based reporting (i.e., the node automatically transmits to the root without being queried). Assuming continuous reporting is too expensive, there are many ways to reduce the number and size of messages, with the goal of maintaining inference accuracy. Reports can be transmitted according to a sampling rate or a suppression scheme, where the rate τ or threshold ϵ, respectively, can be tuned to adjust energy cost. Failures can be handled using retransmissions to increase the probability any single message successfully reaches the root, or by adding application-level redundancy to messages, which increases their payload length. We next present several representative examples that we have investigated along these design axes.

We begin with two basic schemes (with no redundancy added):

- $\mathsf{Samp}(1/\tau)$: ***fixed-rate sampling***. A node reports its current reading every τ timesteps.
- $\mathsf{Supp}(\epsilon)$: ***value-based temporal suppression***. A node maintains its last transmitted reading, y_{last}, and only transmits its current reading y if $|y - y_{\text{last}}| > \epsilon$.

There is an intuitive appeal to $\mathsf{Supp}(\epsilon)$. If there are no failures, we can bound the reading at every timestep within $\pm\epsilon$ with absolute certainty. $\mathsf{Samp}(1/\tau)$, on other hand, offers no such feature, and may miss short-term fluctuations if τ is too long.

We can imagine many other suppression schemes. For example, the node can use a function f over all readings it has transmitted in a recent window. The result of the function serves as a prediction of its current reading. The node only transmits if the actual reading is more than ϵ away from the prediction. We briefly discuss how BaySail inference incorporates such suppression schemes in Section 4.2.

More complex *spatial* suppression schemes, such as *Ken* [5] and *Conch* [27], are also possible, but incorporating them into BaySail is beyond the scope of this paper. To briefly illustrate the challenge, consider the following example. A node u uses temporal suppression to report its value to another node v, which in turn uses this value to spatially suppress its own reports to the root. However, possible failures in u's messages to v would seriously confuse v and affect the correctness of spatial suppression. BaySail is designed to resolve such ambiguities out-of-network, but in this case, v needs to resolve such ambiguities in-network. How to extend BaySail to cope with failures in dependent suppression schemes is still an area of our ongoing work.

Adding Redundancy The basic reporting schemes above can be enhanced with redundancy. The network MAC layer can add receiver acknowledgments (Acks) in response to all transmissions, where the sender re-sends if no Ack is received. We denote this redundancy scheme $\mathsf{Ack}(r)$, where r indicates the maximum number of times the node attempts to transmit before giving up. This scheme can be used to augment both $\mathsf{Samp}(1/\tau)$ and $\mathsf{Supp}(\epsilon)$. Note that while willingness to retransmit lowers effective failure rate, it cannot eliminate failures entirely. Additionally, the scheme can bring significant extra cost due to overhead costs of sending extra messages. All received messages must be followed by an acknowledgment, each its own message. Each retransmission is also an additional message.

Besides $\mathsf{Ack}(r)$, we have spent considerable effort exploring how to add application-level redundancy to $\mathsf{Supp}(\epsilon)$. Any suitable redundancy scheme must satisfy two criteria. 1) It must be simple to deploy in-network and have a low energy footprint (i.e., it must only marginally increase the number and size of messages). 2) The resulting out-of-network inference must be fast (i.e., to provide acceptable response times to the user). Next, we define and discuss a progression of redundancy schemes for $\mathsf{Supp}(\epsilon)$: $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$, $\mathsf{Supp}(\epsilon)/\mathsf{C}$, $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$, and $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$. Each of these are for deployment at individual nodes, independent of others. Quantitative analysis of cost and inference are in Section 5.

$\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$ is the use of receiver acknowledgments and retransmissions described above.

$\mathsf{Supp}(\epsilon)/\mathsf{C}$, ***suppression with counter***, was our initial attempt at application-level redundancy. The node maintains a counter, incremented with each transmission to the root. Each report is augmented with the current counter value. If the root receives consecutive reports with non-consecutive counters, it can calculate how many failures have occurred between. For example, if the root receives the series $\{y_t, \perp_{t+1}, \ldots, \perp_{t+9}, y_{t+10}\}$, with y_t having counter 20 and y_{t+10} counter 23, it knows two failures have occurred in nine timesteps. It cannot, however, determine when they occurred. The main advantage is cost; we require only a small fixed cost per message to encode the counter. The disadvantage, as we will discuss in Section 4, is the difficulty in leveraging the counters in out-of-network inference.

$\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$, ***suppression with last r transmission timestamps***, piggybacks on existing reports, just like $\mathsf{Supp}(\epsilon)/\mathsf{C}$. The node augments each report with a list of the last r timestamps (in addition to the current timestamp) in which it transmitted. Upon receiving such a report, the root uses the timestamps therein to infer causes of previous non-reports. If for any of the r timestamps, no report was received, that time's non-report is labeled as a failure. Any non-reports between these are labeled as suppressions. Note any

ambiguous non-reports preceding the earliest of the r timestamps remain ambiguous. Continuing our $\mathsf{Supp}(\epsilon)/\mathsf{C}$ example, if $r = 4$ and the report listing y_{t+10} includes $\{t + 10, t + 7, t + 2, t\}$, we label $\perp_{t+2}$ and $\perp_{t+7}$ as failures, and all other non-reports as suppressions. While $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$ produces longer messages (r additional numbers rather than one in $\mathsf{Supp}(\epsilon)/\mathsf{C}$), we show in Section 4 how they can be leveraged in inference to better bound the range of possible values for each non-report. Nevertheless, the inference still has poor running time.

$\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ builds just slightly on $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$ by including additional ***direction bits*** to indicate the direction in which each transmitted value moved relative to the previous transmission: 1 indicates the value has increased since the previous transmission, and 0 indicates the value has decreased. At the first glance this redundancy scheme appears to be a very incremental improvement that lets us constrain each failure and subsequent non-reports to one range, rather than two (as in Figure 2). It is not at all intuitive why this is a *necessary* improvement. As we show in Sections 4 and 5, $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ enables much more efficient inference techniques that run orders of magnitude faster than for $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$. With a cost of only $r + 1$ additional bits, and the additional benefit of further constraining non-reports, it is well worth it. An important conclusion of our experiments is that $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ strikes the best balance of the in-network and out-of-network criteria.

4 Data Analysis

Recall that data analysis, which runs at the base station, fulfills two purposes simultaneously: 1) reconstruction of the missing readings, and 2) estimation of the model process parameters. The result is a joint distribution of missing values and model parameters.

In this section, we first present *BayBase*, a basic Bayesian approach that does not utilize the knowledge of suppression. This approach can be applied to data collected using $\mathsf{Samp}(1/\tau)$ and $\mathsf{Supp}(\epsilon)$, as well as versions of these with $\mathsf{Ack}(r)$. It does not apply to $\mathsf{Supp}(\epsilon)$ with C, $\mathsf{T}(r)$, and $\mathsf{TD}(r)$, because these schemes inject redundant information specific to suppression, which BayBase does not exploit.

Next, we show how BaySail additionally incorporates both the knowledge of the suppression scheme and the extra information provided by the redundancy scheme. BaySail works with $\mathsf{Supp}(\epsilon)$ and all its variants with redundancy. We discuss the computational issues that arise in implementing inference, and demonstrate how the choice of redundancy scheme can dramatically affect the computational efficiency of inference.

4.1 Basic Bayesian Inference

Our analysis is Bayesian. Through Bayes' Rule, we combine prior information on the model parameters and other unknowns (captured by *prior distributions*) with the observed data. The result is a distribution, named *posterior*, for the parameters and other unknowns, conditioned on the observed data. For lack of space, we omit a general description; see [16] for more details.

To illustrate, consider the soil moisture model in Example 2. First, we can encode any prior knowledge on model parameters using prior distributions; e.g., the prior distribution for σ^2 (variance) can be specified using a fairly vague inverse-gamma distribution $1/\sigma^2 \sim Gamma(2, 3)$, which reflects our lack of prior knowledge of this parameter.

Let V denote all readings that we have received, and W denote all missing readings (due to either suppression or failure). The posterior distribution of interest is

$$p\left(W, \phi, \sigma^2, \tau | V\right).$$

In order to sample this posterior distribution, we employ the widely used *Gibbs sampling* technique [15]. This technique sequentially draws samples of W, ϕ, σ^2, and τ using their respective *full conditional distributions*. The full conditional distribution is the distribution of one of the unknowns given all the other unknowns and the known data. An iteration is an update of these unknowns. After a sufficient number of iterations, the samples will essentially come from the true posterior distribution $p\left(W, \phi, \sigma^2, \tau | V\right)$. The algorithm is illustrated below:

1. Set $i \leftarrow 1$ and initialize $\phi^{(0)}$, $\sigma^{2(0)}$, and $\tau^{(0)}$ according to the respective prior distributions.
2. Sample $W^{(i)}$ from the distribution
$$p\left(W^{(i)} | V, \phi^{(i-1)}, \sigma^{2(i-1)}, \tau^{(i-1)}\right).$$
3. Sample $\phi^{(i)}$ from the distribution
$$p\left(\phi^{(i)} | V, W^{(i)}, \sigma^{2(i-1)}, \tau^{(i-1)}\right).$$
4. Sample $\sigma^{2(i)}$ from the distribution
$$p\left(\sigma^{2(i)} | V, W^{(i)}, \phi^{(i)}, \tau^{(i-1)}\right).$$
5. Sample $\tau^{(i)}$ from the distribution
$$p\left(\tau^{(i)} | V, W^{(i)}, \phi^{(i)}, \sigma^{2(i)}\right).$$
6. Save $W^{(i)}, \phi^{(i)}, \sigma^{2(i)}, \tau^{(i)}$, set $i \leftarrow i + 1$, and go to step (2).

Once Gibbs sampling converges, we have obtained a collection of samples that together approximate the true posterior distribution $p\left(W, \phi, \sigma^2, \tau | V\right)$. For this model, the technique converges very quickly since all distributions described above are efficient to sample from. As an example, we illustrate the details in step (2) above.

Example 3. *To help illustrate, let us first ignore the spatial aspect and assume that there is only a single node. Suppose the sensor has taken a sequence of m readings. Among these, the base station does not observe k values and correctly receives $m - k$ values. We denote the k-dimensional vector corresponding to the missing values by $\mathbf{w}$, and the $(m - k)$-dimensional vector corresponding to the received values by $\mathbf{v}$. Our goal in step (2) above is to sample the distribution $p\left(\mathbf{w}|\mathbf{v}, \phi, \sigma^2, \tau\right)$, where we have omitted the superscript i (the Gibbs iteration index) for brevity.*

We shall partition $\mathbf{w}$ into r clusters *corresponding to contiguous sequences of missing values, where each sequence is surrounded by two received values. We indicate these clusters by*

$$\tilde{z_j} = [z_{t_j}, z_{t_j+1}, \ldots, z_{t_j+k_j-1}], \quad j = 1, \ldots, r,$$

where cluster j is a sequence of k_j consecutive missing values, and t_j is the timestamp of the first missing value in the cluster. Because the temporal evolution of the process is Markovian, we can sample $\mathbf{w}$ one cluster at a time, conditioned on the two known readings surrounding the cluster. Specifically, cluster j can be sampled from

$$p\left(\tilde{z_j} | y_{t_j-1}, y_{t_j+k_j}, \phi, \sigma^2, \tau\right),$$

where y_{t_j-1} and $y_{t_j+k_j}$ are the two known readings surrounding cluster j. It can be shown that this distribution is a k_j-variate Gaussian, which can be sampled using standard techniques.

Next, we show how to carry out step (2) of Gibbs sampling for the full spatio-temporal model. Recall that W denotes the set of missing readings over time across all nodes. For each node u, let $\mathbf{w}_u$ denote the set of missing readings over time for just node u. Instead of sampling all of W at the same time in step (2), we break

this step down into n mini-steps, where n is the number of nodes. In each step, we draw $\mathbf{w}_u$ for a single node u, conditioned on all the rest of missing values and the model parameters. We use the cluster-based technique described above for the single-node case, except now we take spatial correlation into account. Suppose we need to sample a cluster at node u that starts at time t and ends at time t'. The appropriate full conditional would be

$$p\left(Y[u,t:t']|Y[u,t-1],Y[u,t'+1],Y[-u,t:t'],\phi,\sigma^2,\tau\right),$$

where $Y[i,j]$ denotes the reading at node i at time j, $t:t'$ denotes the timestep range $[t,t']$, and $-u$, when used to index Y, denotes all nodes except u. Again, this distribution is simply a $(t'-t+1)$-variate Gaussian.

Although Gibbs sampling may be unnecessary in special cases (e.g., when closed-form analytic solutions exist), such simulation-based methods will be necessary for complex models. Also, as we will show next, once we incorporate knowledge of suppression and redundancy into analysis, inference will be further complicated by constraints over the posterior distribution; in this case, simulation-based methods become even more crucial.

4.2 Incorporating Suppression/Redundancy

Our novel extension to the basic Bayesian approach toward handling missing data is utilizing the knowledge about the suppression scheme. The key to incorporating such knowledge and any additional information from the redundancy scheme is the observation that we can effectively express them as additional *constraints* when sampling from the posterior. A sample is valid only if it satisfies all constraints derived from the suppression scheme and redundancy. Different types of redundancy translate into different types of constraints, which have different impact on the efficiency of sampling. In the following, we show, in progression, how to obtain the set of constraints $\mathcal{C}$ from the data obtained under different redundancy schemes, and how it affects our sampling strategy and efficiency.

$\mathsf{Supp}(\epsilon)$ and $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$ Consider the sequence from a node, $(y_0,y_1,y_2,y_3,y_4,y_5)$, where the base station receives y_0 and y_5, but y_1 through y_4 are missing. Intuitively, there is little we can do to constrain the possible settings of missing values in this case. Note, for example, we cannot conclude y_5 differs from y_4 by more than ϵ, because suppression is based on the last transmitted reading (as opposed to the reading from the previous timestep), and we have no way of knowing when that occurred.

Nevertheless, it is straightforward to code a constraint (detailed specification is rather tedious and omitted) that checks, given values y_1 to y_4, whether there would be a transmission at timestep 5 according to $\mathsf{Supp}(\epsilon)$. Unfortunately, this constraint is complex and non-linear. We check each sample with this constraint, and reject the sample if the constraint is violated. In practice, we found this additional constraint does not necessarily result in better accuracy (because the constraint is not very tight), but does increase the overhead of sampling significantly. Thus, in experiments (Section 5), we ignore the combination of $\mathsf{Supp}(\epsilon)$ and $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$ with BaySail, and use BayBase instead.

$\mathsf{Supp}(\epsilon)/\mathsf{C}$ Consider the same sequence as above, but now suppose the root receives counter values with y_0 and y_5, with difference between them equal to 3. Therefore, the base station can conclude that two failures have occurred among the four missing values. Unfortunately, we do not which of the missing values correspond to failures. Similar to the case above, we can code a constraint that checks, given values y_1 to y_4, whether there would exactly be two transmissions from timesteps 1 through 4 and one transmission at timestep 5. We found two problems with this approach. First, the probability of generating a valid sample, which requires that values are wide enough apart twice to induce the transmissions, is low; hence, many samples would be rejected. Second, because valid samples can place the two failures anywhere within the cluster, bounds on individual values are still not tight. For these reasons, this redundancy scheme is dominated by the two below, and we do not consider it further in experiments (Section 5).

$\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$ Now, suppose the base station also receives timestamp information piggybacked with y_5 that identifies y_1 and y_2 as suppressions, and y_3 and y_4 as failures. We can derive following constraints on the missing values:

$$\begin{array}{llll} |y_1-y_0|\le\epsilon, & |y_2-y_0|\le\epsilon, & & \\ |y_3-y_0|>\epsilon, & |y_4-y_3|>\epsilon, & |y_5-y_4|>\epsilon. & \end{array}$$

Each generated sample that meets all these constraints is consistent with the locations of suppressions and failures given by redundancy. This redundancy scheme leads to much tighter bounds in inference. Note, however, each constraint of the form $|a-b|>\epsilon$ represents the disjunction of two linear inequality constraints ($a-b>\epsilon \vee b-a>\epsilon$) because of the absolute value. Therefore, we do not quite have a system of linear constraints. Generating a valid sample that satisfies all these constraints is still a difficult task. In practice, we find the rejection rate to be quite high, especially for longer clusters of missing values. On the other hand, this redundancy does reduce uncertainty more than $\mathsf{Supp}(\epsilon)/\mathsf{C}$, so we still study it carefully in experiments (Section 5).

Note that the degree of redundancy r may not be high enough to distinguish all failures from suppressions in long sequences of missing data. In that case, we treat the part for which we have no failure/suppression information as discussed earlier for $\mathsf{Supp}(\epsilon)$ and $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$.

$\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ Finally, suppose that the base station receives direction bits 1 for each of y_3 and y_5 (indicating each of them increased from the previous transmission) and 0 for y_4 (indicating it decreased). We can now derive the following constraints:

$$\begin{array}{lll} -\epsilon\le y_1-y_0\le\epsilon, & -\epsilon\le y_2-y_0\le\epsilon, & \\ y_3-y_0>\epsilon, & y_3-y_4>\epsilon, & y_5-y_4>\epsilon. \end{array}$$

Although sampling from a constrained multivariate distribution in general is a difficult task, in this case the distribution is Gaussian (Example 3) and all constraints are linear. This combination enables us to apply a very efficient sampling method from [26]. Suppose we need to sample a vector $\mathbf{y}$ from a k-variate Gaussian subject to a system $\mathcal{C}$ of m linear inequality constraints, i.e.:

$$\mathbf{Y}\sim N_{\mathcal{C}}(\boldsymbol{\mu},\sigma^2\boldsymbol{\Sigma}),\ \text{where } \mathcal{C}=\{\mathbf{y}\in R^k:\mathbf{B}\mathbf{y}\le\mathbf{b}\}.$$

Here, the $m\times k$ matrix $\mathbf{B}$ and the vector $\mathbf{b}$ of size m together specify all constraints. In the example above, we have

$$\mathbf{B}=\begin{bmatrix}-1&0&0&0\\1&0&0&0\\0&-1&0&0\\0&1&0&0\\0&0&-1&0\\0&0&-1&1\\0&0&0&1\end{bmatrix},\ \mathbf{y}=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix},\ \mathbf{b}=\begin{bmatrix}-y_0+\epsilon\\y_0+\epsilon\\-y_0+\epsilon\\y_0+\epsilon\\-y_0-\epsilon\\-\epsilon\\y_5-\epsilon\end{bmatrix}.$$

The idea is to transform the sample space in such a way that we can easily sample each component of the vector sequentially. To this end, let $\mathbf{A}$ be a $k\times k$ matrix of full rank (obtained using Cholesky decomposition) that satisfies $\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T=\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. We transform $\mathbf{y}$ to $\mathbf{z}$ as follows: $\mathbf{z}=\mathbf{A}\mathbf{y}$. Then, using a

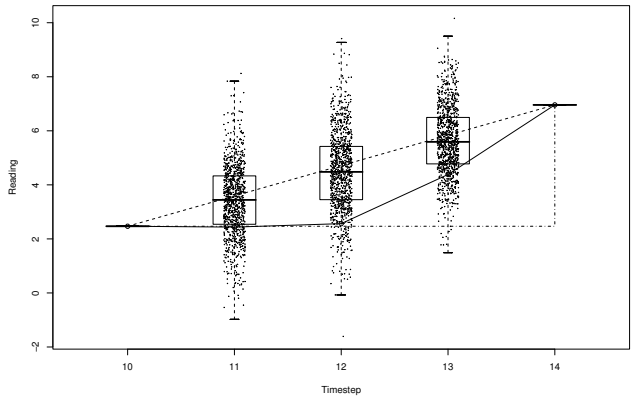

Figure 5: $Suppress(1.0)$**+BayBase.**

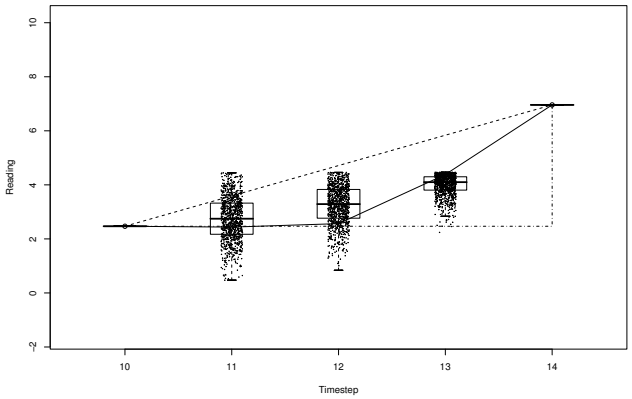

Figure 6: $Suppress(1.0)/\mathsf{TD}(8)$**+BaySail.**

well-known property of the multivariate Gaussian, we have:

$$\mathbf{Z} \sim N_{\mathcal{C}'}(\mathbf{A}\boldsymbol{\mu}, \sigma^2\mathbf{I}), \text{ where } \mathcal{C}' = \{\mathbf{z} \in R^k : \mathbf{B}\mathbf{A}^{-1}\mathbf{z} \leq \mathbf{b}\}.$$

Note that components of $\mathbf{z}$ are now independent of each other, which makes sampling easier. Applying the Gibbs sampling idea again, we sample each component z_j of $\mathbf{z}$ in turn conditioned on all other components. Specifically, we draw z_j from the univariate Gaussian $N(\alpha_j, \sigma^2)$, where α_j is the j-th component of $\mathbf{A}\boldsymbol{\mu}$, subject to $\mathcal{C}'$. With all other components fixed, this system of constraints can be simplified into an interval bound on z_j. Thus, the task boils down to sampling from a truncated univariate Gaussian, which does not involve expensive rejection. Finally, each sample of $\mathbf{z}$ we have obtained is transformed back into the original space with $\mathbf{y} = \mathbf{A}^{-1}\mathbf{z}$.

Beyond Value-Based Temporal Suppression Suppose a node bases suppression not by evaluating $|y_t - \bar{y}_t| \leq \epsilon$, but more generally by $|y_t - f(\bar{y}_t)| \leq \epsilon$, where f predicts the current reading based on $\bar{y}_t$, data that has been transmitted previously. Techniques for deriving constraints can be extended in a straightforward manner. The efficient sampling method for the $\mathsf{TD}(r)$ case also applies when f is linear. However, when f is non-linear we may still have to resort to rejection sampling. We believe that in practice most suppression schemes will be linear, because they run on sensor nodes and therefore cannot be very complex or expensive to evaluate. Nevertheless, it would be interesting to investigate more efficient inference techniques for non-linear constraints.

5 Experimental Evaluation

The goal of this section is to compare the presented reporting and inference schemes. We examine the fundamental trade-off between energy cost and quality of inference to determine how well each scheme exploits its energy spent (i.e., the most "bang-for-the-buck") and to understand the influence of process variance and failure rate. We also compare the improvement made in BaySail when shifting from rejection-based to direct sampling, emphasizing the importance of out-of-network efficiency. Finally, we show results obtained by spatial inference for a spatio-temporal model.

We have evaluated the presented reporting and inference schemes through a combination of implementation and simulation.

Implementation As stated, one advantage of application-level redundancy is its ease of implementation. In fact, we have implemented suppression and redundancy in TinyOS with a matter of less than 50 lines of additional code on top of an existing tutorial application [2], and successfully executed the resulting applications on Mica2 motes [7]. We have also implemented lower-level redundancy with acknowledgments and retransmissions; while also not requiring much additional code, this effort necessitated understanding lower-layer code. Finally, in executing these applications, we have observed message failure rates that justify the settings made in experiments.

We have implemented BayBase and BaySail (using both direct and rejection-based sampling) in R [1]. These run on a standard computer that serves as the base station of the sensor network.

Simulation The above efforts supply all necessary in-network and out-of-network code. Nevertheless, to fully control the experimental environment (i.e., maintaining consistency across runs, verifying inferred results against true readings and process parameters, varying parameters and failure rate, etc.), we simulate raw sensor readings and transmission failures using the soil moisture model given in Section 2. This, rather than an actual sensor node, provides input to the BaySail or BayBase inference code.

We focus on a subset of reporting/analysis schemes:

$\mathsf{Samp}(1/\tau) + \mathsf{BayBase}$, $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r) + \mathsf{BayBase}$,
$\mathsf{Supp}(\epsilon)/\mathsf{T}(r) + \mathsf{BaySail}$, $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r) + \mathsf{BaySail}$.

Packet Sizes In general, our goal is to produce as precise an inference as possible, while transmitting as few bytes as possible. The packet sizes, taken from standard Tiny OS message types and the B-MAC protocol [25], are as follows:

Component	Size (bytes)
Lower-layer overhead	4
Application	6
Acknowledgment	5
Timestamp(r)	r
Direction(r)	$r + 1$ bits

All schemes must transmit at least the overhead and application components for each message. For $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$, the receiver must also transmit an acknowledgment, while the sender potentially must retransmit the overhead and application portions. Finally, $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$ augments its messages with r additional timestamps; $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ adds both timestamps and direction bits.

Missing Data Series As a warm-up, it is helpful to visualize the benefit gained with $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ and BaySail. We compare competing reconstructions of a cluster of three missing values, each of which is a suppression. The two endpoints (at time 10 and 14) have been received by the root. The value-based temporal scheme uses $\epsilon = 1.0$. Figure 5 plots samples of the posterior distribution for each of the three values using BayBase. These are conditioned only on the endpoints and process parameters. Figure 6 plots samples for the same three values, but using BaySail,

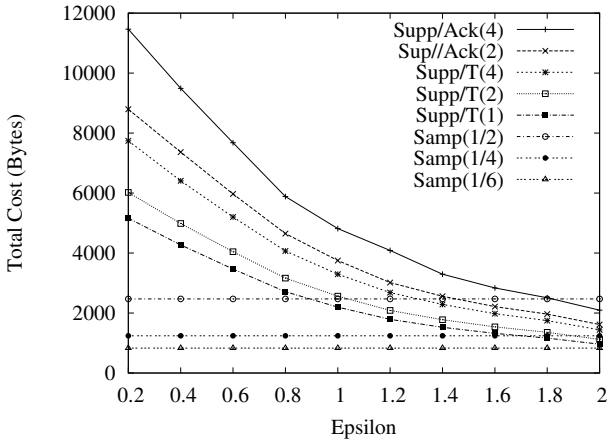

Figure 7: Reporting cost comparison.

with sufficient redundancy to identify each value as a suppression. Again, the depicted boxes stretch from the 25th to 75th quartiles of each posterior distribution. Reconstructions using linear interpolation and stepwise constant interpolation are shown on both figures with dashed black lines (linear the upper, stepwise the lower). The true sequence of values is shown with a solid black line.

The benefit of redundancy is clear between Figures 5 and 6. When the base station knows nothing about the missing values, it must produce a wide range of samples for each timestep. When it knows the values are suppressions, the samples are concentrated in a tighter, bounded range. Note that while no failures occur in this sequence, the possibility that they may have adds uncertainty to BayBase's inference. BaySail, which alone rules out the possibility, dramatically reduces uncertainty.

Linear interpolation assumes the missing readings increase at a constant rate between two received readings. Stepwise constant interpolation assumes the readings remain unchanged until the next received one; it effectively acknowledges the suppression scheme, but ignores failure, assuming none has occurred. The true sequence, as shown in the figure, is somewhere in between. Neither of these approaches gives measures of uncertainty, which is troublesome given they are often simply incorrect.

This example is subjective, and does not reflect that we must pay an additional cost for the redundancy that tells us the missing values are suppressions. To formally evaluate these approaches, we must compare their energy costs and the quality of their inferences. We conduct such comparisons over the remainder of this section.

5.1 Quantitative Analysis

We measure scheme cost as the number of bytes transmitted over the course of time. The energy spent by the network is tied to the number and size of transmissions. Measuring quality of inference is trickier. A simple option for generated samples of either missing readings or process parameters is to compare the error between the mean of the posterior distribution and the actual value. This comparison can be misleading sometimes; for example, the mean of a bimodal distribution is very unlikely to be near the true value. We use the alternative of *high density region (HDR) interval length*, which provides a measure of uncertainty in a distribution for a reading or parameter. Informally, $HDR(f_Z, \theta)$ finds, for the probability density function f_Z of a random variable Z, a set of intervals such that the probability of Z lying in any of these intervals is θ (where $0 < \theta \leq 1$), and the total length of the intervals is minimal. The lower the total length, the more certain the fit.

Reporting Costs We first examine reporting costs absent of subsequent analysis. We use a series of 500 readings, generated with process parameters $\phi = 0.9$ and $\sigma^2 = 1.0$. Figure 7 plots ϵ versus bytes transmitted for a number of reporting schemes. We plot a number of r and τ settings. Our first task is to understand the relationship between sampling and suppression, without redundancy. We see that $\mathsf{Supp}(\epsilon)/\mathsf{T}(1)$, which has almost no redundancy, crosses $\mathsf{Samp}(1/2)$ between $\epsilon = 0.8$ and 1, and continues to decrease in cost as ϵ further increases. In general, the lower ϵ, the more often $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$ and $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$ transmit. $\mathsf{Samp}(1/\tau)$ is obviously unaffected by ϵ, transmitting at a regular interval regardless.

We next consider the marginal cost of redundancy. As evidenced by $\mathsf{Supp}(\epsilon)/\mathsf{T}(r)$, the cost of adding additional timestamps is low compared to varying ϵ. The price of redundancy appears higher for $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$. Our strategy of piggybacking redundancy on existing messages appears cost-effective, though this claim is not complete until we examine quality of inference. Because $\epsilon = 1.0$ appears to be a good point of comparison across reporting schemes, we use it in the next few experiments. Intuitively, lower ϵ values improve inference for suppression schemes, but may induce too much reporting to make a reasonable comparison against sampling.

Cost vs. HDR We now measure the trade-off between reporting cost and HDR length as a measure of inference accuracy. Carrying over the process settings from the previous experiment and $\epsilon = 1.0$, we have produced three time series of 500 readings, and present results averaged over these. The average mean and standard deviation over the readings are 0.94 and 3.19. For suppression, on average, transmissions are attempted in 40% of the timesteps, for a rate slightly lower than $\mathsf{Samp}(1/2)$. Figures 8 and 9 plot the trade-off for the three schemes. The former plots this trade-off averaged over the inferred samples for missing raw readings, while the latter averages over the samples for model parameter ϕ. Note that the number of bytes transmitted, shown on the horizontal axes, is a variable dependent on the redundancy level (r or τ). Since redundancy level settings are not equivalent across reporting schemes, we use bytes transmitted to make a fair comparison. For clarity, we label each plotted point with its redundancy level. HDR covers 80% of the posterior distribution. Finally, note that measuring HDR length is only useful if the inference is correct; we have verified that the HDRs indeed capture the actual value 80% of the time.

Both figures show from best to worst performance, a relative ordering of $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$, $\mathsf{Samp}(1/\tau)$, and $\mathsf{Supp}(1.0)/\mathsf{Ack}(r)$. The magnitude of improvement is far more dramatic for the missing readings than ϕ. In the case of the missing readings, we make two major observations. First, as initially suspected, the use of retransmissions and acknowledgments incurs significant extra cost beyond sampling. Application-level redundancy, piggybacked on existing messages, incurs only slight cost beyond sampling. The second observation is that as we increase the number of allowable retransmissions, and push the effective failure rate toward zero (evidenced by the fact that increasing retransmissions beyond 4 does not greatly increase cost), uncertainty does not continue to decrease. The reason is that even as we intuitively expect most non-reports to be suppressions, the base station still cannot ascertain the cause for any particular non-report. Because of difficulty in devising good failure models, there is no principled way to assign suppression/failure probabilities. In contrast, with $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$, when given a sufficient redundancy level, we know definitively whether non-reports are suppressions or failures. This knowledge reduces uncertainty to a degree that $\mathsf{Supp}(\epsilon)/\mathsf{Ack}(r)$ cannot achieve.

Though Figure 9 shows the same relative order of performance among schemes, the small magnitude of improvement suggests for

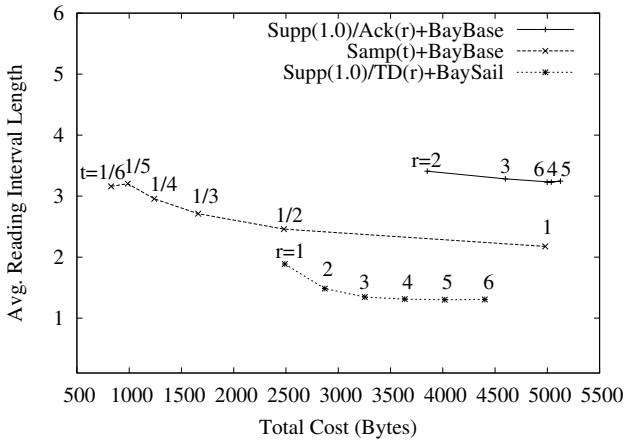

Figure 8: Cost vs. HDR length for missing readings.

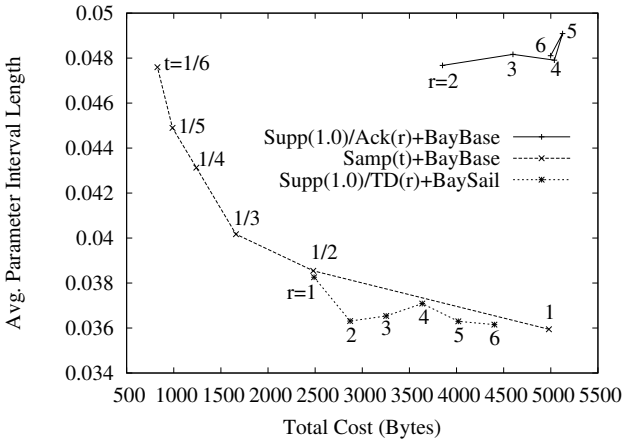

Figure 9: Cost vs. HDR length for ϕ.

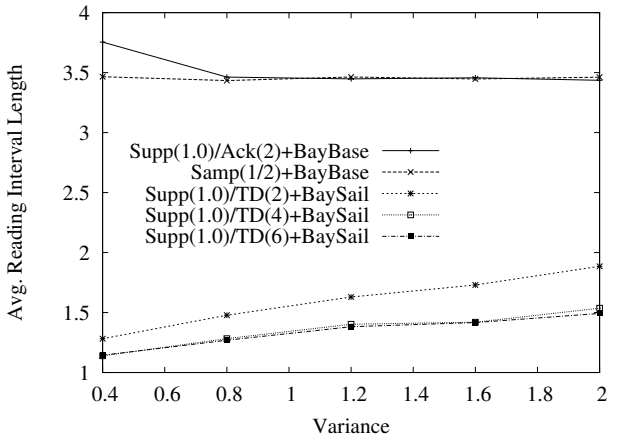

Figure 10: Model variance vs. HDR length.

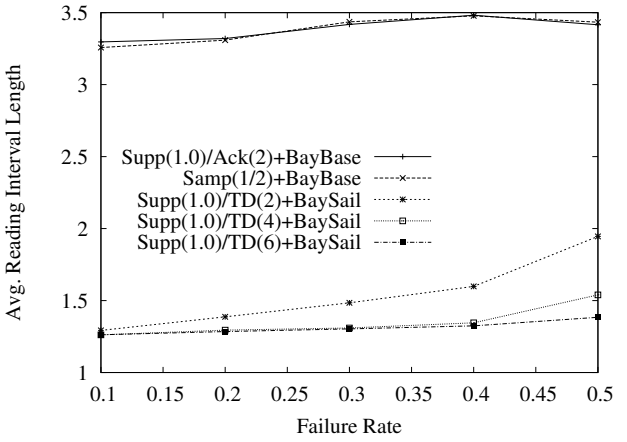

Figure 11: Failure rate vs. HDR length.

purposes of inferring process parameters, it does not matter which is used. We do not yet know if this observation is an artifact of the process in use, or universally true. In either case, if a system user is only interested in process parameters, there is no need to support collection of all readings; we can likely design lower-cost algorithms using in-network processing to just return parameters.

Process Variability and Failure Rate The next set of experiments examine the influence of process variance and transmission failure rate on performance. Intuitively, as they increase, we expect performance to degrade.

Figure 10 plots variance versus average HDR length for a number of $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$ settings, as well as $\mathsf{Samp}(1/2)$ and $\mathsf{Supp}(1.0)/\mathsf{Ack}(2)$ for reference. We see that the advantage of $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$ over the other schemes carries across different variance values. Across all $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$ settings, as variance increases, HDR length increases. Increased variance induces an increased number of transmissions and, as a consequence, failures. Uncertainty is always greater for missing values identified as failures. In addition, when redundancy is not sufficient to identify some missing values, uncertainty increases. Hence, the effect of variance is dampened in higher redundancy settings.

Figure 11 plots failure rate versus average HDR length for $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$, as well as $\mathsf{Samp}(1/2)$ and $\mathsf{Supp}(1.0)/\mathsf{Ack}(2)$. As for variance, $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$'s advantage holds, and HDR length increases with failure rate. Once again, while all schemes are negatively affected by higher failure rates, higher redundancy rates dampen the effect.

5.2 Effect of Redundancy on Inference

As discussed in Section 4.2, one important factor that needs to be considered in choosing the type of redundancy for BaySail is how it affects the efficiency of out-of-network inference. If the type of redundancy added translates into constraints that are difficult to work with (e.g., C and $\mathsf{T}(r)$), we may need to resort to sampling by rejection. As the length of a cluster increases, it becomes increasingly difficult to draw samples that satisfy all constraints, so sampling takes longer. In our implementation, after a specified number of unsuccessful attempts to generate a valid sample (which we call *rejection threshold*), we have to give up and generate a sample that assumes nothing about the missing values. In contrast, $\mathsf{TD}(r)$ adds just an additional byte per message (up to $r = 8$) compared with $\mathsf{T}(r)$, and $\mathsf{TD}(r)$ allows to use a much more efficient direct sampling method without rejection. We now examine the advantages gained by moving from $\mathsf{T}(r)$ to $\mathsf{TD}(r)$.

Figure 12 plots ϵ versus running time (log-scaled) for $\mathsf{Supp}(1.0)/\mathsf{T}(r)$ and $\mathsf{Supp}(1.0)/\mathsf{TD}(r)$. The redundancy is sufficient to identify all non-reports as suppressions or failures. The main point to discern is that the direct sampling method allowed by $\mathsf{TD}(r)$ is at least two orders of magnitude faster than rejection-based sampling used in conjunction with $\mathsf{T}(r)$. Direct sampling typically finishes in 5 minutes. Note that ϵ and running time are directly correlated. As ϵ increases, the number of transmissions decreases, resulting in longer clusters of missing values, and therefore longer processing time. The effect is much more severe for rejection-based sampling. The plotted points for $\mathsf{Supp}(1.0)/\mathsf{T}(r)$

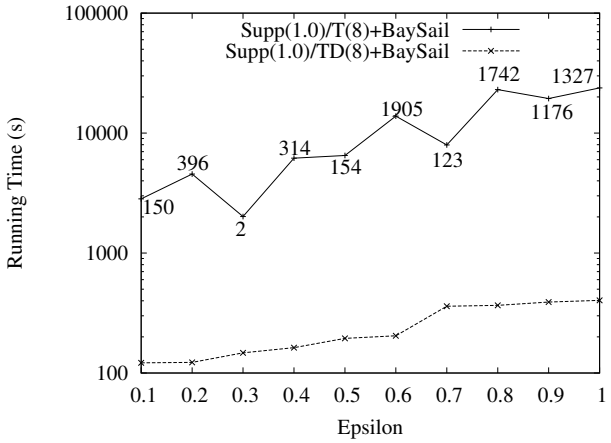

Figure 12: $\mathsf{T}(r)$ and $\mathsf{TD}(r)$: ϵ vs. running time.

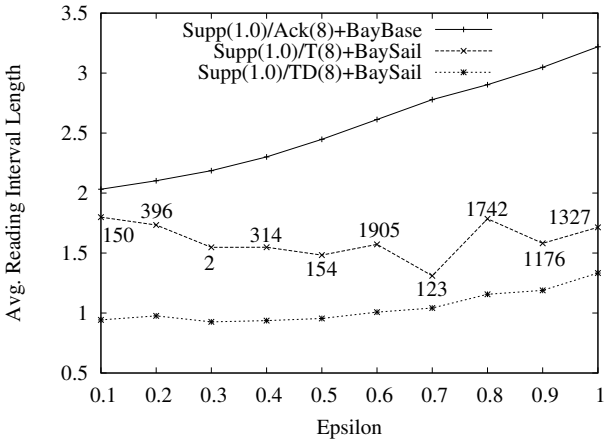

Figure 13: $\mathsf{T}(r)$ and $\mathsf{TD}(r)$: ϵ vs. HDR length.

are labeled with the number of times for which we hit the rejection threshold (10000 in this case). Naturally, running time suffers the longer BaySail is tied up generating samples. We see some "lucky" cases where $\epsilon = 0.3$ and 0.7. Direct sampling does not suffer from this erratic behavior.

$\mathsf{TD}(r)$, through use of direction bits, also has the advantage of reducing the possible ranges in which missing values can be. Figure 13 plots ϵ versus HDR length for the non-reported readings. $\mathsf{TD}(r)$ provides significantly less uncertainty than $\mathsf{T}(r)$. Note both outperform BayBase. Once again, rejection-based sampling is erratic, with "lucky" points providing tighter HDR.

$\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ satisfies the dual criteria of developing redundancy that 1) is lightweight to deploy and results in small number and size of transmissions, and 2) allows for computationally efficient out-of-network statistical inference. We moved to this non-intuitive scheme after progressing through schemes that satisfied the first criterion, but not the second. The improvement is dramatically shown in Figure 12. For minuscule additional in-network cost (still meeting the first criterion), we improve on the second criterion by orders of magnitude.

5.3 Spatial Inference

All experiments presented so far have focused on a single node. We now examine a spatial version of BaySail involving multiple nodes, and verify the intuition that reports from nearby nodes should serve as spatial redundancy and reduce uncertainty. In this experiment, 9 nodes are laid in a 3×3 grid, and each node independently runs $\mathsf{Supp}(1.0)/\mathsf{TD}(8)$. Figure 14 compares average HDR length for missing readings at each of the nodes using standard single-node BaySail and the spatial version. We do observe a improvement across all nodes, achieved through spatio-temporal inference, rather than solely temporal inference. Further, observe the "corner" nodes 1, 3, 6, and 9, with fewer neighbors in the grid, leverage spatial redundancy the least and generally show the least improvement. Meanwhile, the center node 5 shows the most improvement.

Note that while inference is spatial in this experiment, the suppression scheme is not. Recall from Section 3 that we foresee spatio-temporal suppression schemes where nodes communicate in-network to remove redundancy. Work on incorporating such suppression schemes in BaySail is still ongoing.

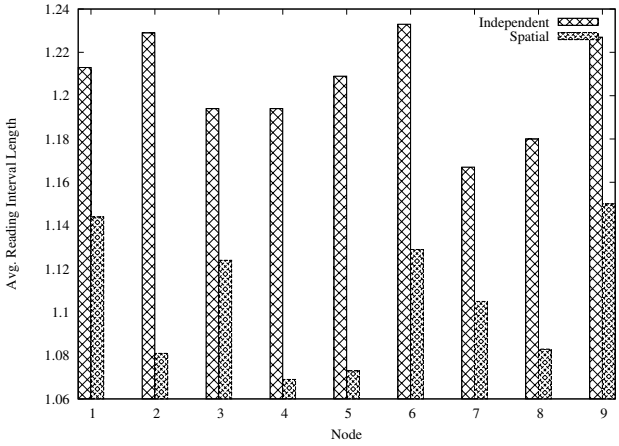

Figure 14: Independent vs. spatial inference.

6 Related Work

Suppression Temporal suppression as described in this paper has roots in adaptive range caching [24]. In this server-cache setting, caches field queries for values, while the server fields value updates. To minimize cache updates, each cache stores a range (simultaneously known at the server) that encompasses a value. The server only transmits updates when a value escapes its cached range. This work is followed by complementary work where the data sources are distributed [23]. Similar work has emerged for sensor networks, now exploiting the spatio-temporal correlation found in deployments [18, 28, 5, 27]. Using different strategies, these approaches avoid transmission when readings can be accurately predicted.

Missing Data There exist a number of ways to address missing data without classifying them as suppressed or failed. The simplest is to interpolate missing values from the reported ones in some fashion. For example, linear regression fits lines using reported values. Data may not realistically transition in this fashion, however. The main weakness of interpolation is the lack of uncertainty measurement it gives in reconstruction. At times, interpolation may provide results that are close to, for example, the posterior mean computed by Bayesian techniques. Nevertheless, interpolation by itself provides no measure of confidence in its results, which is problematic to statistical modelers who need to analyzes the data subsequently.

Conceptual reconstruction [3] seems to also assume that missing data is result of random deletion (at least from the experiments). It assumes that the remaining data is somehow "representative" of the original data, which in general may not be true if the missing data is *engineered*, as with suppression.

The *Ken* [5] framework maintains node models in and out of network, using suppression to only notify the root when there are significant changes. This approach is as vulnerable to failure as ba-

sic suppression. They rely on the Markovian nature of their models to assume any failures will eventually be corrected with future updates, and not be affected by the missed ones. They propose periodic heartbeat updates to ensure models cannot be incorrect indefinitely. This approach is not suitable for raw value reconstruction; for any timestep where the model has suffered from failures and is incorrect, the corresponding raw value samples will be wrong. Our use of redundancy lets us tolerate incorrect stretches of data. For parameter inference, we allow for more general models where past data changes do affect them and cannot be neglected.

Deshpande et al. [10] suggest Bayesian methods for learning a model of network behavior over time, such that the model can eventually be largely substituted for actual network data in query processing. Occasional pull-based acquisitions are needed when the model alone is not sufficient. This can be viewed as an alternative strategy for supporting queries without continuous messaging. Since messaging is pull-based, any non-reports are detected as failures. In BaySail we never trust models enough to use them in lieu of data; rather, we use them in conjunction with the low-cost, but valuable information we get from redundancy.

Constraints Khoussainova et al. [20] suggest an RFID scenario similar to ours. RFID readers provide constant streams of tags in their area, but can sometimes miss reporting some tags. Thus, if a particular tag is read by no readers in some timestep, the tag presumably may be anywhere. They provide constraints to aid in query processing such as, if the tag was present at a reader one timestep in the past, the tag is said to still be present there with 30% probability. These constraints are chosen apart from the data; in our case, the constraints used in reconstruction are drawn from the network itself as redundancy along with data. This approach also does not address historical reconstruction. We argue a temporal model supporting historical reconstruction, in the style of BaySail, would be very useful. For example, in the case a tag is assigned a 30% chance of being present in the first timestep the tag is missing, if it reappears in subsequent timesteps, the probability can be retroactively raised; likewise, if it does not reappear, it can be lowered.

ESP [19] is a language for building methods for cleaning failure-affected, or "dirty," sensor data. The language supplies a series of cleaning methods and the user can declare constraints that should hold during processing. Once again, in our setting constraints are not supplied by the user, but are in the data received from the network. Furthermore, we make use of constraints while doing data reconstruction and model parameter inference simultaneously, instead of performing these tasks in stages, as discussed in Section 2. Nevertheless, the language proposed by ESP may be helpful in further post-processing, e.g., transforming the results of BaySail inference to a format of users' choosing.

Failure Reduction Our approach is to make sensor networks more robust to failure by reducing the system's reliance on reliable message delivery. The complementary approach, typically handled at lower network levels, is to increase the probability transmitted messages reach the root. A fair amount of work has been done on identifying the causes of failure and suggesting methods for reducing it. One approach is to identify reliable communication edges, and limit the network topology to using only them [30, 31]. Hull et al. point to congestion as a major cause of failure and suggest several techniques for lowering nodes transmission rates and ensuring fairness [17]. This latter finding was from an application constantly generating messages, while we expect suppression to reduce the threat of congestion. These lower-level efforts are orthogonal to our approach, and it is certainly possible to use them in conjunction. The important point is all efforts that add cost to deal with failures (i.e. application-layer redundancy and MAC-layer retransmission) must be aware of each other to coordinate their target point in the trade-off between energy cost and accuracy.

Sensor Application Robustness There has been a great deal of work on making sensor network applications more robust to failure. Two examples are the *sketch* [6] and *synopsis diffusion* [22]. These approaches support duplicate-insensitive aggregate computation. While computations such as `MAX` are easy, aggregates such as `SUM` and `COUNT` are non-trivial. Nodes can now transmit multiple copies of values to hedge against failure, while not affecting the final result. These techniques do not lend themselves to non-aggregate data collection. Neither structure can be used to recover the particular values inserted into them. Further, while in-network aggregation allows us to send multiple copies of values, and still bound the size of messages to be constant or quite small compared to network size, we do not have that luxury here.

The *tributary-delta* approach [21] addresses the trade-off between standard tree-based aggregation, which risks losing an entire subtree's contribution due to a single failure, and duplicate-insensitive synopsis-based aggregation, which is more robust against failure but uses larger messages and provides only approximate answers. The intuition is to initially transmit single copies, but send multiple copies near the top of the routing tree, when each aggregate is constituted from many values. The overall number of extra messages amortized over the number of initial single values is low. Once again, this approach is not applicable in our case, where we cannot aggregate. Nevertheless, the approach provides some insight into the idea of choosing different redundancy levels for different nodes, which we currently do not do but plan to investigate as future work.

Database Support for Uncertain Data The emergence of sensor networks, as well as other venues where data collection is itself an uncertain process, has drawn interest in representing probability, uncertainty, and lineage in databases. Some examples include *MistiQ* [9], *MauveDB* [12], and *Trio* [4]. While out of the scope of this paper, determining how to represent data collected and processed with suppression and BaySail is an interesting problem. Our processing gives us a wealth of derived probabilistic data, backed by a small amount of received messages and the suppression scheme itself. The problem of efficiently storing this data and metadata and making them available for query processing is challenging.

7 Conclusion

Continuous data collection is an important problem in sensor networks. Continuous reporting is not energy-efficient. In contrast, suppression can drastically cut down the number of reports, and let the root derive non-reported readings within $\pm\epsilon$ of the actual. Sensor networks are prone to message failure, however, which creates ambiguity at the root: Is a non-report a suppression or failure?

In order to develop a principled approach to resolving uncertainty in the presence of failures, we have proposed BaySail. This framework combines in-network application-level redundancy and out-of-network statistical inference to produce a posterior distribution for missing readings and model parameters we seek to estimate. Our approach is Bayesian and provides a measure of uncertainty that single-point estimates cannot. Novel to Bayesian inference, it exploits redundancy and knowledge of the suppression scheme, leading to much lower result uncertainty. We have investigated several approaches to adding redundancy back to reporting, and found that carefully designed application-level redundancy is simple to implement and more effective than sampling or lower-level acknowledgments and retransmissions. Among the schemes considered, we have found $\mathsf{Supp}(\epsilon)/\mathsf{TD}(r)$ to offer the best combination of low transmission costs and precise, efficient inference.

Future Work We believe the BaySail framework is general enough to support any suppression scheme with redundancy; it is simply a matter of properly setting inference constraints. Nevertheless, as apparent from our experience in the work presented here, finding the right scheme is non-trivial. We plan to address more sophisticated spatio-temporal schemes to leverage their higher potential for reducing communication. These schemes will require new strategies for redundancy and inference. We must be able to cope not only with failure of transmissions to the base station, but also within the network, among nodes that use non-local values in making suppression decisions. In model-based suppression, we may want to dynamically adjust the suppression model, which requires updated model parameters to be transmitted between a node and the root; in this case, we must cope with not only missing readings, but also parameter updates. Our work on these problems is still ongoing.

References

[1] R project for statistical computing. http://www.r-project.org.

[2] TinyOS. http://www.tinyos.net.

[3] C. C. Aggarwal and S. Parthasarathy. Mining massively incomplete data sets by conceptual reconstruction. In *Proceedings of the 2001 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 227–232, San Francisco, California, USA, Aug. 2001.

[4] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *Proc. of the 2006 Intl. Conf. on Very Large Data Bases*, Seoul, South Korea, Sept. 2006.

[5] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Proc. of the 2006 Intl. Conf. on Data Engineering*, Atlanta, Georgia, USA, Apr. 2006.

[6] J. Considine, F. Li, G. Kollios, and J. Byers. Approximation aggregation techniques for sensor databases. In *Proc. of the 2004 Intl. Conf. on Data Engineering*, Boston, Massachusetts, USA, Mar. 2004.

[7] Crossbow Inc. *MPR-Mote Processor Radio Board User's Manual*, 2003.

[8] Crossbow Inc. *Smart Dust Application Note*, 2004.

[9] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *Proceedings of the 2004 International Conference on Very Large Data Bases*, pages 864–875, Toronto, Canada, Sept. 2004.

[10] A. Deshpande, C. Guestrin, and S. Madden. Using probabilistic models for data management in acquisitional environments. In *Proc. of the 2005 Conf. on Innovative Data Systems Research*, Asilomar, California, USA, Jan. 2005.

[11] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proc. of the 2004 Intl. Conf. on Very Large Data Bases*, Toronto, Canada, Aug. 2004.

[12] A. Deshpande and S. Madden. MauveDB: Supporting model-based user views in database systems. In *Proc. of the 2006 ACM SIGMOD Intl. Conf. on Management of Data*, Chicago, Illinois, USA, June 2006.

[13] M. Franklin, S. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu, O. Cooper, A. Edakkunni, and W. Hong. Design considerations for high fan-in systems: The HiFi approach. In *Proc. of the 2005 Conf. on Innovative Data Systems Research*, Asilomar, California, USA, Jan. 2005.

[14] A. Gelfand, S. Banerjee, and D. Gamerman. Spatial Process Modelling for Univariate and Multivariate Dynamic Spatial Data. *Environmentrics*, 16:465–479, 2005.

[15] A. Gelfand and A. Smith. Sampling Based Approaches to Calculating Marginal Densities. *Journal Amer. Stat. Assoc.*, pages 398–409, 1990.

[16] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2003.

[17] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proc. of the 2004 SENSYS*, Baltimore, Maryland, USA, Nov. 2004.

[18] A. Jain, E. Chang, and Y. Wang. Adaptive stream resource management using Kalman filters. In *Proc. of the 2004 ACM SIGMOD Intl. Conf. on Management of Data*, Paris, France, June 2004.

[19] S. Jeffery, G. Alonso, M. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. In *Proc. of the 2006 Intl. Conference on Pervasive Computing*, Dublin, Ireland, May 2006.

[20] N. Khoussainova, M. Balazinska, and D. Suciu. Towards correcting input data errors probabilistically using integrity constraints. In *Proc. of the 2006 ACM Workshop on Data Engineering for Wireless and Mobile Access*, Chicago, Illinois, USA, June 2006.

[21] A. Manjhi, S. Nath, and P. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *Proc. of the 2005 ACM SIGMOD Intl. Conf. on Management of Data*, Baltimore, Maryland, USA, June 2005.

[22] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. of the 2004 SENSYS*, Baltimore, Maryland, USA, Nov. 2004.

[23] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proc. of the 2003 ACM SIGMOD Intl. Conf. on Management of Data*, San Diego, California, USA, June 2003.

[24] C. Olston, B. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In *Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of Data*, Santa Barbara, California, USA, May 2001.

[25] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. of the 2004 SENSYS*, Baltimore, Maryland, USA, Nov. 2004.

[26] G. Rodriguez-Yam, R. Davis, and L. Scharf. Efficient Gibbs sampling of truncated multivariate normal with application to constrained linear regression. Technical report, Colorado State University, 2004.

[27] A. Silberstein, R. Braynard, and J. Yang. Constraint-chaining: On energy-efficient continuous monitoring in sensor networks. In *Proc. of the 2006 ACM SIGMOD Intl. Conf. on Management of Data*, Chicago, Illinois, USA, June 2006.

[28] D. Tulone and S. Madden. PAQ: Time series forecasting for approximate query answering in sensor networks. In *Proc. of the 2006 EWSN*, Zurich, Switzerland, Feb. 2006.

[29] M. West and P. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer, 1997.

[30] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. of the 2003 SENSYS*, Los Angeles, California, USA, Nov. 2003.

[31] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of the 2003 SENSYS*, Los Angeles, California, USA, Nov. 2003.