MIST: Distributed Indexing and Querying in Sensor Networks using Statistical Models

Arnab Bhattacharya
Dept. of Computer Science,
University of California,
Santa Barbara
arnab@cs.ucsb.edu

Anand Meka
Dept. of Computer Science,
University of California,
Santa Barbara
meka@cs.ucsb.edu

Ambuj K. Singh
Dept. of Computer Science,
University of California,
Santa Barbara
ambuj@cs.ucsb.edu

Abstract

The modeling of high level semantic events from low level sensor signals is important in order to understand distributed phenomena. For such *content-modeling* purposes, transformation of numeric data into symbols and the modeling of resulting symbolic sequences can be achieved using statistical models—Markov Chains (MCs) and Hidden Markov Models (HMMs). We consider the problem of distributed indexing and semantic querying over such sensor models. Specifically, we are interested in efficiently answering (i) *range* queries: return all sensors that have observed an unusual sequence of symbols with a high likelihood, (ii) *top-1* queries: return the sensor that has the maximum probability of observing a given sequence, and (iii) *1-NN* queries: return the sensor (model) which is most similar to a query model. All the above queries can be answered at the centralized base station, if each sensor transmits its model to the base station. However, this is communication-intensive. We present a much more efficient alternative—a distributed index structure, *MIST* (Model-based Index STructure), and accompanying algorithms for answering the above queries. MIST aggregates two or more constituent models into a single composite model, and constructs an in-network hierarchy over such composite models. We develop two kinds of composite models: the first kind captures the *average* behavior of the underlying models and the second kind captures the *extreme* behaviors of the underlying models. Using the index parameters maintained at the root of a subtree, we bound the probability of observation of a query sequence from a sensor in the subtree. We also bound the distance of a query model to a sensor model using these parameters. Extensive experimental evaluation on both real-world and synthetic data sets show that the MIST schemes scale well in terms of network size and number of model states. We also show its superior performance over the centralized schemes in terms of update, query, and total communication costs.

1 Introduction

Large scale sensor networks are being deployed for applications such as habitat monitoring [19], seismic monitoring [2], and location tracking systems [17]. As sensors become more inexpensive and more easily deployable, individual measurements will pave the way to high level semantically rich events, directly mined from the raw and noisy sensor data. Given the potentially huge amount of data streamed by sensors, algorithms to extract and interpret the semantics will become an integral part of *content-summarization* in networked sensor applications. For example, the sound, humidity and light sensors on a MICA mote [1] can sense whether a room is empty or occupied; similarly, a temperature and chemical sensor can sense the presence of fire. In the Zebranet project [16], scientists tied acceleration sensors to zebras' collars in order to observe their movements. This enabled the scientists to characterize a zebra's movement in terms of the three main states: grazing, walking and fast moving.

The transformation from sensor readings to symbolic states can be achieved at a central node that collects readings from the entire network [16]. This paper explores a much more efficient alternative: first transform-

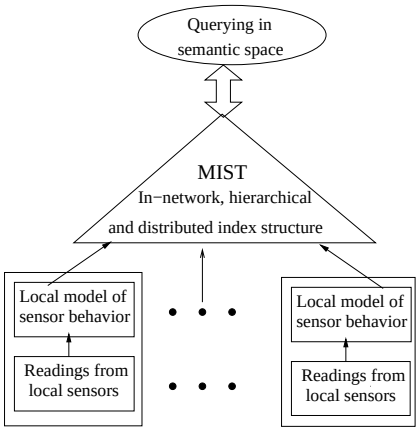

Figure 1: Distributed indexing and querying of sensor models.

ing the readings into symbolic models locally at the sensors through semantic interpretation, and then performing an in-network indexing and aggregation of semantic models to capture the global patterns. This *Model-based Index STructure* (*MIST*) can then answer semantic queries. The setup is shown schematically in Figure 1.

Of the several semantic models relevant in a sensor network context [6, 8, 10], Markov Chains [3] and Hidden Markov Models [23] are the most useful. A *Markov Chain (MC)* captures the underlying dynamics of the physical phenomena or entity by a generative model that emits a sequence of symbols. Figure 2 shows a typical example of an MC. In this example, the speed observed by the sensor on a zebra has been quantized into three symbols G (grazing), W (walking) and F (fast moving). In previous work, MCs were employed by Elnahraway et al. [10] to capture spatial correlations, and by Deshpande et al. [8] to capture temporal correlations.

A Hidden Markov Model (HMM), akin to an MC, is a generative model for a sequence of symbols. However, in an HMM, there exists a set of underlying system states that are not directly observable, but can be inferred from the observation symbols. Figure 3 illustrates an HMM for the Zebranet project. It consists of two hidden states, *Predator Present* and *Predator Absent* which emit the observation symbols G, W and F. Takasu et al. [28] employed HMMs to distinguish the different states of a toy-satellite using sensor data streams. Biologists at UCLA [29] trained an HMM on each acorn woodpecker's vocal signals (measured by acoustic sensor arrays) to recognize the identity

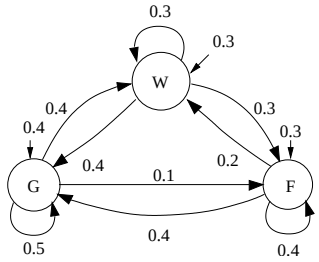

Figure 2: Markov Chain (MC) for mobility model of Zebranet. The states are *G* (grazing), *W* (walking) and *F* (fast moving).

of the individual.

Given such semantic models, either MCs or HMMs, built on the observation sequences at each sensor node, users may be interested in sensors exhibiting a particular behavior. For example, in the Zebranet project, scientists might be interested in identifying all nodes which have observed the $FFFFFF$ sequence (denoting a possible predator attack) with a likelihood of at least 0.8. These nodes can be discovered by asking *range* queries on the sensor network. In addition to the range queries, we also propose *top-1* and *1-NN* queries:

1. *Range query:* Return the sensors that exhibit a particular behavior with a likelihood greater than a certain threshold?

2. *Top-1 query:* Which sensor is most likely to exhibit a given behavior?

3. *1-NN query:* Which sensor model is the most similar to a given pattern (model)?

There can be different ways of answering these queries. The first scheme is a *centralized scheme*, where models are built locally at each node, and then transmitted to the base station (BS). Any query is answered on the models at the BS. To keep the local models and their copy at the BS synchronized, every update to a model's parameters is sent to BS. This is update-intensive. In the second technique, *centralized scheme with slack*, a slack is introduced in updating each parameter. If the query cannot be answered using the cached-models at BS, then it is sent to the models in the network.

We propose a novel distributed indexing-based scheme to answer the above queries. In *MIST*, along with a slack, we construct an in-network hierarchical index structure to answer queries efficiently. MIST exploits the high degree of spatial correlations [9, 15] in environmental sensor net-

works, by performing a spatial aggregation of such correlated symbolic data models into a single *index model*. The index model is built only on the component model parameters and not on the underlying sequences. MIST prunes updates much better than the centralized scheme with slack, as slack is not only maintained at individual nodes but also at every level of the index structure. Queries are first sent to the MIST's hierarchical index. If they are not pruned, they are sent to the local models.

To answer the queries mentioned above, MIST builds two different types of index models, *average models* and *min-max models*. These two models differ in the parameters which are retained and the manner in which the queries are handled. The min-max models have more parameters, hence higher update costs, but prune the query better leading to lower query communication costs.

This paper makes the following contributions:

- We develop a distributed and hierarchical index structure for sensor networks based on statistical models.
- We design two novel methods of aggregating the statistical models into a distributed index structure of models. The first method produces a valid model, the *average model*, that captures the average behavior of the constituent models. Spatial correlation parameters are maintained along with the average models. The second method produces pseudo-models in the form of *min* and *max* models which are used to capture the extreme behaviors of the constituent models.
- We capture the dynamic behavior of the model parameters by introducing a slack at each level of the index hierarchy. We design algorithms to aggregate the index models using both spatial and temporal correlations in a distributed setting.
- We propose two probabilistic sequence-based queries, *range* and *top-1* queries, and one model-based semantic query, *1-NN* query, that are of interest in a distributed sensor network setting. We design algorithms to answer them efficiently. We use the index parameters maintained at the root of a subtree to bound the probability of observation of a query sequence from a sensor in the subtree. The distance of a query model to a sensor model is similarly bounded.

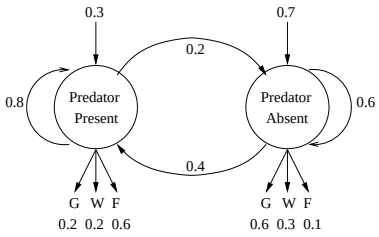

Figure 3: Hidden Markov Model (HMM) for the Zebranet.

- We perform experiments on real and synthetic data sets, using both MCs and HMMs, and show that MIST schemes not only scale well with network size and number of model states but also outperform the competing centralized schemes in terms of update, query and total communication costs.

2 Markov Chains and Hidden Markov Models

A first-order *Markov Chain (MC)* [3] is a discrete time stochastic process with a finite number of states in which the probability of occurrence of a future state depends only on the current state; past states are inconsequential. This property is called the *Markov property*. An MC is defined as:

$$MC = \{n, \pi, \tau\}$$

where n is the *number of states*, π is the *start state probability vector* of length n, and τ is the $n \times n$ *transition matrix*. $\pi(u)$ denotes the probability of starting from state u in the first step; $\tau(u, v)$ denotes the probability of reaching state v from state u in a single step.

In an MC, each observation symbol is modeled as a state. A *Hidden Markov Model (HMM)* [23], on the other hand, models the stochastic process assuming that the internal states cannot be observed directly. Only the observations from these states can be measured. Thus, there is an observation probability vector of the symbols for each state of the HMM, in addition to the transition matrix between the states and the start state probability vector. An HMM is defined as:

$$HMM = \{n, m, \pi, \tau, \xi\}$$

where n, π and τ are defined as in an MC, m is the *number of observation symbols*, and ξ is the $n \times m$ *observation*

matrix. $\xi(u, x)$ denotes the probability of observation of symbol x in state u.

Assume an observation sequence $o = o_1 o_2 \dots o_k$ of length k where each o_i is an observation symbol. For an MC to generate this sequence, it must first start from the state o_1, then transit to state o_2 and so on. Hence, the probability of observation of the sequence o from the MC is:

$$p(o_1 o_2 \dots o_k) = \pi(o_1)\tau(o_1, o_2) \dots \tau(o_{k-1}, o_k). \quad (1)$$

The state path that the sequence follows is the same as the sequence itself. However, in an HMM, the sequence of symbols does not correspond to a particular state path. All state paths of length k can possibly generate the sequence. The probability of observation of sequence o from one such state path $s_1 s_2 \dots s_k$ can be calculated. Adding the probabilities along all the possible paths gives the total probability of observation of o from the HMM:

$$p(o_1 o_2 \dots o_k) = \sum_{\text{all paths } s_1 s_2 \dots s_k} \pi(s_1)\xi(s_1, o_1)\tau(s_1, s_2) \dots \xi(s_k, o_k). \quad (2)$$

The Viterbi algorithm [23] uses dynamic programming to compute the above probability in $O(n^2 k)$ time.

3 Related Work

The general problem of content modeling and semantic querying has received considerable interest in the data mining community. Automated discovery of non-trivial, useful and previously unknown content (or knowledge) from raw data has been based on a few well-established techniques for data analysis such as decision trees [24], linear regression [22] and HMMs [23].

Linear regression models, e.g., ARIMA [22], fit a model to raw data values either to observe the underlying trends or to predict future data values. Lazardis et al. [18] proposed an online algorithm to construct a piecewise constant approximation of a time-series which guarantees that the compressed representation satisfies an error bound on the L_∞ distance. Since sensors do not exhaustively represent data, BBQ [8] proposed to complement raw data readings with a statistical model. BBQ answers queries by returning approximate values with a probabilistic confidence. Deshpande et al. [9] modeled conditional probability distributions of various sensor attributes and introduced the notion of conditional plans for query optimization with correlated attributes. Temporal correlations are captured by a Markov model in BBQ, and by a Kalman filter in Jain et al. [14]. Chu et al. [6] capture spatial correlations using joint probability distributions. Elnahraway et al. [10] also employed Markov models to estimate the current data values at a node based on the last observation at the node and those at its immediate neighbors.

Chu et al. [6], Silberstein et al. [26] and Olston et al. [21] maintain bounded approximations on actual values. Composition of MCs and HMMs has been studied by Minnen et al. [20] in order to cluster sequences. Even though the composite model's recognition performance is good, its poor scalability with the number of constituent models makes it infeasible for large-scale sensor networks. Smyth [27] used an expectation-maximization (EM) algorithm to build such composite models. Zeng et al. [30] proposed a novel fused-HMM model to integrate HMM models from the two different domains of audio and video. Brand [4] and Saul et al. [25] have developed tightly-coupled HMM models by introducing state dependencies between hidden states of the constituent HMMs, but their models do not scale with the number of constituent HMMs.

4 Distributed Index Structure

This section describes the construction of *MIST*, a distributed and hierarchical index structure on statistical models. We assume that every sensor trains an MC or an HMM on its observations. We first capture the notion of spatial correlation in two neighboring MCs.

Definition 1. ***($(1-\epsilon)$-correlation)*** *Models λ_1 and λ_2 are $(1-\epsilon)$-correlated if for all corresponding parameters σ_1 of λ_1 and σ_2 of λ_2, the following relationship holds*

$$(1-\epsilon) \leq \frac{\min\{\sigma_1, \sigma_2\}}{\max\{\sigma_1, \sigma_2\}} \quad (3)$$

For m models having the corresponding i^{th} parameters as σ_1^i, σ_2^i, $\dots$, σ_m^i, the correlation among them can be similarly defined. In this case, the correlation is given by

$$(1-\epsilon) = \min_{\forall i} \left[\frac{\min\{\sigma_1^i, \sigma_2^i, \dots, \sigma_m^i\}}{\max\{\sigma_1^i, \sigma_2^i, \dots, \sigma_m^i\}} \right] \quad (4)$$

Example: Consider the following pair of two-state MCs:

$$\pi_1 = \begin{bmatrix}0.5 & 0.5\end{bmatrix} \qquad \pi_2 = \begin{bmatrix}0.6 & 0.4\end{bmatrix}$$
$$\tau_1 = \begin{bmatrix}0.3 & 0.7\\ 0.6 & 0.4\end{bmatrix} \qquad \tau_2 = \begin{bmatrix}0.4 & 0.6\\ 0.7 & 0.3\end{bmatrix}$$

The correlation $(1-\epsilon)$ between them is the minimum of the following set: $\{\frac{0.5}{0.6}, \frac{0.4}{0.5}, \frac{0.3}{0.4}, \frac{0.6}{0.7}\}$, i.e., $\epsilon = 0.25$.

The correlation signifi es how similar the two models are. When $\epsilon \to 0$, the two models are highly correlated and are very similar to each other. On the other hand, when $\epsilon \to 1$, the models are quite dissimilar. We next defi ne two types of *index models*.

4.1 Average Model

Definition 2. ***(Average Model)*** *Given m MCs λ_1, λ_2, …, λ_m, the* average *MC λ_{avg} can be defined as:*

$$\forall u, \quad \pi_{avg}(u) = \frac{\pi_1(u) + \pi_2(u) + \cdots + \pi_m(u)}{m}$$
$$\forall u, v, \quad \tau_{avg}(u,v) = \frac{\tau_1(u,v) + \tau_2(u,v) + \cdots + \tau_m(u,v)}{m}$$

In addition to these parameters, λ_{avg} also maintains 3 additional parameters: an ϵ' parameter, from which its correlation to the individual models can be computed, and β_{max} and β_{min}, the maximum and minimum among all the parameters of the constituent models.

The next theorem captures the correlation between the average model and any of the constituent models.

Theorem 1. *Given m models $\lambda_1, \lambda_2, \ldots, \lambda_m$ that are $(1-\epsilon)$-correlated, the correlation between λ_{avg}, the average model built from them, and any of the m models is at least $(1-\epsilon')$ where $\epsilon' = \frac{\left(1-\frac{1}{m}\right)\epsilon}{\left(1-\frac{1}{m}\epsilon\right)}$.*

Proof. Let the k^{th} parameter belonging to one of the constituent models λ_i, be σ_i^k; and assume that the minimum of this parameter across all m models is ϕ and the maximum is ψ. Therefore,

$$(1-\epsilon)\psi \le \phi \le \sigma_i^k \le \psi \tag{5}$$

Denote this parameter's average over all models as μ, which is maintained by λ_{avg}. The correlation between ψ and μ is minimized when μ attains its minimum possible value. That happens when all the parameters except ψ are equal to ϕ. Thus,

$$\mu = \frac{\phi + \sigma_1^k + \cdots + \sigma_{m-2}^k + \psi}{m} \ge \frac{(m-1)\phi + \psi}{m}$$
$$\ge \frac{(m-1)(1-\epsilon)\psi + \psi}{m} = \left[1 - \left(1 - \frac{1}{m}\right)\epsilon\right]\psi$$

Also, $\mu \le \psi$. Therefore, the correlation parameter between ψ and μ is $\epsilon_1 = \left(1 - \frac{1}{m}\epsilon\right)$. Similarly, the correlation between ϕ and μ is minimized when μ is maximum. This happens when all the parameters except ϕ are equal to ψ. Thus,

$$\mu = \frac{\phi + \sigma_1^k + \cdots + \sigma_{m-2}^k + \psi}{m} \le \frac{\phi + (m-1)\psi}{m}$$
$$\le \frac{\phi + (m-1)\frac{\phi}{1-\epsilon}}{m}$$
$$\text{or, } \phi \ge \left[1 - \frac{\left(1-\frac{1}{m}\right)\epsilon}{1-\frac{\epsilon}{m}}\right]\mu$$

Also, $\phi \le \mu$. Therefore, the correlation parameter between ϕ and μ is $\epsilon_2 = (1-\frac{1}{m})\epsilon/(1-\frac{\epsilon}{m})$.

The correlation between all other parameters and μ lie within these two extremes. Since, $(1-\epsilon_2) \le (1-\epsilon_1)$, we say that the average model is at least $(1-\epsilon')$-correlated to the individual models where $\epsilon' = \frac{\left(1-\frac{1}{m}\right)\epsilon}{\left(1-\frac{1}{m}\epsilon\right)}$. □

For two models, the correlation parameter ϵ' evaluates to $\epsilon/(2-\epsilon)$, which is roughly half the correlation parameter ϵ between the constituent models, for low values of ϵ.

4.2 Min-Max Model

The min-max model consists of two separate models: the *min*-model, denoted by λ_{min}, and the *max*-model, denoted by λ_{max}.

Definition 3. ***(Min-Max Model)*** *Given m MCs λ_1, λ_2, …, λ_m, the* min *MC and the* max *MC are defined using the following parameters:*

$$\forall u, \quad \pi_{min}(u) = \min\{\pi_1(u), \pi_2(u), \ldots, \pi_m(u)\}$$
$$\forall u, v, \quad \tau_{min}(u,v) = \min\{\tau_1(u,v), \tau_2(u,v), \ldots, \tau_m(u,v)\}$$
$$\forall u, \quad \pi_{max}(u) = \max\{\pi_1(u), \pi_2(u), \ldots, \pi_m(u)\}$$
$$\forall u, v, \quad \tau_{max}(u,v) = \max\{\tau_1(u,v), \tau_2(u,v), \ldots, \tau_m(u,v)\}$$

Note that the min and max models are pseudo-models, since the start state probabilities of all the states and the transition probabilities for each state do not necessarily add up to 1. As these models maintain an upper and lower bound on each parameter, they are employed to provide an upper and lower bound on the probability of observation of a query sequence from the underlying models.

4.3 Hidden Markov Model (HMM)

We assume that the number of states in the constituent HMMs are the same and that there is a one-to-one correspondence between the states of the constituent models. The parameters of a state in the composite HMM will then have a one-to-one correspondence with the parameters of the corresponding states of the constituent models. With this requirement, correlation can be defined by Eq. (4). The average model and the min-max models analogously adopt Definitions 2 and 3. When *a priori* knowledge of hidden states is not available, a state correspondence can be established by considering all possible state mappings.

In this paper, we have assumed *a priori* knowledge of the number of hidden states and their meaning. This establishes state correspondence. When no prior information regarding the hidden states is available, we next present a method of establishing a state correspondence based on the similarity of the states. A notion of distance between two states is defined and then a minimum distance mapping is found between the states of one constituent HMM to the other. This mapping is then used to establish the one-to-one correspondence. The state distance can be defined as the Euclidean distance between start state, transition and observation probabilities of the two states. For more than two models, the correspondence can be built incrementally. MIST schemes are, however, guaranteed to return the correct answers for any state correspondence.

4.4 Hierarchical Index Construction

In this section, we describe how a distributed and hierarchical index structure is built over the entire sensor network. We overlay a tree topology on the network and perform a bottom-up aggregation of the index models. The leaf-level models are the actual models built by the individual sensors, while the internal index nodes (models) summarize the statistical behavior of the models underneath.

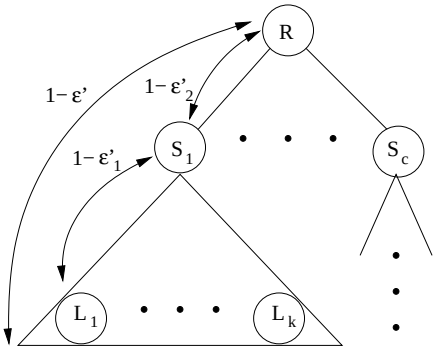

Figure 4: Correlation of the average model R with any leaf model L_i in its subtree: $(1-\epsilon') = (1-\epsilon'_2)(1-\epsilon'_1)$ where ϵ'_1, ϵ'_2 are the correlation parameters between the average models and their children at levels 1 and 2 respectively.

We first explain how average models are indexed. Figure 4 shows an example of a tree topology with 2 levels of index nodes. Model R is the average model of its children S_1 through S_c. Model S_1 is the average model built from the leaf models L_1,...,L_k. Suppose that the correlation between R and its children S_1,...,S_c models is at least $(1-\epsilon'_2)$, and the correlation between any average model S_i and the leaf models under it is at least $(1-\epsilon'_1)$. Theorem 2 shows how to calculate the correlation parameter ϵ' from the model R to any of the leaf models.

Theorem 2. *Consider an average model R. If the correlation between R and its children S_1,...,S_c models is at least $(1-\epsilon'_2)$, and the correlation between any average model S_i and the leaf models under it is at least $(1-\epsilon'_1)$, then the correlation between the average model R and any of the leaf models is at least $(1-\epsilon') = (1-\epsilon'_2)(1-\epsilon'_1)$.*

Proof. Assume that the node R has c children $S_1, \ldots, S_c$. Also assume that each child S_i has a variable number of leaf models $L_{i1}, \ldots, L_{in}$ as shown in Figure 4. Consider a single parameter σ across all the models.

If σ_{min_2} is the minimum across all the parameters σ_{s_1}, $\ldots$, σ_{s_c} at level 2, then

$$\sigma_{min_2} \geq (1-\epsilon'_2)\sigma_R \tag{6}$$

Denoting the maximum correlation parameter across the constituent models of all subtrees S_1 to S_c as ϵ_1, i.e.,

$$\epsilon_1 = \max\{\epsilon_{11}, \ldots, \epsilon_{1c}\} \tag{7}$$

Without loss of generality, assume that the maximum correlation parameter $\epsilon_1 = \epsilon_{11}$; i.e., it is observed between the leaf models in subtree 1. Then, if σ_{min_1} is the minimum parameter across all the models in the subtree 1, then,

$$\sigma_{min_1} \geq (1 - \epsilon'_1)\sigma_{s_1} \tag{8}$$

Combining Eqs. (6) and (8),

$$\begin{aligned} \sigma_{min_1} &\geq (1 - \epsilon'_1)\sigma_{s_1} \\ &\geq (1 - \epsilon'_1)\sigma_{min_2} \\ &\geq (1 - \epsilon'_1)(1 - \epsilon'_2)\sigma_R \end{aligned} \tag{9}$$

Similarly, denoting the maximum across all the children of node R by σ_{max_2}, and the maximum across the leaf level nodes by σ_{max_1},

$$\begin{aligned} \sigma_{max_1} &\leq \sigma_{s_1}/(1 - \epsilon'_1) \\ &\leq \sigma_{max_2}/(1 - \epsilon'_1) \\ &\leq \sigma_R/((1 - \epsilon'_1)(1 - \epsilon'_2)) \end{aligned} \tag{10}$$

From Eqs. (9) and (10), we can conclude that the minimum correlation between R and any of the leaf level sensors under it, is at least $(1 - \epsilon') = 1 - (1 - \epsilon'_2)(1 - \epsilon'_1)$. □

The other index parameters are calculated in the following way. The maximum of the β_{max}'s of the children gives the β_{max} for this node and the minimum of the β_{min}'s is the new β_{min}. Employing these parameters, the average model can estimate the minimum and the maximum probabilities of observation of a sequence for the set of nodes in its subtree.

The min-max models are also aggregated in a hierarchical manner. Each parameter of the min-model is the minimum of all corresponding parameters from the min-models and each parameter of the max-model is the maximum of all such parameters of the max-models.

4.5 Dynamic Maintenance of Models

Once an aggregation of distributed data sources has been carried out, the underlying data distribution may change. This may lead to violations of the existing parameters at the higher levels of the tree, necessitating an expensive rebuilding of composite models. In this section, we discuss how to avoid such expensive update costs by introducing a small slack locally at each node, and at every index node in the tree. Although this may lead to a degradation in query pruning capabilities, and hence higher query communication costs, the amortized benefi ts in communication are large.

We consider the parameters of each model to be a function of time, and denote the model at time t by $\lambda^{(t)}$. As the data distribution changes, the underlying model parameters are recomputed after every small duration d. Assume that the model transmitted by a node to its parent at the last update time $t = u$ is $\lambda^{(u)}$. The child node does not update its parent at time $t + d$, as long as $\lambda^{(t+d)}$ is $(1 - \delta)$-correlated with $\lambda^{(u)}$. The idea of the slack parameter δ is analogous to the correlation parameter ϵ (Defi nition 1).

We now explain how the slack parameter δ is incorporated in maintaining the correlation parameters at every level of the index structure. Consider an average model. It maintains an ϵ which allows it to bound its correlation with any model in its subtree. However, this ϵ has been calculated by observing the correlation of the cached copy of its child models. With time, the child models may be updated. Consider a single parameter σ_{avg} of the average model and the corresponding parameter $\sigma^{(u)}$ from the child's cached model. The correlation parameter ϵ guarantees that $\sigma^{(u)} \geq (1 - \epsilon)\sigma_{avg}$ and the slack parameter δ guarantees that at any other time-point $t + d$, $\sigma^{(t+d)} \geq (1 - \delta)\sigma^{(u)}$. Together, they guarantee that $\sigma^{(t+d)} \geq (1-\delta)(1-\epsilon)\sigma_{avg}$. Similarly, $\sigma^{(t+d)} \leq \sigma_{avg}/((1 - \delta)(1 - \epsilon))$. This relationship is true at any level of the index tree. Using Defi nition 1, the relationship of the correlation parameters with and without slack can be expressed by

$$\epsilon_{slack} = 1 - (1 - \delta)(1 - \epsilon_{noslack})$$

Therefore, aggregating the slack parameters in a bottom-up fashion, as mentioned above, preserves the correctness of the correlation parameter maintained by the index structure. We will see in the next section how the spatial and temporal correlation parameters, ϵ and δ, are employed during query pruning.

5 Query Algorithms

This section describes the processing of *range*, *top-1* and *1-NN* queries using MIST's average and min-max models.

5.1 Range Query

Users may be interested in sensors exhibiting an abnormal behavior. These nodes can be discovered by asking *range queries* of the form: *Return all nodes in the network that have observed a particular sequence of symbols q with a probability greater than a certain threshold χ.* These queries are like *select* queries since they select the set of sensors that satisfy the threshold.

First, we explain how these queries are handled by MIST's average models. Along with the average model, every node in the tree maintains the parameters, β_{max}, β_{min}, and the aggregate ϵ, as mentioned in Section 4. As the index model maintained at a node is $(1-\epsilon)$-correlated with respect to any constituent model in its subtree, the aggregate model can be used to provide lower and upper bounds on the probability of observation of a sequence from any constituent model. The next theorem states the bounds for an MC.

Theorem 3. *Consider an average model λ_{avg}. Assume that the correlation parameter maintained at λ_{avg} is ϵ and the slack parameter maintained is δ. If q is a sequence of length k, the probability of observation of q from λ_{avg} can be expressed as $\prod_{i=1}^{k} \sigma^i_{avg}$. The probabilities of observation of q from any model λ_j in its subtree are then bounded by p_l and p_r:*

$$p_l \leq p_j \leq p_r \tag{11}$$

where

$$p_l = \prod_{i=1}^{k} \left[\max\left\{\sigma^i_{avg}((1-\epsilon)(1-\delta)), \beta_{min}(1-\delta)\right\}\right] \tag{12}$$

$$p_r = \prod_{i=1}^{k} \left[\min\left\{\sigma^i_{avg}/((1-\epsilon)(1-\delta)), \beta_{max}/(1-\delta)\right\}\right] \tag{13}$$

Proof. We consider and examine the bounds of each parameter σ^i_{avg}.

σ^i_{avg} was computed from the corresponding parameters of the models in the subtree of λ_{avg} when the last updates happened for these models. Assume model λ_j's parameter to be σ^i_j at that time. From the assumption of $(1-\epsilon)$-correlation, $\sigma^i_j \geq \sigma^i_{avg}(1-\epsilon)$. However, this bound can be improved by utilizing the β_{min} parameter. No parameter can be less than β_{min}, and hence,

$$\sigma^i_j \geq \max\{\sigma^i_{avg}(1-\epsilon), \beta_{min}\}$$

Since a slack is maintained, the current value $\sigma^{i,(t)}_j$ at time t may not be exactly equal to σ^i_j. However, since the slack is at most δ,

$$\begin{aligned}\sigma^{i,(t)}_j &\geq \sigma^i_j(1-\delta) \\ &\geq \max\{\sigma^i_{avg}((1-\epsilon)(1-\delta)), \beta_{min}(1-\delta)\}\end{aligned}$$

Since p_j is the product of k such $\sigma^{i,(t)}_j$s, we can write

$$p_j = \prod_{i=1}^{k} \sigma^{i,(t)}_j \geq p_l \tag{14}$$

Utilizing the β_{max} parameter and the $(1-\epsilon)$-correlation, the upper bound on each parameter $\sigma^{i,(t)}_j$ can be written as

$$\begin{aligned}\sigma^{i,(t)}_j &\leq \sigma^i_j/(1-\delta) \\ &\leq \min\{\sigma^i_{avg}/((1-\epsilon)(1-\delta)), \beta_{max}/(1-\delta)\}\end{aligned}$$

Therefore,

$$p_j = \prod_{i=1}^{k} \sigma^{i,(t)}_j \leq p_r \tag{15}$$

Together, Eqs. (14) and (15) prove the theorem. □

If the threshold for the range query $\chi < p_l$, all sensors in the subtree are guaranteed to satisfy the query. Similarly, if $\chi > p_r$, no sensor in the subtree can satisfy the query. In these two cases, the entire subtree below the node where λ_{avg} is maintained can be pruned. If neither of the pruning conditions is satisfi ed, the query is percolated down, and this pruning is recursively carried down, if necessary, till the leaf level models. All the results (nodes that satisfy the query) are aggregated in a bottom-up fashion at the base station.

Next, we explain how the base station employs its min-max models to prune the query. Similar to the average models, min-max models compute the bounds on the probability of observation of the query, and use the bounds to prune subtrees. The following theorem states the bounds for MCs.

Theorem 4. *The probability of observing a sequence q from any of the child models of an index node is bounded by p_l and p_r:*

$$p_l = p(q|\lambda_{min})(1-\delta)^k \quad (16)$$

$$p_r = p(q|\lambda_{max})/(1-\delta)^k \quad (17)$$

where λ_{min} and λ_{max} are the min and the max models maintained by the index node.

Proof. Without loss of generality, consider q to be an observation sequence $q_1 q_2 \ldots q_k$ of length k. The probability of observation of q from λ is

$$p(q|\lambda) = \pi(q_1).\prod_{i=1}^{k-1}\Big[\tau(q_i, q_{i+1})\Big]$$

For each parameter σ_i employed by any child model λ to compute the above probability of query generation, there exists a corresponding parameter σ_{min} in λ_{min} and σ_{max} in λ_{max}. Therefore, using Definition 3 and the $(1-\delta)$-correlation property, for any parameter $\sigma^{(t+d)}$ at the current time-point $t+d$, $\sigma^{(t+d)} \geq \sigma^{(t)}(1-\delta) \geq \sigma_{min}(1-\delta)$ and $\sigma^{(t+d)} \leq \sigma^{(t)}/(1-\delta) \leq \sigma_{max}/(1-\delta)$.

Utilizing these inequalities for each of the k terms, we see that the probability of observation of q is bounded by the probabilities of observation of q from the min and max models with a slack factor. □

For the case of HMMs, Theorems 3 and 4 can be used to bound the probabilities of observation of a given query sequence along a single state path. Since the probability of observation of a sequence from an HMM is the sum of such probabilities along all possible paths, the total probability is bounded as well.

5.2 Top-1 Query

Range queries may return all or none of the sensors as the answer set. In order to avoid the difficulties of finding the right threshold, users may be interested in the sensor which best describes a particular behavior. Then, *top-1 queries* of the following form may be posed on the sensor network: *Given a sequence of symbols q, return the sensor that has the highest probability of observing it.* Using the answer of such a query, a threshold can be chosen to retrieve the other sensors via range querying.

MIST answers the top-1 query in the following way. At every level, the parent node calculates the bounds of observation of the sequence q from each of its child models in the same way as described in Section 5.1. This may be done by employing the average model and the index parameters or the min and the max models. Then, it checks whether the maximum value for observation of q from any child is less than the minimum value of observation of q from any other child. If so, the former child model and the subtree below it are pruned. The query is recursively sent to each of the remaining child nodes.

In general, top-1 queries are more communication intensive than range queries. This is because, for range queries, at any level of the index, there is a chance that all children of a particular node may be pruned as none of them satisfy the threshold. However, for top-1 queries, the bounds of the children are compared against each other. Therefore, the query will be sent to at least one child. Further, as the similarities among the children increase, their bounds become identical making the pruning for top-1 queries less likely.

5.3 1-NN or Model Query

Both range and top-1 queries were sequence-based queries. In this section, we will consider a higher level semantic query, the *1-NN query*. Instead of providing a single observation sequence as a query, users may provide a model (or a set of observation sequences from which a model can be built) and ask the following model query: *Return the sensor model that is most similar to the given query model Q.*

To answer the model query, we first define the notion of distance between two Markov Chains. The distance between two MCs λ_1 and λ_2 is defined as

$$d(\lambda_1, \lambda_2) = \sqrt{\sum_{\forall u}\big(\pi_1(u) - \pi_2(u)\big)^2 + \sum_{\forall u,v}\big(\tau_1(u,v) - \tau_2(u,v)\big)^2}$$

This distance is a metric distance. This definition can be extended to HMMs.

We will first explain how 1-NN queries are handled for average models. To find the model nearest to the query model, we employ an M-tree like mechanism [7]. An M-tree is built on the model parameter space but is physically

embedded in the communication graph. Average models at each node maintain a *radius* which is the largest distance from the average model to any of its child models. Between a parameter σ_{avg} of the average model and a corresponding parameter σ of any model in its subtree, the distance can be calculated as follows:

$$\max \begin{cases} \min\{\sigma_{avg}/(1-\epsilon), \beta_{max}\} - \sigma_{avg}, \\ \sigma_{avg} - \max\{\sigma_{avg}(1-\epsilon), \beta_{min}\} \end{cases}$$

The upper bound of σ is given by $\min\{\sigma_{avg}/(1-\epsilon), \beta_{max}\}$ and hence its distance to σ_{avg} can be calculated as shown above. Similarly, the distance of the lower bound of σ to σ_{avg} can be calculated, and consequently the maximum distance can be computed.

Then, *radius* is calculated as the square root of the sum of squares of all the distances defined on each parameter σ. The details of how this radius is used to prune subtrees is shown next.

We show how a particular sensor node S employs M-tree pruning to handle the model query. If the distance from the query model to the composite model maintained at S is d, then triangle inequality guarantees that the distance of query to any of the (composite) models in its subtree will lie between $min\{0, d-rad\}$ and $d+rad$. A priority list of subtrees is maintained according to the ascending order of its minimum possible distance to the query model. If the smallest possible distance to a subtree is greater than the largest possible distance to another subtree, then it is guaranteed that the model nearest to the query cannot reside in this subtree and hence this subtree can be safely pruned. Out of all remaining subtrees, the one for which the minimum distance is the lowest is expanded and explored as before. This subtree is deleted from the priority list and the results are aggregated bottom up, and are added to the list. Then all the subtrees that can now be pruned are deleted from the list. This algorithm is carried on for the rest of the subtrees until there is only one sensor node left. S returns this to its parent. Finally, the priority list of base station contains the model nearest to the query model.

Min-max models answer the model query by utilizing a principle similar to the R-tree [13]. For each parameter of the constituent models, the corresponding parameters in the min-model and the max-model act as the bounds. In the vector space of model parameters, the min and the max models form a minimum bounding rectangle (MBR). The query model is a point in this space. Therefore, to any MBR, it has a minimum and a maximum possible distance. Similar to the average models, these minimum and maximum distances can be used to prune the MBRs (i.e., the min-max models and the entire set of constituent models under it).

5.4 Effect of Dimensionality

The effect of dimensionality (number of model parameters) on MIST is observed only for the model queries and not for the sequence (range or top-1) queries. Even though MIST maintains min and max parameters analogous to the R-tree, there is a significant difference in how these bounds are utilized to prune the sequence queries. In an R-tree, each index is an MBR, and the query is a hyper-rectangle (or a point) in the multi-dimensional space. Query pruning depends on the intersection (or containment) in this high-dimensional space resulting in the curse of dimensionality. However, in the case of MIST, given a sequence query of length k, we compute two values—a lower bound and an upper bound—on the value of the query observation probability. In the case of MCs, the lower bound is obtained as the product of k individual min-model parameters (the corresponding start state and the $k-1$ transition probability parameters). Similarly, the upper bound is obtained from the max-model parameters. For the case of HMMs, the upper and lower bounds can computed as discussed in Section 5.1. Pruning depends on whether the query threshold χ lies within these lower and upper bounds. In other words, pruning takes place in the *single-dimension* of probability space—i.e., on the $(0, 1)$ real number line. Thus, irrespective of the number of dimensions of the underlying models, the upper and lower bounds on query probability depend only on the product of k min-max model parameters. Similarly, the bounds for the average models are also computed on the single-dimensional probability line and therefore, there is no curse of dimensionality.

On the other hand, for model queries, the search is carried over the m-dimensional space of model parameters. Each sensor model becomes a point in the multi-dimensional space and the one nearest to the query model is retrieved. MIST's min-max models employ a straightforward R-tree based 1-NN search and its average models employ M-tree based 1-NN search. As the dimensionality increases, the probability of intersection of the query with the index nodes increases, and thus, the pruning power of MIST decreases.

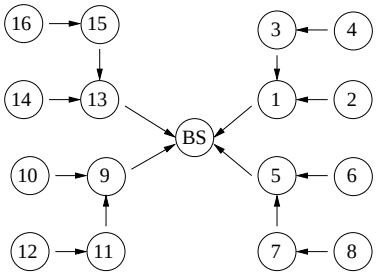

Figure 5: Topology of the laboratory sensor network. There are 4 rooms and 4 sensors in each room. BS denotes the base station.

6 Slack Analysis

A large value of slack minimizes updates, but considerably decreases MIST's query pruning capabilities because of the wider query probability bounds. On the other hand, a small slack will have effi cient query pruning due to tight bounds, but will lead to increased update costs. In this section, we characterize the optimal choice of δ to achieve the minimum total communication cost, comprising both the update and query costs.

We denote the probability that the query will be sent down from an aggregate model to its children by P_q, and the probability that an update (for a parameter) will be sent up from the children to its aggregate by P_u. A detailed analysis (Appendix A) based on the random-walk deviation of a parameter shows that

$$P_u = 16/\left[1/(1-\delta) - (1-\delta)\right]^2.$$

Similarly, the analysis (Appendix A) of P_q for a range query on a sequence of length k yields

$$P_q = (1/2^k)\left[1/(1-\epsilon)^k(1-\delta)^k - (1-\epsilon)^k(1-\delta)^k\right].$$

We assume that the communication cost for sending a query parameter, and the cost of updating a parameter are equal and normalized to 1. Further, assuming that each model has n parameters, and Q queries each of length k are posed during this duration d, the total expected communication cost at each index node is

$$T = \frac{16nd}{[1/(1-\delta) - (1-\delta)]^2} + \frac{Qk}{2^k}.s(k)$$

where $s(k) = \left[1/(1-\epsilon)^k(1-\delta)^k - (1-\epsilon)^k(1-\delta)^k\right]$.

To obtain the optimal δ, we differentiate T with respect to δ and set it to zero. This yields a polynomial equation of degree $2k+5$. The second derivative of T with respect to δ is greater than zero, showing that the solution obtained is indeed a minimum. As the second derivative is continuous in the open interval $(0,1)$, the solution of the equation can be obtained using Newton's method [12], or techniques like fi nite-difference methods for faster convergence [11].

Slack estimated by the above method minimizes the communication cost locally at each index node. The optimal slack δ_i required at each node i to minimize the global communication cost (Appendix A) for the entire network can be evaluated only after considering the cost at all nodes. There are two main bottlenecks involved in this global optimization: (i) all the ϵ_i parameters have to be sent to the base station to evaluate δ_i's, and (ii) it is computationally demanding to solve for δ_i's. Further, as ϵ_i parameters keep evolving, the optimal δ_i parameters will change, and hence recomputing the globally optimal δ_i's is very expensive.

Since the sensors have limited processing capabilities, the iterative numerical operations involved in the calculation of the locally optimal slack are also computationally challenging. Therefore, we use experimental techniques to estimate the slack δ. We estimate the optimal value of the slack parameter *a priori* for a wide range of ϵ, query rate, and query length settings. Each sensor or an index node maintains a table, and uses an appropriate δ based on table-lookup.

7 Fault-Tolerance

Node failures are one of the primary causes of network unreliability. A parent node in the MIST index needs to distinguish between the case where a child node fails and the case that a child's parameters are within the slack δ. To make MIST robust to node failures, every parent maintains an expected update interval and poll the child for updates. Detecting node failures using acknowledgment packets is costly. Therefore, MIST employs periodic heartbeat [6] message exchanges to keep the parent-child nodes synchronized. After a node has failed, its children switch to a new parent, and transmit their parameters to the new parent. This is a one-time cost. Subsequently, the correlation

and the slack-based update protocols are followed.

However, if MIST were to handle queries even during node failures, then replication may be a good alternative mechanism. The index model (or base model) information can be replicated at a chosen *sibling* node. There should exist a path between the sibling and the parent of the node, even if the node fails. The parent and children of a node are informed of the identity of the sibling node. At the sibling node, the replicated index can be maintained with a correlation of $(1 - \phi)$ with respect to the actual indices. If ϕ is a small value, then the replicated index is more up-to-date with the current index, but incurs larger communication costs. Hence, this protocol can be seen as a trade-off between consistency and communication costs, similar to the replication protocols which achieve scalability by providing loose consistency guarantees [5]. Hence, when a node fails, its parent transmits the query to the sibling node. The sibling node attempts to answer the query employing its $(1 - \phi)$-correlated models. If it succeeds, it returns the result; else, it transmits the query to the children of the original node.

Periodic heartbeat messages can be used to discover link failures for updates. For queries, if an answer is not returned by the subtree within a specific time-out, it is assumed that the link has failed. When a node detects link-failure with respect to its parent, it switches to a new parent and transmits its model parameters to the new parent. When a parent detects a link failure to a child, it notifies it through alternate routes. MIST allows transient data inconsistencies until the detection of failures. MIST trades-off these short-lived inconsistencies in favor of the communication savings accrued by avoiding the robust but expensive ACK protocol.

8 Performance Evaluation

In this section, we present the experimental results for all the three queries—range, top-1 and 1-NN—on MIST's min-max and average models built for MCs and HMMs. First, we describe our datasets. Then, we explain the different settings for the measurements of query, update and total communication costs.

8.1 Experimental Setup

Our experiments were conducted on two datasets—a real data set obtained from our laboratory and a synthetically generated data set. In the laboratory data set, sensors were located in four rooms and four sensors were placed in four different corners of each room. The topology is illustrated in Figure 5. The base station is a central server where the queries are posed.

In the laboratory dataset, sensors were used to measure the temperature inside the laboratory. The sensors sensed temperature every 30 seconds for 10 days. The values were quantized into three symbols: C (*cold*) for temperatures less than $25°$C, P (*pleasant*) for temperatures between $25°$C and $27°$C, and H (*hot*) for temperatures higher than $27°$C. Semantic queries which are of interest, such as (i) alternating weather patterns, $HCHC$, (ii) consistently pleasant temperature, $PPPP$, etc., were posed. Sequence queries were generated in random by sampling from a uniform distribution. For 1-NN queries, query models were sampled from a uniform distribution. Markov Chains with 3 states and HMMs with 2 states and 3 symbols were built on each sensor on sequences generated for each day.

The synthetic dataset was generated for different network sizes ranging from 16 to 512. The number of model states was varied from 3 to 11. The base models were generated by controlling the model correlation or the ϵ parameter for 5 different values ranging from 0.001 to 0.5. These models were built for sequences generated over three hour periods, for five days. Updates to data value were generated using the uniform random walk method [3].

Communication costs were measured in number of bytes. Transmission messages encoded a model parameter in 2 bytes, and a k length query string in k bytes. For model queries, the entire model was encoded in $2m$ bytes where m is the number of parameters in the model. The answers for n-size networks were encoded in a bit vector of length n, or equivalently $n/8$ bytes.

8.2 Compared Techniques

We first present the two centralized schemes. Then, we present variations of MIST that are considered in the experimental results.

• **Centralized scheme with no slack:** Each node transmits its models to the base station (BS). Every update to a

model parameter is sent to the BS. Queries are posed at the BS, which always maintains the latest models. As a result, queries are answered using zero communication cost.

• **Centralized scheme with slack:** Initially all models are transmitted to the BS. An update at each node is not transmitted to the BS if the current parameter is within the slack, i.e., within a $(1-\delta)$-correlation of the base station's cached parameter. Query probabilities and distances are bounded using the approximate models at BS. If a query cannot be answered with certainty, it is injected into the network to retrieve the required models.

• **MIST schemes:** Both types of index models—average and min-max—are maintained under slack and no slack conditions.

We evaluate the performance of our index structure with respect to query rate, slack and correlation in terms of communication costs. Our experiments were conducted for the following values of ϵ: $0.001, 0.01, 0.1, 0.2, 0.5$. Unless mentioned otherwise, the slack δ is set to the locally optimal value. We experiment with both MCs and HMMs built from real-life and synthetic data. For brevity, we only report the representative results.

8.3 Scalability with Query Rate

The first experiment evaluates the scalability of the different schemes with query rate. Figure 6 compares MIST's schemes with the centralized schemes on MCs built from synthetic data. The total communication cost, which is the sum of update costs and (range) query costs was measured for varying query rates. The *query rate* is the number of queries posed to the network between two successive model construction time instances. This time interval is 3 hours for our datasets. The centralized scheme with slack performs poorly, as it injected most of the queries into the network. We notice that the rest of the schemes scale well with query rate.

Figure 7 magnifies Figure 6 to compare MIST's average and min-max models with the centralized scheme without slack. For small query rates, MIST's slack-based schemes, which maintain a slack at every level, outperform those without slack by almost a factor of two. Up to query rates of 5, we see that average model with slack performs the best among MIST schemes. At small query rates, updates become the dominating factor of the total costs, and as the average model maintains smaller number of parameters than the min-max model, it outperforms the min-max

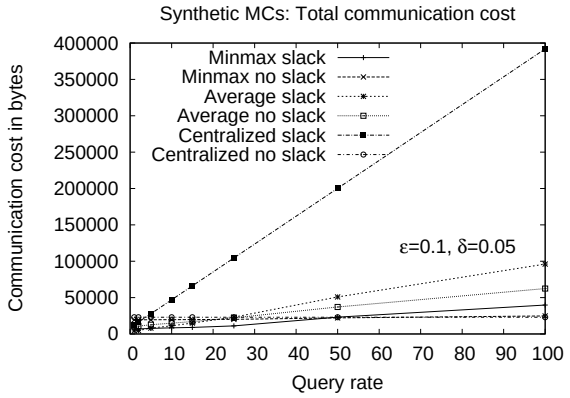

Figure 6: Total communication costs of MIST and the competing schemes for varying query rates.

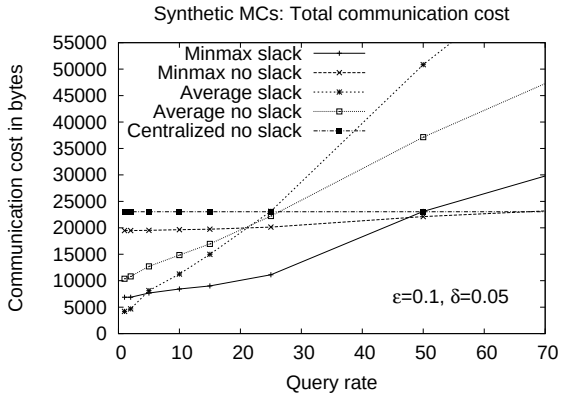

Figure 7: Total communication costs of MIST and the competing schemes for varying query rates.

scheme.

As the query rate increases to 25, the min-max models provide efficient query pruning and better query communication costs, and hence outperform the average model. When the query rate increases further up to 50, the cost of slack-based indexing schemes increases rapidly. This is because those queries which were not pruned by the slack-based index models are drilled down into the network. At these query rates, min-max scheme with no-slack provides the lowest costs. As this scheme maintains up-to-date indices, it provides tight bounds on the probability of query observation from the underlying models, and hence prunes most of the queries at the highest levels of the index structure. At higher query rates, the centralized scheme which

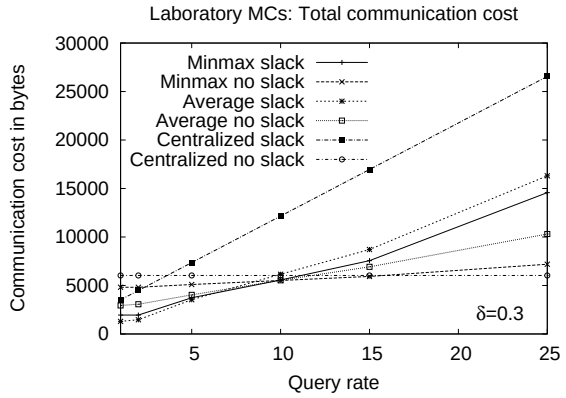

Figure 8: Total communication costs of MIST schemes and the competing schemes for MCs built on laboratory data.

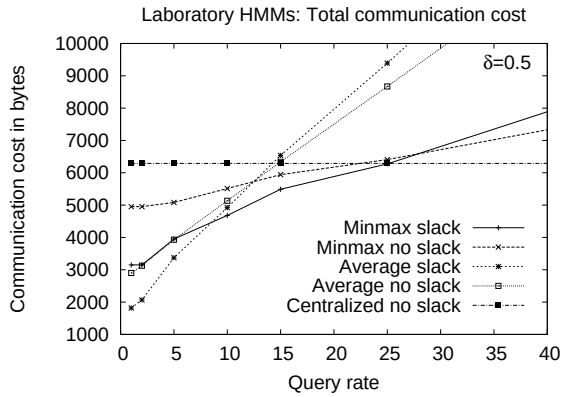

Figure 9: Total communication costs of MIST and the competing schemes for HMMs built on laboratory data.

has zero query costs becomes a viable alternative and ultimately becomes the most effi cient scheme.

Figure 8 depicts the communication costs of the various schemes for MCs built from the laboratory data. The overall trend of the schemes is similar to those observed in synthetic MCs (Figure 7). However, we note that the schemes with no slack outperform the slack based schemes at medium query rates of 10. Although the update costs were reduced due to the high δ value of 0.3, these savings were offset by the large increase in query communication costs because of the reduced pruning power. The pruning power was much lower than the synthetic data set because of the high values of ϵ for the laboratory data.

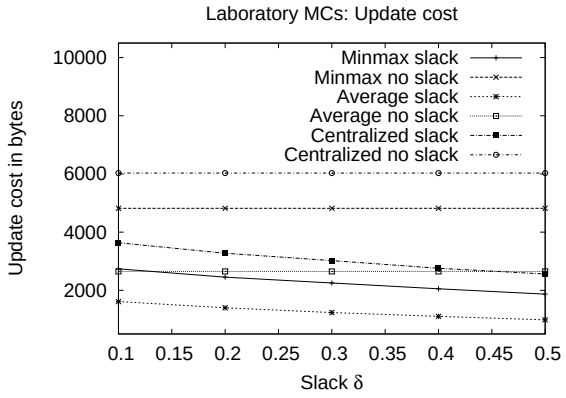

Figure 10: Update costs of MIST and the competing schemes for MCs built on laboratory data for varying slack.

Figure 9 shows that for HMMs built from real data, the average slack scheme is the best for query rates up to 9. After that, the min-max scheme with slack performs the best. On further analysis, we found that query pruning is very little on real-world HMMs, and therefore for low query rates, the update costs become more signifi cant. This explains why the average models with their low update costs perform the best.

8.4 Update Costs

We next compare the update costs of the various schemes. Figure 10 shows the low update costs of MIST-based schemes with increasing slack, for MCs built on laboratory data. We observe that the update costs of slack-based schemes were almost half the costs of the corresponding schemes without slack. We also observe that the costs of slack-based centralized schemes are twice as high as MIST's slack-based indexing schemes. This is because MIST maintains slack at every level of the hierarchy whereas the centralized scheme maintains the slack only for the base models. The average models transmit a single model and three index parameters compared to min-max's two pseudo-models at every level of hierarchy; therefore, the update costs of average models were better than those of min-max models.

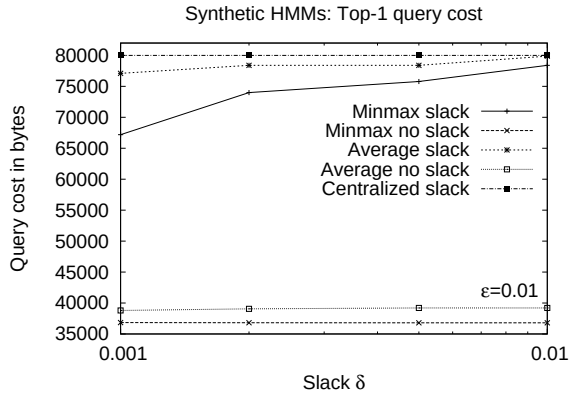

Figure 11: Top-1 query costs for HMMs built on synthetic data for varying slack.

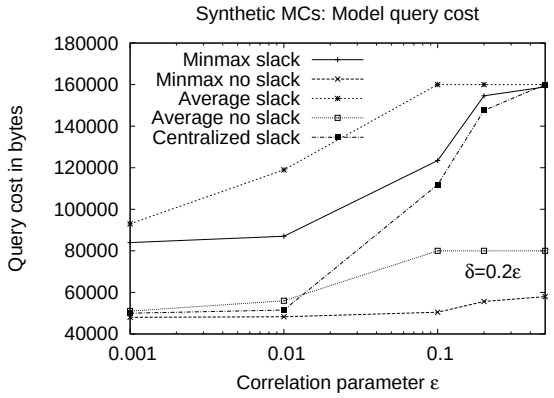

Figure 12: Model query costs for MCs built on synthetic data for varying correlation.

8.5 Query Costs

We finally analyze the costs for answering the three different queries in terms of the total message size. The centralized scheme with no slack always has a cost of 0 and is therefore not shown.

Figure 11 depicts the query costs for top-1 queries on HMMs built on synthetic data. Query costs were plotted against varying slack parameters. The query cost of the slack-based schemes, including the centralized scheme were almost twice as expensive as the schemes without slack. As the slack is maintained at every level, the bounds on the query probability at the top levels of MIST hierarchy are too wide for efficient query pruning. Hence, schemes without slack performed much better. Min-max models outperform the average models since it maintains two pseudo-models, which provide much tighter bounds than a single average model.

Figure 12 depicts query costs for 1-NN queries on MCs built from synthetic data. Here, the communication costs are measured against different values of correlation parameter ϵ. For low values of ϵ, the synthetically generated models are highly correlated and hence the index structure provides very tight bounds on the underlying models. Hence, at these values, the query communication costs for MIST's indexing schemes without slack are very low. It is worth noting that at such low values, even the centralized scheme with slack, can prune queries much more efficiently than the slack-based schemes. When ϵ increased to 0.1, we observed that most of queries could not be pruned by the centralized scheme and hence were injected into the network. With increasing ϵ, the bounds for average models become worse. This explains the increasing costs for the no-slack schemes. The bounds from the min-max models also become larger as the minimum and maximum for each parameter become more varied.

Figure 13 illustrates the effects of different lengths of the query sequence on the query costs for the range queries. The experiments were performed on synthetic MCs. Query length k was varied from 2 to 6 and the threshold set as α^k, with α chosen uniformly between 0 and 1. With increasing query length, the number of bytes needed to encode the query string goes up. However, the bounds for the probability of observation of the query becomes tighter with increasing length. Thus, the chances of a query getting filtered increases. This results in a small increase of query costs, and good scalability of MIST schemes with query length.

8.6 Optimal Slack

We next performed experiments on real data to evaluate the effect of the slack parameter on the total communication cost. Figure 14 shows the total costs for the min-max and the average slack schemes with δ varied from 0.1 to 0.9 when the query rate was set to 5. For low values of δ, the update costs are large; whereas for high values of δ, there is almost no query pruning resulting in prohibitively high query costs. For $\delta = 0.3$, the total communication cost was found to be minimum for both the indexing

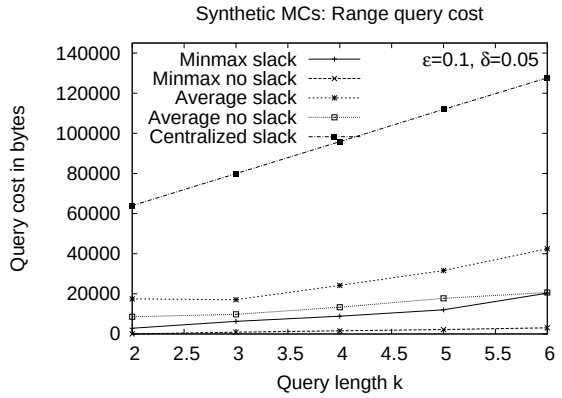

Figure 13: Range query costs for MCs built on synthetic data for varying query length.

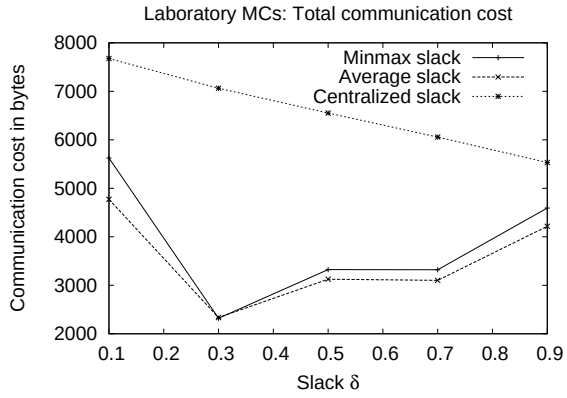

Figure 14: Total communication costs of MIST slack schemes for MCs built on real data for varying slack.

schemes. At this slack value, not many updates happened and many of the queries could be pruned at higher levels. Interestingly, the total costs for the centralized slack scheme kept decreasing with increasing δ. As the query rate was low, update costs dominated the query costs, and hence decreased the total costs with increasing slack.

8.7 Scalability with Network Size

The centralized schemes scale linearly with network size. In the centralized scheme without slack, each update message travels the entire path from the sensor to the BS whereas in the centralized scheme with slack, a query which could not be answered at the base station is sent

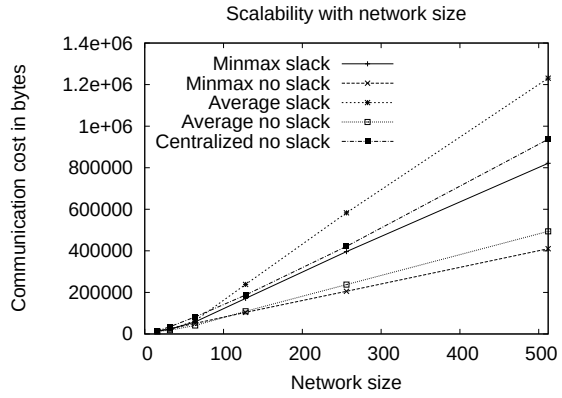

Figure 15: Scalability of MIST and centralized schemes with network size.

to a sensor in order to retrieve the corresponding results. Therefore, the total communication costs of the centralized schemes increase linearly with increasing network size. MIST exploits spatial correlations among the different sensors and both its updates and queries are pruned using a tree-based protocol. Hence, the scalability is better.

Figure 15 depicts the total communication costs of MIST schemes and the competing centralized schemes with varying network size. The experiments employed MCs built on synthetic data with $\epsilon = 0.1$ and $\delta = 0.05$ with (range) query rate 15. The centralized scheme with slack has been omitted from the figure due to its high query communication costs. MIST schemes without slack offer significant savings compared to the slack-based schemes. As the network size increases, the slack-based schemes achieve low update costs due to pruning; however, there is a huge increase in query communication costs due to two reasons: (i) the query pruning decreases considerably at higher levels of the tree due to the slack employed in the parameters, and (ii) the queries traverse a longer distance to retrieve the results. The centralized scheme without slack incurs large update costs since it transmits every model to the base station. In MIST, models are sent up only to the next level and indexed. Therefore, the update costs are small. In sum, this figure depicts the superior scalability of MIST based schemes with network size and shows that MIST maintains the performance gain over the centralized schemes.

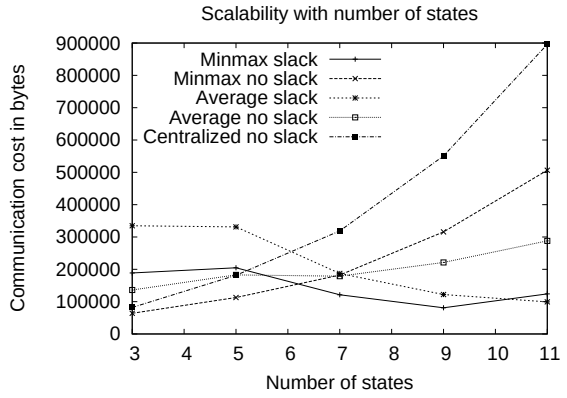

Figure 16: Scalability of MIST and centralized schemes with number of states.

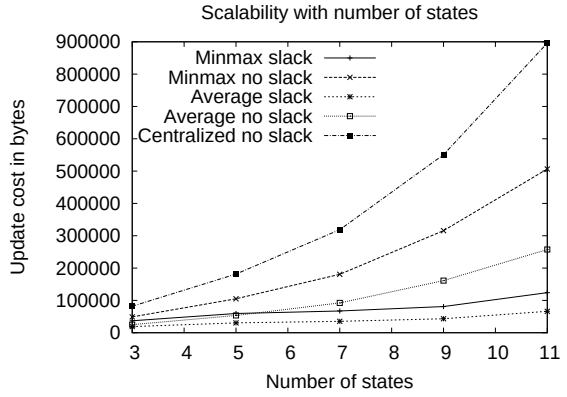

Figure 17: Update costs of MIST with number of states.

8.8 Scalability with Number of Model States

In this section, we describe how MIST and the centralized schemes scale with number of states. Figure 16 depicts the total communication costs with varying number of states for range queries. The experiments were done with synthetic MCs with $\epsilon = 0.1$, $\delta = 0.05$, and query rate 100. We notice an interesting trend. For small state sizes, the schemes without slack outperformed the schemes with slack, whereas at larger state sizes the reverse phenomenon was observed. In order to understand the underlying reasons behind such a trend in total costs, we plotted the individual contributions of the update and the query costs of each scheme in Figures 17 and 18 respectively.

Figure 17 illustrates that MIST's slack-based schemes scale well with increasing state size because of update

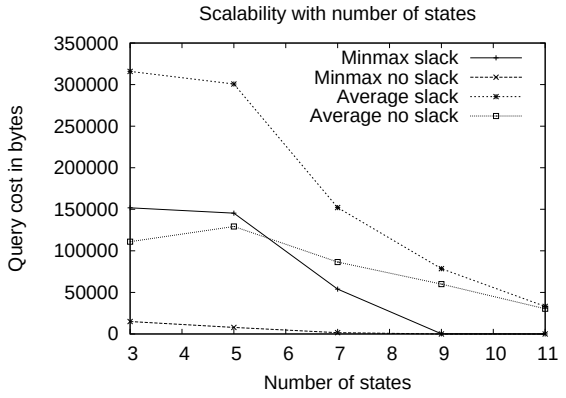

Figure 18: Range query costs of MIST with number of states.

pruning. On the other hand, the schemes without slack incur larger transmission costs due to the increased number of parameters and little update pruning. The centralized scheme with no slack scales poorly.

However, for the range query costs depicted in Figure 18, it is interesting to observe that the query costs of each scheme for 100 random queries decrease with increasing state size. The centralized scheme without slack has zero query costs and is, therefore, omitted. The average transition probabilities and the average start state probabilities are inversely proportional to the number of states. Therefore, the probability of observing a query of length k (which is a product of $k-1$ transition probabilities and 1 start state probability for an MC) decreases with increasing number of states. The bounds are also products of k probabilities. Thus, they become smaller and their difference shrinks resulting in fewer queries surpassing the query threshold. The schemes without slack outperformed those with slack because of the tighter probability bounds. Figure 18 shows the query costs for a very high query rate of 100. For lower query rates, the update costs dominated. At high query rates and large state sizes, the query costs became negligible, and the trend in the update costs was reflected in the total communication costs. Thus, MIST does not suffer from the "curse of dimensionality" from the number of states for range and top-1 queries.

However, the effect of dimensionality is noticed for model queries, where the search is carried over the m-dimensional space of model parameters. Figure 19 shows the degradation of performance in MIST's models with in-

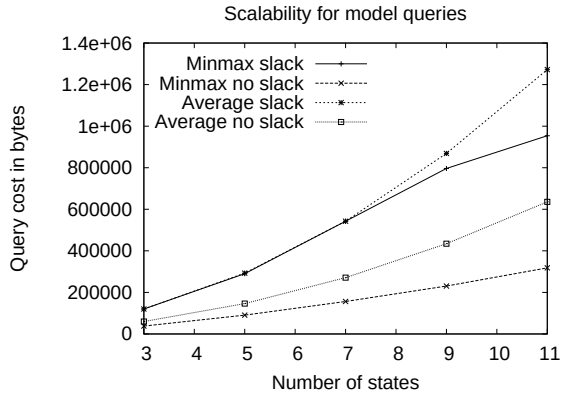

Figure 19: Curse of dimensionality: Model query costs of MIST with number of states.

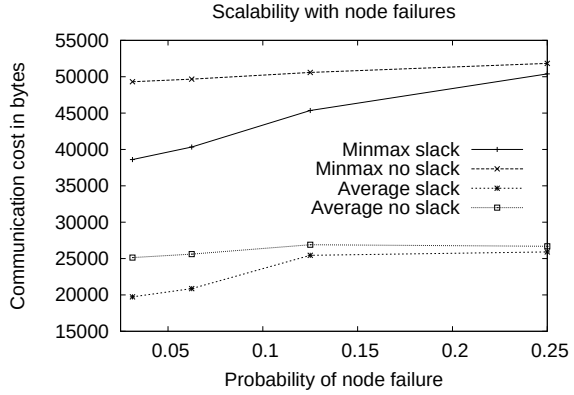

Figure 20: MIST's performance under node failure.

creasing m. Even though there is an effect of dimensionality, the communication costs remain practical up to large state sizes of 11. MIST schemes without slack provide better bounds, and hence, lead to lower communication costs.

8.9 Fault-Tolerance Experiments

Figure 20 depicts the scalability of various schemes with failure probability. Every node fails with a probability of f. The experiments were performed on MCs with $\epsilon = 0.1$, $\delta = 0.05$ and query rate set to 10. At the time of node failures, MIST schemes incur additional costs, as the children of the failed node discover a new parent and propagate updates to the new parent and its ancestors. Therefore, as the probability of node failure increases, MIST schemes show

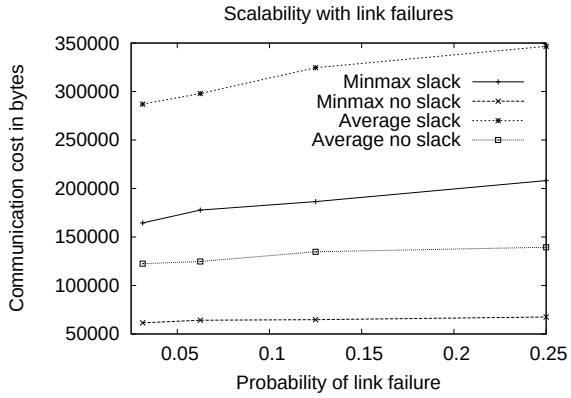

Figure 21: MIST's performance under link failure.

a moderate increase in total communication costs.

Figure 21 depicts the communication costs of MIST schemes with increasing link failure probability. As the link failure probability increases, the percentage of nodes that switch to a new parent and follow the subsequent expensive update protocol increases proportionally. Hence, the costs of each scheme increase.

Summary: When the spatial data is highly correlated, for low query rates, the minmax scheme with slack is the best. For high query rates (over 50), the minmax scheme and the centralized scheme without slack are the best. When the correlations are low, MIST schemes without slack perform the best for query rates of more than 10. For large network sizes of more than 100 sensors, MIST schemes without slack scale the best. For state sizes less than 6, MIST schemes without slack are recommended. For higher state sizes, MIST schemes with slack are preferred. With increasing link and node failure probability, the average scheme with slack scales better than other schemes.

9 Conclusion

In this paper, we developed a distributed and hierarchical index structure for sensor networks, *MIST*, based on Markov Chains and Hidden Markov Models. These statistical models capture the semantics of a sensor system by transforming raw signals into symbols, thus permitting a high-level understanding and analysis.

In order to unearth global semantic patterns, the mod-

els at the individual sensors need to be aggregated. In the light of communication constraints in a sensor network, aggregation should be based on the model parameters. We designed two novel distributed algorithms for model aggregation. The first algorithm produces a valid statistical model, the *average model*, that captures the average behavior of the constituent models. Spatial correlation parameters and two other index parameters were maintained along with the average models. The second algorithm produces pseudo-models in the form of *min* and *max* models which were used to capture the extreme behavior of the constituent models. We also captured the temporal shift of the parameters of a model by considering a slack at each level of the index structure.

We proposed two probabilistic sequence-based queries, *range* and *top-1* queries, and one high-level model-based semantic query, *1-NN* query, that are of interest in a distributed sensor network setting. We designed algorithms to answer these queries efficiently. We used the index parameters maintained at the root of a subtree to bound the probability of observation of a query sequence from a sensor in the subtree. We also bounded the distance of a query model to a sensor model using these parameters.

We designed algorithms to answer them efficiently by bounding the probability of observation of a query sequence from a sensor (as well as the distance of a query model from a sensor model) in a subtree using just the index parameters and the slack parameters maintained at the root of the subtree.

We compared our schemes against two other centralized schemes, one with slack and the other without slack. Extensive experimental evaluation on both real-world and synthetic data sets showed that MIST's models outperform the competing centralized schemes in terms of update, query and total communication costs. The scalability experiments showed that MIST scales well with network size and number of model states.

Acknowledgments: This work was supported in part by grants ITR 0331697 from the National Science Foundation, USA and DAAD19-03-D-004 from the Army Research Organization through the Institute of Collaborative Biotechnologies.

References

[1] Crossbow Wireless Sensor Networks. http://www.xbow.com/.

[2] G. W. Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *EWSN*, 2005.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley, 1998.

[4] M. Brand. Coupled Hidden Markov Models for Modeling Interacting Processes. Technical report, MIT Media Lab Perceptaul Computing/Learning and Common Sense, 1997.

[5] J. B. Carter, J. K. Bennet, and W. Zwaenepoel. Techniques for reducing consistency-related communication in distributed shared-memory systems. *ACM Trans. Comp. Sys.*, 13(3):205–243, 1995.

[6] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *ICDE*, pages 48–60, 2006.

[7] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB*, pages 426–435, 1997.

[8] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-Driven Data Acquistion in Sensor Networks. In *VLDB*, pages 588–599, 2004.

[9] A. Deshpande, C. Guestrina, W. Hong, and S. Madden. Exploiting Correlated Attributes in Acquisitional Query Processing. In *ICDE*, pages 143–154, 2005.

[10] E. Elnahrawy and B. Nath. Context-Aware Sensors. In *EWSN*, pages 77–93, 2004.

[11] D. Faires, R. L. Burden, K. Sandberg, and B. Pirtle. *Numerical Methods*. Thomson Learning, 2002.

[12] R. L. Graham, D. E. Knuth, and O. Patashnik. Concrete Mathematics: A Foundation of Computer Science. *Addison-Wiley*, 1989.

[13] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD*, pages 47–57, 1984.

[14] A. Jain, E. Chang, and Y. F. Wang. Adaptive Stream Resource Management Using Kalman Filters. In *SIGMOD*, pages 11–22, 2004.

[15] A. Jindal and K. Psounis. Modelling spatially-correlated sensor network data. *ACM TOSN*, 2(4):466–499, 2004.

[16] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *ASPLOS-X*, pages 96–107, 2002.

[17] H. T. Kung and D. Vlah. Efficient location tracking using sensor networks. *IEEE Trans. on Wireless Comm. and Networking*, 3(1954-1961), 2003.

[18] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *ICDE*, pages 429–440, 2003.

[19] A. MainWaring, J. Polastre, R. Szewczyck, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. Technical report, Intel Research, 2002.

[20] D. C. Minnen and C. R. Wren. Finding Temporal Patterns by Data Decomposition. In *AFGR*, pages 608–613, 2004.

[21] C. Olston, J. Widom, and B. T. Loo. Adaptive precision setting for cached approximate values. In *SIGMOD*, pages 355–366, 2001.

[22] M. Pourahmadi. Foundations of Time Series Analysis and Prediction Theory. *Wiley*, 2001.

[23] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[24] S. Russell and P. Norvig. *Artifi cial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[25] L. K. Saul and M. I. Jordan. Mixed Memory Markov Model: Decomposing Complex Stochastic Processes as Mixture of Simple Ones. In *ICML*, pages 75–88, 1999.

[26] A. Silberstein, K. Munagala, and J. Yang. Energy-effi cient monitoring of extreme values in sensor networks. In *SIGMOD*, pages 169–180, 2006.

[27] P. Smyth. Clustering Sequences with Hidden Markov Models. *Advances in Neural Information Processing Systems*, 9:648–658, 1997.

[28] A. Takasu and K. Aihara. An Annotation Method for Sensor Data Streams Based on Statistical Patterns. In *IMDA*, pages 95–100, 2006.

[29] Y. Yao, Y. Lin, E. Stabler, and C. Taylor. Vocal Individual Recognition of Acorn Woodpecker. In *CENS, UCLA Research Review Poster*, 2003.

[30] Z. Zeng, J. Tu, B. Pianfetti, M.Liu, T. Zhang, and Z. Zhang. Audiovisual Affect Recognition through Multi-stream Fused HMM for HCI. In *CVPR*, pages 967–972, 2005.

A Slack Analysis

We denote the probability that the query will be sent down from an aggregate model to its children by P_q, and the probability that an update will be sent up from the children to its aggregate by P_u.

We assume that each parameter σ of the model performs a uniform random-walk during the duration d. Hence the probability P_u that σ does not deviate more than w in the duration d can be calculated as $4d/w^2$ [21] using Chebyshev inequality [12]. Assuming σ to be uniformly distributed, the expected w is given by $\int_0^1 \sigma(1/(1-\delta)-(1-\delta))d\sigma = (1/2)[1/(1-\delta)-(1-\delta)]$. Hence, for a single time instance ($d = 1$) and a single parameter,

$$P_u = 16/\left[1/(1-\delta)-(1-\delta)\right]^2 \tag{18}$$

Next we evaluate P_q for range queries and perform an average-case analysis, assuming that the lower and upper bounds are the farthest apart. Consider a query sequence q of length k. If the probability of its observation from the average model is $\gamma = \gamma_1 \ldots \gamma_k$, then the probability of observing q from the children lies in the range $[\gamma_l, \gamma_r] = [\gamma(1-\epsilon)^k(1-\delta)^k, \gamma/(1-\epsilon)^k(1-\delta)^k]$.

We want to evaluate the probability P_q that the query could not be answered by the composite model, i.e., the query threshold χ lies between the $[\gamma_l, \gamma_r]$ bounds. Hence, $P_q = P(\chi \in (\gamma_l, \gamma_r))$. We assume each γ_i parameter to be independent and uniformly distributed between 0 and 1. Similarly, the threshold χ is assumed to be uniformly distributed. Hence, P_q can be evaluated by

$$\begin{aligned}
P_q &= \int P(\chi \in (\gamma_l, \gamma_r))\, p(\gamma) d\gamma \\
&= \int \left[\int_0^1 (\gamma_r - \gamma_l)\, p(\chi)\, d(\chi)\right]\, p(\gamma)\, d\gamma \\
&= \int (\gamma_r - \gamma_l)\, p(\gamma)\, d\gamma \; [\because p(\chi) = 1] \\
&= \int \gamma[1/(1-\epsilon)^k(1-\delta)^k - (1-\epsilon)^k(1-\delta)^k]\, p(\gamma)\, d\gamma
\end{aligned}$$

Each parameter γ_i can vary uniformly between 0 and 1. Therefore, $\int \gamma \;\; p(\gamma)\, d\gamma = \int_0^1 \cdots \int_0^1 \gamma_1 \ldots \gamma_k \;\; p(\gamma_1) \ldots p(\gamma_k)\, d\gamma_1 \ldots d\gamma_k = 1/2^k$. Hence,

$$P_q = (1/2^k)\left[1/(1-\epsilon)^k(1-\delta)^k - (1-\epsilon)^k(1-\delta)^k\right] \tag{19}$$

We now estimate the total expected cost of both query and updates. We assume that the communication cost for sending a query parameter, and the cost of updating a parameter are equal and normalized to 1. Further, assuming that each model has n parameters, and Q queries each of length k are posed during this duration d, the total expected communication cost is

$$T = \frac{16nd}{[1/(1-\delta)-(1-\delta)]^2} + \frac{Qk}{2^k}.s(k)$$

where $s(k) = \left[1/(1-\epsilon)^k(1-\delta)^k - (1-\epsilon)^k(1-\delta)^k\right]$.

To obtain the optimal δ, we differentiate T with respect to δ and set it to 0. This yields a polynomial equation of degree $2k+5$. The second derivative of T with respect to δ is greater than zero, showing that the solution obtained is indeed a minimum. As the second derivative is continuous in the open interval $(0, 1)$, the solution of the equation can be obtained using Newton's method [12], or other techniques like fi nite-difference methods for faster convergence [11].

Since the sensors have limited computational capabilities to perform such iterative numerical calculations, we instead use experimental techniques to estimate the optimal δ. We estimate the optimal value of the slack parameter *a priori* for a wide range of ϵ, query rate, and query length settings. Each sensor or an index node maintains a table, and uses an appropriate δ based on the table-lookup. In Section 8.6, we discuss the experimental procedure.

In the MIST index structure, every internal node in a tree will estimate the δ locally, as mentioned in Section 6, by employing the query settings and the ϵ-correlation parameter. This is a greedy choice, as the optimal δ_i required at each level i to minimize the global communication cost for the entire network can be evaluated only after considering the cost at all levels. Extending the previous analysis to multiple levels of the tree, we now determine QC, the *query communication cost*, and UC, the *update cost* for all the updates.

Let C_l denote the total number of children at level l. Since a query injected at a particular level l is conditionally dependent on the fact that it escaped pruning at all the levels above l, the total query cost

$$QC = Qk \sum_{l=1}^{h} \left(\frac{C_l}{2^k} \prod_{i \in l} s_i(k) \right)$$

where $s_i(k) = \left[1/(1-\epsilon_i)^k(1-\delta_i)^k - (1-\epsilon_i)^k(1-\delta_i)^k\right]$. Similarly, extending the above discussion on updates to multiple levels, we get the expected update cost as

$$UC = \sum_{l=1}^{h} \left(C_l \sum_{i \in l} \left[16nd/\left[1/(1-\delta_i) - (1-\delta_i)\right]^2 \right] \right)$$

Hence, the total communication cost is

$$TC = QC + UC \tag{20}$$

Given ϵ_i's at every level, TC should be minimized over all δ_i's. This function is non-convex, and Monte-Carlo techniques [11] can be employed to solve for δ_i values. There are two main bottlenecks of this procedure: (i) all ϵ_i parameters have to be sent to the base station to evaluate δ_is, and (ii) it is computationally demanding to solve for δs. Further, as ϵ_i parameters keep evolving, the optimal δ_i parameters will change, and hence recomputing the globally optimal δ_is is a very expensive procedure. Thus, we employ our greedy algorithm to calculate the δ_i at each index.