eSENSE: Energy Efficient Stochastic Sensing Framework for Wireless Sensor Platforms

Haiyang Liu, Abhishek Chandra and Jaideep Srivastava
Dept. of Computer Science and Engineering, University of Minnesota
Minneapolis, MN 55414
{hliu, chandra, srivasta}@cs.umn.edu

ABSTRACT

Energy is a precious resource in wireless sensor networks as sensor nodes are typically powered by batteries with high replacement cost. This paper presents *eSENSE: an energy-efficient stochastic sensing framework* for wireless sensor platforms. eSENSE is a node-level framework that utilizes knowledge of the underlying data streams as well as application data quality requirements to conserve energy on a sensor node. eSENSE employs a stochastic scheduling algorithm to dynamically control the operating modes of the sensor node components. This scheduling algorithm enables an adaptive sampling strategy that aggressively conserves power by adjusting sensing activity to the application requirements. Using experimental results obtained on Power-TOSSIM with a real-world data trace, we demonstrate that our approach reduces energy consumption by 29-36% while providing strong statistical guarantees on data quality.

Categories and Subject Descriptors: C.3 [Special-purpose and Application-based Systems]: Real-time and embedded systems

General Terms: Algorithms, Management

Keywords: Scheduling, Energy Management, Sensor networks

1. INTRODUCTION

1.1 Sensor Energy Management

Energy efficiency has been widely recognized as a key issue that presents major challenges [1] in wireless sensor networks. Recent advances in sensor platform hardware design have made sensor nodes more energy-aware. Many sensor platforms now allow their main components, such as the CPU, the radio, and the sensor, to have multiple operating modes with significantly different power levels [2, 3, 4]. Even low-end sensors such as temperature/humidity sensors on the Telos platform [3, 5] now allow automatic mode switching.

Extensive research has been conducted on achieving energy-efficient operations through intelligent mode switching of main components of sensor platforms. Most existing research efforts in sensor energy management have focused on optimizing the power consumption of the radio and the CPU [1, 6]. These efforts have been driven largely by the conventional wisdom that these components consume most of the power on a sensor node [1]. In reality, the operation of sensors can be critical in determining the lifetime of a sensor node for the following reasons. First, specialized sensors can be energy consuming. For example, the heading sensor offered by xBow [7] can consume a power of about 375 mW, which is much higher than the 60 mW consumed by the mica2 radio transmitting at full power. Second, after common CPU and radio energy management, even low-power sensors, if not well-managed, could account for a significant fraction of the total energy consumption. Our experiments, presented in Section 4, reveal that the SHT series temperature sensor integrated on the Telos platform, that uses only 1.65 mW of power while sampling, could consume up to 38% of the total energy at a modest sampling rate of 0.1 Hz. Excluding the idle energy consumption, which can be improved only through better hardware design, the percentage of sensing is even higher (about 45-90%). Thus, effective modulation of the sensor operating modes is crucial for better energy conservation. Moreover, reduced sensing activity enables the CPU and the radio to spend more time in sleep mode, thus resulting in even higher energy savings for these components. Therefore, we believe that *sensor power control is not only desirable, but essential for sensor platform energy management*.

1.2 Dynamic Data Quality Requirements

Sensor platforms support versatile applications, which have widely varying data quality requirements from the sensor data streams [8]. For instance, a Heating, Ventilation and Air-conditioning (HVAC) application might require fine-grained temperature readings of a building, while coarse-grained readings may be sufficient for a fire monitoring application that is only concerned with the temperature exceeding a pre-defined threshold. In addition, data quality requirements may change even for the same application over different time periods and for different value ranges [8]. For example, the HVAC application may require more precise readings during daytime when offices are occupied, while only coarse measurements might be sufficient at night when offices are empty. System support for dynamic data quality on sensor nodes also provides applications with an effective means to achieve graceful performance degradation [9] when the network is congested or the sensor nodes are constrained. In case of such constraints, the application can throttle the data sensing and transmission rates by reducing its data fidelity requirement. As a result, *sensor platforms must be able to satisfy dynamic data quality requirements*.

1.3 Adaptive Data Sampling

Due to dynamic data quality requirements, the determination of the data sampling rate on a sensor must be driven by application semantics and the dynamics of the measured data. Existing sensor network applications such as TinyDB [10] do not account for these requirements, and the conventional sampling rates used in such applications are static user-supplied parameters. Statically-defined sampling rates result in either energy wastage under stable conditions, or unsatisfactory sample quality when the physical phenomenon experiences rapid changes. It is thus desirable to *provide adaptive sampling as a system service to end applications*, which only need to supply semantic data requirements. This distinction between the semantic and the actual sampling rates would benefit both low as well as high data rate applications by achieving a better tradeoff between energy and data quality.

The demand for dynamic data quality and support for adaptive sampling on sensor platforms calls for a novel approach to energy management for sensor platforms. Since sensor data streams are measurements of physical phenomena, correlations within data streams are inherent. For instance, the temperature variation in a room is governed by heat transfer laws, which limit the amount of variation that can occur between two successive temperature readings. Such temporal correlations can be exploited for energy management by taking sensor measurements only when large variations are expected in the underlying data values. In this paper, we present *eSENSE: an energy-efficient sensing framework* that utilizes knowledge of the underlying data streams to conserve energy on a sensor node, while satisfying the application data quality requirements. Coupled with data stream prediction models and data quality models, our scheduling framework dynamically controls the operating modes of the sensor node components. The core of our approach is a stochastic scheduling algorithm that performs data sampling in a probabilistic manner. Using experimental results obtained on an enhanced version of PowerTOSSIM [2], we demonstrate that our approach reduces energy consumption by 29-36% while providing strong statistical guarantees on data quality.

1.4 Research Contributions

This paper makes the following research contributions. (a) We propose a *semantic sensing framework* that distinguishes semantic sampling rate from actual system sensing operations. This framework provides a generic sensor scheduling service that integrates system operations with application semantics. (b) We propose a *stochastic sensor scheduling* algorithm that combines data stream predictions with application data quality requirements to produce a sampling schedule for each sensing device. Our scheduling algorithm provides statistical guarantees on data quality while substantially reducing the energy consumption of the sensor platform.

2. SYSTEM ARCHITECTURE

Figure 1 shows the eSENSE framework architecture. In this section, we describe the various components of the framework and their models. We describe the stochastic scheduling algorithm in Section 3.

2.1 Data Quality Model

The quality of measurement data can be generally quantified in terms of temporal resolution, measurement resolution, and sampling quality. Temporal resolution refers to the maximum available sampling frequency, which determines the granularity of temporal changes that can be captured in the data stream. We define this maximum sampling frequency as the *base sampling frequency*. The base sampling frequency could depend on the physical limitations of the sensing device, the available communication bandwidth, or the highest temporal resolution required by the application. Thus the sampled data sequence at the base sampling frequency represents the closest approximation to the underlying process that could be achieved by a sensor node in an application. We refer to this data sequence obtained by sensing at the base sampling frequency as the baseline data sequence (illustrated in Figure 2).

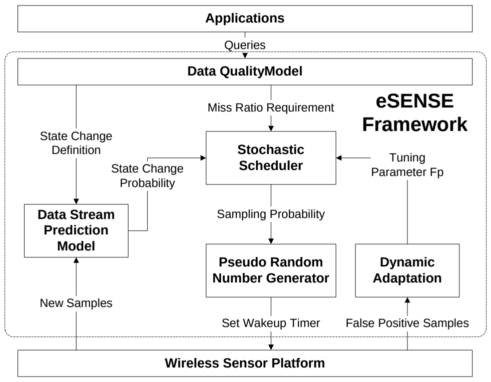


Figure 1: The eSENSE framework.

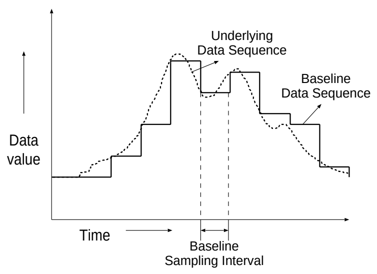
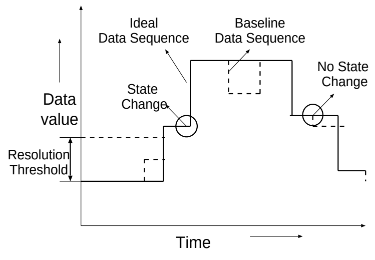


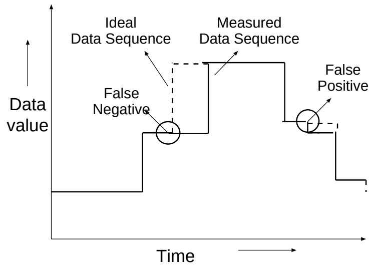
Figure 2: The baseline data sequence approximates the underlying process by sampling at the baseline frequency.

The concept of measurement resolution refers to a data range around the measured value that contains the actual data value. For example, a measurement resolution of 2^oC for a measurement value of 100^oC means that the true value is bounded in the range (98^oC, 102^oC). We refer to this measurement resolution as the *relative resolution threshold* δ, capturing the relative error range. In addition, we use the term *absolute resolution threshold* to denote the absolute difference between the measured value and an absolute threshold. Absolute resolution threshold is commonly used in predicate-based filtering operations in sensor network queries. We then define *state change* to be a change in the data value exceeding the resolution threshold. Intuitively, a state change corresponds to an interesting sensor measurement that needs to be reported to the application.

In the proposed eSENSE framework, unnecessary sensing operations are avoided through adaptive sampling, i.e., data is sensed only when a state change is expected. With this approach, it is possible to miss certain state changes if data was not sampled at those time instances. We refer to such missed state changes as *false negatives or misses*, as they correspond to a false expectation of not having a state change when actually there is one. Non-zero misses are commonly acceptable to many monitoring and aggregation estimation applications in sensor networks. Similarly, it is possible for this sensing approach to make a measurement when there is no actual state change. We refer to such redundant sensing events as *false positives or false hits*, as they correspond to a false expectation of observing a state change when there is none. Figure 3 illustrates



(a) Ideal data sequence



(b) Measured data sequence

Figure 3: Data sequences produced by using different adaptive sampling strategies: (a) An ideal strategy samples data only when there is a state change based on the resolution threshold δ. (b) A practical strategy tries to emulate the ideal strategy, but has some false negatives and some false positives.

the notion of false negatives and false positives for a data sequence by comparing it to an ideal data sequence (that consists of samples taken only at state change points).

A sensing strategy should strive to minimize *both* false negatives as well as false positives: while false negatives result in degraded data quality, false positives result in energy wastage. We define two quantities, the *miss ratio* $\gamma = \frac{n_f}{n}$ and the *false hit ratio* $\rho = \frac{n_p}{n}$, to quantify the degradation in data quality and wasteful sampling respectively. Here, n_f and n_p denote the number of misses and false hits respectively, and n denotes the total number of sampling points (corresponding to the base sampling frequency).

2.2 Data Stream Prediction Model

eSENSE employs a data stream model to predict future sensor readings from historical data. Statistical models are particularly suitable for sensor network applications [11]. While several sophisticated statistical models [12, 13] can be used, we used a biased random walk model in our experiments. This model is a type of first-order Markov model that we chose for its computational efficiency and compact representation. This simple model can readily capture the intrinsic correlations in data streams and is sufficient to evaluate the effectiveness of our scheduling algorithm.

In this model, a k-step prediction is given by: $X_{i+k} = X_i + N(\mu_k, \sigma_k)$, where, X_i denotes the data value at time instance i, X_{i+k} denotes the predicted data value at time $i + k$, i.e., k time steps forward from step i, and $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. The possibly non-zero mean value μ or the bias, captures the systematic trend in the data stream, while σ captures the process random noise and non-linear error components. More details on the model can be found in [14].

Given the data quality model with a resolution threshold δ, the state change probability can be computed by looking up the probability value in a locally stored unit normal distribution table and applying appropriate transformations. The model can be constructed from a training data stream and updated with new data samples. The initial construction and subsequent updates could be carried out at base stations, similar to [11], taking advantage of the storage and computing power of base stations in addition to more complete view of measurement data streams. Since the prediction models take small number of parameters and are updated infrequently, the amortized communication cost of model updates on a sensor node is expected to be negligible.

3. SCHEDULING ALGORITHM

In this section, we present a stochastic scheduling algorithm that employs the underlying data stream model and the data quality requirement to determine sampling instants for the sensor. The goal of this algorithm is to minimize the sensor energy consumption while meeting the desired data quality requirements. The intuition behind our algorithm is to sample with high probability at instants when state change probabilities are expected to be high. Note that our stochastic scheduling algorithm does not depend on the specific prediction model and data quality model being used, and can be used in conjunction with any kind of models as long as they can estimate state change probabilities at future time instants. Next, we formalize the scheduling problem and present our solution.

3.1 Problem Formulation

We formulate the stochastic scheduling problem as an optimization problem that minimizes the total energy consumption while providing statistical guarantees on data sampling quality.

Let us assume that the baseline data sequence consists of N data samples, and the probability of state change at a sampling instant i (determined using the underlying data stream model and the application's resolution threshold δ) is q_i. Further assume that the average energy spent for each measurement is e_{avg} (this includes the average energy spent by the sensor, CPU, and the radio). Finally, let the application's data quality requirement be expressed as a *tolerance level* $F_N \in [0, 1]$, such that its miss ratio $\gamma \leq F_N$. Then, the goal of the stochastic scheduling algorithm is to determine a probability of sensing $p_i \in [0, 1]$ for each sampling instant such that it minimizes the total energy $E = \sum_{i=1}^{N} p_i \cdot e_{avg}$ under the constraint

$$\frac{\sum_{i=1}^{N} (1 - p_i) \cdot q_i}{N} \leq F_n. \quad (1)$$

The constraint given by Inequality 1 satisfies the statistical data quality requirement of the application as we require the *expected* miss ratio to be less than the application-specified tolerance level. Recall from Section 2.1 that the expected miss ratio, γ, is evaluated as the expected number of false negatives divided by the total number of data samples. Thus, given the false negative probabilities f_{n_i} over all sampling instances, we have,

$$\gamma = \frac{\sum_{i=1}^{N} f_{n_i}}{N}. \quad (2)$$

To catch state changes more effectively, the higher the probability of state change q_i, the higher the probability of sensing p_i should be. Therefore, we assume that p_i is, by design, positively correlated to the probability of state change q_i at each sampling instant. Hence, the false negative probability f_{n_i} at each scheduling instant is less than what would be obtained by assuming independence between p_i and q_i. In other words,

$$f_{n\,i} \leq (1 - p_i) \cdot q_i.$$

Thus, the miss ratio (Equation 2) reduces to

$$\gamma = \frac{\sum_{i=1}^{N} f_{n\,i}}{N} \leq \frac{\sum_{i=1}^{N} (1 - p_i) \cdot q_i}{N}$$

Thus, the constraint (Inequality 1) satisfies the data quality requirement $\gamma \leq F_n$. In fact, the constraint is a conservative bound on the data quality requirement, such that if a schedule satisfies the constraint, it must also satisfy the data quality requirement.

3.2 Scheduling Algorithm

Having presented the problem formulation, we now present a stochastic scheduling algorithm that closely approximates the optimization problem. The goal of the scheduling algorithm is to determine the sensing probability p_i for each sampling instance given the state change probability q_i for that scheduling point. Given q_i, solving for the precise value of p_i would require the joint distribution of the random processes of sampling and state changing. This distribution is neither available nor desirable due to its high storage and computational overhead. Instead, we simplify the computation of p_i as follows: we first determine the upper and lower bounds for p_i, and the scheduling algorithm then chooses a value from this range based on a heuristic we describe later. Intuitively, the upper bound of p_i specifies a limit such that selecting values higher than it would only waste energy for providing unnecessary data quality improvement. On the other hand, the lower bound of p_i corresponds to a limit, such that going below it would always result in violation of the application's tolerance level.

Determining the Sensing Probability Upper Bound

To determine the upper bound on the value of p_i for a given q_i value, our scheduling algorithm performs local optimization instead of global optimization. Note that local optimization meets a stricter requirement since satisfying the constraint at each sampling instance automatically satisfies the constraint over all sampling instances. In other words, the optimization problem is reduced to minimizing p_i at each scheduling instant under the constraint $(1 - p_i) \cdot q_i \leq F_n$, which yields the solution

$$p_i^{ub} = \begin{cases} 0, & \text{if } 0 \leq q_i \leq F_n \\ 1 - \frac{F_n}{q_i}, & \text{if } F_n < q_i \leq 1 \end{cases}$$

In other words, p_i^{ub} is the *minimum* value of p_i that guarantees the satisfaction of the data quality requirement for *each* sensing instance. This value of p_i is an upper bound on the value of the sensing probability, because, any sensing probability value higher than p_i^{ub}, while always satisfying the local optimization constraint, would be more wasteful of energy.

Determining the Sensing Probability Lower Bound

To determine the lower bound on the value of p_i, we consider the most optimistic scenario where every sample catches a real state change, i.e., there are no false positives. In this scenario, the data quality requirement can be satisfied only if $q_i - p_i \leq F_n$, or $p_i \geq q_i - F_n$, which provides us with the following lower bound:

$$p_i^{lb} = \begin{cases} 0, & \text{if } 0 \leq q_i \leq F_n \\ q_i - F_n, & \text{if } F_n < q_i \leq 1 \end{cases}$$

This value of p_i is the lower bound because any sensing probability value smaller than p_i^{lb} would always result in violating the data quality requirement. Thus, the value p_i^{lb} corresponds to the *smallest* value of p_i given q_i, that meets the data quality constraint.

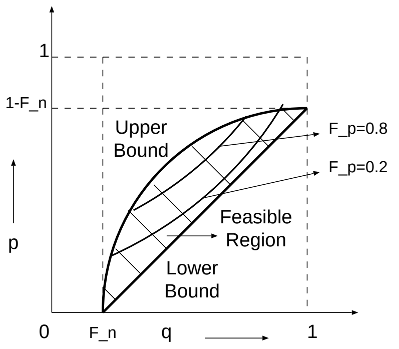


Figure 4: The relation between sensing probability p_i and the state change probability q_i. F_n is the tolerance level that determines the bound on data quality, while F_p is the false hit ratio threshold that is used as a tuning parameter to select a value from the feasible region bounded by the upper and lower bounds of p_i.

Selecting the Sensing Probability Value

Given the upper and lower bounds on the value of p_i given q_i, we present a heuristic to select the actual value of p_i. Note that the application uses a miss ratio bound F_n to limit the data quality degradation. Analogously, we can bound the energy wastage by using a false hit ratio limit F_p. Our heuristic uses this limit F_p as the tuning parameter to determine the p_i value from the region bounded by p_i^{lb} and p_i^{ub}. Lower values of F_p correspond to more aggressive energy saving, while higher values of F_p provide better data quality at the expense of higher power consumption. F_p can be approximated as $F_p = p_i \cdot (1 - q_i)$, which yields

$$p_i = \frac{F_p}{(1 - q_i)},$$

subject to the two bounds derived above. The stochastic scheduling algorithm then uses this p_i value to probabilistically schedule a sensing event at a sampling point. Note that this formula also satisfy the design principle that the higher state change probability is, the higher sampling probability would be.

Figure 4 shows the relation between the sensing probability p_i and the state change probability q_i for a given value of the data quality threshold F_n. The figure also shows the intermediate values that p_i would take based on the value of the tuning parameter F_p.

Dynamic Adaptation

While major trend change of the measurement data stream can be captured by model updates, local fluctuations and inaccuracy in estimation of q_i may lead to poor scheduling decisions affecting the sample data quality. Thus, it is important to ensure that our scheduling scheme adapts to sudden or unforeseen data variations. While it is not possible to directly observe false negatives (corresponding to missed state changes), we can measure false positive rates to estimate the dynamism in the underlying data. Intuitively, a low rate of false positives implies that most of the sensing events result in state changes, suggesting the possibility of missing other significant changes. Thus, a low false positive rate could be taken as an indication of more dynamic data values, and the number of sampling events should be increased in this case to catch possibly significant state changes. On the other hand, if we observe a high rate of false positives, it means that we are taking large number of redundant samples, many of which are non-informative. Such a high rate indicates a relatively stable data process, and the sampling probability should be decreased in this case to save energy. We use the tuning parameter F_p to achieve this dynamic adaptation of the sampling probability p_i. A Multiplicative Increment Additive Decrement algorithm DynamicAdapt is illustrated in Algorithm 1 in hope of fast responding to sudden events.

Practical Considerations

While sampling decision must be made for each time instant, it is inefficient to compute at each instant. Instead, at each scheduled sensing instant (when the CPU is turned on anyway for the sensing operation), the stochastic scheduler determines the next sampling instant using a pre-generated random number sequence and the sequence of sampling probabilities. Algorithm 2 shows the pseudo-code for our stochastic scheduling algorithm.

Algorithm 1 DynamicAdapt (F_p, $sample$)

```
 1: GlobalVariables: w windowSize, ρ falseNegThreshold,
    BM windowBitMap, m multiplier, step
 2: BM ≪ 1
 3: if $sample$ is a false hit then BM = BM OR 1
 4: $\rho_o$ = number of 1's in BM / $w$
 5: if $\rho_o \geq \rho$
 6:    then $F_p$ = $F_p$ - $step$
 7:    else $F_p$ = $multiplier$ * $F_p$
 8: endif
 9: if $F_p < 0$ then $F_p$ = 0
10: if $F_p > 1$ then $F_p$ = 1
11: return $F_p$
```

Algorithm 2 Schedule (F_n, F_p)

```
 1: $k$ = 0, $P_k$ = 1.0, $rand$ = 0.0
 2: $sample \Leftarrow$ take a sample from the sensor
 3: $F_p \Leftarrow$ DynamicAdapt($F_p$, $sample$)
 4: while ($P_k > rand$)
 5:    k = k+1
 6:    ($\mu_k$, $\sigma_k$) $\Leftarrow$ apply data stream prediction model
 7:    $q_k \Leftarrow$ apply data quality model on ($\mu_k$, $\sigma_k$)
 8:    $P_k^{ub} \Leftarrow$ apply formula in Section 3.2
 9:    $P_k^{lb} \Leftarrow$ apply formula in Section 3.2
10:    $P_k \Leftarrow$ apply formula in Section 3.2
11:    if $P_k > P_k^{ub}$ then $P_k = P_k^{ub}$
12:    if $P_k < P_k^{lb}$ then $P_k = P_k^{lb}$
13:    $rand \Leftarrow$ apply [0,1] pseudo random number generator
14: endwhile
15: return k
```

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

We have implemented an eSENSE prototype on TinyOS and evaluated its performance for the Telos platform [4] in PowerTOSSIM [2]. The energy model is extracted from [3] and summarized in [14]. In order to enable accurate sensing power estimation, we extended PowerTOSSIM by generating debug messages, carrying the channel numbers and the current time-stamp, whenever an ADC reading request is issued. An ADCDataLoading plugin was also developed for the TinyViz visualization tool to allow dynamic loading of data traces for sensor readings. This ADCDataLoading plugin also enabled catching all sensing-related debug messages to compute the corresponding sensor energy consumption.

We use real-world temperature readings to test the effectiveness of our prototype. This data was sampled in an air-conditioned storage room using an SHT11 temperature sensor at sampling frequency of 0.1Hz for two days. In order to capture greater temperature variations, the sensor was placed close to a ventilation exit. The collected data trace is shown in Figure 5. The simulation time period starts at the data point corresponding to about 6 am on the second day in the trace, when air conditioning is configured to turn on in the room. This choice of data set results in richer variations in the test data. A simulation period of 10,000 seconds (corresponding to 1000 sample points) was selected for each run. In order to reduce the artifact of pseudo randomness, each simulation run was repeated multiple times with different random seeds and the arithmetic mean is reported.

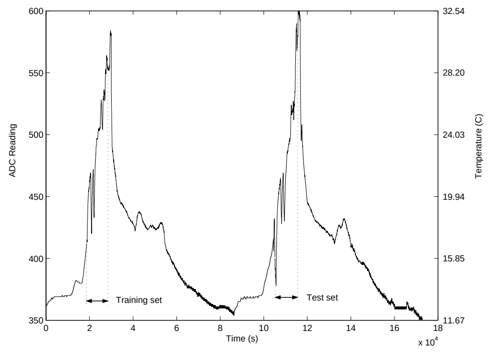


Figure 5: Data trace consisting of two days of temperature variation in a room.

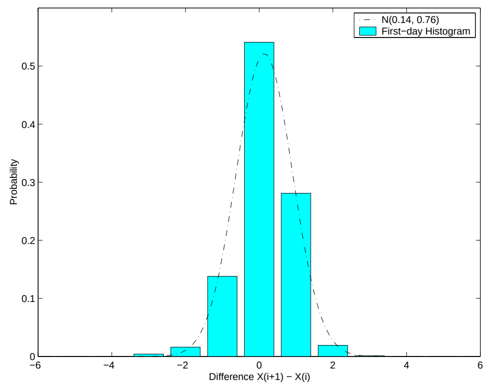


Figure 6: Gaussian approximation to the histogram of a single-step predictor (k=1)

4.2 Model Construction

Environmental data such as temperature readings usually exhibits time-of-day patterns as seen from Figure 5. Therefore, we use the same time period of 10,000 seconds from the first day as the training set for constructing the data stream model (described in Section 2.2). Figure 6 shows the data histogram and the corresponding Gaussian distribution approximation for single-step prediction (k=1). The close approximation seen in this figure validates our selection of the biased random walk model. Overall, parameters of up to 20-step prediction models were computed in this experiment.

4.3 Performance Results

We begin the performance evaluation by measuring the overhead of eSENSE. We then present experimental results obtained by varying the different experimental parameters such as the application's resolution threshold and the miss ratio confidence level. We present two cases: one with relative resolution thresholds and the other with absolute threshold specification. For each experiment, we examine the energy savings and the level of data quality provided.

In our experiments, we compare the following five sampling strategies. (a) *Baseline sampling:* baseline sampling corresponds to taking sensor readings at the baseline frequency. We use baseline sampling for comparison as it reflects the current best practice in sensor energy management. Note that baseline sampling represents the worst-case sampling strategy in terms of the number of samples taken, while achieving zero miss ratio. (b) *Ideal sampling:* ideal sampling is a clairvoyant sampling strategy based on an application's data resolution threshold that takes samples only when there is a state change. Thus ideal sampling consumes the least possible energy while achieving zero miss ratio. (c) *eSENSE with upper bound sampling probability:* this is our stochastic scheduling algorithm that employs the upper bound of the sampling probability (described in Section 3.2) to determine the sampling points. Recall that this choice of sampling probability corresponds to the most conservative choice for meeting the application's miss ratio requirement. (d) *eSENSE with lower bound sampling probability:* this is similar to the previous strategy except that it employs the lower bound of the sampling probability. This strategy is the most energy-efficient scheme while providing the most relaxed guarantees on data quality. (e) *Dynamic eSENSE:* this strategy employs the stochastic scheduling algorithm with dynamic adaptation. It adapts to the current data stream conditions and thus provides a tradeoff between energy consumption and data quality guarantees.

Note that the same send-on-change data transmission strategy is employed for all of the above sensing schemes. Also, when there is no need for sensing, the CPU, the sensor and the radio are always in sleep mode.

eSENSE Overhead

The energy overhead in eSENSE consists of two main components. First, at each sampling instant, eSENSE incurs a fixed energy overhead due to the extra CPU time spent computing the state change probability, sampling probability, and generating a pseudo-random number for making a sampling decision. To compute this overhead, we used the CPU profiling tools in PowerTOSSIM to measure the number of CPU cycles used for these computations. It turned out to be negligible as, on average, only 660 cycles were needed for each time point, corresponding to a total energy overhead of 0.89 mJ for the simulation period (less than 0.5% of the total energy consumption). The second energy overhead component is the energy needed for model update. However, as we discussed in Section 2.2, the model update would be typically done on a base station, and the model would then be loaded on the sensor node. Thus the only overhead incurred on the sensor node by this component is the energy spent in downloading the model, which is extremely small as, in practice, these models would either be pre-loaded or infrequently updated (weekly or monthly). This is true even for many applications monitoring fast changing environments such as vehicle tracking since model updates are needed only when long term trends change significantly. Local fast changing variations will be picked up by local dynamic tuning of scheduling parameters. We also measured the eSENSE memory overhead to be 2310 bytes in ROM and 242 bytes in RAM. These results show that eSENSE implementation is compact and efficient.

Relative state change threshold

We first present results for experiments that use a relative state change threshold. In this case, a state change occurs when the difference between the new data value and the most recently sent value is greater than the threshold. For each experiment, the application specifies a *confidence level*, whose corresponding *tolerance level* (equal to *1 - confidence level*) specifies the maximum miss ratio it is ready to suffer.

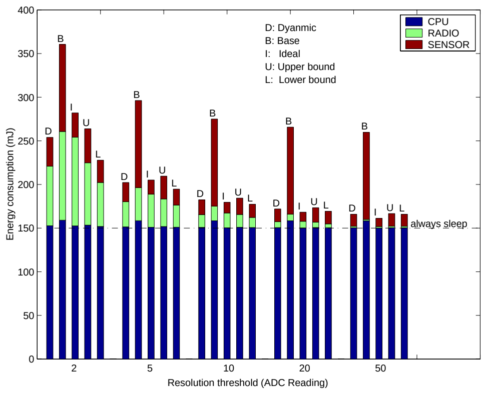


Figure 7: Energy consumption for confidence level = 90%.

Figure 7 shows the energy consumption of the various scheduling strategies. The horizontal line in the figure denotes the minimum energy needed when there is no activity. This energy corresponds to the absolute minimum energy required just to keep the sensor node powered on (with all components in sleep modes). This energy reflects the hardware limit imposed on any power management scheme.

We make several interesting observations from Figure 7. First, after employing the commonly used send-on-change strategy, the radio component is no longer the most energy-consuming component (consuming only about 0.7-28% of the total energy). Second, the CPU now consumes the largest amount of energy (about 44-61% of the total), most of which is spent in sleep mode. Third, sensing now consumes a significant percentage of the total power. For instance, with baseline sampling, sensing accounts for about 28% and 38% of the total energy for state change thresholds of 2 and 50 respectively. These results verify our contention that even low-power modality sensors can account for a large percentage of energy consumption.

We also see from Figure 7 that the energy consumption of the dynamic scheduling algorithm (166-254 mJ) lies between those of the upper bound (167-264 mJ) and the lower bound (166-228 mJ) strategies. Dynamic scheduling can reduce the baseline energy consumption by about 29% to 36%. In addition, the dynamic scheduling algorithm consumes almost the same amount of energy as the ideal sampling strategy. In some cases (threshold = 2 and 5), the dynamic strategy even consumes *less* energy than ideal sampling. Note that this is possible because dynamic sampling is allowed a 10% miss ratio, while ideal sampling has zero miss ratio.

Figure 8 shows the miss ratios associated with the three eSENSE sampling strategies. As can be seen from Figure 8, the dynamic (1-9.4%) and upper bound (0.7-6.8%) sampling strategies always stay within the tolerance level (10%) for all values of resolution threshold δ. The lower bound strategy yields lower miss ratios for high δ values (corresponding to coarser accuracy requirement), but goes above the tolerance level to 17.4% and 13% respectively for low δ values (fine-grained accuracy) of δ = 2 and 5.

Figures 9 and 10 show the performance of the various strategies for a fixed resolution threshold of 5, with varying miss ratio tolerance levels. Figure 9 shows upto an additional 18% energy saving

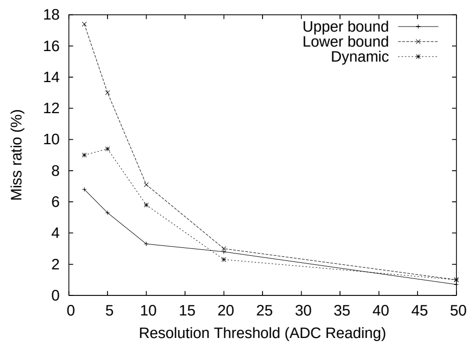


Figure 8: Miss ratio for confidence level = 90%.

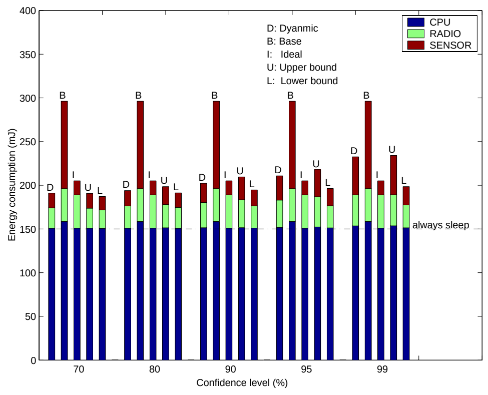


Figure 9: Energy consumption for resolution threshold = 5.

for dynamic strategy when varying the confidence tolerance level from 99% (tolerance = 1%) to 70% (tolerance = 30%). Figure 10 again verifies that the dynamic and upper bound strategies always guarantee the required confidence level.

As described in Section 3.2, in the dynamic eSENSE scheduling strategy, the false positive threshold F_p is used as a dynamic tuning parameter to adjust sampling probabilities p_i following variations in the underlying processes. The values of the sampling probability p_i obtained in this manner are an indicator of how energy consumption is distributed temporally. Figure 11 shows the variation of the tuning parameter F_p and the sampling probability p_i over the time period of the test data trace. As can be seen from the figure, higher values of F_p, and thus higher sampling probabilities (corresponding to higher expected energy consumption) occur at time periods

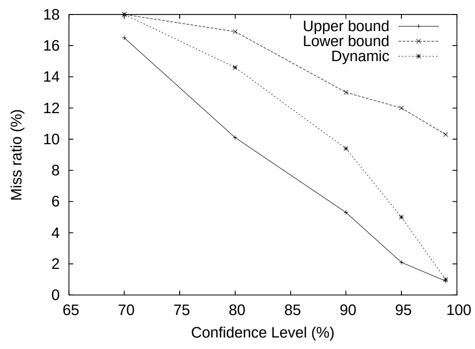


Figure 10: Miss ratio for resolution threshold = 5.

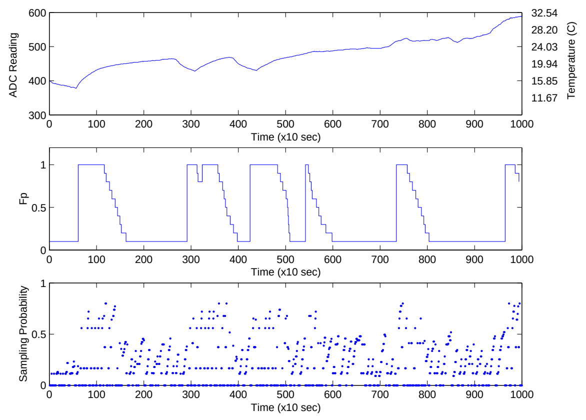


Figure 11: The variation of the tuning parameter F_p and the sampling probability p_i, relative threshold = 5 and confidence level = 90%.

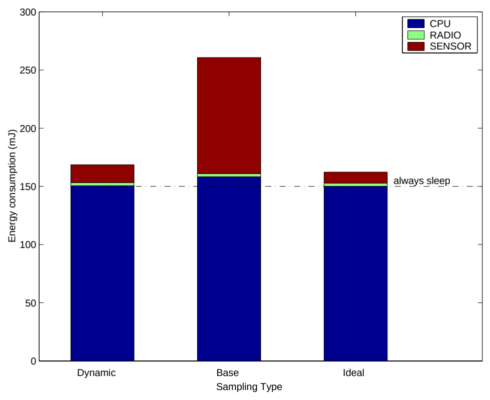


Figure 12: Energy consumption for absolute resolution threshold = 450 and confidence level = 99%.

with larger data variation. This result indicates that the dynamic algorithm is able to adapt to changes in the underlying phenomenon and spends more energy during stages of flux. This figure also shows the ability of the dynamic scheduling algorithm to achieve adaptive sampling without any model updates.

Absolute state change threshold

With an absolute threshold, a state change is defined to occur only when the data value crosses the threshold. The same data source and prediction models are used for these experiments. Figures 12 and 13 present results generated from an experiment using an absolute threshold of 450 and confidence level of 99%. Figure 12 shows that eSENSE can achieve a 35% energy saving compared to baseline sampling and consumes only about 4% more than the ideal sampling strategy. The miss ratio for eSENSE scheduling is measured at 0.8%, satisfying the confidence level requirement. Figure 13 shows that higher sampling probabilities (more energy) are generated in the period when data values are close to the threshold.

5. RELATED WORK

Recently, several research efforts have focused on energy-efficient operations on wireless sensor platforms. These efforts include minimizing data transmission through data compressions [13] and saving energy by turning redundant nodes off while maintaining re-

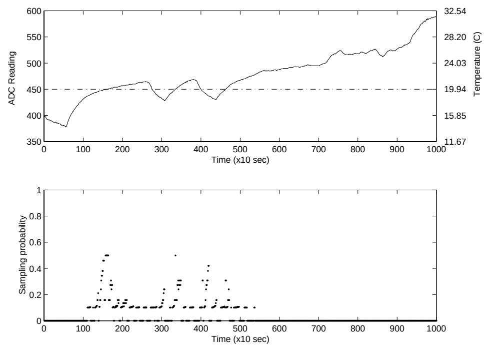


Figure 13: The variation in the sampling probability p_i, absolute threshold = 450 and confidence level = 99%.

quired field coverage [15]. [16] introduces a node-level energy allocation scheme to maximize the overall gain to multiple applications running on a node. [11] uses a statistical model at the base station to optimize query plans by picking the optimum set of attributes and nodes for data acquisition. However, it considers only communication cost in its cost model and does not provide real time scheduling service in response to sudden events. eSENSE is complementary to these approaches in that our sensor scheduling framework is a node-level real time approach that primarily targets sensing energy conservation. eSENSE thus can be used in conjunction with existing work to further reduce energy consumption.

Dynamic Power Management (DPM) [17] and multiple sensing unit scheduling (MSUS) [18] also attempt to control the operating modes of sensor node components in response to different workloads. Prediction-based dynamic power control has also been used in energy management in mobile systems [19, 20] and embedded systems [21, 22]. However, unlike our approach, they are completely unaware of application semantics.

Stochastic sensor scheduling has also been used in target-tracking applications [23] for maximizing estimation accuracy. It is fundamentally different from our goal of minimizing energy consumption given an accuracy requirement.

6. CONCLUDING REMARKS

In this paper, we presented eSENSE: an energy-efficient sensing framework for wireless sensor platforms, that can achieve significant energy savings in response to dynamic data quality requirements. Our scheduling framework dynamically controls the operating modes of the sensor node components using a stochastic scheduling algorithm coupled with data stream prediction models and data quality models. Using experimental results obtained on PowerTOSSIM with a real-world data trace, we showed that our approach reduces energy consumption by 29-36% while providing strong statistical guarantees on data quality.

As part of future work, we intend to extend our work to schedule data transmissions on the network by modeling the radio component as a pseudo-sensor and to multiple sensors by exploring their correlations.

7. REFERENCES

[1] D. Estrin, A. Sayeed, and M. Srivastava, “Wireless Sensor Networks,” Sept. 2002, tutorial at MobiCom’02.

[2] V. Shnayder, M. Hempstead, B. Chen, G. Werner-Allen, and M. Welsh, “Simulating the Power Consumption of Large-Scale Sensor Network Applications,” in *Proceedings of SenSys’04*, Nov. 2004.

[3] “Telos/Tmote Datasheet,” Dec. 2004, http: //www.moteiv.com/products/docs/tmote-sky-datasheet.pdf.

[4] J. Polastre, R. Szewczyk, and D. Culler, “Telos: Enabling Ultra-Low Power Wireless Research,” in *Proceedings of IPSN’05*, Apr. 2005.

[5] “SHT1x/SHT7x Humidity and Temperature Sensor Datasheet,” July 2004, http: //www.sensirion.com/en/pdf/Datasheet_SHT1x_SHT7x.pdf.

[6] A. Boulis, S. Ganeriwal, , and M. B. Srivastava, “Aggregation in sensor networks: a energy-accuracy trade-off,” in *Proceedings of SNPA’03*, May 2003.

[7] xBow Technology Inc., “Heading Sensor Datasheet,” July 2004, http://www.xbow.com/Products/Product_pdf_files/ Mag_pdf/6020-0030-01_A_C%HS110.pdf.

[8] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, “Load Shedding in a Data Stream Manager,” in *Proceedings of VLDB’03*, Sept. 2003.

[9] A. Deshpande, C. Guestrin, and S. Madden, “Resource-aware Wireless Sensor-Actuator Networks,” *Data Engineering Bulletin*, vol. 28, no. 1, pp. 40–47, Mar. 2005.

[10] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks,” in *Proceedings of OSDI’02*, Dec. 2002.

[11] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, “Model Driven Data Acquisition in Sensor Networks,” in *Proceedings of VLDB’04*, Aug. 2004.

[12] R. Vilalta, C. Apte, J. L. Hellerstein, S. Ma, and S. M. Weiss, “Predictive algorithms in the management of computer systems,” *IBM Systems Journal*, vol. 41, no. 3, pp. 461–474, 2002.

[13] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, “Compressing Historical Information in Sensor Networks,” in *Proceedings of SIGMOD’04*, June 2004.

[14] H. Liu, A. Chandra, and J. Srivastava, “dSENSE: Data-driven Stochastic Energy Management for Wireless Sensor Platforms,” Dept. of CSE, Univ. of Minnesota, Tech. Rep. TR 05-018, May 2005.

[15] Z. Abrams, A. Goel, and S. Plotkin, “Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks,” in *Proceedings of IPSN’04*, Apr. 2004.

[16] A. Boulis and M. B. Srivastava, “Node-level Energy Management for Sensor Networks in the Presence of Multiple Applications,” in *Proceedings of PerCom’03*, Mar. 2003.

[17] A. Sinha and A. Chandrakasan, “Dynamic Power management in Wireless Sensor Networks,” *IEEE Design and Test*, vol. 18, no. 2, pp. 62–74, Mar. 2001.

[18] H. Cam, R. Poornachandran, and H. Ahmad, “Energy-efficient Task Scheduling for Wireless Sensor Nodes with Multiple Sensing Units,” in *Proceedings of IPCCC’05*, Apr. 2005.

[19] X. Liu, P. Shenoy, and W. Gong, “A Time Series-based Approach for Power Management in Mobile Processors and Disks,” in *Proceedings of NOSSDAV’04*, June 2004, pp. 74–81.

[20] J. Lorch and A. Smith, “Operating System Modifications for task-based speed and voltage scheduling,” in *Proceedings of MobiSys’03*, May 2003, pp. 215–229.

[21] D. Li, P. H. Chou, and N. Bagherzadeh, “Mode Selection and Mode-Dependency Modeling for Power-Aware Embedded Systems,” in *Proceedings of the 7th Asia and South Pacific Design Automation Conference*, Jan. 2002, pp. 697–704.

[22] M. Srivastava, A. Chandrakasan, and R. Brodersen, “Predictive System shutdown and other architectural techniques for energy efficient programmable computation,” *IEEE Transactions on VLSI Systems (TVLSI)*, vol. 4, no. 1, pp. 42–55, Mar. 1996.

[23] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray, “Sensor Scheduling Algorithms Requiring Limited Computation,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2004.