Adaptive Probing and Communication in Sensor Networks

Iftach Ragoler, Yossi Matias, and Nimrod Aviram

School of Computer Science, Tel Aviv University
{ragoleri, matias, aviramni}@post.tau.ac.il

Abstract. Sensor networks consist of multiple low-cost, autonomous, ad-hoc sensors, that periodically probe and react to the environment and communicate with other sensors or devices. A primary concern in the operation of sensor networks is the limited energy capacity per sensor. As a result, a common challenge is in setting the probing frequency, so as to compromise between the cost of frequent probing and the inaccuracy resulting from infrequent probing.
We present adaptive probing algorithms that enable sensors to make effective selections of their next probing time, based on prior probes. We also present adaptive communication techniques, which allow reduced communication between sensors, and hence significant energy savings, without sacrificing accuracy. The presented algorithms were implemented in Motes sensors and are shown to be effective by testing them on real data.

1 Introduction

Sensor networks consist of multiple low-cost, autonomous, ad-hoc sensors, that periodically probe and react to the environment and communicate with other sensors or devices. A primary concern in the operations of sensor networks is the limited energy capacity per sensor. There has been extensive research aimed to reduce energy consumption in the application itself, network level, Datalink level and physical level. A few examples are works on network aggregation, improved routing algorithms, efficient communication, synchronizing wake-up time between nodes, data compression and efficient tracking algorithms.

This paper presents two primitives for energy saving in the service level: *adaptive probing* and *adaptive communication*. These primitives assume very little about the system settings or the behavior of the measured data, and thus many applications may gain significant energy savings simply by using the new primitives instead of raw probing and communication.

Adaptive Probing

In current settings, the probing rate of sensors is typically fixed, and is set as a compromise between the cost of frequent probing and the inaccuracy resulting from infrequent probing. In contrast, *adaptive probing* is based on variable rate

probing so that the base rate is lower than the typical one, and is increased as required when the probed measurements change more rapidly, or are approaching the vicinity of a predefined threshold. The *adaptive probing* primitive enables sensors to make effective selections of their next probing times, based on what they have learned so far. This results with reduced energy consumption while improving accuracy.

The energy saving obtained when using adaptive probing is three fold: (1) reduced probing: as probing is the most frequent operation in sensor networks and thus reduction in the average probing rate could be significant; (2) reduced communication, as a result of reduced probing - the communication step is typically the most expensive operation in sensor networks; and (3) reduced wake-up time: the average sleep time is increased resulting with potentially reduced energy consumption.

The problem of devising a suitable adaptive probing can be formalized as an optimization problem with respect to cost versus approximation error: given a stream of data, our goal is to probe the stream so as to obtain as good approximation as possible, with minimum number of probes. We denote by *Approximate Probing* (*AP*) a method for online scheduling of probes with the goal of learning the measured values with minimum error and minimum cost.

Approximate probing is suitable for settings in which the entire pattern of a measured value over time is of interest. There are other situations of interest in which the sensor network has to alert when certain values exceed a predefined threshold. Rather than having a constant rate probing, it is more effective to have an adaptive probing approach: the next probe should be set according to the approximate time to reach the threshold. We denote by *Threshold Probing* (*TP*) a method for efficiently alerting on a value reaching a threshold in minimum detecting latency and at a minimum cost.

Adaptive Communication

The second optimization primitive is *Adaptive Communication* (*AC*), which is a method for efficiently adapting the communication to the actual values of the data stream.

In *Adaptive Communication*, a sender and its receiver maintain each a synopsis of the stream history and compute each a prediction for the next value. When a node needs to send the next value, it sends a message only in case the value deviates significantly from its prediction, which is also shared (approximately) by its receiver. Otherwise this can be thought of as an *Implicit Send* operation, in which no value is actually transmitted and its receiver will compute the next value by itself based on its own prediction. In this setting, we show additional significant energy savings, as in many cases communication is a dominant factor in energy consumption.

In the heart of all our methodology is computing an instant, low cost model of the changes in the environment and issuing a tiny prediction from this model for the next desired probe or send time.

The requirement to obtain an instant prediction implies using a simple model to compute that prediction, so that it can be computed efficiently. There is extensive literature about prediction computations such as adaptive filters which include Weiner and Kalman filters (e.g., [4]), Box-Jenkins modeling (e.g., [1]) and time series analysis (e.g., [16]) to name a few. Those methods usually involve complex calculations that can be computed on a powerful server such as a base station, but are too costly to be computed on the sensors themselves. In addition, those methods typically compute the expected value in a particular time, rather than the expected time to get to some value as our method requires. It is possible to find an approximation to this inverse function using doubling and binary search, but this will be even more costly. We further elaborate on related works in Section 6. In contrast, our method is built on computing a tiny model in the severe constraint of the sensor itself and not in a powerful base station. Adding to this constraint the fact that the calculation should be done in real-time for the next probing or sending epoch, limits the possible complexity of the prediction. Our model makes no assumptions about the model of the environment (e.g., sinusoid or any other periodic model), thus it can serve as a basic primitive in large number of applications.

To fully exploit the advantages of adaptive probing and adaptive communication, a tiny low cost non-volatile memory is sufficient. In particular, about a dozen of registers that remain non-volatile in "sleep" period, and for which storing and retrieval is low-cost would be sufficient to hold the history synopses.

We devise basic concrete algorithms and validate their potential impact with experimentation. We tested the algorithms by simulation on data gathered from the Great Duck Island (GDI) [9]. We also implemented the algorithms on Mica motes and tested the performance of these implementations. For instance, at the GDI case we obtain up to 55% improvement in the error for the same number of probes (AP), up to 95% improvements in latency of threshold detection (TP), and up to 75% reduction in the number of messages needed for the same error (AC). In the Mica motes experiments we got up to up to 48% improvement in the error for the same cost (AP) and up to 70% reduction in the number of messages needed for the same error (AC).

2 Basic Prediction Techniques

We begin by presenting some basic low-cost prediction techniques that will later be used by both the adaptive probing and adaptive communication algorithms. Given a target value τ, one of our objectives is to predict the time in which the measured value will be τ, based on the recorded information. The prediction techniques, presented in increasing complexities, are used for computing the next probing time based on the recorded history of the measured stream, and also estimate the value of the stream in a future time.

Distance prediction. This technique is based on the distance between the current measured value and a given threshold. While the value is far from the thresh-

old, the node can probe at a slow rate, but as the value advances towards the threshold, the node should increase the probing rate.

Let Δt_{i+1} be the time between probe i and probe $i+1$, Let x_i be the value measured at time t_i. Our objective is to predict the time t_{i+1} at which the measured value will satisfy $x_{i+1} = \tau$. The time till next probe $\Delta t_{i+1} = t_{i+1} - t_i$ will be taken as linear in the distance from the threshold: $\Delta t_{i+1} = \Theta(\tau - x_i)$

There are no memory requirements for calculating the distance prediction, except for τ, as the next probing time is calculated using the current value only.

Linear Prediction. This technique adapts the local probing rate not only to the distance from the target value, but also to the rate of change in the measured values. Intuitively, while the probed value is changing fast, the node should probe at a high rate, but when the change is slow the node can probe at a slower rate and still detect the essential properties.

Computing the estimated value at different points is based on linear extrapolation. Given two probe points (t_{i-1}, x_{i-1}) and (t_i, x_i), the predicted value x_{i+1} in time t_{i+1} is computed as :

$$x_{i+1} = x_i + (t_{i+1} - t_i) \cdot \frac{x_i - x_{i-1}}{t_i - t_{i-1}} \tag{1}$$

Based on this prediction, the time t_{i+1} in which the value is expected to get to x_{i+1} is hence:

$$t_{i+1} = t_i + (x_{i+1} - x_i) \cdot \frac{t_i - t_{i-1}}{x_i - x_{i-1}} \tag{2}$$

We can then compute Δt_{i+1} by substituting x_{i+1} with τ. If the computed Δt_{i+1} is not positive, then we can probe as far in the future as desired.

The memory requirement consists of additional two registers, holding t_{i-1} and x_{i-1}, which are updated at each epoch.

Quadratic Prediction. This technique extends the linear prediction by accounting for the acceleration / deceleration in the measured value; i.e., to its second derivative. Intuitively, while the measured function is accelerating significantly, the node should correspond by accelerating the probing frequency, whereas while the acceleration is moderate, the prediction can be done using linear prediction.

Computing the estimated value at different points is based on quadratic extrapolation. Given three probe points (t_{i-2}, x_{i-2}), (t_{i-1}, x_{i-1}) and (t_i, x_i), computing the value x_{i+1} in time t_{i+1} can be computed using the Lagrange Equation (LE) for three points:

$$\begin{aligned} LE = x_{i-2} \cdot \frac{(t_{i+1} - t_{i-1})(t_{i+1} - t_i)}{(t_{i-2} - t_{i-1})(t_{i-2} - t_i)} + x_{i-1} \cdot \frac{(t_{i+1} - t_{i-2})(t_{i+1} - t_i)}{(t_{i-1} - t_{i-2})(t_{i-1} - t_i)} \\ + x_i \cdot \frac{(t_{i+1} - t_{i-2})(t_{i+1} - t_{i-1})}{(t_i - t_{i-2})(t_i - t_{i-1})} \end{aligned} \tag{3}$$

We provide an improved progressive computation, by leveraging on previous computations: let $x'_i = \frac{x_i - x_{i-1}}{t_i - t_{i-1}}$, $t'_i = \frac{t_i - t_{i-1}}{2}$ be the values as computed for linear prediction, x'_{i-1} and t'_{i-1} be these values as computed in the previous epoch. In

our *progressive estimation* (PE) the number of basic arithmetic operations is 13 compared to 23 in the Lagrange computation:

$$PE = x_i + x'_i(t_{i+1} - t_i) + \frac{(x'_i - x'_{i-1})(t_{i+1} - t_i)(t_{i+1} - t_{i-1})}{t'_i - t'_{i-1}} \tag{4}$$

It can be proved using basic arithmetic that LE as computed by the Lagrange Equation (Eq. 3) equals to PE as computed by the progressive estimation (Eq. 4). The detailed proof can be found in the extended paper.

Given the next desired value $x_{i+1} = \tau$, the time t_{i+1} can be computed by solving the quadratic equation. There are a few possibilities for the equation results, depending on the number of solutions that specify a time in the future. We take the smallest t_{i+1} such that $t_{i+1} > t_i$. If no such solution exists, we can probe again very far in the future, such that Δt_{i+1} is very large. Quadratic prediction requires two more non-volatile registers for x'_i and t'_i, bringing the total number of required registers to four.

Higher Dimension Prediction. It is possible to extend to higher prediction levels using the general Lagrange equation. However, the additional cost may not be justified by the possible improvement, which is expected to be diminishing.

3 Adaptive Probing

We define two types of Adaptive Probing: Approximate Probing (Section 3.1) and Threshold Probing (Section 3.2).

3.1 Approximate Probing (AP)

The basic function of sensors is to approximate their environment as they probe it in a periodic manner. Our goal is to improve this approximation, so that we probe as infrequently as possible with minimum approximation error.

The AP algorithm uses extrapolation for predicting future values and determining the right time for the next probe. It may utilize any of the prediction techniques, described in Section 2. We present algorithms with linear and quadratic predictions. The $Naive$ algorithm can be viewed as probing in equal *time* difference between probes, therefore the changes in the probed environment do not influence its execution. In contrast, the AP algorithm will aim to probe in equal *value* differences between probes, adapting to the changes in the environment. At each epoch the algorithm should decide *when* the next probe should take place.

For a given desired *value* difference c between the last probe and the next probe, the goal of the algorithm is to decide the *time* difference, Δt until next probe.

Recall that x_{i-1} is the probed value at time t_{i-1} and x_i is the value in current time t_i. By defining $c = x_{i+1} - x_i$ and using Eq. 2 for linear prediction, we get the next probing time:

$$t_{i+1} = t_i + c \cdot \frac{t_i - t_{i-1}}{x_i - x_{i-1}} \tag{5}$$

For quadratic prediction, substituting $c = x_{i+1} - x_i$ in Eq. 4 gives:

$$c = x'_i(t_{i+1} - t_i) + \frac{(x'_i - x'_{i-1})(t_{i+1} - t_i)(t_{i+1} - t_{i-1})}{t'_i - t'_{i-1}} \tag{6}$$

Calculating quadratic equation for both c and $-c$ (for increasing values) may result with four possible solutions. The selected solution is taken as discussed for the quadratic prediction of Section 2.

The Error Metric: Let E be the error, N be the number of probes, RV_i be the real value and AV_i be the value estimated by the algorithm at epoch i then the error is computed as: $E_{abs} = \frac{1}{N}\sum_{i=1}^{N}|RV_i - AV_i|$ or $E_{rel} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{RV_i - AV_i}{RV_i}\right|$. Higher moments of errors can be defined in a similar manner.

3.2 Threshold Probing (TP)

In many cases, the function of a sensor network is to alert when the measured value reaches a certain level. That would be the case, for instance, in fire detection. In such cases, we do not want to approximate an entire measured signal, but rather to alert when reaching at the desired threshold, with minimum cost and minimum latency error.

The algorithm for TP is based on computing Estimate Time of Arrival (ETA): the predicted time in which the threshold value is expected to be reached based on the stream history.

Let τ be the threshold value; we desire to predict the time t_{i+1} in which the measured value will become $x_{i+1} = \tau$. The TP algorithm based on linear prediction, and using Eq. 2, gives:

$$t_{i+1} = t_i + \frac{(\tau - x_i) \cdot (t_i - t_{i-1})}{x_i - x_{i-1}} \tag{7}$$

Eq. 7 might return time t_{i+1} in the past (i.e., $< t_i$), when the value is getting farther from the threshold. In that case the next probe can be determined using a predetermined bound, as discussed in Section 3.3.

The TP algorithm is based on quadratic prediction, using Eq. 4, gives:

$$x_{i+1} = x_i + x'_i \Delta t_{i \to i+1} + \frac{\left(x'_i - x'_{i-1}\right) \Delta t_{i \to i+1} \Delta t_{i-1 \to i+1}}{t'_i - t'_{i-1}} \tag{8}$$

As for AP, solving the quadratic equation might return two possible results, in order to be conservative and thus not lose important events, the algorithm should take a point in time t, such that t is the minimal solution that is larger than the current time t_i. If the equation does not have any result, we use the adjustment parameters as discussed in Section 3.3.

The Error Metric: The goal of Threshold Probing is to detect an event as soon as possible, therefore the error measure is the delay between the time in which real event occurs and the time in which the algorithm detects this event.

Let $Error_{TP} \geq 0$ be the error of the algorithm, T_a be the algorithmic detected time of event e and T_e the time of event Let T_e the time of an event e, in which the measured value reaches a predefined threshold, and let T_a be the detection time based on the TP algorithm. Then the latency error is defined as: $Error_{TP} = T_a - T_e$

Note that for any algorithm, but especially for the $Naive$ algorithm the value of the error in each run is influenced by the starting time of the probing (i.e. the synchronization) as it is defined by the time difference between the last probe and the threshold achievement. The average error in the $Naive$ case is half of the time between probes. For the TP algorithm, the starting point is less important as the algorithm adjusts itself to the value changes.

3.3 Realization

We consider the realization of the AP and TP algorithms. The basic algorithms, as described so far, only treat the anticipated case where a very small synopsis of history perfectly predicts the future, and they do not guarantee neither error nor cost. As a result, there may be quite reasonable adversarial scenarios in which the algorithm predicts poorly or costly. In order to overcome those problems, we utilize two methods: (1) using adjustment parameters for computing Δt; (2) incorporating a fall back methodology in which the algorithm monitors its own performance and transforms to a $Naive$ algorithm in case it performs poorly. Due to space limitation, we only describe briefly the method of adjustment parameters and leave the discussion on the fall back method to the full paper.

Let Δt_{i+1} be the estimated time difference so that $x_{i+1} = \tau$, when letting $t_{i+1} - t_i = \Delta t_{i+1}$, as computed by a prediction method. To address the imperfection of this prediction, we conservatively enforce the next probe to be earlier, based on two *adjustment parameters*: $\alpha \geq 0$, and $\beta \geq 0$:

$$\Delta t_a = \frac{\Delta t_{i+1}}{(1+\alpha)} - \beta \tag{9}$$

The parameter α is multiplicative and it would mostly affect the mean relative error; the parameter β is additive and it would mostly affect the mean absolute error. We also use two *adjustment parameters*: $\overline{\Delta}$ and $\underline{\Delta}$. The upper-adjustment parameter $\overline{\Delta}$ provides an upper bound on the time difference between two probes, and hence on the latency error. The lower-adjustment parameter $\underline{\Delta}$ provides a lower bound on the time difference, to avoid costly probing that is beyond the required granularity. The waiting time for the next probe is computed as a function of the adjusted prediction and the two adjustment parameters:

$$\Delta t = Min\left(\mathrm{Max}\left(\Delta t_a, \underline{\Delta}\right), \overline{\Delta}\right) \tag{10}$$

4 Adaptive Communication (AC)

One of the most common uses of communication in sensor networks is the transfer of probed values of a sensor node to other nodes (e.g., base station). The value

can be passed as raw data or aggregated along the communication path. In either case, a node transfers data to its parent node, which in turn transfers data onwards. The base assumption for our methodology is that there is a temporal correlation between subsequent values.

4.1 AC Protocol

The sender maintains a synopsis that represents the stream of values sent so far, and a prediction of the value that is to be sent next, based on the synopsis. Similarly, the receiver maintains a synopsis that represents the stream of values received so far, and a prediction of the value that is to be received next, based on the synopsis. The synopses and the corresponding predictions can be computed using the prediction techniques of Section 2.

If the next probe is as predicted, within a small allowed error, then the sender does not send it. If the receiver does not receive a value at a designated time, it assumes that it equals its predicted value. Thus we manage to avoid transmission and still transfer the appropriate information. This event is therefore called *implicit send*.

In cases in which the time of transmission is unknown, as is the case for probing with a variable rate, the receiver does not compute any prediction. Its synopsis is then only used for the purpose of interpolation for reporting values. The protocol is described in Table 1, with an allowed error $\varepsilon(s)$ per value transmission.

4.2 AC Implementation

In many installations, the base station needs all the probed information from each node in a multi-hop network of nodes (e.g., [9]). In such scenarios the *AC* protocol can be applied between each node and the base station. It has two advantages: (1) each message saving implies savings along all the routing to the root; and (2) the base station might be a powerful node with 'unlimited' memory, energy and computation power for issuing the *Receiver* protocol.

There are cases in which the user is interested only in aggregation of results or the aggregation is desired from an energy saving perspective (e.g., [8]). In such scenarios the *AC* protocol is applied between each node and its parent, and the *Receiver* saves k states for its k children. If this is too much for its constrained memory, it is used for the maximum number of children which can use the protocol, while other children will continue propagating all their values as before.

In sensor networks there might be extensive loss of communication due to collisions, synchronization issues, lack of resources and other interferences. In order to overcome these losses we propose two methods: (1) transmit redundant information in which the sensor sends the subsequent probe after each sent probe. In that case we would send more than the lossless *AC* but still less or equal to the $Naive$; (2) in applications that use acknowledgment, piggyback on this acknowledgment for AC, resulting with no losses.

ADAPTIVE COMMUNICATION PROTOCOL - SENDER

```
    Save Synopsis and prediction Pred(v_{i+1})
 1  while TRUE
 2        do
 3           Probe a value v_i    // Probe a new value
 4           if |v_i − Pred(v_i)| > ε
 5              then    //If this value is deviating
                        // Send the new value to the Receiver
 6                   Send v_i
 7                   Update Synopsis
 8           compute Pred(v_{i+1}) //Compute next prediction
 9           i++
10           sleep()            // Sleep until next probe
```

ADAPTIVE COMMUNICATION PROTOCOL - RECEIVER

```
    Save Synopsis
 1  while TRUE
 2        do
 3           if Receive New value v_i
 4              then    // If received new value
 5                   Update Synopsis
 6              else    // There was no new value
                        // Compute value using extrapolation
 7                   Compute v_i
 8           Report v_i  // Report the new value
 9           i++
10           sleep()  // Sleep until next probe
```

Table 1. Adaptive Communication (AC) protocol, the *Sender* sends its value only when it deviates by more than ε from its predicted value. The Receiver reports the next value either by using the received value or by extrapolating using previous values.

5 Experimental Results

In the experimental study, we demonstrate the efficiency of the algorithms proposed in this paper. For each type of algorithm (i.e., AP, TP and AC) we conducted the following experiments: (1) run the algorithm over real nodes traces, which were taken from the Great Duck Island (GDI) project [9]; (2) run the algorithms on Mica motes [12] using TinyOS [15]. Due to place limitations, we present here a subset of the experiments, the full experiments can be found in the extended paper.

5.1 Framework

Traces from GDI - The algorithms were implemented using an ad-hoc simulator written in Java. All our experiments were held with the adjustment parame-

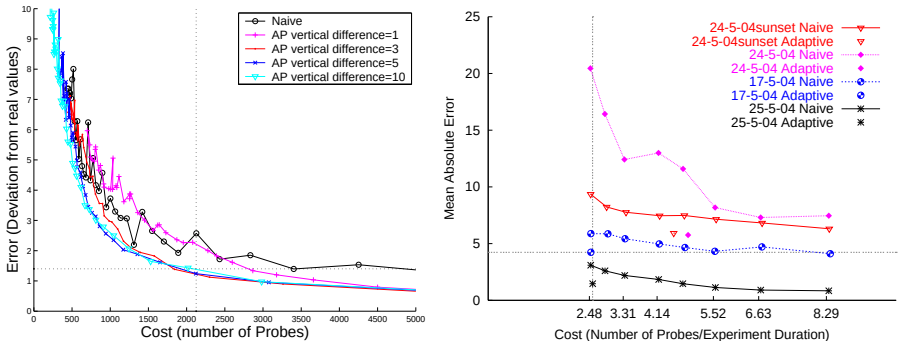
ters. We plugged the real motes traces taken from the GDI project [2] into the simulator. The traces include a few attributes from each mote, and consist of probes taken about once every 5 minutes; we chose the temperature measure. The *Naive* free variable was the time between probes, we simulated with different values and for each such value we found the error and the cost (number of probes) for drawing the plot. For the adaptive algorithm, the free variable was used in order to generate the lower and upper bounds, which were division and multiplicity of the *Naive* rate by a constant factor.

Mica Motes - We implemented the algorithms on Mica2 Motes [12] using the Nesc language and over the TinyOS [15] OS. Our test bed includes two motes in which one was attached to the base station in order to present the results, the second ran *Naive* algorithm and adaptive algorithm in parallel. We chose this setup in order to measure the same values using the same sensing calibration. In order to know the 'real function', the *Naive* algorithm served two purposes: 1) determining the 'real function', which means to probe very fast; 2) simulating the *Naive* algorithm. In order to do that, the sensor probed and sent in a fixed, very high frequency. For the simulation of the *Naive* algorithm we took only one of every k probes. Thus for every experiment we get several results of error vs. cost, depending on k. In our experiments we measured using the light sensing device. Most of the experiments measured at least one sunset or sunrise, in order to test the algorithms on a changing environment.

5.2 Approximate Probing (AP)

As evident from the results for the GDI case (Fig.1a), the *AP* is superior over the *Naive*, especially if a low error rate is desired or a higher cost is allowed; the reason for that is that the adaptive algorithm can use additional probes better. It can be seen that for the same cost we obtained up to 54% improvements in the error (the horizonal line in the figure, which compares between the *Naive* and *AP* with desired value difference between probes of 5). For the same error we obtained up to 43% reduction in probes (the vertical line in the figure, which compares between the *Naive* and *AP* with desired value difference between probes of 5). We tried in various other possible metrics and got similar results, details can be seen in the extended paper.

For the real Motes experiments (Fig.1b), we ran the algorithms a few times. As said earlier, for each experiment we got a few *Naive* results that are represented as a plot and one *AP* result which is presented as a point with the same shape and color. We can see that for the same cost we get up to 48% improvement in the error (vertical line in the figure, which compares between the *Naive* and *AP* on the 25th of May). For the same error we get up to 51% improvements in cost (horizontal line in the figure, which compares between the *Naive* and *AP* on the 17th of May).



(a) (b)

Fig. 1. Error vs. Cost. (a) AP algorithm simulated over the measured data of GDI of node 101. The adjustment parameters $\underline{\Delta}$ and $\overline{\Delta}$ are 1/3 and 3 times of the naive rate. Plots for vertical difference of $1, 3, 5, 10^\circ$ Celsius (b) AP algorithm run over Mica Motes. For each day there are a few $Naive$ experiments and one adaptive experiment

5.3 Threshold Probing (TP)

For node 101 of the GDI, around time $2,030,000$ there is a single area in which the value climbs over 37°, it stays over this value for 3900 seconds (reaching even to 39°). We defined 37° as the Threshold. The results for the comparisons are shown in Fig. 2. It is evident that TP with first derivative is quite superior to the $Naive$. Positive values of α and β improve the results, which can reach in some cases up to 90-95% improvements for the same cost.

It is notable that the TP algorithm is much more stable in its results, whereas the $Naive$ results are somewhat arbitrary; this is due to the fact that the $Naive$ is much more vulnerable to the staring time relative to the threshold crossing time. The TP in contrast is adapting to the measures and thus "prepares itself" to the threshold achievement in advanced.

5.4 Adaptive Communication (AC)

We ran both the $Naive$ and the AC algorithms with equal time between probes. The $Naive$ is always sending while the AC is sending on a needed basis only.

For the GDI project (Fig. 3a), the free variable was the time between probes; for each such value, we ran a simulation and obtained the error and the cost (number of probes) for the plot. We can see that for the same cost we get up to 75% error improvements compared to the $Naive$ (see vertical line, which compares between the $Naive$ and AC with $\varepsilon = 0.5$). For the same error we get up to 52% cost saving compared to the $Naive$ (see horizonal line, which compares between the $Naive$ and AC with $\varepsilon = 1$).

For the Mica Motes experiments (Fig. 3b), we can see that for the same cost we get up to 70% error improvements compared to the $Naive$ (see vertical line, which compares between the $Naive$ and AC on the 29th of May) . For the same

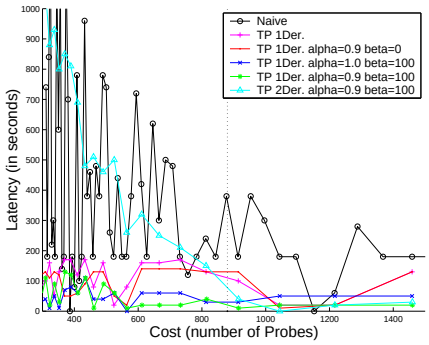
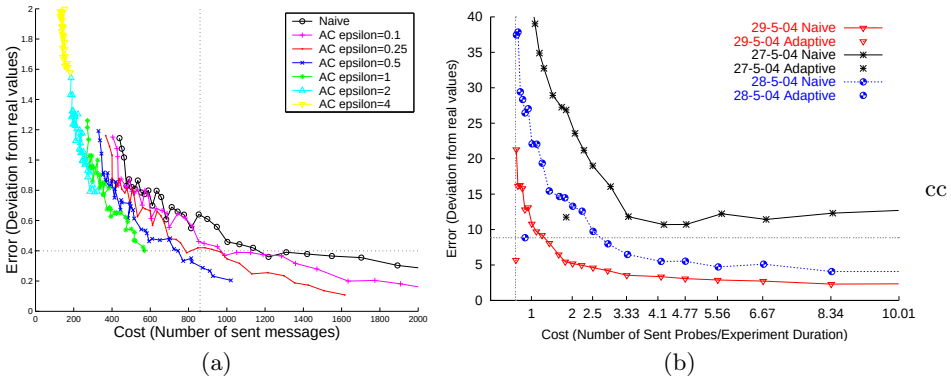


Fig. 2. Latency vs. Cost. TP algorithm simulated over node 101 of GDI. The Threshold was set to 37° Celsius. The adjustment parameters $\underline{\Delta}$ and $\overline{\Delta}$ are 1/3 and 3 times of the naive rate. α and β are precaution parameters.

error we get up to 70% reduction in the number of messages needed for the same error compared to the $Naive$ (see horizontal line, which compares between the $Naive$ and AC on the 28th of May).



(a) (b)

Fig. 3. Error as function of the Cost. AC algorithm (a) simulated over the measured data of GDI over node 101. The plots describes various ε values. (b)AC algorithm run over Mica Motes. For each day there are a few $Naive$ experiments and one adaptive experiment

6 Related Work

There is rich literature on forecasting and adaptive techniques. The typical questions considered in these works is typically computing the expected value of an

observed sequence at a future time. We are mostly interested in the inverse question of when is the expected time to get to a certain value. It is possible to find an approximation to this inverse function using doubling and binary search but this will be rather costly in our context. Note that our adaptive methods should be computed every epoch in runtime manner towards the next epoch; therefore it should be done very cheaply and efficiently. We highlight a few adaptive techniques that were considered in various scenarios, and relevant works on sensor networks.

Adaptive Filters theory is a well researched domain with huge literature (e.g., [4]). The idea is to apply filters on a signal for filtering, smoothing and prediction. Weiner filters are used over stationary data. Kalman filters are used for ongoing updated data where the computation is done incrementally. The idea of predicting using Adaptive Filters is to find the next expected value based on n previous values, assuming the value is changing linearly.

Box-Jenkins modeling [1] is a mathematical modeling of time series used for forecasting. It involves identifying an appropriate ARIMA process, fitting it to the data, and then using the fitted model for forecasting. It has three iterative processes: Model selection, Parameter estimation and Model checking. Recent improvements in the process [10] added two more steps of Data preparation and Forecasting. All those methods are too computation and memory intensive for our limited devices and online computation involved in our goal.

In [13] the authors propose an automatic environmental model construction, but their sensors ("PDA-like devices") are order of magnitude more powerful than the sensors considered in our work (e.g., Mica Motes). They are trying to construct a global model for future forecast whereas we are forming a local model for forecasting the very near future.

In [11] the authors propose to adapt the sampling rate based on an a priory model such as sinusoid. In contrast, our model is changing over time and adapts to the changing measured stream. As a result, we do not assume an a priory model which makes our solution more general and adequate for larger set of applications. As far as we know, except for this work, existing probing algorithms use constant rate probing frequency. In [3] the authors propose to create a temporal model by a base station. Each sensor in this case sends a message only if the value deviates from this model. In our AC algorithm, the sender maintains its own model. Thus, the base station does not need to issue and send a model which is a costly communication step by its own.

The notion of a single node that uses only its own resources for event detection was considered in [7, 5]. However, [7] define an event as a value over some predefined threshold but continued to probe in constant rate. Paper [5] propose to put a special hardware in designated nodes, which has cost and need a priory knowledge of the topology.

The idea of balancing the tradeoff between accuracy and lifetime was introduced in [6]. They use this tradeoff in building a synopsis that represents the data for queries of aggregation purposes, while our approach is to use this tradeoff for reducing the number of probing and basic communication. In [14]

the authors propose the idea of sending messages on a changing basis; however, this method is proposed in the context of aggregations only, and in case it deviates from previous values. In contrast, *AC* deals with any communication and sends in case the value deviates from previous prediction and not just from the previous value.

7 Conclusions

In this paper we present the first study of adaptive probing and adaptive communication for improved utilization of sensor networks. We present basic algorithms that effectively realize the adaptive methods. The potential impact of our approach is demonstrated using experimentations. There are many open questions with regard to adaptive methods in sensor networks. Below we mention a few.

It would be interesting to consider various objective functions for approximation and their corresponding error metrics. We described some basic mathematical models that guide the adaptive algorithms. It would be interesting to further develop mathematical models for measured functions that provide a good approximation for real data and at the same time are manageable with low overhead, given the limited computing and communication resources available in sensor networks.

Prediction of future patterns based on past information is the heart of the proposed adaptive methods. There may be significant promise in incorporating methods from other disciplines that address related questions. We expect that due to the computational constraints in sensor networks, many known techniques would require considerable adaptation in order to be useful here. Some of them might be derived from fields like Adaptive Filters and Box-Jenkins modeling.

This work focuses on adaptive methods for single sensors, or for pairs of sensors. Applying more advanced adaptive methods for the entire network may be a rich area for research with a potentially significant impact.

Acknowledgments: We thank Christos Faloutsos for helpful pointers to relevant literature in the forecasting domain.

References

1. G. Edward, P. Box, and G. M. Jenkins. *Time Series Analysis: Forecasting and Control.* Prentice Hall PTR, 1994.
2. Habitat monitoring on great duck island. http://www.greatduckisland.net/.
3. S. Goel and T. Imielinski. Prediction-based monitoring in sensor networks: Taking lessons from mpeg. In *ACM Computer Communication, Vol. 31, No. 5*, 2001.
4. S. Haykin. *Adaptive Filter Theory.* Prentice Hall, 3rd edition, 1996.
5. J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, March 2003.
6. K. Kalpakis, V. Puttagunta, and P. Namjoshi. Accuracy vs. lifetime: Linear sketches for approximate aggregate range queries in sensor networks. available as umbc cs tr-04-04, February 11, 2004.

7. J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao. Distributed group management for track initiation and maintenance in target localization. In *IPSN*, 2003.
8. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *THE MAGAZINE OF USENIX and SAGE April 2003*, page 8, 2003.
9. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM WSNA*, Atlanta, GA, Sept. 2002.
10. S. Makridakis, S. Wheelwright, and R. J. Hyndman. *Forecasting : Methods and Applications*. John Wiley & Sons., 1998.
11. A. D. Marbini and L. E. Sacks. Adaptive sampling mechanisms in sensor networks. In *London Communications Symposium*, 2003.
12. Berkley mica motes. $http://www.xbow.com/Products/Wireless\_Sensor\_Networks.htm$.
13. S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. In *VLDB*, 2003.
14. M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Tina: a scheme for temporal coherency-aware in-network aggregation. In *ACM workshop on Data engineering for wireless and mobile access*, pages 69–76. ACM Press, 2003.
15. Tinyos operating system. $http://webs.cs.berkeley.edu/tos/$.
16. B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. In *16th International Conference on Data Engineering*, page 13. IEEE Computer Society, 2000.