3-2006

CAPS: Energy-Efficient Processing of Continuous Aggregate Queries in Sensor Networks

Wen HU
University of New South Wales

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Rajiv SHOREY
IBM India Research Laboratory

DOI: https://doi.org/10.1109/PERCOM.2006.14

Citation

HU, Wen; MISRA, Archan; and SHOREY, Rajiv. CAPS: Energy-Efficient Processing of Continuous Aggregate Queries in Sensor Networks. (2006). *2006 IEEE International Conference on Pervasive Computing and Communications 4th PerCom: March 13-17, Pisa, Italy: Proceedings*. 190-199. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/686

CAPS: Energy-Efficient Processing of Continuous Aggregate Queries in Sensor Networks

Wen Hu
University of New South Wales
National ICT Australia Limited
wenh@cse.unsw.edu.au

Archan Misra
IBM T J Watson Research Center,
Next-Gen Web Infrastructure Dept.
archan@us.ibm.com

Rajeev Shorey
IBM India Research Laboratory
rajeev.shorey@gm.com

Abstract

*In this paper, we design and evaluate an energy efficient data retrieval architecture for continuous aggregate queries in wireless sensor networks. We show how the modification of precision in one sensor affects the sample-reporting frequency of other sensors, and how the precisions of a group of sensors may be collectively modified to achieve the target **Quality of Information (QoI)** with higher energy-efficiency. The proposed Collective Adaptive Precision Setting (CAPS) architecture is then extended to exploit the observed temporal correlation among successive sensor samples for even greater energy efficiency. Detailed simulations with synthetic and real data traces demonstrate how the combination of weak consistency semantics and temporal correlation can dramatically lower the energy consumption in practical sensor environments.*

1 Introduction

Energy-efficient operation of the sensor infrastructure is critical in many pervasive or context-sensitive environments where such battery-powered sensors provide the required knowledge of the environmental state. A promising approach for drastically reducing the communication and/or sensing overhead centers on exploiting an application's acceptable tolerance of imprecise and inaccurate data. Such tolerance is expressed in terms of a Quality of Information (***QoI***) metric, and is especially useful for applications issuing *aggregation queries* (such as min, max, *sum*, *mean*, etc.) over a set of sensors. The QoI bounds may be either deterministic(e.g., [1]) or statistical (e.g., [2,3]).

In this paper, we present a novel architecture, called Collective Adaptive Precision Setting (**CAPS**), for energy-efficient support of *continuous aggregate* queries in a sensor network. In the CAPS framework, the sink is able to *always* ensure, over the entire query lifetime, that the aggregate value computed by it (and reported to the application) does not diverge from the true reading by more than a specified "tolerance". The key is to have the sink communicate a *precision range* or interval to an individual sensor, with the idea that a sensor need not report its samples back to the sink as long as they fall within this specified range. This bounded-divergence approach transforms the conventional *sink-initiated, polling-based* model of data collection to a more energy-efficient *source-initiated, event-driven* framework. This idea of using a precision range was first introduced in [1], and subsequently extended in [4,5]. However, all prior work addresses only the *instantaneous* query model, where applications issue synchronous, snapshot queries.

Our focus is on the *continuous queries*, which is a better fit for many long-running pervasive environment monitoring applications. This introduces several new interesting features and challenges. First, the application's QoI requirement is usually be expressed as a tolerance on the aggregate statistic (e.g., "I want the minimum temperature within a range of ± 5 units"), which a sink must decompose into individual precision intervals for each sensor. Second, the sink must continually ensure conformance to the aggregate QoI bound, even as the values reported by individual sensors, or even the set of sensors itself, changes over the lifetime of the query. Finally, in many operating environments, there is a significant temporal correlation across the successive samples reported by an individual sensor (a fact demonstrated and exploited in [2] for snapshot queries). For example, an outdoor thermal sensor *usually* reports increasing readings during the morning, and gradually decreasing readings as the afternoon progresses. The architecture should exploit such *expected* or *predictable* variations to further reduce the communication overhead.

To make our examples more concrete and practically relevant, consider one specific distributed energy resource applications for *smart* indoor environments, that adaptively adjust the operation of power-hungry equipment, such as air conditioners. Assume that thermal sensors in a room generate temperature samples once every $T = 60$ seconds. A smart controller that automatically activates or shuts off air-conditioners would want to monitor the Maximum ($\max$) and Minimum ($\min$) temperature in a room/building, every T secs, with a specified precision range $R = 1^{\circ}$, i.e., the application is satisfied as long as the sink reports within ± 1 Celsius of the true MIN and MAX values computed by the sensors.

There are three key contributions of our work as follows:

First, we first demonstrate how R, the composite precision bound specified by an application, may be decomposed into individual precision ranges (for each individual sensor) for certain representative aggregation functions. Unlike prior work, the precision range for a sensor cannot be determined in isolation, since an adjustment of the precision range for sensor A may necessitate a change in the precision setting for another sensor B (to maintain the QoI), which indirectly affects the reporting frequency of both nodes.

Second, we extend the range adjustment technique to accommodate sensor heterogeneity, in terms of either variable sensor-to-sink communication overhead, or individual resource constraints. This allows more-constrained sensors to have larger precision ranges (permit greater divergence) and lower reporting frequency.

Third, we additional demonstrate how predictive algorithms may be used in tandem at the sink and source nodes to exploit temporal correlation among successive samples, and significantly reduce the reporting overhead. To our knowledge, this combination of temporal correlation with collective weak consistency metrics has not been explored before.

The rest of this paper is organized as follows. Section 2 discusses prior work in this area. In Section 3, we introduce the notion of collective range adjustment for continuous aggregate queries, and then express and solve the energy efficiency objective as a non-linear optimization problem. Section 4 describes the CAPS architecture (without temporal prediction) and the resulting adaptive algorithms. Section 4.2 then describes how simple linear prediction algorithms can provide even better performance. Section 5 evaluates and studies CAPS via both extensive simulations and *empirical sensor data* collected from an operational Motes-based sensor testbed. Finally, Section 6 concludes the paper.

2 Related work

Recent work on data management in sensor networks has focused primarily on techniques for either in-network aggregation of sensor data or exploitation of the spatio-temporal correlation among sensor data samples. For robust aggregation of data in sensor networks, [6] presented architecture called TAG, where a routing tree centered at the sink is developed during the query dissemination phase. [7] extended this model to provide fault-tolerance using a directed acyclic graph that effectively routes the individual or aggregate data samples across multiple paths toward the sink. However, these protocols do not exploit the fact that applications tolerate slightly inaccurate data.

Most relevant to our work is the work in [1], which studied the optimal trade-off between sink-initiated fetching from individual data sources and the source-initiated refresh of data (based on precision bounds). The resulting algorithm considered the costs of data retrieval and query dissemination to each candidate data source, and then presented a technique by which the central sink would decide on an efficient combination of these approaches (across multiple nodes) to satisfy the QoI of an “aggregate” query. However, since [1] focuses on snapshot queries, there is *no attempt to adjust the precision ranges of individual nodes collectively to ensure conformance to the QoI.* In [1], if the QoI of a new snapshot query could not be satisfied by the existing precision ranges, the sink would simply poll an appropriate subset of nodes. The CAPS approach of *collective precision setting* is, however, the first to ensure *continual* adherence to the QoI bounds The idea of adjusting the precision range adaptively based on the observed ratio of sink-initiated fetching and source-initiated refreshes was first reported for generic Web-based data sources in [4]. More recently, [5] applied the adaptive precision approach specifically to the sensor network setting, considering in detail the various operational modes of individual sensors and their impact on the overall energy consumption. However, both [4] and [5] focus on the adaptation of the precision range of an individual sensor, and do not consider the case of aggregate, long-lived queries.

For exploiting the potential temporal correlation among successive data samples of a single sensor, we shall use a generic closed control loop architecture that predicts the expected value of future data samples. The use of closed-loop control for tracking time-varying parameters of a sensor network has been reported in some earlier work. For example, [3] uses a predictive controller to adjust the number of activated nodes in a sensor network with a time-varying node population. Recently, [8] proposes a predictive storage architecture for sensor networks. However, it neither introduces a detailed predictive algorithm, nor does it evaluate the architecture.

Table 1. Mathematical Notations

Symbol	Definition
s_i	Sensor i
S	A set of sensors $\{1, 2, ... n\}$
$e_{i,j}$	An edge between sensor s_i and s_j
E	A set of edges
R	Application QoI requirement
T	Monitoring time serial $\{1, 2, ... m\}$
t_j	Time j
v_i^j	The reading of sensor i at time j
r_i^j	The interval of sensor i at time j
H_i^j	Upper bound on s_i's reading at t_j
L_i^j	Lower bound on s_i's reading at t_j
Hop_i^j	Hop-count from s_i to the sink at t_j
Cu_i^j	The update cost of sensor i at time j.

3 The Collective Range Adjustment Problem and Solution

In this section, we first present the generic problem of collective range adjustment for aggregate queries. We then express how energy-efficient processing of such an aggregate query may be formulated as an optimization problem, and subsequently present a necessary and sufficient condition for achieving optimality. This insight will allow us to develop the functional blocks of the CAPS architecture in Section 4.

Let us first introduce the necessary mathematical notation, all of which are collectively listed in Table 1. We assume that the application issues an aggregation query with its QoI specified by a precision range R–this implies that the aggregate value computed at the sink at any instant should be accurate to within $\pm R$. In other words, the application is assured that, if the value reported by the sink is V, the *true value* of the aggregate state (which would require each sensor to report each sample to the sink) lies within $[V - R, V + R]$. The sensor network itself is modeled as undirected graph $G =< S, E >$, where S is a set of sensors, and E is a set of edges. There is an edge ($e_{i,j}$) between sensors s_i and s_j if their distance is less than the transmission range Γ.

Further, we assume a common sensing period T, with each sensor generating a new data sample every T time units. Accordingly, we index time to be integer-valued, with $t = j$ referring to the j^{th} sample. The CAPS architecture assumes that the sink communicates (mostly implicitly) the *precision interval* (L_i^j, H_i^j) to sensor s_i, indicating that the sensor needs to report its sample value v_i^j at time j only if it lies outside this specified range. Accordingly, at all instants, each sensor is associated with a precision interval defined by the relation $2 * r_i^j = H_i^j - L_i^j$. In practice (we initially consider the simple case without temporal prediction), the range (L_i^j, H_i^j) is not communicated by the sink at each sampling period, but is instead set by the sink at the instants when a sensor reports a new sample to the sink (i.e., when its value lies outside the last specified interval). Accordingly, when a sensor reports a sampled value to the sink at time j, it receives either a new value r_i^j (for implicitly computing L_i^j and H_i^j based on its present reported value v_i^j) or an explicit specification of $\{L_i^j, H_i^j\}$. This implies that s_i guarantees that $v_i^k \geq L_i^j$ and $v_i^k \leq H_i^j$ when $k > j$; otherwise, it will send an updated value to the sink. When the new bound is implicitly communicated through the interval r_i^j, a sensor calculates L_i^j and H_i^j as follows:

$$L_i^j = v_i^j - r_i^j \quad (1)$$

$$H_i^j = v_i^j + r_i^j \quad (2)$$

To accommodate the heterogeneity in the sensor nodes or the network topology, let Cu_i^j denote the update cost (communication overhead) incurred if indeed s_i has to report its sample value at time j to the sink. Similarly, Cq_i^j denotes the cost (to the sink) of having to issue a polling request to sensor s_i at time j; this request modifies the range r_i^j, and in turn, may cause s_i to report its data sample back to the sink (since it no longer lies in the new interval (L_i^j, H_i^j).

3.1 Assumptions on Sensor Network Operation

For our optimization model, we make the following simplifying assumptions on the behavior of the sensor network. First, sensors (except for the sink) use a fixed transmission power.

Second, the sensing frequency is high enough to capture any temporal correlation in the usually slowly-varying underlying environmental state of interest [9].

Third, we ignore the cost of polling from the sink to an individual sensor. In other words, we assume that $Cq_i^j = 0$ $\forall i, j$. This simplifying assumption is acceptable in real wireless operating environments where the sink and the sensors can use different transmission powers. While the sink can use the maximum permitted transmission power (e.g., level $0X99$ in mica motes [10]) since it is typically not energy-constrained, the sensors use the minimum transmission power level (e.g. the level $0X00$ in mica motes) to conserve energy. Accordingly, while all downlink traffic (from sink to sensors) is one-hop,the sink itself is potentially multiple hops away from the sensors (for uplink traffic). In addition, in environments where a broadcast-based diffusion algorithm [11] is used to set-up and maintain the network routing topology, new r_i^j values may be piggybacked on the existing network control packets with minimal additional overhead.

3.2 The Optimization Formulation for Candidate Aggregate Queries

Our primary objective is to reduce the total communication energy consumption in the sensor network, while ensuring conformance to the application's QoI bound. Accordingly, we can express our objective **P1** for the case of sum and $mean$ queries as:

$$minimize \quad \sum_{j \in T} \sum_{i \in S} Cu_i^j * I_{i,j}, \tag{3}$$

where $I_{i,j}$ is an indicator function taking on the value 1 only if the sensor s_i actually reports it value to the sink at time j. Note that this minimization implicitly assumes that $Cq_i^j = 0$, i.e., we ignore the transmission cost on the downlink.

The update cost Cu_i^j itself is a function of the multi-hop transmission cost from sensor s_i to the sink, and, in a network with identical transmission power, should be a linear function of Hop_i^j, the length of the uplink path from s_i to the sink[1].(We maintain the time index j in the hop count, even if the sensor network is itself static, to accommodate the possibility that the actual uplink path may be dynamic.) Now, since objective P1 involves an infinite summation over time, we simplify the formulation to replace the time axis by the *average reporting rate* of a sensor. Accordingly, our optimization problem can be rewritten as **P2**:

$$minimize \quad \sum_{i \in S} \widehat{Cu}_i^j(r_i, H_i), \tag{4}$$

where $\widehat{Cu}_i^j(r_i^j, Hop_i^j)$ denotes the expected average reporting cost (until the r_is are again modified) and explicitly indicates its dependence on both the path length and the specified precision interval. At any point, $\widehat{Cu_i}$ and r_i should intuitively be inversely related: at any instant of time j, if the interval r_i^j is wider, the probability of having the sampled value of the sensor lie outside the permitted range is smaller. We will discuss the precise relationship between $\widehat{Cu_i}$ and r_i for a simple scenario in Section 3.3.

Of course, the minimization problem P2 is subject to additional constraints on the r_is–these constraints ensure that the resulting computed statistic always satisfies the application's QoI requirement (R). This is precisely the heart of the collective adaptation problem. We now define the additional constraints for our four representative aggregate operators, min, max, $mean$ and sum.

For sum and $mean$ aggregation functions, the constraints can be expressed as Equation (5) and (6) respectively, with $count(S)$ indicating the number of sensors in the reporting set.

$$\sum_{i \in S} r_i^j \leq 2R, \forall t_j \tag{5}$$

$$\frac{\sum_{i \in S} r_i^j}{count(S)} \leq 2R, \forall t_j \tag{6}$$

For the case of min and max aggregation queries, the objective function cannot be expressed in the form of Equation 4, since the update rate also depends on the *evolution of the sensor values*. To begin with, the constraints for min and max aggregation functions are slightly more involved and are expressed via Equations (7) and (8) respectively. The proofs of these constraints, and the derivation of similar constraints for other commonplace aggregation functions is available in [1].

$$\min_{i \in S}(H_i^j) - \min_{i \in S}(L_i^j) \leq 2R, \forall t_j \tag{7}$$

$$\max_{i \in S}(H_i^j) - \max_{i \in S}(L_i^j) \leq 2R, \forall t_j \tag{8}$$

To see why Equation 4 does not hold in these cases, consider the case of a min query, when a sensor s_i that reports a new minimum value v_i. Since this is reported back to the application as a new minimum, the sink now guarantees that the true minimum will subsequently lie between $[v_i - R, v_i + R]$. Since sensor s_i must have its $H_i = v_i + R$, to ensure that Equation 7 is satisfied, the sink can modify the reporting range for other sensors to $[v_i - R, \infty]$, even if their hop count has not changed. Accordingly, unlike sum or $mean$ queries, *the range for a sensor is not independent of the actual sensor values*. Consequently, the objective function for min and max queries may be expressed (at each instant a new value is generated) in the form **P3**:

$$minimize \quad \max\ \widehat{Cu}_i(L_i, H_i), i \in S, \tag{9}$$

subject to the constraints of Equation 7 and 8 respectively.

3.3 Optimization Under Random Walk Model

To solve the optimization problem P2, expressed by Equation 4, we need to define the explicit relationship between $\widehat{C_u}$ and r_i. In general, this will clearly depend on the actual evolution of the successive data samples at the sensor. *In the absence of any knowledge of temporal correlation among a sensor's samples*, we can model the variation of the sensor's sampled values as a random walk [12], i.e., as the evolution of a time series that adds uncorrelated zero-mean noise to the previous sample, centered at the last reported sample v_i^j. For such a model, it is well known that the *expected hitting time to a boundary* r_i, i.e., the time till the sampled value first deviates from the mean by more

[1]The generic formulation of Cu_i^j can reflect other forms of sensor heterogeneity, besides topological distance to the sink. For example, if we want to reduce the update burden on sensor nodes nearing battery exhaustion, we can make Cu_i^j an inverse function of the residual battery energy of the node. We do not explore such alternatives further in this paper.

than $\pm r_i$ is $\propto \frac{1}{r_i^2}$. We thus have $\widehat{Cu_i} \approx \frac{Hop_i}{(r_i)^2}$ (except for a proportionality constant), resulting in the following simplification of the optimization function P2:

$$minimize \quad \sum_{i \in S} \frac{Hop_i}{(r_i)^2}. \tag{10}$$

For the case of the sum aggregation query, Equation 10 is to be minimized subject to the additional constraints:

$$\sum_{i \in S} r_i \leq R, \tag{11}$$

$$r_i > 0, i \in S. \tag{12}$$

Lemma 1 *For the sum and mean aggregation queries, the objective function (10) is minimized when:*

$$r_1 : ... : r_n = \frac{Hop_1}{(r_1)^2} : ... : \frac{Hop_n}{(r_n)^2}. \tag{13}$$

(For the $mean$ aggregation query, the constraint of Equation 11 is replaced by a nearly identical constraint, with only a different scaling factor $count(S)$).

For compactness, the proof of this property is provided in the Appendix. Based on Equation 13, we can easily compute the optimal allocation of r_i among the set of constituent sensors, given an application's QoI bound R.

For aggregation query min, the sink needs to monitor the bounds of those sensors (set η) whose reading are minimum (V_{min}) only. To conform to the application QoI requirement R, the sink must set the bounds of the sensors in set η with the necessary stringency, while the other sensors can have looser bounds.

Lemma 2 *For the min aggregation query, the objective function (9) is solved by setting:*

$$H_i = \begin{cases} V_{min} + R & \textit{if } i \in \eta \\ \infty & \textit{otherwise} \end{cases}$$

and,

$$L_i = V_{min} - R, \forall i \tag{14}$$

Aggregation query max is symmetric to min.

4 Architecture Description

We describe base CAPS architecture, as well as the enhanced one that incorporates temporal prediction, and the related algorithms in this section.

4.1 Basic Architecture Overview

Figure 1 shows the architecture of CAPS (ignoring, for now, the role of temporal prediction). The CAPS middleware takes the application's QoI requirement (R) as external input. Based on these inputs, the Bounds Setting Component (BSC) computes and informs each individual sensor of its precision range. At time j, sensor i transfers a new value v_i^j to the sink when $v_i^j > H_i^{j-1}$ or $v_i^j < L_i^{j-1}$. These updates are processed in the Aggregate Value Evaluator (AVE). The AVE simply returns the new aggregation result to the application if it continues to satisfy application's QoI requirement; otherwise, it informs the BSC of the most recent updated sensor values. The BSC then calculates and distribute a new set of bounds to the sensors, using the underlying sensor routing infrastructure.

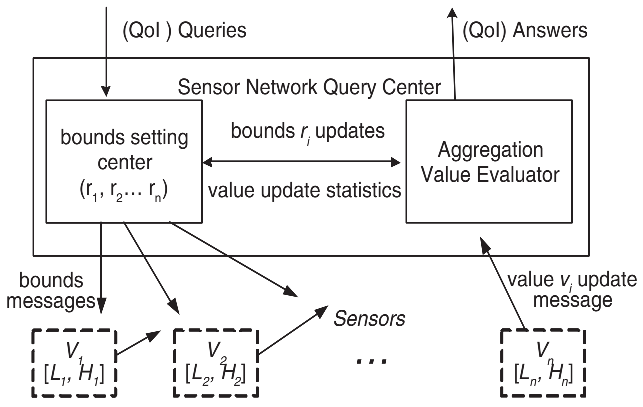

Figure 1. The basic architecture of CAPS

4.1.1 Aggregation Functions SUM and AVG

Based on the optimum conditional equation (13), we design our CAPS algorithm for aggregation functions SUM as Algorithm 1. Initially, the bounds are set statically (all the sensors have the same bounds R). Once the initial network topology has been learned via the first round sensing value updates, function $set\_adaptive\_bounds$ assigns different bounds for different hop-counts, which are then broadcast by the sink. Sensors reporting back to CAPS with a data sample also indicate their current bound. CAPS continually normalizes the bounds that it broadcasts at subsequent time instants to adapt to topology changes without knowing the exact network topology. A topology change (such as a new sensor coming on line, or a sensor moving closer or farther from the sink) is implicitly detected whenever the cumulative bounds $\sum_{i \in\ set} r_i$ reported back deviates from the application's QoI requirement. If the network topology changes, function $reset\_bounds$ re-normalizes

Algorithm 1 Algorithm of CAPS (aggregation function SUM)

```
procedure $CALCULATE\_BOUNDS(state, R)$
    if $state = first\_round$ then    ▷ network topology is unknown
        $set\_static\_bounds(R)$;
        return;
    else if $state = second\_round$ then
        $set\_adaptive\_bounds(R)$;
        return;
    end if
    if network topology has changed then
        $reset\_bounds(sum\_of\_bounds/R)$;
    end if
end procedure

procedure $set\_adaptive\_bounds(R)$
    $total\_bounds \leftarrow max\_hops * R$;
    for $i \leftarrow 1, max\_hops$ do
        $ratio_i \leftarrow pow(i, 0.33333)$;    ▷ normalized bound ratio of each hop
        $total\_ratio \leftarrow total\_ratio + ratio_i$;
    end for
    for $i \leftarrow 1, max\_hops$ do
        $bound_i \leftarrow total\_bounds * ratio_i/total\_ratio$;
    end for
end procedure

procedure $reset\_bounds(ratio)$
    for $i \leftarrow 1, max\_hops$ do
        $bound_i \leftarrow bound_i/ratio$;
    end for
end procedure
```

the sum_of_bounds, and thus ensures continued adherence to the QoI (without knowing how many sensors are located at various hop counts from the sink). The algorithm for aggregation function AVG is similar to SUM except that the result aggregation bound needs to be divided by the number of the sensors.

4.1.2 Aggregation Functions MIN and MAX

Based on Property 2, we design our CAPS algorithm for aggregation function min as Algorithm 2. Aggregation function MAX is symmetric to MIN.

4.2 Architecture with Temporal Prediction

The basic architecture described in Section 4.1 fails to exploit the *predictable* temporal variation among successive sensor node samples. To illustrate this, consider a sum aggregation query issued on two sensors s_1 and s_2 with the QoI requirement $R = 5$. Now, let the sensors *deterministically* evolve such that s_1's value decreases by 2 and s_2's value decreases by 1 at each sample. Suppose, at time $t = 0$, both sensors report identical values of $v_i^0 = 100,\ i = 1, 2$. Moreover, let Hop_1=8 and $Hop_2 = 1$, so that the optimum ranges computed by the random-walk model (Equation 13) are given by $r_1 = 6.7$ and $r_2 = 3.3$. It is clear that the temporal evaluation of the two sensors evolves as $\{\ v_1^1 = 102, v_2^1 = 99\}$, $\{v_1^2 = 104, v_2^2 = 98\}$, $\{v_1^3 = 106, v_2^3 = 97\}$, ... In this case, it is easy to see that s_1 generates a proactive update at time $j = 5$ (as $108 > 106.7$), while s_2 generates an update at $j = 4$. Thus, both s_1 and s_2 generate updates at more or less similar rates, although their update costs Cu_i are vastly different.

The problem really lies in the inability of the basic bound-adjustment algorithm to *factor in the different temporal rates or patterns exhibited by the samples of different sensors*. If the sink *could have predicted, say from past behavior, this deterministic evolution*, then it can proactively set (without any communication with the sensor) the range on each sensor as: $\{L_1^1 = 98.7, H_1^1 = 105.3, L_2^1 = 97.33, H_2^1 = 100.33\}$, $\{L_1^2 = 100.7, H_1^2 = 107.3, L_2^2 = 96.33, H_2^2 = 99.33\}$, $\{L_1^3 = 102.7, H_1^3 = 109.3, L_2^3 = 95.33, H_2^3 = 98.33\}$, ..., thus bounding the query response to $\{196, 206\}$ at time $j = 1$, $\{197, 207\}$ at $j = 2$, and $\{198, 208\}$ at $j = 3$.

While no sensor will evolve completely deterministically, smart temporal prediction algorithms at the sink should thus be able to significantly lower the need for proactive updates, *by dynamically adjusting the expected range of values* (L_i^j, H_i^j) *for each sensor*. For our simplified operating model, where $Cq_i = 0$, we can simply assume that the sink transmits new precision ranges at each instant. However, in practice, Cq_i will not be exactly zero; additionally, communicating new precision intervals at each sampling period may lead to unacceptable latency or network congestion on the downlink. We can avoid this overhead by noting that the sensor can independently *infer* the exact

Algorithm 2 Algorithm of CAPS (aggregation function MIN)

```
procedure $CALCULATE\_BOUNDS(state, R)$
    if $state = first\_round$ then    ▷ network topology is unknown
        $set\_static\_bounds(R)$;
        return;
    end if
    $m\_l\_b \leftarrow calculate\_min\_lower\_bound()$;
    if $m\_l\_b$ has changed then
        $reset\_bounds(R, m\_l\_b)$;
    end if
end procedure

procedure $reset\_bounds(R, min\_lower\_bound)$
    for $i \leftarrow 1, max\_hops$ do
        if $lower\_bound_i = min\_lower\_bound$ then    ▷ sensor $i$ has minimum lower bound
            $bound_i \leftarrow R$;
        else
            $upper\_bound_i \leftarrow MAX$;
            $lower\_bound_i \leftarrow min\_lower\_bound$;
        end if
    end for
end procedure
```

ranges computed at the sink, by *simply running an identical copy of the sink's predictor algorithm locally*. This may be implemented by either the sink transmitting the predictor module to the sensor (which loads it dynamically into its OS), or if the predictor structure is pre-determined (e.g., a 3^{rd}-order AR predictor), by simply transmitting the computed coefficients.

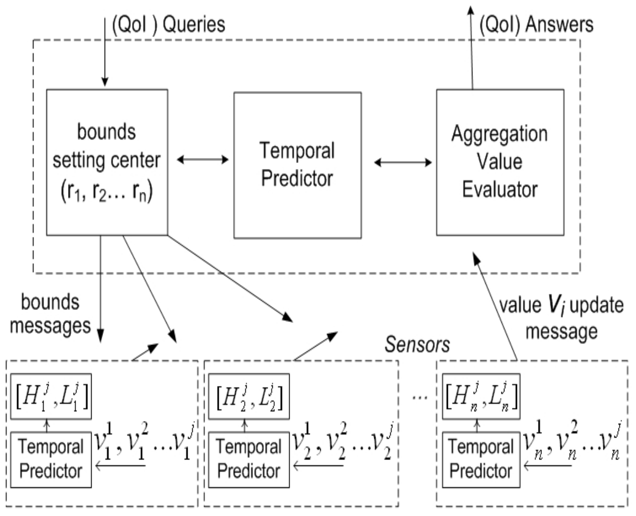
(QoI) Queries
(QoI) Answers
bounds setting center (r1, r2... rn)
Temporal Predictor
Aggregation Value Evaluator
bounds messages
value V_i update message
Sensors
$[H_1^j, L_1^j]$
Temporal Predictor
$v_1^1, v_1^2 \ldots v_1^j$
$[H_2^j, L_2^j]$
Temporal Predictor
$v_2^1, v_2^2 \ldots v_2^j$
$[H_n^j, L_n^j]$
Temporal Predictor
$v_n^1, v_n^2 \ldots v_n^j$

Figure 2. CAPS Architecture with Temporal Prediction

The temporal predictor-enhanced CAPS architecture is shown in Figure 2. It is identical to Figure 1, except for a) an additional TemporalPredictor component in the sink that operates on the past data samples to generate new $\{L_i^j, H_i^j\}$ values in tandem with the BSC, and b) a similar component at the sensor. In general, the architecture can accommodate any predictor (e.g., least-squares, Kalman, etc.). However, a predictor must not be overly-complicated, to ensure that the additional computational and storage (since past samples must be stored while needed) overhead on the sensor does not prove prohibitive. For our initial investigations, we choose a simple linear least-squares (AR) predictor. This is also justified by the observation [9] that samples of many microclimate data exhibit linear correlation under sufficiently high sampling rates. Accordingly, Figure 3 shows the internal operation of the Temporal Predictor at a sensor. A sufficiently long history of the true past samples is used to continually adjust the predictor coefficients (using linear least-squares regression), which are then used to predict the next sample value $\hat{v}_i^{j+1}$, and then adjust the range (L_i^j, H_i^j) (using $L_i^{j+1} = \hat{v}_i^{j+1} - r^i$ and $H_i^{j+1} = \hat{v}_i^{j+1} + r^i$). Note that the multi-step linear predictor uses the past predicted values $\hat{v}_i^j$, rather than the true samples v_i^j (except when v_i^j falls outside $[L_i^j, H_i^j]$) to ensure consistency with the sink (which usually does not receive the intermediate samples v_i^j).

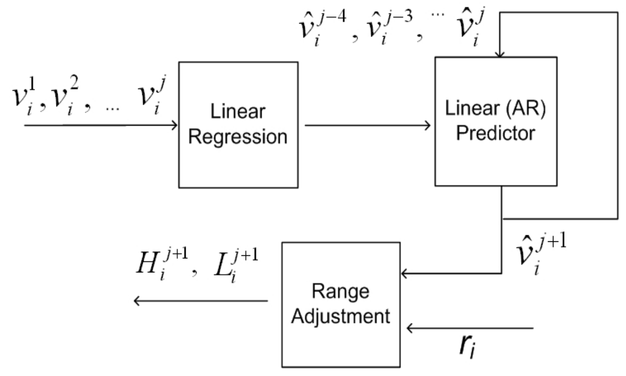

Figure 3. Internal structure of Temporal Predictor

5 Evaluation

We now present simulation and experimental results on the performance of CAPS. In particular, we are interested in the degree to which CAPS can reduce the communication energy in sensor networks, and how this performance is affected by changes to the network topology, sink placement, choice of aggregation function and data generation models. For a comprehensive comparison, we focus on the communication energy consumption under three different techniques for satisfying the QoI requirements of aggregate queries:

- **BASE:** This models the case where each sensor is required to update the sink with every data sample. Conceptually, this captures the case where the application specifies a QoI range of $R = 0$ and provides a baseline for comparing the performance of queries with non-zero tolerance bounds.
- **STATIC:** Here, the aggregate QoI bound is identically split across all the sensors in the constituent set, ignoring the variation in their individual communication costs. Thus, given an application QoI R, each sensor's range is set to R for $\min$, $\max$ and $mean$ queries, and to $R/count(S)$ for the sum query.
- **CAPS**: We shall compare the performance of both basic CAPS, as well as CAPS with temporal predictors.

For the simulation results using synthetic data, we built a custom simulator, using the number of transmissions to indirectly measure the transmission energy overhead (since all sensor nodes use a fixed transmit power).

5.1 Data Generation Models and Topology

We shall study the performance of all three schemes for both *synthetic* and *real data* data traces. For synthetic data, we shall consider data generated according to the random walk model, with each individual sensor assigned an initial data value from a zero-mean binomial distribution. The real data traces are based on data collected over a period of three hours from Mica-2 light sensors deployed in our lab. The sampling period for all our studies is set to $T = 30$ seconds. Except for Figure 6 and 7, all the other performance results are based on synthetic data.

The performance of the three different algorithms on synthetic data is based on random network topologies generated from a uniform node distribution. We perform studies for networks comprising $N = \{100, 200, 300\}$ nodes. For studies with real data, we use an 8-hop linear network, reflecting the alignment of 8 motes along a lab corridor.

5.2 Importance on Non-zero Tolerance on Energy Consumption

Figure 4 shows the network energy consumption with different application QoI requirements (R) for the base CAPS algorithm (without temporal prediction). In studies based on synthetic data generated according to a random walk model, the impact of an application's QoI is best represented by the *normalized precision bound*, defined as the ratio of R to the step-size s by which a sensor's value varies. Clearly, the frequency with which an individual sensor violates it specified value r_i will be directly related to the normalized bound. Accordingly, Figure 4 plots the total communication energy overhead (for the $mean$ aggregation query) for base CAPS as a function of this normalized bound for random networks of three different sizes. The experiments show that there is *a sharp initial drop* in the total energy consumption. By recalling that a normalized bound equal to 0 implies the BASE model, we see that CAPS with a normalized bound of 2 can reduce the energy consumption by $\sim 90\%$ over the BASE approach. This validates our hypothesis that even reasonably tight (but non-zero) tolerance bounds in aggregate queries can provide very significant operational savings in sensor networks.

5.3 Performance Benefits of CAPS Without Temporal Prediction

Figure 5 shows the ratio of the network energy consumption between the STATIC and CAPS approaches for different normalized precision bounds, again for the case of synthetic data generated from a random walk model over a zero-mean binomial distribution. A ratio larger than 1 reflects the potential for energy savings via CAPS over the

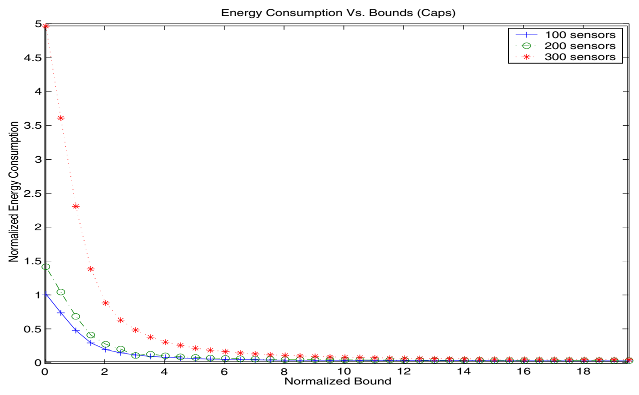

Figure 4. Network Energy Consumption Vs. QoI (normalized R) for $mean$ query

STATIC approach. We can see that CAPS can reduce network energy consumption by around$\sim 10\%$ when the normalized bounds are reasonable large (e.g. 6), regardless of network size. However, when the normalized bounds are too large, then CAPS actually results in higher energy overhead. This can be explained by realizing that at such high values, each individual sensor has a very large precision range r_i and is thus unlikely to ever generate data outside its range, especially when the range is identically distributed for all sensors. Indeed, for cases when the application tolerance is very loose, CAPS may actually impose tighter bounds on sensors close to the sink, causing them to occasionally generate updates (while STATIC never generates an update)! However, for such large bounds, the *communication energy overhead* (not the normalized ratio) for both CAPS and STATIC is anyway very low, so the relative performance gains are of much lesser interest.

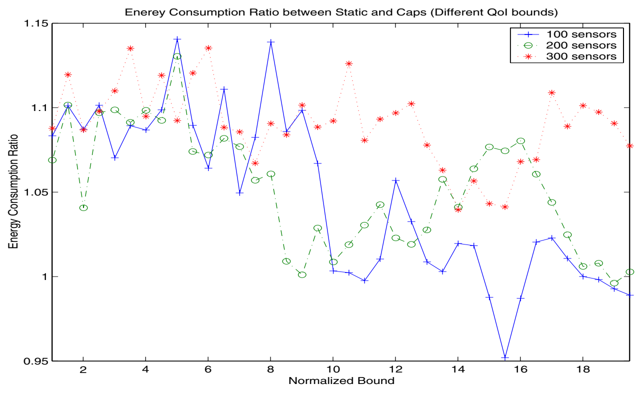

Figure 5. Relative Energy Overhead of STATIC and CAPS Vs. QoI (normalized R)

Figure 6 shows the network energy consumption ratio between STATIC and CAPS with different application QoI requirements (R) (not normalized bounds) in an 8-hop linear network with *real light intensity data*. It shows that the behavior of CAPS is reasonably consistent across synthetic and real data. In particular, CAPS provides energy savings in the realistic portion of the curve, where the precision range is neither too small nor infeasibly large.

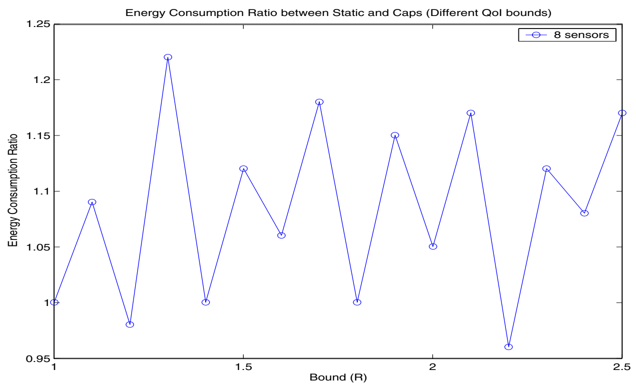

Figure 6. Relative Energy Overhead of STATIC and CAPS Vs. R (with Real Motes Data)

We also studied the relative performance of STATIC and CAPS for different locations of the sink. A change of the location of the sink influences the hop count of different sensor nodes to the sink, and thus influences the precision ranges computed by CAPS. Our studies show that the energy savings achieved by CAP do vary by $\sim 10\%$ based on the sink location (for example, for the 300 node network, the energy savings varied from a min of 4% to a max of 16%).

5.4 Impact of Temporal Prediction on CAPS Performance

Figure 7 shows the network energy consumption ratio between basic CAPS and CAPS enhanced with a temporal predictor for the $mean$ aggregation query. The performance results are derived from the real data traces obtained over the 8-node network in our lab. For the temporal predictor, we implemented a 5^{th} order linear (AR) predictor of the form:

$$\hat{v}^j = a_1\hat{v}^{j-1} + a_2\hat{v}^{j-2} + a_3\hat{v}^{j-3} + a_4\hat{v}^{j-4} + a_5\hat{v}^{j-5} \quad (15)$$

where the coefficients (a_i) of the predictor are continuously updated based on a linear regression over the most recent 50 data samples. We believe that this predictor is efficient in terms of both computational and storage overhead, and can be easily implemented even on existing resource-constrained sensor node platforms.

The figure demonstrates the power of the temporal prediction approach, as it can reduce the network energy consumption by *as much as a factor of* 5 (when the QoI bound (R) is 1.5). In other words, combining predictive filtering with adaptive precision range adjustment can provider **an order of magnitude increase in the operational lifetime** in practical sensor network environments. As expected, when the QoI bound R is too small (e.g. 1), sensors need to transfer data back to the sink at almost each time unit even in the presence of prediction, thus providing relatively smaller savings in communication energy. Similarly, at the other extreme, for applications with very loose QoI bounds, the total number of transmissions needed becomes small, even without prediction. Clearly, in such cases, the benefit of predictive algorithms is comparatively smaller. Of course, larger precision bounds imply progressively less accuracy in the state reported by the sink to the application. The experiments confirm our initial hypothesis that real sensor data exhibits significant temporal correlation, that needs to be effectively exploited.

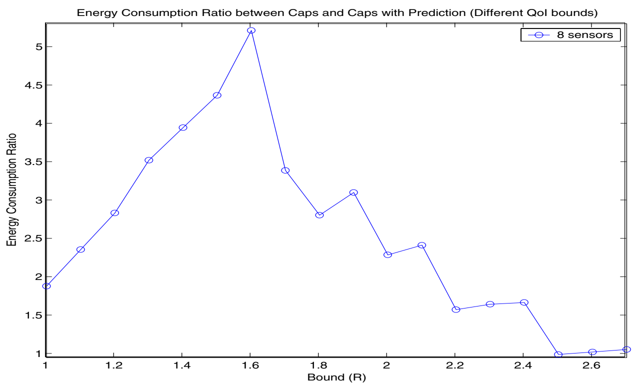

Figure 7. The network energy consumption ratio between CAPS, without and with temporal prediction Vs. QoI (R)

6 Conclusions and Future Work

In this paper, we proposed, implemented and evaluated an energy efficient data retrieval architecture (called CAPS) for continuous aggregate queries in wireless sensor networks. We demonstrated how an application's QoI bound may be collectively decomposed into individual sensor precision ranges, while accommodating the variation in the sensor update costs. More importantly, the CAPS architecture is extended to include temporal predictors that further exploit the correlation among successive data samples. Simulation studies with simulated and synthetic data demonstrate that, while CAPS can reduce the energy overhead by $10 - 20\%$, a combination of temporal prediction and weak consistency can dramatically reduce the operational energy consumption (e.g., by *a factor of 5*).

Having validated CAPS by extensive simulations, we are now working to implement the CAPS algorithms on the Motes platform. For future work, we shall investigate how the features of CAPS may be integrated into in-network data processing architectures.

To make CAPS scalable to future sensor networks that might have thousands of sensors, it is desirable to have hierarchical network model instead of flat network model [13]. We are planning to investigate how to distribute the bounds to clusters/micro-servers while meeting application's QoI requirement.

References

[1] C. Olston and J. Widom, "Offering a precision-performance tradeoff for aggregation queries over replicated data," in *Proceedings of VLDB*. Margan Kaufmann Publishers Inc., 2000, pp. 144–155.

[2] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model driven data acquisition in sensor networks," in *Proceedings of VLDB*. Margan Kaufmann Publishers Inc., 2004, pp. 144–155.

[3] I. Hwang, Q. Han, and A. Misra, "Mastaq: A middleware architecture for sensor applications with statistical quality constraints," in *Proceedings of PERCOMW*. IEEE Computer Society, 2005, pp. 390–395.

[4] C. Olston, B. T. Loo, and J. Widom, "Adaptive precision setting for cached approximate values," in *Proceedings of SIGMOD*. ACM Press, 2001, pp. 355–366.

[5] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy efficient data collection in distributed sensor environments," in *Proceedings of ICDCS*. IEEE Computer Society, 2004, pp. 590–597.

[6] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[7] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proceedings of ICDE*. IEEE Computer Society, 2004, p. 449.

[8] P. Desnoyers, D. Ganesan, H. Li, M. Li, and P. Shenoy, "Presto: A predictive storage architecture for sensor networks," in *Proceedings of HotOS X*, 2005.

[9] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin, "Lightweight temporal compression in microclimate data," in *Proceedings of EmNetS-I*, 2004.

[10] "Crossbow technology, inc. http://www.xbow.com."

[11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 2–16, 2003.

[12] R. Graham, D. Knuth, and O. Patashnik, *Concrete mathematics: a foundation for computer science*. Addison-Welsley Longman Publishing Co., Inc., 1989.

[13] W. Hu, N. Bulusu, and S. Jha, "A communication paradigm for hybrid sensor/actuator networks," in *Proceedings of the IEEE PIMRC*, Barcelona, Spain, Sept. 2004, pp. 201– 205.

Appendix: The Optimal Condition of Objective Function (10)

We prove that the objective function (10) reaches optimum when equation (13) is satisfied in this appendix.

Firstly, we prove that the objective function (10) reaches optimum when

$$\sum_{i\in S} r_i^j = 2R, \forall j \tag{16}$$

We prove it by contradiction. At any specific time j, assumed there exists a positive value Δ, such that the objective function (10) reaches optimum value (α) when $\sum_{i\in S} r_i = 2R - \Delta$. If Δ is added to any r_i (e.g. r_1), we can get a new value (β) from the objective function (10) where $\alpha - \beta = \frac{Hop_i}{(r_1)^2} - \frac{Hop_i}{(r_1+\Delta)^2}$.

It is easy to get $\alpha - \beta > 0$ since Δ is a positive number. Therefore, α is not the optimal value of the objective function (10). Secondly, let us rewrite equation (16) as (17).

$$r_2^j = 2R - \sum_{i\in S}^{i\neq 2} r_i^j, \forall j \tag{17}$$

Then, we can get equation (18) when we substitute r_2^j with $2R - \sum_{i\in S}^{i\neq 2} r_i^j$ in equation (10).

$$minimize \sum_{j\in T} \frac{Hop_1^j}{(r_1^j)^2} + \frac{Hop_2^j}{(2R - \sum_{i\in S}^{i\neq 2} r_i^j)^2} + ... + \frac{Hop_n^j}{(r_n^j)^2} \tag{18}$$

Let us take partial derivative of equation (18) with respect to r_1^j, and equal it to 0:

$$\begin{aligned} \frac{\partial}{\partial r_1^j}\left(\frac{Hop_1^j}{(r_1^j)^2}\right) + \frac{\partial}{\partial r_1^j}\left(\frac{Hop_2^j}{(2R - \sum_{i\in S}^{i\neq 2} r_i^j)^2}\right) = 0 \\ \Rightarrow \quad -\frac{2Hop_1^j}{(r_1^j)^3} + \frac{2Hop_2^j}{(2R - \sum_{i\in S}^{i\neq 2} r_i^j)^3} = 0 \\ \Rightarrow \quad r_1^j : r_2^j = \frac{Hop_1^j}{(r_1^j)^2} : \frac{Hop_2^j}{(r_2^j)^2} \end{aligned} \tag{19}$$

Similarly, we can prove that the objective function (10) reaches partial optimum when:

$$r_1^j : r_3^j = \frac{Hop_1^j}{(r_1^j)^2} : \frac{Hop_3^j}{(r_3^j)^2}, ..., r_1^j : r_n^j = \frac{Hop_1^j}{(r_1^j)^2} : \frac{Hop_n^j}{(r_n^j)^2} \tag{20}$$

Therefore, equation (10) reaches optimum when equation (13) is satisfied.