Adaptive Data Collection Strategies for Lifetime-Constrained Wireless Sensor Networks

Xueyan Tang, *Member, IEEE*, and Jianliang Xu, *Senior Member, IEEE*

Abstract—Communication is a primary source of energy consumption in wireless sensor networks. Due to resource constraints, the sensor nodes may not have enough energy to report every reading to the base station over a required network lifetime. This paper investigates data collection strategies in lifetime-constrained wireless sensor networks. Our objective is to maximize the accuracy of data collected by the base station over the network lifetime. Instead of sending sensor readings periodically, the relative importance of the readings is considered in data collection: the sensor nodes send data updates to the base station when the new readings differ more substantially from the previous ones. We analyze the optimal update strategy and develop adaptive update strategies for both individual and aggregate data collections. We also present two methods to cope with message losses in wireless transmission. To make full use of the energy budgets, we design an algorithm to allocate the numbers of updates allowed to be sent by the sensor nodes based on their topological relations. Experimental results using real data traces show that, compared with the periodic strategy, adaptive strategies significantly improve the accuracy of data collected by the base station.

Index Terms—data collection, energy efficiency, network lifetime, data accuracy, sensor network.

1 Introduction

The primary functions of wireless sensor networks are to observe and analyze physical phenomena [1], [2], [3]. A wireless sensor network typically consists of a base station and a group of geographically distributed sensor nodes. The sensor nodes are responsible for sampling real-world phenomena such as temperature and solar radiation. They also communicate with each other and the base station through radios to exchange information. The base station, on the other hand, collects the data acquired by the sensor nodes for relevant applications. The data collected by the base station may include individual sensor readings or an aggregate form of sensor readings. Primarily designed for monitoring purposes, many sensor applications request continuous collection of up-to-date sensor data.

In wireless sensor networks, the sensor nodes are usually battery powered. Replacing the batteries is not only costly but also inconvenient in many situations. Thus, many sensor networks are deployed to operate for a designated time period called *network lifetime* [4]. Due to resource constraints, however, a sensor node may not have enough energy to report every reading to the base station over the required network lifetime, since communication is a primary source of energy consumption [5], [6]. Therefore, the node has to decide which readings to send to the base station on the fly.

A straightforward method is to let the sensor nodes periodically report readings at the maximum rate subject to the energy constraint [7]. However, this approach is not effective. Consider, for example, a series of solar radiation readings 369, 330, 264, 266, 274, 279, 260, 233, 225 (W/m^2) logged in the LEM project[1] at 9 successive time units [8]. Suppose the energy budget is sufficient for a sensor node to send only 3 updates to the base station. This implies the maximum report rate is once every 3 time units. If the node reports periodically, the 1st, 4th and 7th readings would be sent to the base station. Then, the readings observed by the base station in real time are 369, 369, 369, 266, 266, 266, 260, 260 and 260 over the 9 time units (see Figure 1(a)). So, the instantaneous deviations from up-to-date sensor readings are 0, 39, 105, 0, 8, 13, 0, 27 and 35. As a result, the cumulative deviation is 227. In contrast, if the sensor node sends the 1st, 3rd and 8th readings to the base station, the base station would observe 369, 369, 264, 264, 264, 264, 264, 233 and 233 over the 9 time units (see Figure 1(b)). The instantaneous deviations are thus 0, 39, 0, 2, 10, 15, 4, 0 and 8. Therefore, the cumulative deviation is 78 — a 66% reduction compared to the periodic approach. This example motivates us to consider the relative importance of sensor readings when making update decisions. In the above example, the 1st, 3rd and 8th readings are more important because they differ more substantially from the previous readings. It is desirable to update the base station with these readings to reduce the deviation of the data observed by the base station.

In this paper, we investigate data collection strategies in lifetime-constrained wireless sensor networks. Given a network lifetime requirement, we are interested in determining which sensor readings to send to the base station with an objective of minimizing the deviations of the readings observed by the base station over the network lifetime. Our contributions are as follows:

- *X. Tang is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798. E-mail: asxytang@ntu.edu.sg.*
- *J. Xu is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: xujl@comp.hkbu.edu.hk.*

1. Please refer to Section 6.1 for a description of the traces collected by the LEM project [8].

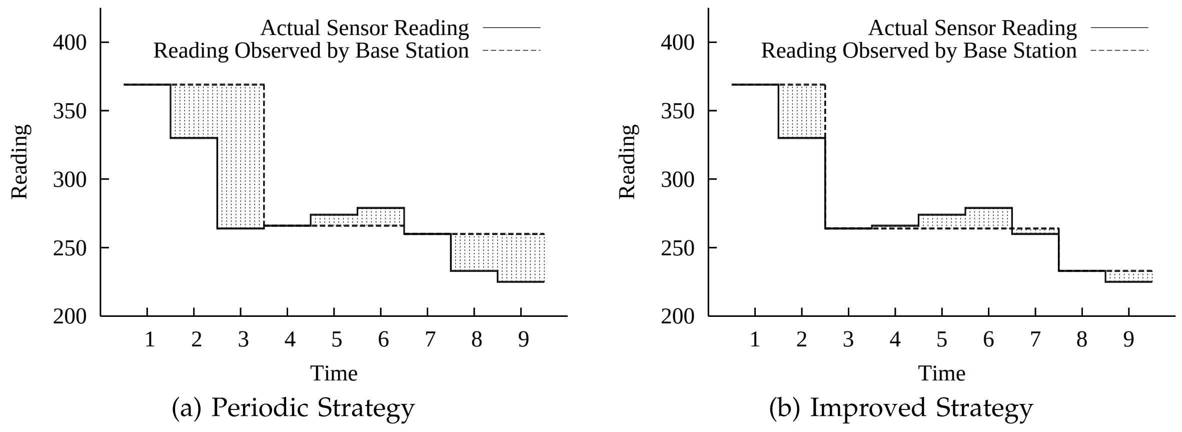


Fig. 1. Deviation of Data Collected by Base Station

- We formulate the lifetime-constrained data collection problem in sensor networks. An offline algorithm is developed to compute the optimal data update strategy.
- We propose an adaptive strategy that makes data update decisions on the fly based on sensor readings to meet network lifetime requirements. The basic strategy applies directly to individual data collection where the application monitors the reading of an individual sensor node. It is also extended to deal with aggregate data collection where the application continuously requests an aggregate form of sensor data (e.g., the average reading of all sensor nodes).
- We develop two methods, History and Expected, for the adaptive strategy to cope with message losses in wireless transmission. The key idea is to take into consideration the possibility of update losses in estimating the importance of sensor readings.
- In connection with the adaptive strategy for aggregate data collection, we develop an algorithm to allocate the numbers of updates allowed to be sent by the sensor nodes based on their topological relations. The goal is to make full use of the energy budgets of the sensor nodes to improve the quality of collected data.
- We conduct an experimental evaluation using a wide range of real data traces for both individual and aggregate data collections. The results show that, compared to the periodic strategy, the proposed adaptive strategy significantly improves the accuracy of data collected by the base station over the network lifetime.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 describes the system model. Section 4 analyzes the optimal data update strategy and proposes a basic adaptive update strategy for individual data collection. Section 5 extends the adaptive strategy to aggregate data collection. The experimental setup and results are discussed in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Several approaches have been proposed to trade the quality of data collection for energy efficiency in wireless sensor networks. One approach is to relax data semantics to allow a specified degree of error to be tolerated in the collected data. Studies have been carried out for individual data collection [9], [10], [11], aggregate data collection [12], [13], [14], [15], [16] and quantile tracking [17], [18], [19]. In our earlier work, we designed a two-tier storage scheme for one-shot individual data collection in object tracking sensor networks [20]. We also developed precision allocation schemes to extend network lifetime for continuous aggregate data collection [21], [22]. Different from existing work, in this paper, we target at improving the accuracy of continuous data collection given the requirements of network lifetime. Without a priori knowledge of the changing pattern of sensor readings, it is difficult to pre-set appropriate precision constraints on sensor data collection to meet a given network lifetime requirement. Moreover, a static precision constraint is insufficient if the changing pattern of sensor readings keeps evolving over time. In this paper, we propose techniques to dynamically adjust the precision of data collection on the fly.

Another approach exploits the spatial correlation between sensor readings [23], [24]. In this approach, groups of sensor nodes that are geographically clustered and have similar sensor readings are identified. Data are then collected from an elected representative node in each group only, thereby saving the energy at the other nodes. Our approach complements this one in that we exploit the temporal correlation between sensor readings. We instruct the sensor nodes to send fewer data updates when the physical phenomena in their immediate surroundings change slowly. The saved energy is then used to send more updates when the physical phenomena change rapidly to improve the quality of data collected in real time.

In addition, Considine *et al.* [25] and Nath *et al.* [26] implemented approximate data aggregation under multipath routing by means of sketches and synopses. However, they did not make use of temporal locality to suppress data updates. Work has also been done on compressing historical sensor readings for transmission [27], [28]. These methods are applicable to archival data collection where the application wants to log historical sensor readings and analyze them at a later time. In contrast, we consider monitoring applications such as environmental and structural monitoring that continuously request up-to-date sensor readings.

3 System Model

We consider a network of sensor nodes that periodically sample local measurements (e.g., temperature and solar radiation) at a designated rate. Without loss of generality, the period between two successive samplings is assumed to be 1 time unit. The base station gathers data from the sensor nodes. The sensor network has a lifetime requirement of T time units. Due to the energy constraints of sensor nodes, not all sensor readings can be sent to the base station over the required network lifetime. Thus, the data collected by the base station are likely to deviate from up-to-date sensor readings at some time units. Let $B(t)$ be the data value observed by the base station at time t, and $Q(t)$ be the exact data value at time t. The instantaneous deviation at time t is then given by

$$|B(t) - Q(t)|.$$

Note that the definitions of $B(t)$ and $Q(t)$ vary with the type of data collection. In individual data collection, they refer to the reading of an individual sensor node. In aggregate data collection, they refer to an aggregate form of the readings acquired by all sensor nodes (e.g., the maximum or the average reading of all sensor nodes). We measure the quality of data collection as the root mean square error of collected data with respect to exact data over the network lifetime [26], [7], i.e.,

$$\sqrt{\frac{\sum_{t=1}^{T} |B(t) - Q(t)|^2}{T}}.$$

Root mean square error is a well-known metric to measure deviation in statistics. The smaller is the root mean square error, the higher is the accuracy of collected data. Our objective is to determine for the sensor nodes which readings to send to the base station (called *the data update strategy*) so as to minimize the root mean square error over the network lifetime.

4 Basic Data Update Strategies

We start by investigating the simple case where a single sensor node sends its readings to the base station directly in individual data collection. The data update strategies developed for this simple case serve as a building block for the strategies we shall propose for aggregate data collection in a network of sensor nodes (Section 5).

4.1 Problem Formulation

For simplicity, we assume the base station maintains the reading last updated by the sensor node until the next update. Our analysis and algorithms can be extended in a straightforward manner to include more sophisticated prediction models [23], [27] for the base station to extrapolate sensor readings on the fly over inter-update periods (see Section 4.3).

Let $d_1, d_2, \cdots, d_T$ be the T readings acquired by a sensor node over the network lifetime, i.e., for any $1 \le t \le T$,

$$Q(t) = d_t.$$

Assume the sensor node can send at most $M \le T$ data updates to the base station due to its energy constraint. Suppose the data updates are sent at times $v_1, v_2, \cdots, v_M$ where $1 = v_1 < v_2 < \cdots < v_M \le T$.[2] Then, for each $1 \le i < M$, the data observed by the base station[3] from time v_i to $v_{i+1} - 1$ is d_{v_i}, and that from time v_M to T is d_{v_M}, i.e.,

$$B(t) = \begin{cases} d_{v_i} & \text{if } v_i \le t < v_{i+1}, \\ d_{v_M} & \text{if } v_M \le t \le T. \end{cases}$$

Thus, the root mean square error is given by

$$\begin{aligned} \mathcal{D}(T : v_1, v_2, \cdots, v_M) &= \sqrt{\frac{\sum_{t=1}^{T} |B(t) - Q(t)|^2}{T}} \\ &= \sqrt{\frac{\sum_{i=1}^{M-1} \sum_{j=v_i}^{v_{i+1}-1} |d_j - d_{v_i}|^2 + \sum_{j=v_M}^{T} |d_j - d_{v_M}|^2}{T}}. \end{aligned}$$

The data collection problem can therefore be formulated as finding $v_1, v_2, \cdots, v_M$ that minimize $\mathcal{D}(T : v_1, v_2, \cdots, v_M)$.

4.2 Optimal Data Update Strategy

We develop an optimal data update strategy assuming that all sensor readings are known a priori. It will be used as a yardstick (lower bound) in performance evaluation (Section 6).

The data collection problem defined above can be solved by a dynamic programming algorithm. Note that given a network lifetime requirement, to minimize the root mean square error, it is equivalent to minimize the total square error. Consider a more generalized problem of finding $1 = v_1 < v_2 < \cdots < v_m \le t$ (where $m \le M$ and $t \le T$) that minimize

$$\begin{aligned} & T \cdot \mathcal{D}(t : v_1, v_2, \cdots, v_m)^2 \\ & \quad = \sum_{i=1}^{m-1} \sum_{j=v_i}^{v_{i+1}-1} |d_j - d_{v_i}|^2 + \sum_{j=v_m}^{t} |d_j - d_{v_m}|^2. \end{aligned}$$

We shall call it the (t, m)-optimization problem.

2. We stipulate that $v_1 = 1$ because the data collected by the base station are undefined initially. Thus, the sensor node must send its reading to the base station at the first time unit.

3. We neglect the transmission delay in the network since it simply shifts the data observed by the base station by a time offset that is independent of any data update strategy.

Let $v_1^*, v_2^*, \cdots, v_m^*$ be an optimal solution to the (t, m)-optimization problem. We show that $v_1^*, v_2^*, \cdots, v_{m-1}^*$ must be an optimal solution to the $(v_m^* - 1, m-1)$-optimization problem. Assume on the contrary that there exists a better solution $u_1^*, u_2^*, \cdots, u_{m-1}^*$ to the $(v_m^* - 1, m-1)$-optimization problem, i.e.,

$$\begin{aligned} &T \cdot \mathcal{D}(v_m^* - 1 : u_1^*, u_2^*, \cdots, u_{m-1}^*)^2 \\ &\quad < T \cdot \mathcal{D}(v_m^* - 1 : v_1^*, v_2^*, \cdots, v_{m-1}^*)^2. \end{aligned}$$

It follows that

$$\begin{aligned} &T \cdot \mathcal{D}(t : u_1^*, u_2^*, \cdots, u_{m-1}^*, v_m^*)^2 \\ &= T \cdot \mathcal{D}(v_m^* - 1 : u_1^*, u_2^*, \cdots, u_{m-1}^*)^2 + \sum_{j=v_m^*}^{t} |d_j - d_{v_m^*}|^2 \\ &< T \cdot \mathcal{D}(v_m^* - 1 : v_1^*, v_2^*, \cdots, v_{m-1}^*)^2 + \sum_{j=v_m^*}^{t} |d_j - d_{v_m^*}|^2 \\ &= T \cdot \mathcal{D}(t : v_1^*, v_2^*, \cdots, v_{m-1}^*, v_m^*)^2, \end{aligned}$$

which contradicts the optimality of $v_1^*, v_2^*, \cdots, v_m^*$. Therefore, the optimal solution to the (t, m)-optimization problem must contain optimal solutions to some subproblems.

Let $A(t, m)$ be the minimal achievable total square error in the (t, m)-optimization problem, and let $B(t, m)$ be the time of the last data update in the optimal solution. The recurrences for dynamic programming are then given by

$$A(t,m) = \begin{cases} \min\limits_{m \le i \le t} \Big(A(i-1, m-1) + \sum_{j=i}^{t} |d_j - d_i|^2 \Big) \\ \hfill \text{if } m > 1, \\ \sum_{j=1}^{t} |d_j - d_1|^2 \hfill \text{if } m = 1, \end{cases}$$

and

$$B(t,m) = \begin{cases} \arg\min\limits_{m \le i \le t} \Big(A(i-1, m-1) + \sum_{j=i}^{t} |d_j - d_i|^2 \Big) \\ \hfill \text{if } m > 1, \\ 1 \hfill \text{if } m = 1. \end{cases}$$

Starting from $A(t, 1)$'s and $B(t, 1)$'s, we can compute all $A(t, m)$'s and $B(t, m)$'s in increasing orders of t and m. To solve the problem defined in Section 4.1, on obtaining all A- and B-entries, the optimal times for sending data updates are calculated by tracing back the B-entries:

$$v_M = B(T, M),$$

and for each $1 \le i < M$,

$$v_i = B(v_{i+1} - 1, i).$$

Given any i, the computation complexity of $\sum_{j=i}^{t} |d_j - d_i|^2$ for all different t's is $O(T)$. Hence, $\sum_{j=i}^{t} |d_j - d_i|^2$ for all pairs of i and t can be computed in a pre-processing stage in $O(T^2)$ time. Then, the time complexity to compute each A-/B-entry is given by $O(T)$. Since there are a total of $O(M \cdot T)$ A-/B-entries, the total time complexity of the dynamic programming algorithm is $O(MT^2)$.

4.3 Adaptive Data Update Strategy

The dynamic programming solution presented above is an offline algorithm — all sensor readings over the network lifetime are required in the computation. In real-time monitoring applications, however, future sensor readings are not known a priori and data update decisions must be made on the fly. So now, we propose an adaptive online data update strategy.

The basic idea is to let the sensor node update a new reading with the base station only when the new reading substantially differs from the last update to the base station. The rationale behind is that these sensor readings, as shown by the example in Section 1, are more effective in reducing the instantaneous deviations of the data observed by the base station. It is similar in spirit to the idea of prioritizing sensor data delivery based on their differences from the data most recently transmitted when the radio queue of a sensor node overflows [7].

Specifically, the sensor node maintains the reading U last updated with the base station. When a new reading V is generated, the difference between V and U is computed. The sensor node updates the new reading with the base station only if the difference is greater than a threshold W.

It is intuitive that the rate of data updates sent by the sensor node depends on the threshold W. We propose to dynamically adjust the threshold to meet the network lifetime requirement. Our design is inspired by the work of Olston *et al.* [13] which used thresholds to filter streams of data updates and adapted the thresholds to control stream rates. Their purpose of adaptation, however, was to minimize the total communication cost between a set of data sources and the data sink. In contrast, our objective here is to adjust the threshold of a data source over time to meet the lifetime requirement. To this end, the sensor node measures the data update period I (i.e., the duration between two successive data updates to the base station) using an exponential aging method. At each data update, the estimate of I is recomputed as

$$I = \alpha \cdot (T_c - T_l) + (1 - \alpha) \cdot I_{old},$$

where T_c is the current time, T_l is the time of the last data update to the base station, I_{old} is the estimate of I at the last data update, and α is a factor weighing the importance of the current update period against past ones. On the other hand, the expected data update period under the network lifetime requirement is computed as

$$I_E = \frac{T - T_c}{R},$$

where $T - T_c$ is the remaining network lifetime and R is the remaining number of data updates allowed. If the total allowable number of updates due to energy constraints is M, then R is given by $M - C$, where C is the number of data updates sent so far.

I is compared with I_E whenever the sensor node updates its reading with the base station. If I is greater than I_E by a factor ϵ (i.e., $I > I_E \cdot (1+\epsilon)$), the threshold W is reduced by a factor δ: $W = W \cdot (1-\delta)$ to increase the data update rate and hence improve the accuracy of data collected by the base station. On the other hand, if I is less than I_E by a factor ϵ (i.e., $I < I_E \cdot (1-\epsilon)$), the threshold W is increased by a factor δ: $W = W \cdot (1+\delta)$ to extend the data update period. We shall investigate the impact of algorithm parameters α, δ and ϵ with simulation experiments in Section 6.

If we check (and adjust if necessary) the threshold each time the sensor node updates with the base station, the adaptive algorithm would react quickly to increase the threshold when the changing of physical phenomena becomes more intensive. This is because when the changes increase in magnitude, the data update rate increases, thereby giving more chances of adjustment. However, when the changing of physical phenomena becomes less intensive, the algorithm would react slowly to reduce the threshold. This is because when the changes decrease in magnitude, the data update rate decreases, leading to fewer chances of adjustment. To remedy it, we deliberately make some adjustments to the threshold in addition to those performed when the sensor node updates its reading with the base station. Specifically, if there has been no data update to the base station for twice the expected update period I_E, we decrease the threshold W by a factor of δ.

To initialize the threshold, the sensor node is instructed to send data updates periodically (at the expected data update period I_E) for a small number of h times at the beginning of data collection. The average difference between successively updated readings is used to initialize the threshold. The adaptive data update strategy is summarized in Algorithm 1.

We remark that this basic adaptive strategy applies directly to individual data collection that requests the reading of an individual sensor node. If the communication between the source sensor node and the base station has to go through multiple hops in a wireless sensor network, the energy constraints at the intermediate nodes should be taken account of in calculating the total allowable number of data updates M.

The basic adaptive strategy can also be tailored to include prediction models for the base station to extrapolate sensor readings on the fly over inter-update periods. To do so, the sensor node maintains the same prediction model as that used by the base station and applies the threshold W to the difference between the actual sensor reading and the reading predicted by the model. That is, the sensor node updates a new reading with the base station only if the new readings differs from the predicted reading by more than W. Again, the threshold W is adjusted dynamically to meet the network lifetime requirement. A good extrapolation model is expected to reduce the threshold W, thereby improving the accuracy of data observed by the base station.

Algorithm 1 Adaptive Data Update Strategy

```
1: set $W \leftarrow 0$;
2: for each time unit $T_c = T/M * i + 1$ $(0 \le i \le h-1)$ do
3:     let $V$ be the sensor reading acquired at time $T_c$;
4:     send a data update $V$ to the base station;
5:     if $i \neq 0$ then
6:         set $W \leftarrow W + |V - U|$;
7:     end if
8:     set $U \leftarrow V$, $T_l \leftarrow T_c$;
9: end for
10: set $W \leftarrow W/(h-1)$;
11: set $C \leftarrow h$, $I \leftarrow T/M$, $I_E \leftarrow T/M$;
12: for each time unit $T_c > T/M * (h-1) + 1$ do
13:     let $V$ be the sensor reading acquired at time $T_c$;
14:     if $|V - U| > W$ and $C < M$ then
15:         send a data update $V$ to the base station;
16:         set $C \leftarrow C + 1$;
17:         set $I \leftarrow \alpha \cdot (T_c - T_l) + (1-\alpha) \cdot I$;
18:         set $I_E \leftarrow (T - T_c)/(M - C)$;
19:         if $I > I_E \cdot (1+\epsilon)$ then
20:             set $W \leftarrow W \cdot (1-\delta)$;
21:         else if $I < I_E \cdot (1-\epsilon)$ then
22:             set $W \leftarrow W \cdot (1+\delta)$;
23:         end if
24:         set $U \leftarrow V$, $T_l \leftarrow T_c$;
25:     else if $T_c - T_l > 2 \cdot I_E$ then
26:         set $W \leftarrow W \cdot (1-\delta)$;
27:         set $T_l \leftarrow T_c$;
28:     end if
29: end for
```

4.4 Coping with Message Losses

In general, messages transmitted over wireless links are subject to losses due to environmental interference, packet collision and low signal-to-noise ratios [25], [26]. So far, we have assumed reliable transmission of data updates. The adaptive strategy described in Section 4.3 is directly applicable if a reliable transfer protocol is used by the sensor network to guarantee the delivery of every single data update [29]. If the transfer protocol is not reliable, however, data updates may be lost in transmission. Losing a data update in the adaptive strategy has an adverse effect on the accuracy of data collection that may be even more severe than losing an update in the periodic strategy. This is because update decisions in the adaptive strategy are made based on sensor readings. For example, if the physical phenomena measured by a sensor node undergo a dramatic change and then remain stable for a long time period, the adaptive strategy would transmit only one data update. In case the update is lost, the base station would retain obsolete and incorrect data until the phenomena change again significantly to trigger the next data update. In contrast, under the periodic strategy, the sensor readings are transmitted periodically even if the phenomena are stable. As a result, the data observed by the base station would be corrected sooner.

In this section, we propose two methods to cope with message losses in the adaptive strategy. The key idea is to take into consideration the possibility of update losses in calculating the difference of a new sensor reading with respect to previously updated readings. Let p be the message loss rate from a sensor node to the base station.[4] Our first method maintains the last k updated readings U_k, U_{k-1}, $\ldots$, U_2, U_1 sent by the sensor node, where k is a given number and U_i is the ith most recently updated reading. The data currently observed by the base station would be U_i $(i = 1, 2, \ldots, k)$ if and only if the updates of U_{i-1}, U_{i-2}, $\ldots$, U_2, U_1 were all lost and the update of U_i was successful. Assuming that message losses occur independently, the probability of the base station observing U_i at present is then $p^{i-1}(1-p)$. When a new sensor reading V is generated, the expected difference of V from the data at the base station is given by

$$\frac{\sum_{i=1}^{k} p^{i-1}(1-p)\cdot|V-U_i|}{\sum_{i=1}^{k} p^{i-1}(1-p)} = \frac{\sum_{i=1}^{k} p^{i-1}(1-p)\cdot|V-U_i|}{1-p^k}. \quad (1)$$

The sensor node compares the expected difference (1) with the threshold W and updates the new reading with the base station only if the expected difference is greater than W. On sending a data update, the sensor node updates the set of last k updated readings. We shall refer to this method as History. In fact, the original adaptive strategy of Section 4.3 is a special case of the History method with $k = 1$. Note that in calculating the expected difference (1), we do not take into consideration the situation where all the last k updates were lost. This is because the sensor node maintains the readings of the last k updates only. We remark that the effect of such simplification is insignificant since the probability of losing all the last k updates is p^k, which decreases exponentially with increasing k. A k value of 8 would make this probability lower than 0.4% even if the message loss rate p is as high as 50%.

It is intuitive that the accuracy of the History method improves with increasing k (i.e., maintaining a longer history of updated readings). However, the storage cost as well as the computation cost of (1) both increase with k. Our second method attempts to reduce these costs by maintaining at the sensor node only one updated reading — the expected updated reading. Let U_i be the ith most recently updated reading sent by the sensor node. Taking all past updated readings into consideration, the expected updated reading is computed as

$$\sum_i p^{i-1}(1-p)\cdot U_i.$$

The expected updated reading can be maintained incrementally at the sensor node. Let U_e be the expected updated reading. On sending a data update V, the sensor node simply updates U_e by setting

$$U_e = U_e \cdot p + V \cdot (1-p).$$

On generating a new reading, the sensor node computes its difference from the expected updated reading U_e. The new reading is updated with the base station only if the difference is greater than the threshold W. We shall refer to this method as Expected. Note that the Expected method trades the accuracy of estimation for storage and computation complexities. In general, the difference between a new sensor reading V and U_e is smaller than the expected difference (1) defined in the History method (with a complete history of updated readings maintained) since

$$|V-U_e| = \left|V-\sum_i p^{i-1}(1-p)\cdot U_i\right| \le \sum_i p^{i-1}(1-p)\cdot|V-U_i|.$$

The two differences are equivalent only if V together with all past updated readings U_1, U_2, U_3, $\ldots$ increase or decrease monotonically.

The adaptive data update strategy can be augmented with the History or Expected method to cope with update losses. Like that in the original adaptive strategy of Section 4.3, the threshold W in the History and Expected methods is also dynamically adjusted to meet the network lifetime requirement.

5 Adaptive Aggregate Data Collection

Now, we consider aggregate data collection that requests an aggregate form of sensor data over a network of sensor nodes (e.g., the maximum or the average reading of all sensor nodes).

5.1 Applying Adaptive Data Update Strategy

Due to limited radio transmission range, a routing infrastructure has to be established to transport data from the sensor nodes to the base station. A common practice is to organize the sensor nodes into a tree structure rooted at the base station (e.g., by flooding a routing request over the network) [7], [11], [18], [31]. To collect data, each intermediate node is responsible for forwarding the data updates received from its children to its parent.[5] *In-network aggregation* is often used to cut down the volume of data sent over the upper-level links in the tree for aggregate data collection [34]. That is, the data update sent by an intermediate node to its parent is a partial aggregate result of the sensor readings in the subtree

4. Message losses can be inferred by tracking the sequence numbers of the messages successfully received at the destination. A number of efficient estimators exist for link reliability based on message losses observed [30]. The effect of imperfect loss rate estimation will be investigated by simulation experiments in Section 6.

5. A number of MAC protocols exist to coordinate the switches between sleep and active modes among the sensor nodes [32], [33].

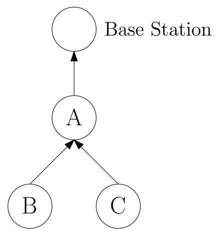


Fig. 2. A Simple Network

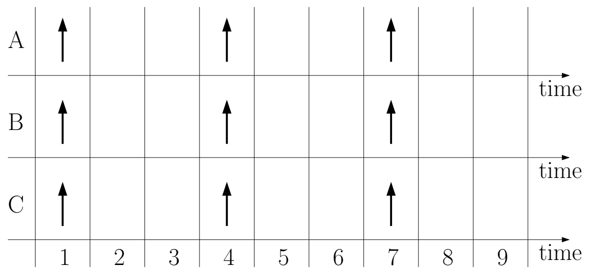


Fig. 3. Periodic Update Strategy

rooted at the intermediate node. For example, at a node i, the partial result for maximum aggregation is the maximum sensor reading in the subtree $\mathcal{T}_i$ rooted at node i; the partial result for average aggregation has the form of $\langle S_i, |\mathcal{T}_i| \rangle$, where S_i is the sum of the sensor readings in $\mathcal{T}_i$, and $|\mathcal{T}_i|$ is the size of $\mathcal{T}_i$ [34].

If all sensor nodes send data updates periodically and at the same rate, the same number of data updates are sent over each link in the tree. Assuming there is one update every 3 time units, Figure 3 shows some sample update behaviors of the sensor nodes in Figure 2. Each row in the figure shows the behavior of one node (specified at the left side) and each arrow represents a data update sent by the node to its parent. As seen from Figure 3, over the 9 time units, each sensor node sends 3 updates at times 1, 4, and 7. Note that sending and receiving data updates both consume energy. Thus, under the periodic update strategy, the energy bottleneck in each subtree rooted at a child of the base station is the node with the highest degree in the subtree. Let i be the bottleneck node in the subtree rooted at a child node j of the base station. Suppose node i has an energy budget e_i. Then, the number of data updates each node in $\mathcal{T}_j$ can send is given by $e_i/(s+|\mathcal{C}_i|\cdot v)$, where $\mathcal{C}_i$ is the set of i's children, $|\mathcal{C}_i|$ is the number of i's children, s and v are the energy costs for a sensor node to send and receive a data update respectively.

However, unlike individual data collection, we cannot simply apply the adaptive update strategy in Section 4.3 to the local readings of each sensor node for aggregate data collection. This is because if each sensor node makes update decisions based on its local readings independently, the data updates initiated by different nodes may not be synchronized. If an intermediate node forwards a data update to its parent immediately upon receiving an update from any child, it may end up in sending as many updates as the total of those sent by its children, thus defeating the purpose of in-network aggregation. Figure 4 shows some sample update behaviors of the sensor nodes in Figure 2. Assume that node B sends data updates at times 1, 5 and 8 based on its local readings, and node C sends updates at times 3, 8 and 9. The row marked "A(local)" shows the data updates initiated by node A based on its local readings. Integrating the three update streams of "A(local)", "B" and "C", the intermediate node A would need to send 8 updates over the 9 time units. The situation deteriorates if an intermediate node has more descendants.

To leverage the advantages of both adaptive update strategy and in-network aggregation, we propose to let each intermediate node apply the adaptive update strategy for its partial aggregate results rather than its local readings. Specifically, in addition to the local reading, an intermediate node also maintains the latest data value reported by each child. The data value maintained for a child is refreshed when the intermediate node receives a new update from the child. When this happens or when the intermediate node acquires a new local reading, it re-aggregates the data values to produce a new partial aggregate result. The result is then sent to the parent if it differs from the last updated data value to the parent by more than a threshold W. In general, not every data update from the children of an intermediate node leads to an immediate update from the intermediate node to its parent. Figure 5 shows some sample update behaviors of the sensor nodes in Figure 2. Using the adaptive strategy in Section 4.3, the number of updates sent by an intermediate node is controlled by adjusting

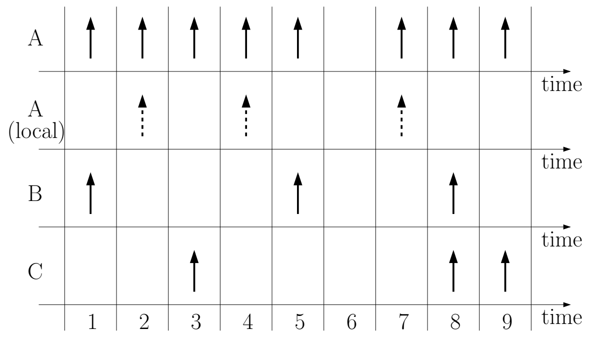


Fig. 4. Improper Use of Adaptive Update Strategy

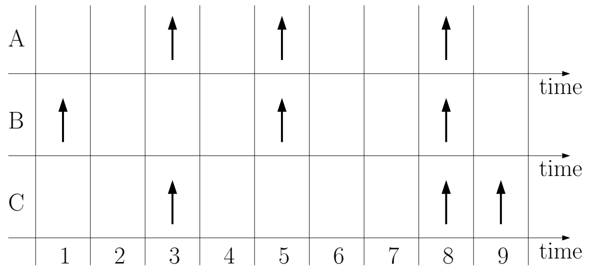
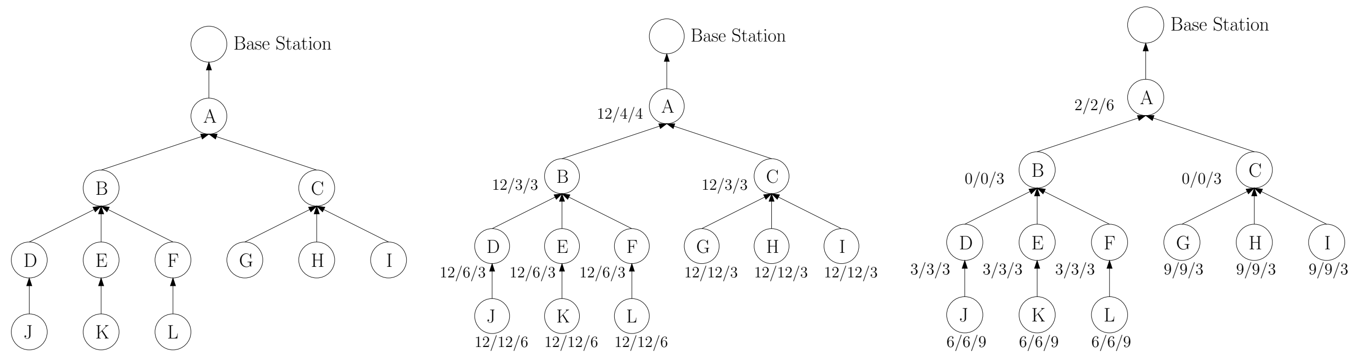


Fig. 5. Proper Use of Adaptive Update Strategy



(a) An Example Network (b) Update Allocation (Round 1) (c) Update Allocation (Round 2)

Fig. 6. An Example Network and Its Update Allocation

the threshold dynamically. The intermediate node can also employ the History or Expected method developed in Section 4.4 to cope with update losses. In this case, the loss rate p refers to the link loss rate between the intermediate node and its parent.

5.2 Allocating Numbers of Updates

In addition to the energy constraints of individual sensor nodes, aggregate data collection over a network of sensor nodes is also restricted by the topological relations among the nodes. We now study the number of updates each sensor node can send in adaptive aggregate data collection. Note that in periodic data collection, since the update decisions are not made based on sensor readings, there is no need for a sensor node to send more updates to its parent than those sent by its parent to its grandparent. As a result, the rate of updates sent by a sensor node does not exceed that by its parent – it either equals the parent's rate or is an integral divisor of the parent's rate [7]. However, adaptive data collection is different. Consider an example topology in Figure 6(a). Suppose all nodes have the same energy budget e. Following the periodic strategy, since B and C have a degree 3, each of them can send at most $e/(s+3\cdot v)$ data updates to their parent A, where s and v are the energy costs for a sensor node to send and receive a data update respectively. If each of the remaining nodes sends the same number of updates as B and C, their energy budgets are underutilized. Node A consumes a portion $(s+2\cdot v)/(s+3\cdot v)$ of the energy budget, nodes D to F each consumes a portion $(s+v)/(s+3\cdot v)$ of the budget, and nodes G to L each consumes a portion $s/(s+3\cdot v)$ of the budget only. Thus, nodes J, K, L can send more data updates to D, E, F respectively without breaking the energy constraints. This would improve the accuracy of data values maintained at D, E, F for J, K, L respectively, and hence improve the quality of data update decisions made by D, E, F. It in turn improves the accuracy of data collected by the base station. Similarly, A can also send more data updates to the base station to improve the quality of data collection without breaking the energy constraints.

In the following, we propose an algorithm to allocate the numbers of updates allowed to be sent by the sensor nodes based on their topological relations. Our objective is to let the sensor nodes send as many updates as possible subject to the energy constraints.

The allocation algorithm works in iterations. Starting from a zero allocation for all nodes, the algorithm continues to increase their allowable numbers of updates through energy reservation until no further update can be added. Algorithm 2 shows the pesudocode. We maintain an unreserved energy budget u_i for each node i and a total allowable number of updates c_i allocated to node i. u_i is initialized with the energy budget e_i of node i (step 4), and c_i is initialized with 0 (step 5). For any set of nodes $\mathcal{I}$, let $f(\mathcal{I})$ be the subset of nodes in $\mathcal{I}$ whose energy budgets have not been fully reserved, i.e., $f(\mathcal{I}) = \{i \mid i \in \mathcal{I} \text{ and } u_i > 0\}$. In each round of allocation, we first compute the number of updates, x_i, each node i can send under its unreserved energy budget (step 9). Let $\mathcal{C}_i$ be the set of i's children and p_i be i's parent. Then, i is able to send $x_i = u_i/(|f(\{p_i\})|\cdot s + |f(\mathcal{C}_i)|\cdot v)$

Algorithm 2 Update Allocation Algorithm

```
1: set $s \leftarrow$ energy cost to send a data update;
2: set $v \leftarrow$ energy cost to receive a data update;
3: for each $i$ do
4:     set $u_i \leftarrow$ energy budget of node $i$;
5:     initialize its allowable number of updates: $c_i \leftarrow 0$;
6: end for
7: repeat
8:     for each $i$ do
9:         set $x_i \leftarrow u_i/(|f(\{p_i\})|\cdot s + |f(\mathcal{C}_i)|\cdot v)$;
10:    end for
11:    for each $i$ do
12:        set $\Delta_i \leftarrow \min(x_i, x_{p_i})$;
13:    end for
14:    for each $i$ do
15:        set $c_i \leftarrow c_i + \Delta_i$;
16:        set $u_i \leftarrow u_i - \Delta_i \cdot s - \sum_{j\in\mathcal{C}_i} \Delta_j \cdot v$;
17:    end for
18: until $\Delta_i = 0$ for all $i$
```

updates to its parent p_i if p_i's energy budget has not been fully reserved, and is able to receive x_i updates from each child whose energy budget has not been fully reserved. Taking into consideration the energy constraint at i's parent p_i, node i is added an allowable number of updates $\Delta_i = \min(x_i, x_{p_i})$ (step 12), where x_{p_i} is the number of updates p_i is able to receive from i (x_{p_i} is set to infinity if node i is a child of the base station). After incrementing the total allowable number of updates c_i (step 15), the unreserved energy budget of each node is updated according to the number of updates allocated to itself (for sending energy cost) and to its children (for receiving energy cost) before the next round of allocation (step 16). The algorithm terminates when $\Delta_i = 0$ for all nodes (step 18). Since the allocation for each node requires the information of its parent and children only, it is easy to execute Algorithm 2 in a distributed fashion. The algorithm is executed only once at the beginning of data collection, so the associated overhead, amortized over network lifetime, is minimal.

We show an example execution of Algorithm 2 on the tree in Figure 6(a). Assume each node has an energy budget of 12 units, and the costs for a sensor node to send and receive a data update are 1 unit of energy each. Figure 6(b) shows the allocation results of the first round, where each node i is labelled $u_i/x_i/c_i$. Since nodes B and C are the highest degree nodes, their energy budgets are fully reserved in the first round. B, C and their children are each allocated 3 allowable updates. Nodes J, K, L, however, are not constrained by the energy budget of B. Each of them is allocated 6 allowable updates because $x_D = x_E = x_F = 6$. Node A has two children, so it is allocated $12/3 = 4$ allowable updates. Figure 6(c) shows the allocation results of the second round. Since nodes D, E, F each has 3 units of energy left unreserved, nodes J, K, L each is allocated 3 more allowable updates in this round. Node A, on the other hand, has 2 units of energy left unreserved. As a result, it is allocated 2 more allowable updates. At the end of the second round, each node either has fully reserved its own energy budget or its parent has fully reserved the energy budget. Therefore, no more update can be allocated in the third round and the algorithm terminates. It is seen from the final allocation results that nodes A, J, K and L are allowed to send more updates than the highest degree nodes B and C. This helps improve the accuracy of data collected by the base station.

6 Performance Evaluation

6.1 Experimental Setup

We have developed a simulator to evaluate the proposed adaptive data collection strategy. We considered, for the sensor nodes, the energy costs of sending data updates, receiving data updates and acquiring sensor readings. Following [35], [24], Table 1 summarizes the power requirements for different activities of sensor motes.

TABLE 1
Energy Consumption for Different Activities

Activity	Energy Consumption
Transmit a data update	20 nAh
Receive a data update	8 nAh
Acquire a sensor reading	1.08 nAh

We simulated a network of 100 sensor nodes. The network topology was generated as follows. We randomly placed the base station and 100 sensor nodes in a 1×1 area. The sensor nodes were assumed to have a radio transmission range of 0.2. If two sensor nodes were within the radio range of each other, they were considered neighbors in the network connectivity graph. The breadth first search tree rooted at the base station was then computed from the connectivity graph and used as the routing infrastructure for data collection [18], [7]. We have experimented with many randomly generated network topologies and observed similar performance trends. Due to space limitations, we shall only report the results of a sample network topology in this paper. The layout of the topology is shown in Figure 7, where the solid circle represents the base station, the remaining circles represent the sensor nodes and the lines represent the links in the routing tree.

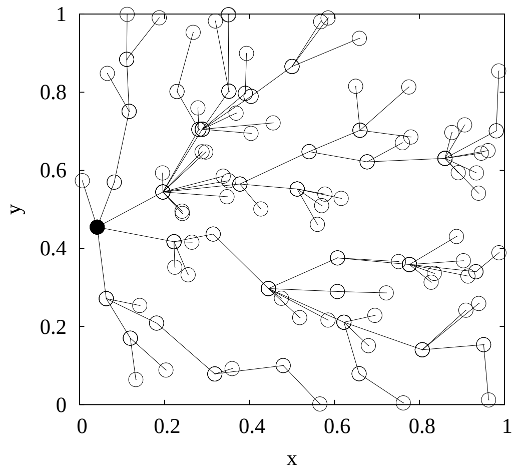


Fig. 7. A Sample Network Topology in the Experiments

We made use of the data provided by the Live from Earth and Mars (LEM) project [8] at the University of Washington to simulate the physical phenomena in the immediate surroundings of sensor nodes. Weather data were collected in the LEM project from several stations in the Washington and Oregon states. We used the temperature (TEMP), solar radiation (SOLAR) and cumulative rain (RAIN) traces logged by the station at the University of Washington from August 2004 to August 2005 in our experiments. Each trace consisted of more than 500,000 readings acquired at a sampling interval of 1 minute. The data in these traces have different changing patterns. Figure 8 shows some representative segments of these traces. Both TEMP and SOLAR data fluctuate over time

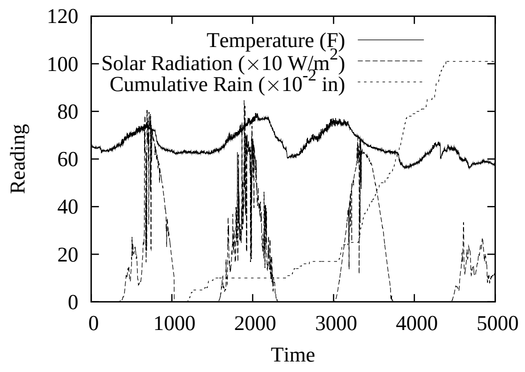


Fig. 8. Sample Data Traces

– their readings are higher in the daytime and lower at night. As seen from Figure 8, the SOLAR data vary more widely than the TEMP data. A remarkable feature of the SOLAR data is that the SOLAR readings remain unchanged regularly because the solar radiation is 0 at night. The RAIN data differ from the TEMP and SOLAR data in that the (cumulative) RAIN readings increase monotonically over time. There also exist periods in which the RAIN readings remain unchanged because there is no rainfall. However, unlike the SOLAR trace, the occurance of these periods is not regular in the RAIN trace. For each of the TEMP, SOLAR and RAIN traces, we extracted 100 different subtraces starting at the same timepoint of different days and associated them with the sensor nodes in our simulated network. Each subtrace contained 20,000 readings. The period between two successive readings in the trace was assumed to be 1 time unit.

We simulated the periodic, adaptive, and optimal data update strategies under different requirements of network lifetime. As defined in Section 3, we measured the root mean square error of the data collected at the base station with respect to the up-to-date sensor readings over the required network lifetime. Recall that the adaptive strategy has three parameters: α, δ and ϵ. The following values were chosen as the default settings: $\alpha = 0.8$, $\delta = 0.1$, and $\epsilon = 0.1$. As shall be shown in Section 6.2, the performance of the adaptive strategy is generally not sensitive to the parameter settings.

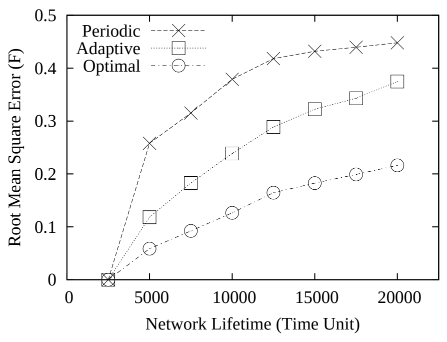
6.2 Performance for Individual Data Collection

First, we evaluate the performance of different strategies for individual data collection in which the base station collects the reading of an individual sensor node. For simplicity, we selected a source sensor node that is a child of the base station in the routing tree (i.e., the source node sends its readings to the base station directly). The initial energy budget of the source node was set to allow it to acquire 2500 local sensor readings and send 2500 data updates to the base station, i.e., 2500 $\times$1.08 + 2500$\times$20 = 5.27$\times 10^4$ nAh.

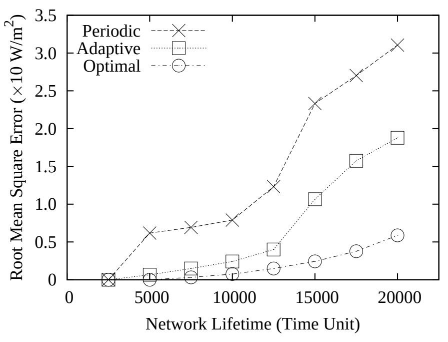
In the periodic strategy, the data update decisions are not made based on sensor readings. Hence, there is no need for a sensor node to acquire more readings than the number of updates it sends for its local readings. So, we instructed the source sensor node to periodically acquire readings and send updates at the maximum rate subject to the energy constraint. For example, given a network lifetime requirement of 10,000 time units, the node acquires and sends one reading every 4 time units, i.e., at times 1, 5, 9, 13, 17, $\cdots$. The adaptive and optimal strategies, on the other hand, are capable of selecting and sending a subset of the acquired readings on the fly based on their relative importance. In the experiments, we instructed the source sensor node to acquire one reading every time unit over the network lifetime (i.e., acquire all readings in the trace). As a result, more energy was spent in acquiring sensor readings in these two strategies than in the periodic strategy. The energy left over was used for data updates.

Figure 9 shows the root mean square error as a function of network lifetime requirement (from 2500 to 20,000 time units) for different strategies and traces. At a network lifetime requirement of 2500 time units, the source sensor node is able to acquire one reading every time unit and send all readings to the base station. The root mean square error, as shown in Figure 9, increases with network lifetime requirement for all strategies. Since the optimal strategy assumes a priori knowledge of all sensor readings over the network lifetime, it is used as a yardstick (lower bound) on root mean square error. As seen from Figure 9, there is a substantial performance gap between the periodic and optimal strategies. The optimal strategy is able to cut down root mean square error by over 50% and 80% for the TEMP and SOLAR traces respectively. It maintains zero error for the RAIN trace throughout the range of network lifetime requirement tested. This strongly motivates the consideration of the relative importance of sensor readings in data collection. By making update decisions on the fly based on sensor readings, the proposed adaptive strategy significantly reduces root mean square error against the periodic strategy. As seen from Figure 9, the relative improvement is up to 54%, 89% and 100% for the TEMP, SOLAR and RAIN traces respectively. This demonstrates the effectiveness of the adaptive strategy in selecting more important sensor readings to update with the base station.

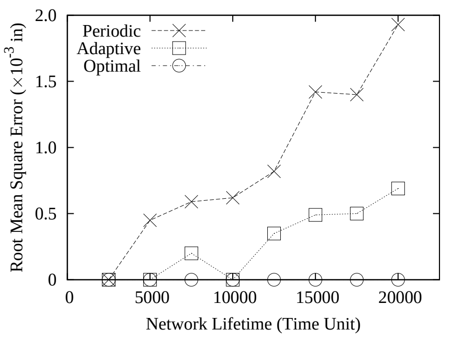
Figure 10 shows the effect of parameter settings in the adaptive strategy for the TEMP trace. We varied each of the three parameters α, δ, and ϵ while keeping the remaining two at their default settings. As shown in Figure 10, the adaptive strategy is not very sensitive to the parameter settings. It produces similar results throughout the ranges of parameter settings tested. These performance trends have been consistently observed for both individual and aggregate data collections over different traces and network topologies. Therefore, we shall report only the experimental results for the default parameter settings in the remainder of this paper.



(a) TEMP Trace

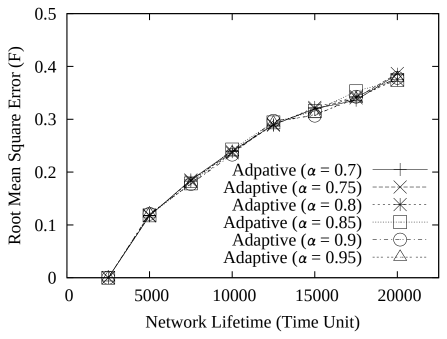


(b) SOLAR Trace

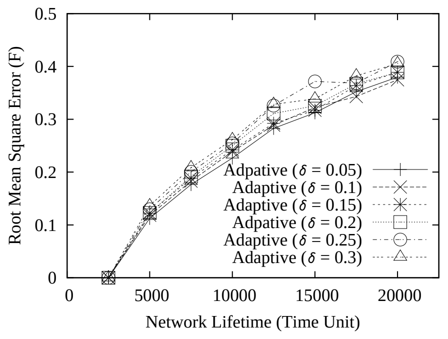


(c) RAIN Trace

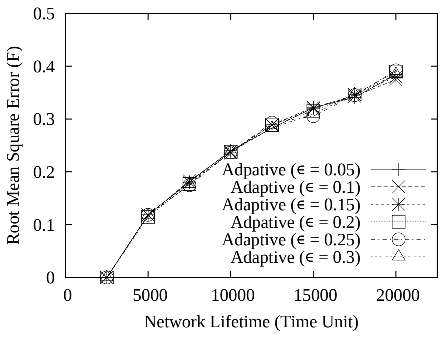
Fig. 9. Root Mean Square Error vs. Network Lifetime Requirement (Individual Data Collection)



(a) Impact of α

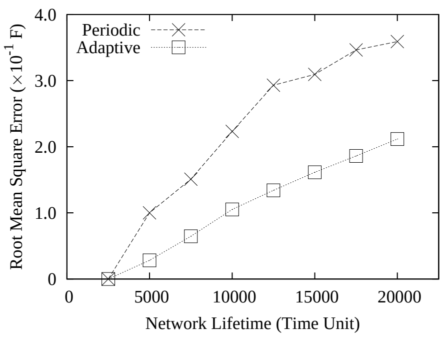


(b) Impact of δ

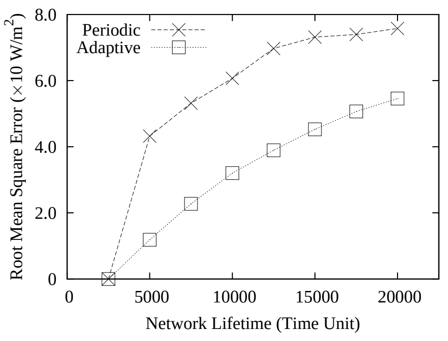


(c) Impact of ϵ

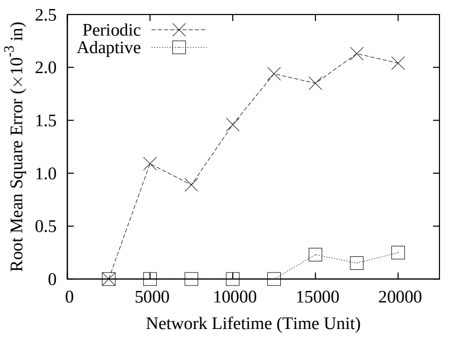
Fig. 10. Performance for Different Parameter Settings in the Adaptive Strategy (TEMP Trace)



(a) TEMP Trace

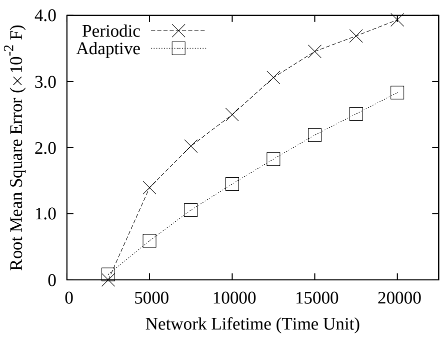


(b) SOLAR Trace

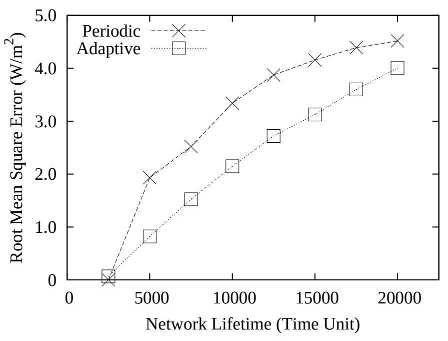


(c) RAIN Trace

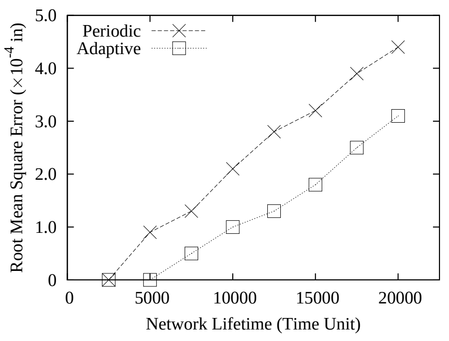
Fig. 11. Root Mean Square Error vs. Network Lifetime Requirement (MAX Aggregation)



(a) TEMP Trace



(b) SOLAR Trace



(c) RAIN Trace

Fig. 12. Root Mean Square Error vs. Network Lifetime Requirement (AVG Aggregation)

6.3 Performance for Aggregate Data Collection

Now, we compare the adaptive strategy against the periodic strategy for aggregate data collection. The same initial energy budget was assigned to each sensor node in the simulated network. The budget was set to allow the bottleneck node (i.e., the highest-degree node in the routing tree, as discussed in Section 5) to acquire 2500 local sensor readings, send 2500 data updates to its parent, and receive 2500 updates from each of its children. For example, the highest-degree node in Figure 7 has 11 children, so the initial energy budget was set at $2500\times1.08 + 2500\times20 + 11\times2500\times8 = 2.727\times10^5$ nAh.

Similar to the experimental methodology in Section 6.2, for the periodic strategy, we instructed each sensor node to acquire readings and send updates at the same rate. We computed the rates for different sensor nodes subject to their energy budgets. The rate of updates sent by a sensor node either equals the parent's rate or is an integral divisor of the parent's rate [7]. For the adaptive strategy, we instructed each sensor node to acquire one reading every time unit (i.e., acquire all readings in the trace). On reserving the energy for acquiring sensor readings over the required network lifetime, the remaining energy budget was used for data updates. The number of updates each sensor node can send to its parent was computed by the allocation algorithm described in Section 5.2.

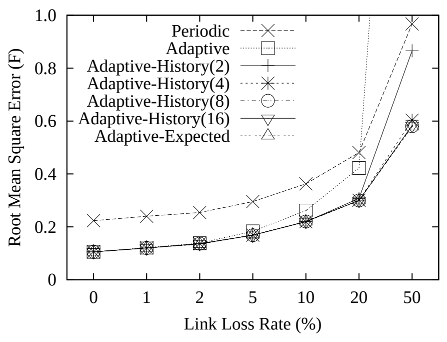
We tested two commonly used aggregates: MAX and AVG, which refer to the maximum and average of the readings at all sensor nodes respectively. Figures 11 and 12 show the root mean square errors of MAX and AVG aggregations respectively. At a network lifetime requirement of 2500 time units, the sensor nodes are each able to acquire one reading every time unit and send all readings to the base station through in-network aggregation. The root mean square error increases with network lifetime requirement for both strategies. By making update decisions based on sensor readings, the adaptive strategy would not send any update when the physical phenomena change slowly. The saved energy is then used to send more updates when the physical phenomena change rapidly. The periodic strategy, on the other hand, does not take sensor readings into consideration. It continues to send updates periodically even when the physical phenomena do not change, thereby wasting much energy. Therefore, as seen from Figures 11 and 12, the adaptive strategy significantly outperforms the periodic strategy. For example, for MAX aggregation, the adaptive strategy cuts down root mean square error by 53%, 47% and 100% for the TEMP, SOLAR and RAIN traces respectively at a network lifetime requirement of 10000 time units.

6.4 Impact of Update Losses

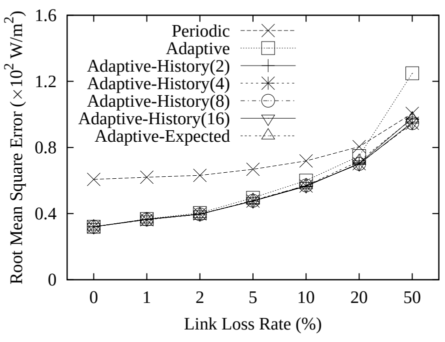
So far, we have evaluated the data collection strategies under reliable transmission of data updates. Now, we investigate the impact of message losses. In addition to the periodic strategy and the original adaptive strategy (Section 4.3), we also simulated the adaptive strategies augmented with the History and Expected methods (Section 4.4) to cope with update losses. They shall be called Adaptive-History and Adaptive-Expected strategies. In our experiments, we randomly determined whether each data update transmitted over a link was lost based on the link loss rate. We conducted 20 different simulation runs for each loss rate value. The average performance of these 20 simulation runs are plotted for performance comparison.

Figures 13 and 14 show the root mean square error as a function of link loss rate for a network lifetime requirement of 10000 time units, where Adaptive-History(k) denotes the Adaptive-History strategy that maintains the last k updated data values at the sensor nodes. Note that all adaptive strategies perform the same in the absence of message losses (the leftmost points in Figures 13 and 14) since the Adaptive-History and Adaptive-Expected strategies degenerate to the original adaptive strategy at a link loss rate of 0. As expected, the quality of data collection deteriorates with increasing link loss rate for all strategies. Comparing the periodic and original adaptive strategies, it is seen that the performance of the original adaptive strategy normally degrades more rapidly than the periodic strategy. This verifies that losing a data update in the adaptive strategy has a more adverse effect on data accuracy than that in the periodic strategy. In particular, at high link loss rates (above 20%), the adaptive strategy often collects data that are even less accurate than the periodic strategy. In contrast, by taking possible update losses into consideration in making update decisions, the Adaptive-History and Adaptive-Expected strategies significantly outperform the original adaptive strategy in terms of root mean square error. Figures 13 and 14 show that the performance of the Adaptive-History strategy improves with increasing k up to 8 (i.e., maintaining a longer history of updated readings at the sensor nodes) and remains quite stable when k exceeds 8. The Adaptive-Expected strategy performs similarly to Adaptive-History(8). Both of them are able to maintain substantial improvement over the periodic strategy in terms of data accuracy even if the link loss rate is as high as 50%. The performance results for other network lifetime requirements have similar trends and are not shown here due to space limitations.

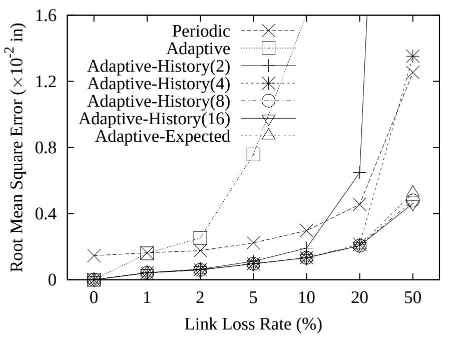
Finally, we study the effect of imperfect loss rate knowledge. In this set of experiments, we used $k = 8$ as the history length maintained by the Adaptive-History strategy. We tested two scenarios in which the link loss rate was overestimated and underestimated relatively by 50% in the augmented adaptive strategies. These two scenarios are identified by "(+)" and "(−)" in Figures 15 and 16 (e.g., Adaptive-History(+)/Adaptive-History(−) denotes the Adaptive-History strategy that overestimates/underestimates the link loss rate relatively by 50% in calculating the difference of a new data value with respect to previously updated data values). As



(a) TEMP Trace

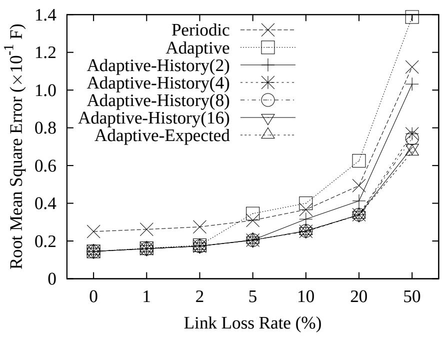


(b) SOLAR Trace

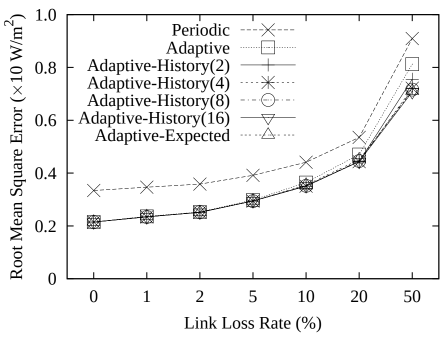


(c) RAIN Trace

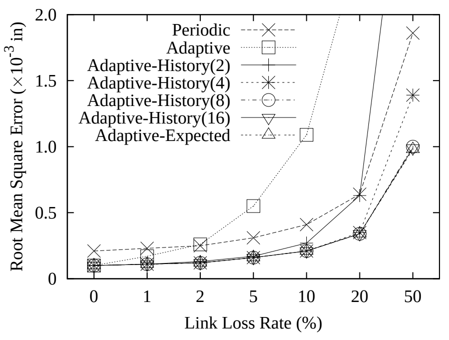
Fig. 13. Root Mean Square Error vs. Link Loss Rate (MAX Aggregation)



(a) TEMP Trace

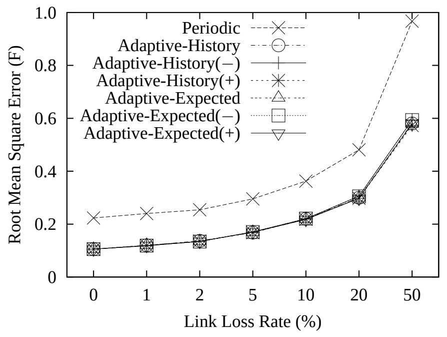


(b) SOLAR Trace

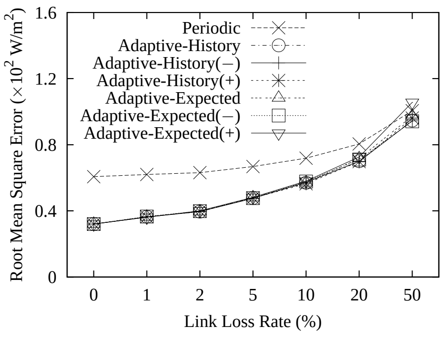


(c) RAIN Trace

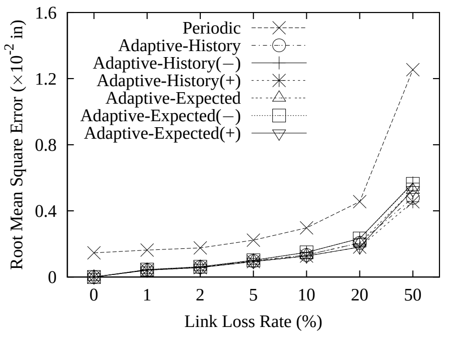
Fig. 14. Root Mean Square Error vs. Link Loss Rate (AVG Aggregation)



(a) TEMP Trace

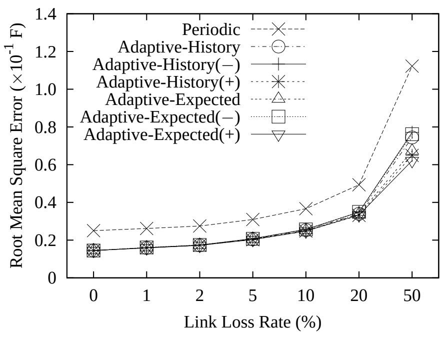


(b) SOLAR Trace

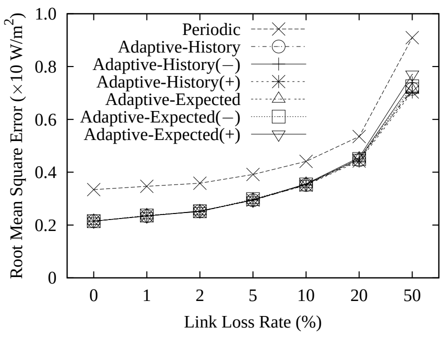


(c) RAIN Trace

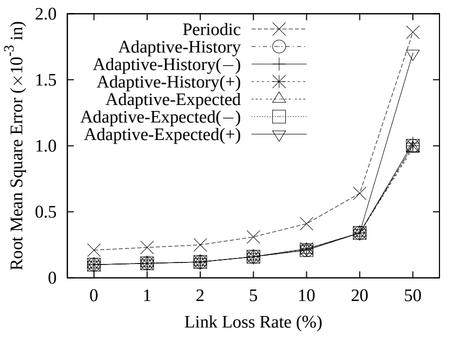
Fig. 15. Performance Results in the Absence of Perfect Loss Rate Knowledge (MAX Aggregation)



(a) TEMP Trace



(b) SOLAR Trace



(c) RAIN Trace

Fig. 16. Performance Results in the Absence of Perfect Loss Rate Knowledge (AVG Aggregation)

seen from Figures 15 and 16, the Adaptive-History and Adaptive-Expected strategies are generally not sensitive to the error in the knowledge of link loss rate. They continue to outperform the periodic strategy in terms of the accuracy of data collection even in the absence of perfect loss rate knowledge.

7 Conclusion

In this paper, we have studied adaptive data collection strategies for lifetime-constrained wireless sensor networks. Instead of collecting sensor readings periodically, the relative importance of the readings is considered in data collection. The sensor nodes send data updates to the base station when the new readings differ more substantially from the previous ones. We have developed adaptive strategies for both individual and aggregate data collections. To make full use of the energy budgets, we have designed an algorithm to allocate the numbers of updates allowed to be sent by the sensor nodes based on their topological relations. We have also presented two methods to cope with message losses in wireless transmission. Experimental results using real data traces show that, compared to the periodic strategy, adaptive strategies significantly improve the accuracy of data collected by the base station over the network lifetime.

Acknowledgments

This work was supported in part by a grant from Nanyang Technological University (Project No. RG47/06). Jianliang Xu's work was supported in part by the Research Grants Council of Hong Kong under Projects HKBU211505, HKBU211307, FRG/05-06/II-65.

References

[1] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Proc. MDM'01*, Jan. 2001, pp. 3–14.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 34–40, June 2004.

[4] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 45–55, January–March 2004.

[5] G. J. Pottie and W. J. Kaiser, "Wireless intergrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[6] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. ACM SenSys'04*, Nov. 2004, pp. 239–249.

[7] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[8] "Live from Earth and Mars (LEM) Project," http://www-k12.atmos.washington.edu/k12/grayskies/.

[9] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy efficient data collection in distributed sensor environments," in *Proc. IEEE ICDCS'04*, Mar. 2004, pp. 590–597.

[10] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proc. IEEE ICDE'06*, Apr. 2006.

[11] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: On energy-efficient continuous monitoring in sensor networks," in *Proc. ACM SIGMOD'06*, June 2006, pp. 157–168.

[12] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, "TiNA: A scheme for temporal coherency-aware in-network aggregation," in *Proc. ACM MobiDE'03*, Sept. 2003, pp. 69–76.

[13] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proc. ACM SIGMOD'03*, June 2003, pp. 563–574.

[14] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Processing approximate aggregate queries in wireless sensor networks," *Information Systems*, vol. 31, no. 8, pp. 770–792, Dec. 2006.

[15] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *Proc. ACM SIGMOD'06*, June 2006, pp. 145–156.

[16] M. Li, Y. Liu, and L. Chen, "Non-threshold based event detection for 3d environment monitoring in sensor networks," in *Proc. IEEE ICDCS'07*, June 2007.

[17] M. B. Greenwald and S. Khanna, "Power-conserving computation of order-statistics over sensor networks," in *Proc. ACM PODS'04*, June 2004, pp. 275–285.

[18] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: New aggregation techniques for sensor networks," in *Proc. ACM SenSys'04*, Nov. 2004, pp. 188–200.

[19] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, "Holistic aggregates in a networked world: Distributed tracking of approximate quantiles," in *Proc. ACM SIGMOD'05*, June 2005, pp. 25–36.

[20] J. Xu, X. Tang, and W.-C. Lee, "A new storage scheme for approximate location queries in object tracking sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 2, pp. 262–275, Feb. 2008.

[21] X. Tang and J. Xu, "Optimizing lifetime for continuous data aggregation with precision guarantees in wireless sensor networks," *IEEE/ACM Transactions on Networking*, accepted to appear, 2008.

[22] M. Wu, J. Xu, X. Tang, and W.-C. Lee, "Top-k monitoring in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 962–976, July 2007.

[23] Y. Kotidis, "Snapshot queries: Towards data-centric sensor networks," in *Proc. IEEE ICDE'05*, Apr. 2005, pp. 131–142.

[24] G. Hartl and B. Li, "infer: A bayesian inference approach towards energy efficient data collection in dense sensor networks," in *Proc. IEEE ICDCS'05*, June 2005, pp. 371–380.

[25] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proc. IEEE ICDE'04*, Mar. 2004, pp. 449–460.

[26] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proc. ACM SenSys'04*, Nov. 2004, pp. 250–262.

[27] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *Proc. IEEE ICDE'03*, Mar. 2003, pp. 429–440.

[28] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Compressing historical information in sensor networks," in *Proc. ACM SIGMOD'04*, June 2004, pp. 527–538.

[29] S. Kim, R. Fonseca, and D. Culler, "Reliable transfer on wireless sensor networks," in *Proc. IEEE SECON'04*, Oct. 2004, pp. 449–459.

[30] A. Woo and D. Culler, "Evaluation of efficient link reliability estimators for low-power wireless networks," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-03-1270, 2003.

[31] C. Buragohain, D. Agrawal, and S. Suri, "Power aware routing for sensor databases," in *Proc. IEEE INFOCOM'05*, Mar. 2005, pp. 1747–1757.

[32] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated, adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[33] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM SenSys'04*, Nov. 2004, pp. 95–107.

[34] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks," in *Proc. USENIX OSDI'02*, Dec. 2002, pp. 131–146.

[35] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, "Wireless sensor networks for habitat monitoring," in *Proc. ACM WSNA'02*, Sept. 2002, pp. 88–97.