Mobile Filter: Exploring Filter Migration for Error-Bounded Continuous Sensor Data Collection

Dan Wang, *Member, IEEE,* Jianliang Xu, *Senior Member, IEEE,* Feng Wang, *Student Member, IEEE,* and Jiangchuan Liu, *Senior Member, IEEE*

***Abstract*—In wireless sensor networks, filters, which suppress data update reports within predefined error bounds, effectively reduce the traffic volume for continuous data collection. All prior filter designs, however, are *stationary* in the sense that each filter is attached to a specific sensor node and remains stationary over its lifetime. In this paper, we propose *mobile filter*, a novel design that explores migration of filters to maximize overall traffic reduction. A mobile filter moves upstream along the data collection path, with its residual size being updated according to the collected data. Intuitively, this migration extracts and relays unused filters, leading to more proactive suppressing of update reports.**

While extra communications are needed to move filters, we show through probabilistic analysis that the overhead is outrun by the gain from suppressing more data updates. We present an optimal filter migration algorithm for a chain topology. The algorithm is then extended to general multi-chain and tree topologies. Extensive simulations demonstrate that, for both synthetic and real data traces, the mobile filtering scheme significantly reduces data traffic and extends network lifetime against a state-of-the-art stationary filtering scheme. Such results are also observed from experiments over a Mica-2 sensor network testbed.

***Index Terms*—Sensor Network, Data Collection, Mobile Filter.**

I. Introduction

Wireless sensor networks have recently been used for many applications, such as habitat monitoring, military surveillance, and terrain discovery, where traditional wired/wireless networks are not appropriate or available. The primary task of a sensor network is to continuously collect the sensed data in the operational field, so that the field's properties of interest can be monitored. In this paper, we are interested in continuously gathering data distribution of the sensor field. For example, we are interested in the following queries:

Query 1: *Get the temperature distribution of the sensor field every other hour for the next 6 months.*

Query 2: *Monitor the population of wildlife at difference places every 4 hours for the next 12 months.*

Such complex queries, though clearly more difficult to answer, reveal richer information than a simple aggregate such as sum or average. For example, a (consistent) change of the population distribution of the wildlife may be an indication of the change of the surrounding environment [1].

D. Wang is with The Hong Kong Polytechnic University, Email:csdwang@comp.polyu.edu.hk; J. Xu is with Hong Kong Baptist University, Email:xujl@comp.hkbu.edu.hk; F. Wang and J. Liu are with Simon Fraser University, Canada, Email:{fwa1,jcliu}@cs.sfu.ca

A preliminary version of this paper appeared in Proc. the 28th International Conference on Distributed Computing Systems (IEEE ICDCS'08).

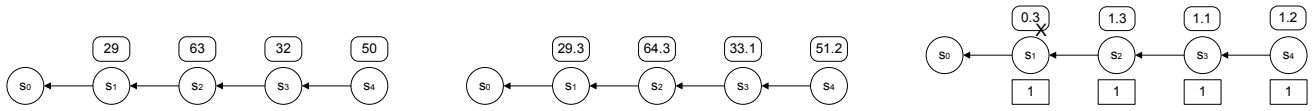
In sensor networks, energy is a severely limited resource, and communication dominates energy consumption. To obtain the distribution information aforementioned, the base station needs to continuously collect data from each sensor node. This is obviously very energy expensive. Fortunately, approximate results are usually acceptable as long as the error is bounded by a certain threshold. Thus, a trade-off between energy consumption and data quality can be explored. Data filtering, by exploring temporal data correlation, is an effective in-network processing scheme towards this goal. Intuitively, if the difference between the new reading and the previous reported reading in a sensor node is small, the node should not report the new reading. Olston *et al*. [2] first generalizes this idea to a filter design for continuous data collection. In their work, a filter is allocated to each sensor node such that the total filter size obeys the user-specified error bound. In each round of data collection, a node will *suppress* its data update report if the difference from the previous report is less than its filter size. There have been a flourish of follow-ups with more intelligent filter allocation strategies (e.g., [3][4]).

All these prior filter designs, however, are *stationary* in the sense that each filter is attached to a specific node and remains stationary during a round of data collection. Thus, unused filters in the current round of data collection might be wasted, limiting the filtering capability.

In this paper, we propose *mobile filter*, a novel design that explores migration of filters to reduce network traffic for error-bounded data collection. A mobile filter moves upstream along the data collection path, with its residual size being updated according to the collected data. Intuitively, this migration extracts and relays unused filters, leading to more proactive suppressing of data reports. While extra communications are needed to move filters, we show through probabilistic analysis that the overhead is outrun by the gain from suppressing more data transmissions. The overhead can be further reduced by piggy-backing the filter information in data update reports[1].

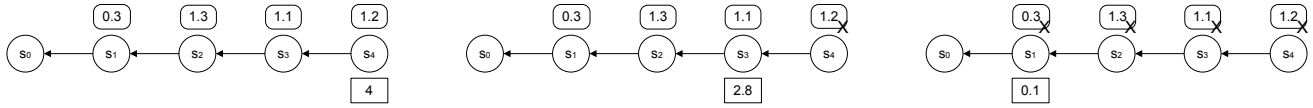
An Example. To illustrate the effect of our mobile filtering scheme, we compare it with a basic stationary filtering scheme in a toy example in Figs. 1 and 2. Consider a sensor network of chain topology (s_4 through s_0). The base station s_0 needs to record the data for each sensor node in each round (or use the previously recorded data if it does not hear from the node). Assume L_1 distance is used for bounding data errors

[1]The filter information is a few bytes. It can be accommodated in an update report within a data packet (the packet size is 60 bytes for a Mica-2 mote).



(a) Previously reported data readings. (b) Data readings of the current round. (c) Stationary filter suppresses one data report from s_1.

Fig. 1. An example of a stationary filtering scheme. Total user allowed filter size (error bound) is 4. Node s_0 is the base station.



(a) Mobile filtering scheme at the start of the current round. (b) Mobile filter moves and suppresses data reports. (c) In total, all four data reports are suppressed.

Fig. 2. An example of a mobile filtering scheme. Total user allowed filter size (error bound) is 4. Node s_0 is the base station.

[5],[2] and the total user-allowed filter size (error bound) is 4. The previously reported reading of each sensor is shown in Fig. 1(a). In the current round, each sensor acquires a new reading, as shown in Fig. 1(b). Using the stationary filtering scheme, filters are allocated to each node and one possible (uniform) allocation is shown below each sensor in Fig. 1(c). We can see that the stationary filters can suppress only one data update report from s_1. All other updates need to be reported, and overall it incurs 2+3+4 = 9 link messages. As a contrast, we now employ the mobile filtering scheme for the same scenario. The entire filter is assigned to s_4 at the beginning of the current round, as shown in Fig. 2(a). The filter suppresses s_4's data update report and the residual filter moves upstream as shown in Fig. 2(b), which further suppresses s_3's update report. In general, the filter suppresses update reports while it moves along the path. Eventually, all four update reports are suppressed, as shown in Fig. 2(c). The total number of link messages incurred is 3 (for the mobile filter transmission).

Intrinsically, one may consider the filter (i.e., the error bound allowed by the user) a valuable resource that can be exploited for conserving energy. In the stationary filtering scheme, each filter has to make an independent decision about data suppressing. The filters have no knowledge of how other filters are used by other sensor nodes. Therefore, the utilization of the filter resource is not optimized; for example, the filters on s_2 through s_4 are wasted in the above example. The mobile filtering scheme, on the other hand, is able to adapt to the current data readings and allocates filters on the fly to optimize the utilization. This intuition will be formalized in our analysis.

There are, however, many design issues left to be addressed; for example, a formal error bound model is needed for the data collection and filtering scheme; filter migration and data filtering algorithms should be developed to maximize the overall traffic reduction. We shall address these issues in detail in the rest of this paper. Our contributions are summarized as follows. First, we propose a novel mobile filtering scheme. Second, we develop an optimal filter migration and data filtering algorithm for a chain topology. We extend our algorithm to general multi-chain and tree topologies for sensor data collection. Third, our scheme is validated through extensive simulations using both synthetic and real-world traces, as well as experiments on a Mica-2 sensor testbed.

The rest of the paper is organized as follows. We review related work in Sec. II. The system model is described in Sec. III. Sec. IV is devoted to our mobile filter design. We show the simulation results in Sec. V, followed by our preliminary experimental results using Mica-2 sensors in Sec. VI. Finally, Sec. VII concludes the paper and discusses future work.

II. Related Work

Wireless sensor networks have been extensively studied in recent years; a survey can be found in [6]. Many sensor networks are designed for continuous data collection applications over a long period of time. Real-world examples include the sensor network deployed on Great Duck Island [7] to monitor the habitat of birds, ZebraNet in Africa [8] to monitor the behavior of wildlife, and the volcano monitoring system [9].

As sensor nodes are usually constrained by a limited power supply, energy efficiency is a key consideration in sensor network designs. A pioneer work [10] has suggested various in-network processing techniques to reduce the network traffic. One effective in-network processing scheme is in-network aggregation. By exploring the query's characteristics, an intermediate sensor node can compute a partial aggregate of its own value and the values of the downstream nodes before reporting to its upstream nodes. A number of aggregate functions, such as MAX, MIN, SUM, AVG, and MEDIAN, have been studied [11][12][13]. Another effective in-network processing scheme is to make use of spatial data correlation, and the studies include clustering [14], sampling [15] and overhearing [16]. Our work falls into an orthogonal category where temporal data correlation is explored [17], and we are interested in *non-aggregate* data. Non-aggregate data can provide a fine-grained analysis of the phenomena in the sensor field, which is requested by many applications [18][19][20]. For example, in the Sonoma Redwoods project [18], the biologists would like to receive detailed data for model analysis and hypotheses testing. These in-network processing techniques can also be combined to achieve higher energy efficiency; see [3][21].

To explore temporal data correlation, data filtering is a commonly used technique that trades data quality for energy efficiency. In [2], a filter is allocated to each sensor node where the total filter size is constrained by the user error bound. The filters shrink periodically and the server will re-allocate the left-over error bound to the sensor nodes based on *burden*

[2] L_1 distance is the sum of the absolute difference over all paired values in the two datasets. Note however that the general framework of mobile filtering does not depend on specific data error models.

scores. The burden score of a node is calculated based on a set of parameters involving the number of update packets generated by the sensor node since the last filter reallocation, the current filter size, and the data reporting cost. The work in [3] further incorporates in-network aggregation into filter designs, where an intermediate node computes partial aggregates from its descendants. A more intelligent filter adjustment scheme is proposed in [4]. In contrast to the previous studies where the filters are reallocated mainly based on data changing patterns, the optimization in [4] explicitly takes the residual energy of the sensor nodes into consideration.

Note that filter re-allocation is a costly operation and is done infrequently so that the cost can be amortized. Thus, though these prior studies [3][2][4] have different filter (re)allocation mechanisms, they share a common assumption: the filter attached to a specific sensor node will be used for suppressing data reports for this node only. In other words, filters are stationary and only data traverse inside the network. The novelty of our work is that we allow the filters to move in each round of data collection, and we show that given the same error bounds, the migration of filters suppresses significantly more data transmissions, making the system more energy efficient.

III. System Model

In our system, the readings from individual sensors are periodically collected by the base station to evaluate complex distribution queries; we call each data collection a *round*. In the first round, all the sensor nodes report their readings. In the subsequent rounds, the sensor nodes report readings that are not suppressed. If the base station does not receive a report from a sensor node, its previously reported reading will be treated as collected data and used for current query evaluation.

A. The Error Bound Model

To facilitate our presentation, in this paper we employ L_1 distance as the error bound model. Specifically, let the true readings of the sensor nodes be $x_1, x_2, \ldots, x_N$ and let the readings collected by the base station be $x'_1, x'_2, \ldots, x'_N$; the L_1 distance is then $L_1 = \sum_{i=1}^{N} |x_i - x'_i|$. If the user-specified precision requirement is E, the error-bounded data collection must guarantee $L_1 = \sum_{i=1}^{N} |x_i - x'_i| \leq E$. L_1 distance is commonly used to measure the distance between complex distributions [5]. The smaller is the L_1 distance of two distributions, the closer are the two distributions. More formally, if the L_1 distance is small, any event will happen with similar probability in the two distributions.

It is worthwhile to note that our mobile filtering scheme is not limited to the L_1 model. It is straightforward to show that it can work with L_k distance where $L_k = \sqrt[k]{\sum_{i=1}^{N} |x_i - x'_i|^k}$ for any $k = 0, 1, 2, \cdots$. In general, the mobile filtering scheme is workable for any aggregate error bound model where the overall error bound is a function of the error introduced from individual sensor node. Additional examples are weighted L_k distance, KL-divergence, etc.

To bound the error of data collection, data filters are installed (either statically or dynamically) on sensor nodes in the network. Each filter is associated with a deviation bound (hereafter referred to as *filter size*) and the total filter size should not exceed the bound E. During a round of data collection, a sensor node reports its data to the base station only if the deviation between the current reading and the last reported reading exceeds the filter size.

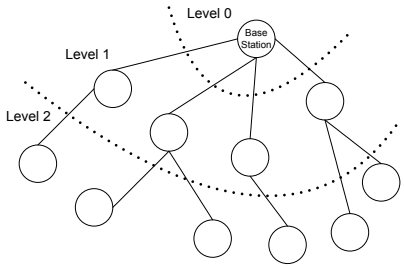


Fig. 3. Underlying communication/routing structure.

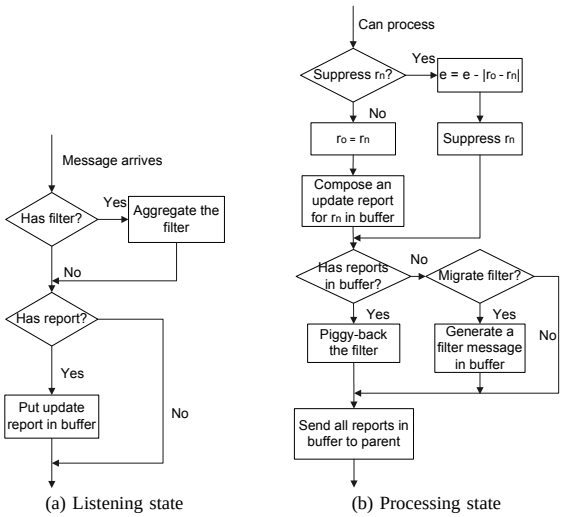
We would like to remark that there are applications that explicitly specify an error bound for each sensor node (e.g., tolerating the error of the readings of each node to be bounded within 1), instead of an aggregate error bound. For these applications, the filter size of each node can be just set as requested and no filter allocation/re-allocation is necessary. Notice that the stationary filtering scheme also aims to work with aggregate error bounds and periodically (re)-allocates filter sizes based on system workload. We emphasize that both the stationary and mobile filtering schemes do not control the user error model and the error bound. Given the error model and the error bound, they both try to optimize the system performance and we will show that mobile filtering outperforms the stationary filtering scheme.

B. Data Collection Model

For each round, we use a data collection model similar to TAG [11]. The underlying network is structured as a tree and the data is propagated from the leaf nodes to the root. Specifically, each sensor node is associated with a *level* in the tree, which indicates the number of hops the node is away from the base station (i.e., the root) (see Fig. 3). To avoid transmission collisions, the time is divided into slots, and a sensor node is kept in a *sleeping* state for most of the time in a round. In each time slot, starting from the leaf level, the sensor nodes at one level are activated to enter into a *processing* state, and the sensor nodes at the level with one hop closer to the sink enter into a *listening* state. Upon being in the processing state, a sensor node acquires a new reading, processes it together with the data received from its children, and possibly transmits some data to its parent node. A sensor node in the listening state monitors the wireless channel and buffers all incoming packets for further processing. Various synchronization techniques can facilitate this state transition [22][11]. A round of data collection is completed when the processing state propagates to the root. In our data collection, we assume that reliable transmission protocol [23] is used for underlying routing and no packet is lost.

IV. Mobile Filtering: Design and Optimization

The objective of our mobile filtering scheme is to minimize the total data transmission cost while maintaining the user specified error bound. In this section, we first outline a



(a) Listening state (b) Processing state

Fig. 4. Operations of a sensor node in each round.

practical mobile filter design. We then analyze this scheme for a chain routing topology. We show that it outperforms stationary filtering and derive an optimal offline migration strategy, together with an efficient online heuristic. We further extend the algorithm to multi-chain and tree topologies.

A. Operations of Mobile Filters

In stationary filtering schemes, each filter only needs to suppress the newly sensed data if it can. In mobile filtering schemes, a mobile filter may not suppress a newly sensed data in the sensor node it travels. The intuition here is that suppressing the data consumes its filter size and may restrict the mobile filter's ability to suppress more data updates upstream. In addition, a mobile filter needs to decide whether to travel to the next sensor node. The intuition here is that if the residual filter size is small, a mobile filter may not travel further to reduce the overhead it incurs.

Formally, in each round of data collection, each sensor node s first senses a new reading r_n and then operates as follows. In the listening state, s receives message(s) sent from its children. Let e be its current filter size (we will show later how this size is initialized). If the incoming message contains an unused filter e_{in}, s updates the filter as $e = e + e_{in}$. If the message contains an update report, it is buffered for forwarding later. Detailed operations for this stage are shown in Fig. 4(a).

When the sensor node s enters into its processing state, a *data filtering strategy* first decides whether the current filter is to suppress r_n. Let r_o be the last reading reported to the base station. If r_n is suppressed, a filter size of $|r_o - r_n|$ is consumed and the residual filter size is updated to $e = e - |r_o - r_n|$. Otherwise if r_n is not suppressed, an update report is composed and buffered, and the residual filter size remains e. The second decision is whether to migrate the residual filter upstream. If there are update reports (either its own or the reports forwarded for its descendants) to be sent to the parent, the residual filter can be piggy-backed. Otherwise, a *filter migration strategy* will decide whether to migrate the residual filter using a separate message. Finally, the sensor node forwards all update reports in the buffer to its parent. Detailed operations for this stage are summarized in Fig. 4(b).

By the end of each round of data collection, each node resets the filter size. It is easy to see that under this operational model, the sum of the data changes suppressed does not exceed the total error bound in each round of data collection. Thus, the user-specified precision requirement is guaranteed. The remaining task is to design data filtering and migration strategies so as to minimize the overall data transmission cost.

B. Filter Migration in Chain Topology

We start our discussion with a simple chain topology. We first show that the mobile filter should initially be placed at the leaf node.

Theorem 1: For a chain topology, the filter should be allocated as a whole to the leaf sensor node in order to minimize the total communication cost.[3]

Proof: Denote the sensor nodes on a chain by s_0, s_1, ..., s_{N-1}, s_N, where s_0 is the base station and s_N is the leaf sensor node. We prove the theorem by induction.

Assume that we do not allocate any filter to s_{N-1}. If there is a data change for s_N, the update must be reported from s_N to s_0. The cost of this update is N. The overall communication cost should also include the cost with a filter installed on the sub-chain from s_{N-1} to s_0.

If we allocate any fraction of filter size F to the leaf node s_N, there are two cases: 1) migrate the filter to s_{N-1} without suppressing the update at s_N. Since this filter migration can be piggy-backed by the update report, it does not incur extra cost. The overall cost is also the sum of s_N's update cost (i.e., N) and the cost with a filter of size F installed on the sub-chain from s_{N-1} to s_0. 2) suppress the update at s_N. Thus, the cost of allocating a filter size of F to s_N is the minimum of the costs of 1) and 2). Notice that 1) has the same cost with allocating the buffer to s_{N-1}. Therefore, allocating F to s_N will result in no worse performance than allocating it to s_{N-1}. It is easy to see that we can do induction both in terms of F and s_{N-1}. As such, this completes the proof. ■

Following this theorem, given a total error bound of E, the filter size allocated to the leaf node is E and the filter sizes allocated to all other nodes are zero. The filter then follows the operations described in Section IV.A. By the end of each round, the leaf node resets the filter size to E and all other nodes reset the filter sizes to zero. Note that resetting the filter sizes does not incur any communication cost.

1) Mobile versus Stationary Filtering: We now give a formal cost analysis of the mobile filtering scheme, assuming the data changes follow a standard normal distribution. While this analysis is necessarily simplified, it provides a clearer view of the benefit of mobile filtering. That is, in stationary filtering scheme, the total error bounds have to be divided among all sensor nodes; and each node may have small filters, making the probability of filter size violation high. The mobile filtering scheme, on the other hand, fully exploits the filter size.

Let E be the total filter size, and $X_i (i = 1, 2, \cdots, N)$ be the random variable for the sensor value change of node s_i. Assume that the change for each sensor node is i.i.d. and that

[3]Here, we assume that the sensor readings always change between two consecutive rounds of data collection.

X_i follows a normal distribution of $N(0,1)$. Without loss of generality, we consider the case where only the upper bound of the filter is violated.

For stationary filtering, each node will be assigned a filter size of $\frac{E}{N}$ under uniform allocation. The probability that the filter is violated at node s_i is $p_i = Pr[X_i > \frac{E}{N}] = 1 - Pr[X_i \leq \frac{E}{N}] = \frac{1}{2}(1-\text{erf}\left(\frac{E}{\sqrt{2}N}\right))$. Define an indicator random variable Y_i such that

$$Y_i = \begin{cases} 1 & \text{if } X_i > \frac{E}{N}; \\ 0 & \text{otherwise.} \end{cases}$$

We have $E[Y_i] = p_i \times 1 + (1-p_i) \times 0 = p_i = \frac{1}{2}(1 - \text{erf}\left(\frac{E}{\sqrt{2}N}\right))$. Given node s_i's update cost of i, the expected transmission cost is $E[\sum_{i=1}^{N} Y_i \times i] = E[Y_i]\sum_{i=1}^{N} i = \frac{N(N+1)}{2}E[Y_i]$.

For mobile filtering, the filter migrates upstream and suppresses the data reports as long as it can and the filter migration stops when the residual filter size is not enough to suppress a data update. The probability that the filter is violated at node s_i is $p_i = Pr[X_i > E - \sum_{j=i+1}^{N+1} X_j] = Pr[\sum_{j=i}^{N+1} X_j > E]$ (Define $X_{N+1} = 0$). Let $Z_i = \sum_{j=i}^{N} X_j$. Since X_i's are i.i.d, Z_i is also a normal distribution of $N(0, N-i+1)$. Define an indicator random variable Y_i such that

$$Y_i = \begin{cases} 1 & \text{if } Z_i > E; \\ 0 & \text{otherwise.} \end{cases}$$

We have $E[Y_i] = p_i = \frac{1}{2}(1 - \text{erf}\left(\frac{E}{\sqrt{2}(N-i+1)}\right))$. The expected cost of mobile filtering is then $E[\sum_{i=1}^{N} Y_i \times i] = \sum_{i=1}^{N} E[Y_i]i$. If the filter migration is not piggy-backed with data reports, there is at most an additional cost of N.

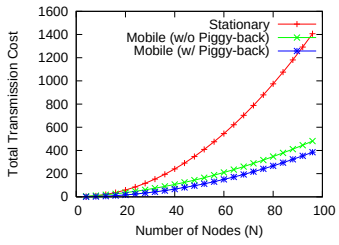


Fig. 5. Expected cost as a function of N. $E = \frac{1}{2}N$.

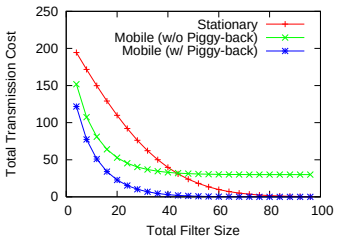


Fig. 6. Expected cost as a function of the total filter size E. $N = 30$.

In Figs. 5 and 6, we plot numerical results for the two filtering schemes. For mobile filtering, the results for two versions are shown, namely, with and without piggy-back. Their performance differs by at most N. In Fig. 5, the total error bound is fixed to $\frac{N}{2}$ and a filter of size 0.5 is attached to each node for stationary filtering. Since the data changes follow a standard normal distribution, in this setting each filter will suppress the updates with a probability of approximately 40%. We can see that mobile filtering greatly outperforms stationary filtering. We also see that when N increases, the cost of stationary filtering increases much faster than that of mobile filtering, implying that mobile filtering is more scalable.

In Fig. 6, we fix the number of sensors to $N = 30$ and vary the total error bound E from 0 to 100. With piggy-back, mobile filtering performs better than stationary filtering for all cases tested. Even without piggy-back, only after $E = 50$ does mobile filtering perform worse than stationary filtering. Note that when $E = 50$, each node obtains a filter size of $50/30 = 1.67$ for stationary filtering, which implies that the probability of a new reading being suppressed is as high as 90%. In other words, the error bound is extremely large in this case, which may not provide meaningful results and is not desirable for most applications.

We state again that data change distributions depend on specific applications. One may question the performance for the distributions where there are a few data changes that are significantly large; and these data changes may consume the filter size if the mobile filter suppresses the data update reports as long as it can. This is why more advanced data filtering and filter migration strategies are needed. The algorithm we show in next subsection will advisably omit these few data changes so as to suppress significantly more data updates; and thus fits for all distributions. In our experiments, we also use real world traces that match a wide range of application scenarios.

2) Filter Migration and Data Filtering Strategies: Recall that our objective is to minimize the number of update reports transmitted in the network given a total filter size of E. In this section, we first develop an optimal offline solution (through dynamic programming) with all data changes known a priori. Let i be the distance (in terms of hops) between the ith node and the base station. Let v_i be the data change (against the last reported value) at sensor node s_i, and e be the residual filter size. Let $G_i(e)$ be the gain from placing a filter of size e at sensor node s_i, which represents the cost difference between suppressing the data update at node s_i and migrating the residual filter size upstream.

$$G_i(e) = \max \begin{cases} i & & (1) \\ G_{i-1}(e), & \text{piggy-back} & (2) \\ i + G_{i-1}(e - v_i), & \text{piggy-back} & (3) \\ i + G_{i-1}(e - v_i) - 1, & \text{no piggy-back} & (4) \end{cases}$$

There are four possible choices that s_i can execute, as shown in (1)-(4). With the first choice, the data update is suppressed, and the residual filter is not sent upstream. With the second choice, the data update is not suppressed and reported to the base station. In this case, the unused filter size e is piggy-backed upstream to node s_{i-1}. With the third choice, the data update is suppressed. The gain consists of two parts. The first part is a saving of i transmissions for this data update. The second part is a potential gain of $G_{i-1}(e - v_i)$ when the residual filter with size $e - v_i$ migrates to the parent. The fourth choice is similar to the third except that the filter migration is not piggy-backed with the data reports (of s_i's descendants) and incurs one extra cost for sending this filter upstream to sensor node s_{i-1}. A sensor node should select the one with the highest gain among the four choices.

We also initialize the $G_i(\cdot)$ function for special cases:

$$\begin{array}{lll} \forall i, \quad G_i(0) = 0, & & (5) \\ \forall i, \quad G_i(< 0) = -\infty, & & (6) \\ \forall e, \quad G_0(e) = 0, & & (7) \\ \forall e, \quad G_1(e) = 0, & \text{no piggy-back} & (8) \\ \forall e, \quad G_1(e) = 1, & \text{piggy-back} & (9) \end{array}$$

where condition (5) means there is no gain if the filter is used up; condition (6) states that negative filters are strictly prohibited; condition (7) states that there is no gain if the filter has arrived at the base station; and conditions (8) and (9) specify the gains for node s_1.

Algorithm CalGain ()

$G_i(e,+)$: the gain at node i with residual filter size e with piggy-back; $G_i(e,-)$: the gain without piggy-back.

1 Initialization;
2 **for** $\forall i, e, \{+,-\}$

3 $$G_i(e,+) = \max \begin{cases} i + G_{i-1}(e - v_i, +), \\ G_{i-1}(e,+) \end{cases}$$

4 $$G_i(e,-) = \max \begin{cases} i + G_{i-1}(e - v_i, -) - 1, \\ G_{i-1}(e,+), \\ i \end{cases}$$

5 **end for**

Output: $G_N(E,-)$ and the filter migration and data filtering strategies.

Fig. 7. Calculate Gain Algorithm.

$G_i(\cdot)$ can then be iteratively calculated using dynamic programming, see Fig. 7. Note, however, that this optimal algorithm needs prior information about the data changes, which is difficult to obtain. We thus develop a greedy online heuristic as follows. Let T_R and T_S be two thresholds used for filter migration and data filtering. If the residual filter size is smaller than T_R, the filter is not sent upstream unless the filter is piggy-backed; if the data update at a sensor is greater than T_S, the filter will not suppress this update. Intuitively, a small residual filter T_R means that the chance of suppressing upstream data reports is small, and thus the filter should not be sent upstream. The threshold T_S means that if a data change is very large, suppressing this update will significantly reduce the chance of suppressing future reports. As such, even if the current residual filter size is able to suppress this update, it leaves the opportunities to suppress updates upstream.

We will examine the performance of this greedy heuristic against the optimal algorithm as well as the impact of T_R and T_S by simulation in Section V.

C. Filter Migration in Multi-Chain Trees

The chain structure provides us with a basic understanding of mobile filtering. In this section, we consider a more general routing structure, a multi-chain tree consisting of multiple chains, which appears in the networks with disjoint multi-path routing or star-like networks. An example is shown in Fig. 8.

In a multi-chain tree, the initial filters will also be assigned to the leaf sensor nodes. Since there are multiple leaf nodes, a filter size allocation strategy among the leaf nodes is needed. Note that if we treat each chain of the tree as a single node, the tree can be considered as the one-hop network studied in [2][4]. Thus, we adapt our filter allocation scheme reported in a previous study [4] and devise our algorithm as follows.

The total error bound is first allocated uniformly to the leaf sensor node of each chain. The filters are re-allocated every UpD rounds. Intuitively, our algorithm re-allocates larger filters to the chains with larger number of update packets and smaller residual energy. Let E_i be the filter size assigned to chain c_i in the current round. Each chain maintains the number of update messages W_i and the minimum residual energy p_i of the sensor nodes on the chain for the recent UpD rounds. Each chain also maintains a set of sampling filter sizes $E_{i,*}$ and the estimated $W_{i,*}$ under these sampling filter sizes. After every UpD round, each chain informs the base station of $W_{i,*}$ and p_i for each of the sampling filter sizes. This information can be submitted by sending a message from the leaf node through the chain topology. In this message, there is a counter W_i for each of the sampling filter sizes. When this message passes an intermediate node, the node will add the number of updates recorded by itself to the respective W_i. This message also marks the minimum residual energy of the sensor nodes. Based on these information, the optimal filter re-allocation algorithm [4] is adopted by the base station to calculate the filters to be allocated to each chain for the next UpD rounds. For the clarify of the paper, we put this algorithm in the Appendix.

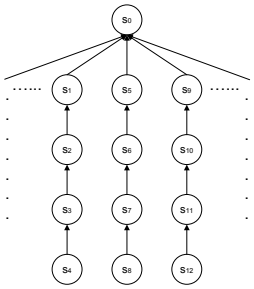


Fig. 8. An example of a multi-chain tree.

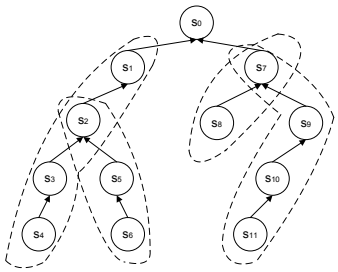


Fig. 9. An example of a tree, divided into multiple chains.

D. Filter Migration in General Trees

Finally, we extend our filter migration scheme to accommodate general tree structures for data collection. Note that the general data collection tree (the routing structure) can be built by some standard protocol (e.g., TAG [11]). Our strategy is to partition the tree into multiple chains and then apply the algorithm for multi-chain trees. Unlike the simple multi-chain tree, however, we need to decide where a chain ends in a general tree (the starting point is always a leaf node). We propose to use the intersection of two tree branches as a natural ending point. An example of such partitioning is shown in Fig. 9. A detailed description for a binary tree partitioning can be found in Fig. 10, which can be easily extended to trees of arbitrary degrees.

Algorithm TreeDivision ()

1 **for** each leaf s_i **do**
2 $s_k = \text{parent}(s_i)$
3 **while** s_i is the only child of s_k **or**
4 s_i is the left child of s_k
5 $s_k = \text{parent}(s_k)$
6 construct a chain from s_i to parent(s_k)
7 **end for**

Fig. 10. Tree Partitioning Algorithm

After partitioning, the tree topology can be treated as a multi-chain structure, except that residual filters are aggregated at the end of a chain (e.g., s_2 and s_7 in Fig. 9). The filter allocation and migration algorithms are the same as those discussed in the previous sections.

V. Simulation Results

A. Simulation Setup

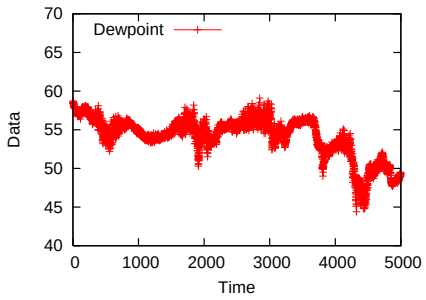


Fig. 11. Dewpoint trace from LEM project.

We have implemented our mobile filtering scheme in ns-2 [24]. Three typical topologies, namely, a chain, a cross, and a grid topology, have been used for performance evaluation. The cross topology is a multi-chain topology with four equal-length branches. In the grid topology, we set the base station at the center and a routing tree is built by broadcasting. For all these topologies, the distance between two neighboring sensor nodes is set to 2m and the transmission power on the physical layer is set to 2.5×10^{-6}dBm.

We adopt the same energy settings as those used in the Great Duck Island project [7] (we assume the voltage is the same in all compared cases). The power required for the operation of transmitting and receiving a packet are set to 20nAh (Ampere-hour) and 8nAh respectively. The power required for the operation of sensing a sample is 1.438nAh. The energy capacity for a sensor node is set to 80mAh. We omit the energy for sensors spent in sleeping state. The system lifetime is defined as the lifetime of the first dying node (in terms of operation rounds), which is widely adopted [14][4].

We test two different data traces in our simulation. The first is a synthetic data trace, where the readings are randomly and uniformly generated in the range of [0, 10] for each sensor. The second is a real world trace obtained from the Live from Earth and Mars (LEM) project [25] at the University of Washington. We used the dewpoint trace logged by the station at the University of Washington from August 2004 to August 2005, which consists of more than 500,000 sensor readings. For illustration purposes, we plot the first 5000 data points of the trace in Fig. 11. We have evaluated our algorithm against other traces in LEM, and similar performance trends are obtained. Each data point in a figure is an average of 10 randomly generated experiments.

We compare our mobile filtering scheme with a state-of-the-art stationary filtering algorithm [4]. It has been shown that this algorithm outperforms other existing stationary filtering algorithms ([3][2]) under various configurations.

B. Simulation Results

In Fig. 12(a), we show the results under a chain topology where the synthetic data are used. The total filter size is set to $2 \times N$; that is, each node on average can get a filter size of 2 (hereafter called the normalized filter size, as opposed to the total filter size $2N$). In this figure, we plot the mobile filtering scheme under both the greedy heuristic and the optimal offline algorithm. In the greedy heuristic, we set $T_R = 0$ and $T_S = 18\%$ of the total filter size. We will show how we choose T_R and T_S shortly. The optimal algorithm (Fig. 7) is used to serve as the performance upper bound in which all data updates on a chain are known a priori.

We can see that the more sensor nodes we have, the smaller the system lifetime for both the mobile and stationary filtering schemes. This is because the total filter size is smaller than the total data change. Thus, with more nodes, the number of data packet transmissions increases. We can make two other observations: First, mobile filtering always performs better than stationary filtering. Second, as the number of nodes increases, the superiority of mobile filtering becomes more substantial. For example, for 12 nodes, the system lifetime of mobile filtering is 2.5 times longer than that of stationary filtering, whereas for 28 nodes, a three time difference is observed. We also compare our scheme with stationary filtering using the dewpoint trace. The filter size is set to $0.2 \times N$. As shown in Fig. 12(b), similar results are found. In both sets of simulations, our greedy heuristic performs very close to the optimal solution. Thus, in the remaining simulations, we will present the results of the greedy heuristic only.

We then study the effect of the precision setting E as well as the impacts of the two parameters T_R and T_S on the greedy heuristic of our mobile filtering scheme. In these simulations, the number of nodes in a chain is fixed at 16.

With the synthetic data trace, we vary the normalized filter size from 0.8 to 3.6. We can see from Fig. 13(a) that allowing larger error bound can significantly improve the network lifetime. In this figure, we show three different T_R settings. Mobile-0% represents the case where the filter always migrates upstream as long as its size is greater than 0. Mobile-20% and Mobile-50% represent that the filter should stop if there is only 20% or 50% of the normalized filter left, unless it is piggy-backed. It can be seen that T_R does not have a big impact on the system lifetime. This is because, filter migration incurs a small cost compared to data reporting; and when the residual filter is small, it will be piggy-backed by the data packet it fails to suppress, making the cost even lower. Thus, we set $T_R = 0$ for the rest of our simulations.

In Fig. 13(b), we test the impact of T_S. Mobile-6, Mobile-8, and Mobile-10 represent the T_S settings of 6, 8, and 10 respectively. Notice that Mobile-10 implies that the filter should suppress all the updates if it can. As can be seen, when the filter size is small, the system lifetime is longer with a small T_S; and when the filter size is large, the system lifetime is longer with a large T_S. This is because when the filter size is small, suppressing a large update may significantly affect the ability of the mobile filter to suppress more future updates upstream. Thus, Mobile-6 performs better than Mobile-8 and Mobile-10. On the other hand, when the filter size is large, the mobile filter should have a greater budget to absorb large data changes. Setting a low T_S in this case will make the mobile filter suppress all the data changes that are less than T_S; yet with residual filter budget but cannot suppress larger data changes due to the small T_S constraint. This is the reason Mobile-6 performs poor when the filter size is greater than 2.5.

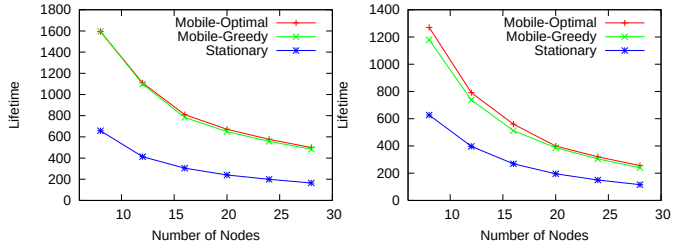


Fig. 12. Lifetime as a function of number of nodes for chain topology under (a) synthetic data and (b) dewpoint trace.

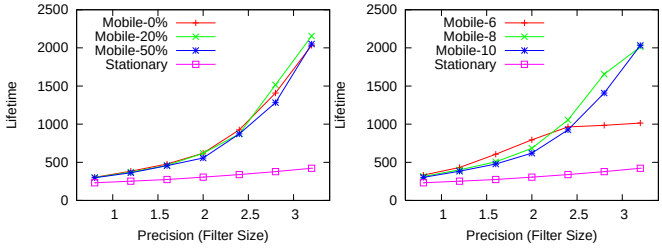


Fig. 13. Lifetime as a function of precision under synthetic data, (a) impact of T_R, (b) impact of T_S.

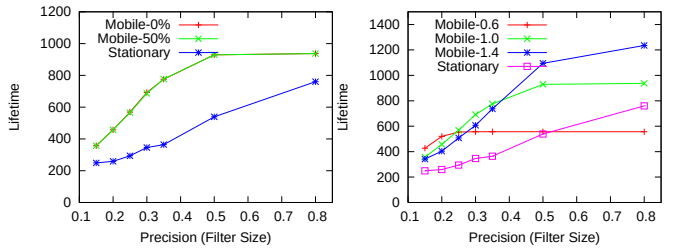


Fig. 14. Lifetime as a function of precision under dewpoint trace, (a) impact of T_R, (b) impact of T_S.

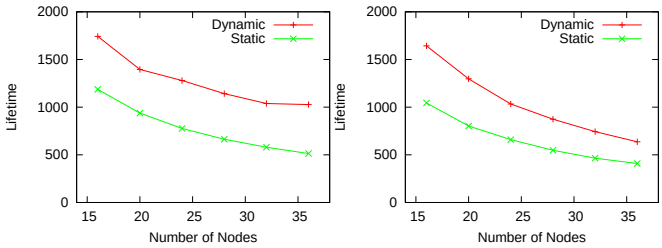


Fig. 15. Lifetime as a function of number of nodes for cross topology under (a) synthetic data and (b) dewpoint trace.

From these simulations, we can see that parameter T_S has a stronger impact than T_R. We next examine the impacts of T_R and T_S against the real dewpoint trace to obtain more insights. As shown in Fig. 14(a), T_R, again, has very little impact on the performance. On the other hand, from Fig. 14(b), the impact of T_S is more obvious for the dewpoint trace than for the synthetic data trace. This is because the data change for the synthetic trace is at most 10. However, for the dewpoint trace, there are occasionally larger data changes. We may call these large data changes *outliers* and they have a larger performance impact on mobile filtering. For both our synthetic data trace and the dewpoint trace, we find that if we set $T_S = 15\%-22\%$ of the total filter size, the system performance is relatively good. Therefore, in the rest of our experiments, we use $T_R = 0$ and $T_S = 18\%$ of the total filter size as our default settings.

We next examine the cross topology. We first consider the lifetime under different numbers of nodes. The results for the synthetic data trace and the dewpoint trace are shown in Fig. 15(a) and Fig. 15(b). Again, our mobile filtering scheme performs consistently better than stationary filtering by 50% to 100%. We also study the parameter UpD, the number of rounds between successive re-allocation of the filters for different chains. The results for the synthetic data trace and the dewpoint trace are shown in Fig. 16(a) and Fig. 16(b), where the total number of nodes is set to 24. We observe that as UpD increases, the system lifetime generally improves. The system will become stabilized sooner for a smaller precision. This is because it takes a shorter time to correctly predict the data changing pattern for smaller filters. The synthetic data trace shows a larger performance variation than the dewpoint trace; the changes of the later are more predictable.

Finally, we examine our mobile filtering scheme for a 7×7 grid topology. From Figs. 17(a) and 17(b), it can be seen that our mobile filtering scheme outperforms the stationary filtering scheme for both the synthetic and the dewpoint trace.

VI. MICA-2 EXPERIMENT RESULTS

To further validate the effectiveness of mobile filtering, we have conducted a series of experiments over our Mica-2 sensor network testbed. In the experiments, we deploy 10 Mica-2 motes to form a chain topology. In each mote, a photoconductive sensor is attached to monitor the light data. An additional mote, which is directly connected to a PC, serves as the base station. For comparison, we implement both the mobile and the stationary filtering algorithms.

Fig. 18 shows the key modules of our implementation. The *Main* module controls our program. The *Timer* module generates time events, so that the *Light Sensing* module can periodically access the Analog Digital Conversion (ADC) hardware to get data from the light sensor. The *Cmd Processor* module accepts and processes commands from the base station, including parameter initialization. The *Mobile/Stationary Filtering* module suppresses the data and passes the unsuppressed data to the *Communication* module for transmission.

In the experiment, we monitor the light data in our research lab. The statistics of a sample set of light changes can be seen from Fig. 19. About 60% of the updates are within one ADC unit, but the update can be as large as 20 ADC units.

Fig. 20 shows the lifetime results with different precision settings for both mobile and stationary filtering schemes. The normalized filter size varies from 1 to 7 ADC units. In our experiments, T_R and T_S are again set to 0% and 18%. We can see that our mobile filtering consistently outperforms stationary filtering, by 55% to 80%. This confirms our simulation results. Furthermore, when the normalized filter size increases, the gain of mobile filtering increases faster than that of stationary filtering. This is because, when the normalized filter size increases, T_S also increases, which offers more opportunities for mobile filtering to suppress more data updates.

We have also performed experiments for other environmental data (such as temperature) of our lab environments, and similar results have been observed.

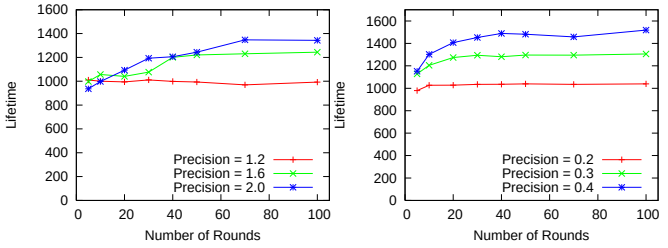


Fig. 16. Lifetime as a function of number of nodes for cross topology under (a) synthetic data and (b) dewpoint trace.

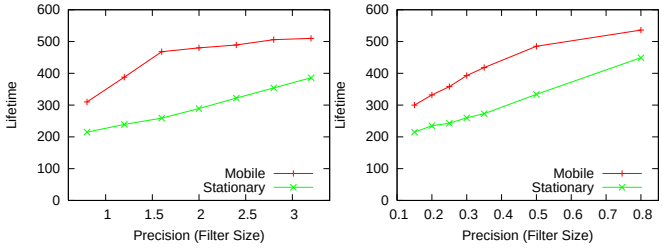


Fig. 17. Lifetime as a function of filter allocation for cross topology under (a) synthetic data and (b) dewpoint trace.

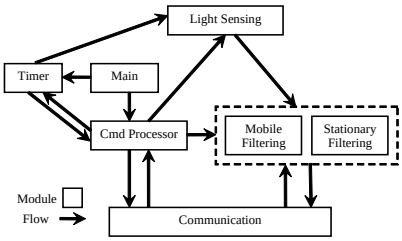


Fig. 18. Modules for Mica-2 sensor experiments.

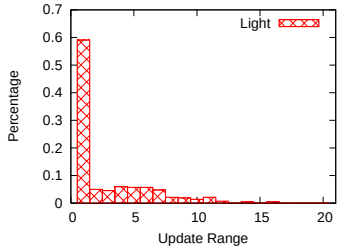


Fig. 19. Data changes of the ADC readings of light.

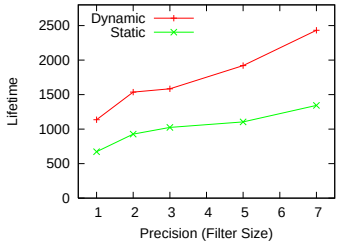


Fig. 20. Lifetime as a function of precision setting in the Mica-2 network for light data.

VII. Conclusion and Future Work

In this paper, we have proposed a novel mobile filtering scheme for error-bounded non-aggregate data collection in sensor networks. By exploring the migration of filters, a mobile filter extracts and relays unused filters in the network to suppress as many data update reports as possible.

An analytical study has been performed to quantify the performance benefit of mobile filtering against the conventional stationary filtering. We have also presented the detailed mobile filter designs for a chain routing topology. An optimal offline filter migration algorithm as well as a greedy online heuristic were developed. The algorithm was further extended to general multi-chain and tree topologies. Extensive simulations showed that: i) a small error allowed in data collection can significantly improve network lifetime, which verifies the importance of this study; ii) our mobile filtering scheme performs close to the optimal offline algorithm under a chain topology; and iii) the mobile filtering scheme substantially extends the network lifetime against the state-of-the-art stationary filtering scheme under various system configurations. Our preliminary experimental results based on a Mica-2 sensor testbed further validated our simulation results.

We believe many future work can be done. We are working on a more advanced mobile filter migration strategy in general graphs. Another interesting direction is to investigate in more depth of the T_S and T_R for general data.

Acknowledgments

Dan Wang's work is supported by grant Hong Kong PolyU/G-YG78, A-PB0R, A-PJ19, 1-ZV5W, and RGC/GRF PolyU 5305/08E. Jianliang Xu's work was supported by the Research Grants Council of Hong Kong under Projects HKBU211307 and HKBU210808. Jiangchuan Liu's work is supported by an NSERC Discovery Grant, an NSERC DAS Grant, an NSERC Strategic Project Grant, and an MITACS Project Grant.

Appendix

The following filter allocation algorithm (Fig. 21) is from [26]. Each chain maintains a set of sampling filter sizes $\frac{1}{2}E_i$, $\frac{3}{4}E_i$, ..., $\frac{2^K-1}{2^K}E_i$, $\frac{2^K+1}{2^K}E_i$, ..., $\frac{5}{4}E_i$, $\frac{3}{2}E_i$. We use $E_{i,*}$ to denote this set. The sensors will count the total number of updates for each of these candidate filter sizes. Every UpD rounds, the filter size of each chain is re-calculated by this algorithm. As specified in [26], the motivation of using exponentially spaced candidate filter sizes is to adjust the error bounds at coarse granularity when they are far away from the optimum, and adjust them at fine granularity when they are close to the optimum.

Algorithm The Optimal Error Bound Allocation

$E, E_{i,*}$: total and candidate error bounds for chain i
$W_{i,*}, p_i$: update message rate for $E_{i,*}$ and the minimum residual energy of chain i

1 $\quad \forall i, j,\ r_{i,j} = \frac{W_{i,j}}{p_i}$
2 $\quad \forall i,\ x_i = 1$
3 **while** $\min_{1 \le i \le n} x_i \neq m$
4 $\quad j = \arg\max_{1 \le i \le n, x_i \neq m} r_{i,x_i}$
5 $\quad$ **if** $E_{j,x_j+1} + \sum_{i \neq j} E_{i,x_i} > E$ **then**
6 $\quad\quad$ **break**
7 $\quad$ **end if**
8 $\quad x_j = x_j + 1$
9 **end while**

Output: optimal error bound E_{i,x_i} allocated for each chain

Fig. 21. Optimal candidate precision allocation in a single-hop network

Line 1 is to compute the normalized energy consumption rate $r_{i,j}$ for each candidate $E_{i,j}$. The initialization in Line 2 starts from the smallest error bound for all chains. In each iteration of Line 3-9, the error bound of the chain with the highest energy consumption rate is replaced with its next smallest candidate. Finally, the error bound of each chain is calculated and will be used for the next UpD rounds.

REFERENCES

[1] T. He, S. Ben-David, and L. Tong, "Nonparametric change detection and estimation in large scale sensor networks," *IEEE Trans. Signal Processing*, vol. 54, no. 4, pp. 1204–1217, 2006.

[2] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proc. ACM SIGMOD'03*, San Diego, CA, USA, Jun. 2003.

[3] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical in-network data aggregation with quality guarantees," in *Proc. IEEE EDBT'04*, Heraklion, Greece, Mar. 2004.

[4] X. Tang and J. Xu, "Extending network lifetime for precision-constrained data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM'06*, Barcelona, Spain.

[5] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White, "Testing that distributions are close," in *Proc. IEEE FOCS'00)*, Redondo Beach, CA, USA, Nov. 2000.

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

[7] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. ACM WSNA'02*, Atlanta, GA, USA, 2002.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Proc. ACM ASPLOS'02*, San Jose, CA, USA, Oct. 2002.

[9] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. USENIX OSDI'06*, Seattle, WA, USA, Nov. 2006.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proc. ACM MOBICOM'99*, Seattle, WA, USA, Aug. 1999.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad hoc sensor networks," in *Proc. USENIX OSDI'02*, Boston, MA, USA, Dec. 2002.

[12] N. Shrivastava, C. Buragohain, S. Suri, and D. Agrawal, "Medians and beyond: New aggregation techniques for sensor networks," in *Proc. ACM SENSYS'04*, Baltimore, MD, USA, Nov. 2004.

[13] Y. Yao and J. Gehrke, "Query processing in sensor networks," in *Proc. CIDR'03*, Asilomar, CA, USA, Jan. 2003.

[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. HICSS'00*, Wailea Maui, HI, USA, Jan. 2000.

[15] D. Wang, Y. Long, and F. Ergun, "A layered architecture for delay sensitive sensor networks," in *Proc. IEEE SECON'05*, Santa Clara, CA, USA, 2005.

[16] A. Silberstein, K. Munagala, and J. Yang, "Energy-efficient monitoring of extreme values in sensor networks," in *Proc. ACM SIGMOD'06*, Chicago, IL, USA, Jun. 2006.

[17] 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*, San Diego, CA, USA, 2003.

[18] D. Chu, A. Deshpande, J. Hellerstein, , and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proc. IEEE ICDS'06*, Atlanta, GA, USA, Apr. 2006.

[19] J. Gao, L. Guibas, and J. Hershberger, "Sparse data aggregation in sensor networks," in *Proc. ACM IPSN'07*, Cambridge, MA, USA, Apr. 2007.

[20] B. Sheng, Q. Li, and W. Mao, "Data storage placement in sensor networks," in *Proc. ACM MOBIHOC'06*, Florence, Italy, May 2006.

[21] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *Proc. ACM SIGMOD'06*, Chicago, IL, USA, Jun. 2006.

[22] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proc. ICDCS'02*, Vienna, Austria, Jul. 2002.

[23] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. ACM SENSYS'03*, Los Angeles, CA, USA, Nov. 2003.

[24] The network simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[25] Live from earth and mars (lem) project. [Online]. Available: http://www-k12.atmos.washington.edu/k12/grayskies

[26] X. Tang and J. Xu, "Optimizing lifetime for continuous data aggregation with precision guarantees in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 904–917, 2008.

PLACE PHOTO HERE

Dan Wang (S'05-M'07) received the B. Sc degree from Peking University, Beijing, China, in 2000, the M. Sc degree from Case Western Reserve University, Cleveland, Ohio, USA, in 2004, and the Ph. D. degree from Simon Fraser University, Burnaby, B.C., Canada, in 2007; all in computer science. He is currently an assistant professor at the Department of Computing, The Hong Kong Polytechnic University. His research interests include wireless sensor networks, Internet routing, and peer-to-peer networks. He is a member of the IEEE.

PLACE PHOTO HERE

Jianliang Xu is an Associate Professor in the Department of Computer Science, Hong Kong Baptist University. He received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, and the PhD degree in computer science from Hong Kong University of Science and Technology. His research interests include data management, mobile computing, wireless sensor networks, and distributed systems. He is a senior member of the IEEE.

PLACE PHOTO HERE

Feng Wang (S'07) received both his Bachelor's and Master's degree in Computer Science and Technology from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently a Ph.D. student in School of Computing Science at Simon Fraser University. His research interests include wireless sensor networks, peer-to-peer live streaming and distributed computing. In summer 2006, he interned in Microsoft Research Asia. In spring 2008 and spring 2009, he was a visiting Ph.D. student in Department of Computing at The Hong Kong Polytechnic University. In spring 2010, he visited the Wireless Ad Hoc Networks Lab in the Institute of Software at Chinese Academy of Sciences. He was awarded Tsinghua University Scholarship for Excellent Student in 1998, 2000 and 2001. At Simon Fraser University, he was awarded the SFU-CS Graduate Entrance Scholarship in 2005 and the Graduate Fellowship in 2007, 2008 and 2009. He was also awarded the Chinese Government Scholarship for Outstanding Self-financed Students Studying Abroad in 2009. He is a Student Member of IEEE and IEEE Communications Society.

PLACE PHOTO HERE

Jiangchuan Liu (S'01-M'03-SM'08) received the BEng degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the PhD degree from The Hong Kong University of Science and Technology in 2003, both in computer science. He is a recipient of Microsoft Research Fellowship (2000), Hong Kong Young Scientist Award (2003), and Canada NSERC DAS Award (2009). He is a co-recipient of the Best Student Paper Award of IEEE IWQoS'2008, the Best Paper Award (2009) of IEEE ComSoc Multimedia Communications Technical Committee, and the Best Paper Award Runner-up of IEEE IWQoS'2010.

He is currently an Associate Professor in the School of Computing Science, Simon Fraser University, British Columbia, Canada, and was an Assistant Professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong from 2003 to 2004.

His research interests include multimedia systems and networks, wireless ad hoc and sensor networks, and peer-to-peer and overlay networks. He is a Senior Member of IEEE and a member of Sigma Xi. He is an Associate Editor of IEEE Transactions on Multimedia, and an editor of IEEE Communications Surveys and Tutorials. He is TPC Vice Chair for Information Systems of IEEE INFOCOM'2011.