Mobile Filtering for Error-Bounded Data Collection in Sensor Networks

Dan Wang
Hong Kong Polytechnic Univ.
Hung Hom, Hong Kong
csdwang@comp.polyu.edu.hk

Jianliang Xu*
Hong Kong Baptist Univ.
Kowloon Tong, Hong Kong
xujl@comp.hkbu.edu.hk

Jiangchuan Liu, Feng Wang †
Simon Fraser University
Burnaby, BC, Canada
{jcliu, fwa1}@cs.sfu.ca

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. We start by presenting 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.*

1 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:

Q1: *Get the temperature distribution of the sensor field every other hour for the next 6 months.*

Q2: *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 [6].

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 reading in a sensor node is small, the node should not report the new reading. Olston *et al*. [13] 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][17]).

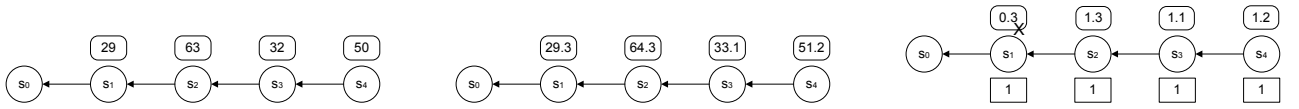
All these prior filter designs, however, are *stationary* in the sense that each filter is attached to a specific node and remains stationary during a round of data collection. Thus, unused filters in the current round of data collection might be wasted, limiting the filtering capability.

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, the overhead is outrun by the gain from suppressing more data trans-

*Jianliang Xu's work was supported by the Research Grants Council of Hong Kong under Project No. HKBU211307.

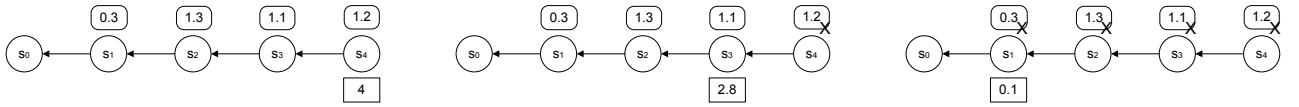
†J. Liu and F. Wang's work is supported by an NSERC Discovery Grant and an NSERC Strategic Project Grant.

1063-6927/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDCS.2008.41



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

Figure 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.

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

missions. The overhead can be further reduced by piggybacking the filter information in data update reports.

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],[1] 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 later in the paper.

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 made in this paper 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.

The rest of the paper is organized as follows. We review related work in Section 2. The system model is described in Section 3. Section 4 is devoted to our mobile filter design. We then show the simulation results in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Wireless sensor networks have been extensively studied in recent years and many sensor networks are designed for continuous data collection applications over a long period of time; see examples in [9][11].

Energy efficiency is a key consideration in sensor network designs. A pioneer work [4] 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 properties, 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

[1] 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.

studied [10][15]. Another effective in-network processing scheme is to make use of spatial data correlation, and the studies include clustering [7], sampling [19] and overhearing [16] techniques. Our work falls into an orthogonal category where temporal data correlation is explored [14], 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 [2][5]. These in-network processing techniques can also be combined to achieved 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 [13], 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 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 is responsible for computing partial aggregates from its descendants. A more intelligent filter adjustment scheme is proposed in [17]. In contrast to the previous studies where the filters are reallocated mainly based on data changing patterns, the optimization in [17] explicitly takes the residual energy of the sensor nodes into consideration.

Although these prior studies [3][13][17] have different filter (re)allocation mechanisms, they share a common fundamental 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.

3 System Model

In our system, the readings from individual sensors need to be periodically collected by the base station to evaluate complex distribution queries, and 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.

3.1 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 [1]. The smaller is the L_1 distance of two distributions,[2] 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.

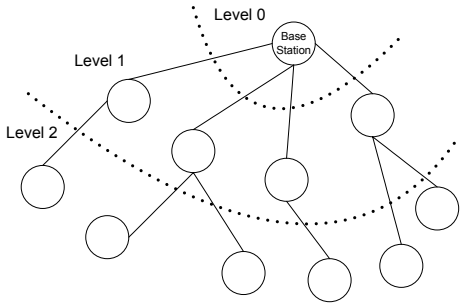


Figure 3. Underlying communication/routing structure.

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 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 senor 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.

3.2 Data Collection Model

For each round, we use a data collection model similar to TAG [10]. 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 en-

[2]The sensor readings can be easily normalized to probabilities.

ter into a *processing* state, and the sensor nodes one level higher 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 [8][10]. A round of data collection is completed when the processing state propagates to the root.

4 Mobile Filtering: Design and Optimization

The objective of our mobile filtering scheme is to minimize the total communication cost while maintaining the user error bound. In this section, we first outline a practical mobile filter design. We analyze this scheme for a chain routing topology. We derive an optimal offline migration strategy, together with an efficient online heuristic. We then extend the algorithm to multi-chain and tree topologies.

4.1 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).

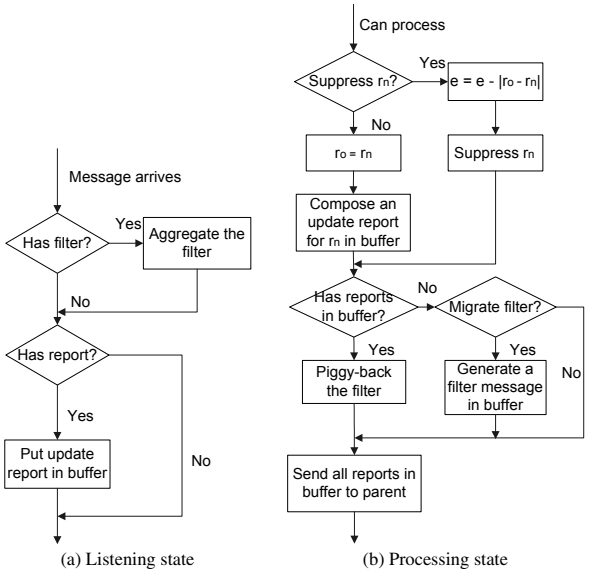


Figure 4. Operations of a sensor node in each round.

It is easy to see that under this operation model, the sum of the data changes suppressed does not exceed the total error bound. 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.

4.2 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: Due to space limitation, we omit the proof. See our technical report for details [20]. □

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 4.1. By the end of each round of data collection, the leaf node resets the filter size to E and all other nodes reset the filter sizes to zero. It is worth noting that resetting the filter sizes does not incur any communication cost.

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

4.2.1 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.

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

There are four possible choices that s_i can execute, as shown in (1)-(4). With the first choice, the data update is suppressed. 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. If the filter migration is not piggy-backed, there is one extra cost for sending this filter upstream to sensor node s_{i-1}. The second choice is similar to the first except the filter migration is piggy-backed with the data reports (of s_i's descendants) and incurs no extra overhead. With the third 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 fourth choice, the data update is suppressed, and the residual filter is not sent upstream. 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:

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

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.

$G_i(\cdot)$ can then be iteratively calculated using dynamic programming, as shown in Fig. 5. Note, however, that this optimal algorithm needs prior information about the data changes, which is difficult to obtain. We therefore 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.

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** $\forall i, e, \{+, -\}$

$$3 \quad G_i(e, +) = \max \begin{cases} i + G_{i-1}(e - v_i, +), \\ G_{i-1}(e, +), \\ 0 \end{cases}$$

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

5 **end for**

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

Figure 5. Calculate Gain Algorithm

4.3 Filter Migration in Multi-Chain Trees

The chain structure has provided 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. 6.

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 [13][17]. Thus, we adapt our filter allocation scheme reported in a previous study [17] 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 lower residual energy. Let E_i be the filter size assigned to chain c_i in the current round. Each chain main-

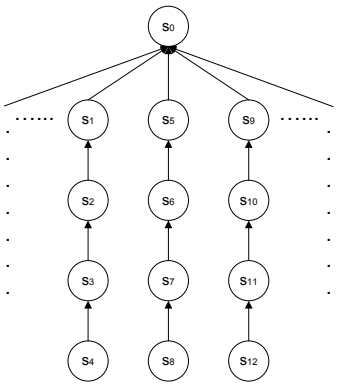


Figure 6. An example of a multi-chain tree.

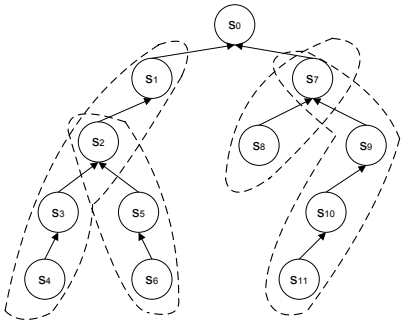


Figure 7. A tree, divided into multiple chains.

tains 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 $\frac{1}{2}E_i$, $\frac{3}{4}E_i$, ..., $\frac{2^K-1}{2^K}E_i$, $\frac{2^K+1}{2^K}E_i$, ..., $\frac{5}{4}E_i$, $\frac{3}{2}E_i$. We also estimate W_i and p_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 sensor 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 estimates the minimum residual energy of the sensor nodes. Based on this information, the optimal filter re-allocation algorithm [17] is adopted by the base station to calculate the filters to be allocated to each chain for the next UpD rounds to maximize the minimum energy of the sensor nodes.

4.4 Filter Migration in General Trees

Finally, we extend our filter migration scheme to accommodate general tree structures for data collection. 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. 7. A detailed description for a binary tree partitioning can be found in Fig. 8, 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**

Figure 8. 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. 7). The filter allocation and migration algorithms are the same as those discussed in the previous sections.

5 Simulation Results

We have implemented our mobile filtering scheme in ns-2 [12]. 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 [11]. The energy costs of transmitting and receiving a packet are set to 20nAh (Ampere-hour) and 8nAh respectively. The energy cost for sensing a sample is 1.438nAh. The energy budget reserved for a sensor node is set to 80mAh. We omit the energy for sensors in sleeping state. The system lifetime is defined as the lifetime of the first dying node, which is widely adopted [7][17].

We test two different data traces in our simulation. The first is a synthetic data trace, whose readings are randomly generated in the range of [0, 10]. The second is a real world trace obtained from the Live from Earth and Mars (LEM) project [18] 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. 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.

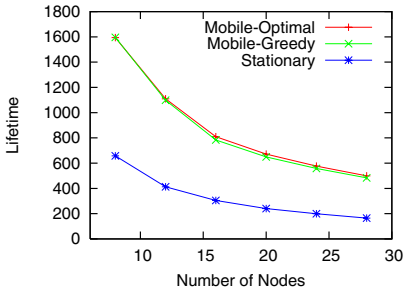


Figure 9. Lifetime as a function of number of nodes for chain topology under synthetic data.

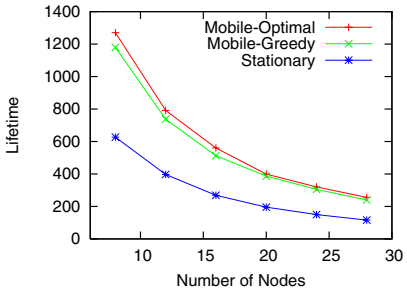


Figure 10. Lifetime as a function of number of nodes for chain topology and dewpoint trace.

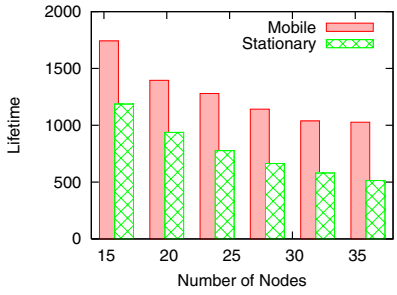


Figure 11. Life time as a function of number of nodes for cross topology under synthetic data.

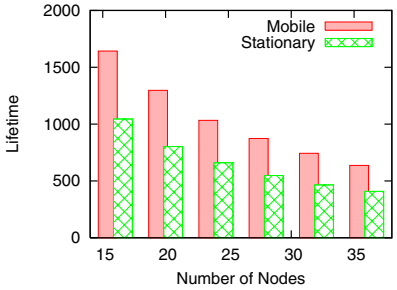


Figure 12. Lifetime as a function of number of nodes for cross topology under dewpoint trace.

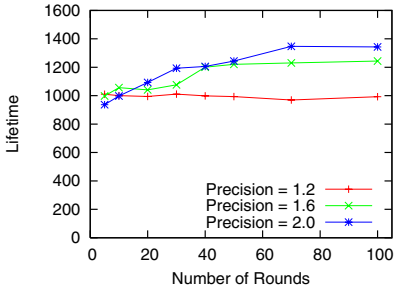


Figure 13. Lifetime as a function of filter allocation for cross topology under synthetic data.

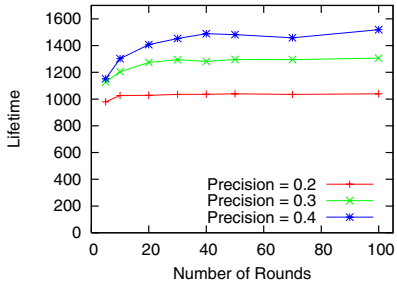


Figure 14. Lifetime as a function of filter allocation for cross topology under dewpoint trace.

We compare our mobile filtering scheme with a state-of-the-art stationary filtering algorithm [17]. It has been shown that this algorithm outperforms other existing stationary filtering algorithms ([3][13]) under various configurations.

Simulation Results

In Fig. 9, 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. Due to the limitation of space, readers may find how we choose T_R and T_S in [20]. The optimal algorithm (Fig. 5) 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 shorter is 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. 10, similar results are found. In both sets of simulations, we can see that 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 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. 11 and Fig. 12. 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 that the filters should be re-allocated for different chains. The results for the synthetic data trace and the dewpoint trace are shown in Fig. 13 and Fig. 14, 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

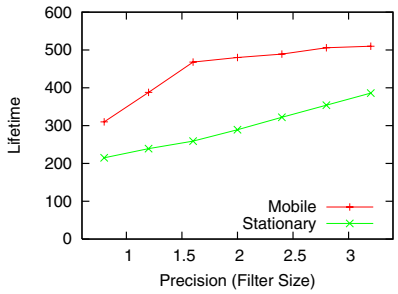


Figure 15. Life time as a function of precision for grid topology under synthetic data trace.

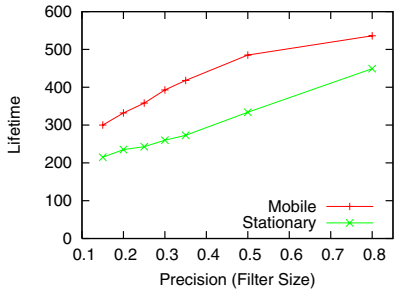


Figure 16. Lifetime as a function of precision for grid topology under dewpoint trace.

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. 15 and 16, it can be seen that our mobile filtering scheme outperforms the stationary filtering scheme for both the synthetic and the dewpoint trace.

We have conducted experiments on a Mica-2 sensor network testbed and such results are also observed, see our technical report [20] for details.

6 Conclusion

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.

We have 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.

References

[1] T. Batu, L. Fortnow, R. Rubinfeld, W. Smith, and P. White, "Testing That Distributions Are Close", in *Proc. IEEE FOCS'00*, Redondo Beach, CA, Nov. 2000.

[2] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks using Probabilistic Models", in *Proc. IEEE ICDE'06*, Atlant, GA, Apr. 2006.

[3] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical In-Network Data Aggregation with Quality Guarantees", in *Proc. EDBT'04*, Heraklion, Greece, Mar. 2004.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", in *Proc. ACM MOBICOM'99*, Seattle, WA, Aug. 1999.

[5] J. Gao, L. Guibas, and J. Hershberger, "Sparse Data Aggregation in Sensor Networks", in *Proc. ACM IPSN'07*, Cambridge, MA, Apr. 2007.

[6] 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, Apr. 2006.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", in *Proc. HICSS'00*, Wailea Maui, HI, Jan. 2000.

[8] 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, July 2002.

[9] 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*, Oct. 2002.

[10] 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, Dec. 2002.

[11] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring", in *Proc. ACM WSNA'02*, Atlanta, GA, Sept. 2002.

[12] "The Network Simulator ns-2", http://www.isi.edu/nsnam/ns/.

[13] C. Olston, J. Jiang, and J. Widom, "Adaptive Filters for Continuous Queries over Distributed Data Streams", in *Proc. ACM SIGMOD'03*, San Diego, CA, June 2003.

[14] 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, Sept. 2003.

[15] 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, Nov. 2004.

[16] A. Silberstein, K. Munagala, and J. Yang, "Energy-Efficient Monitoring of Extreme Values in Sensor Networks", in *Proc. ACM SIGMOD'06*, Chicago, IL, June 2006.

[17] X. Tang and J. Xu, "Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks", in *Proc. IEEE INFOCOM'06*, Apr. 2006. (*An extended version is to appear in ACM/IEEE Trans. Networking*)

[18] Live from Earth and Mars (LEM) Project, http://www-k12.atmos.washington.edu/k12/grayskies, 2006.

[19] D. Wang, Y. Long, and F. Ergun, "A Layered Architecture for Delay Sensitive Sensor Networks", in *Proc. IEEE SECON'05*, Sept. 2005.

[20] D. Wang, J. Xu, J. Liu, and F. Wang, "Mobile Filtering for Error-Bounded Data Collection in Sensor Networks", *Technical Report*, Simon Fraser University, Nov. 2007.

[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, June, 2006.