IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 13 July 2014; revised 20 December 2014; accepted 14 February 2015; date of publication 5 March 2015;
date of current version 7 December 2016.

Digital Object Identifier 10.1109/TETC.2015.2411215

Data Sweeper: A Proactive Filtering Framework for Error-Bounded Sensor Data Collection

DAN WANG[1], JIANGCHUAN LIU[2], JIANLIANG XU[3], HONGBO JIANG[4], AND CHONGGANG WANG[5]

[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[2]School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada
[3]Department of Computer Science, Hong Kong Baptist University, Hong Kong
[4]Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[5]InterDigital Communications, InterDigital Communications, Princeton, NJ 08540 USA

CORRESPONDING AUTHOR: D. WANG (csdwang@comp.polyu.edu.hk)

The work of D. Wang was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61272464 and in part by the Research Grants Council/General Research Fund under Grant PolyU 5264/13E. The work of J. Liu was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, in part by the NSERC Strategic Project Grant, and in part by the Major Program of International Cooperation through the NSFC under Grant 61120106008. The work of J. Xu was supported by the Hong Kong Research Grants Council under Grant HKBU12202414 and Grant HKBU12200114. The work of H. Jiang was supported by the NSFC under Grant 61271226.

ABSTRACT This paper presents data sweeper—a novel framework that attempts to reduce network traffic for error-bounded data collection in wireless sensor networks. Unlike existing passive filters, a data sweeper migrates in the network and proactively suppresses data updates while maintaining the user-defined error bound. Intuitively, the migration of a data sweeper learns the data change of each sensor node on the fly, which helps to maximize the filtering capacity. We design the data sweeper framework in such a way that it can accommodate diverse query specifications and be easily incorporated into the existing sensor network protocols. Moreover, we develop efficient strategies for query precision maintenance, sweeper migration, and data suppression within the framework. In particular, in order to maximize traffic reduction and adapt to online data updates, a Lagrangian relaxation-based algorithm is proposed for data suppression. Extensive simulations based on real-world traces show that the data sweeper significantly reduces the network traffic and extends the system lifetime under various network configurations.

INDEX TERMS Data sweeper, sensor data collection.

I. INTRODUCTION

Wireless sensor networks have been widely used nowadays to capture the physical phenomena in a field of interest. To better understand and analyze the properties of the field, many sensor applications expect continuous data collection. For example, scientists may be interested in monitoring the population distribution of certain nomadic animals in a wild region; a sensor network deployed in the region thus has to periodically measure and report the density of the animals at different locations. In sensor networks, the most limited resource is energy and communication dominates energy consumption. Obviously, requesting every sensor node to report its sensed data is prohibitively expensive. Thus, various in-network processing techniques have been proposed; that is, the sensed data can be filtered, merged, and sampled to reduce the traffic volume [7].

Clearly, with in-network processing, the monitoring results evaluated based on collected data might deviate from the exact results. Fortunately, if the error is bounded by a certain threshold, it is acceptable for many sensor applications. One well-known technique for error-bounded data collection is data filtering, which explores temporal data correlation to suppress data updates. A concrete scheme was proposed in [18]. The data collection is divided in *rounds*, i.e., periodically and each period of time is a round. A filter is installed on each node and the total filter size is constrained

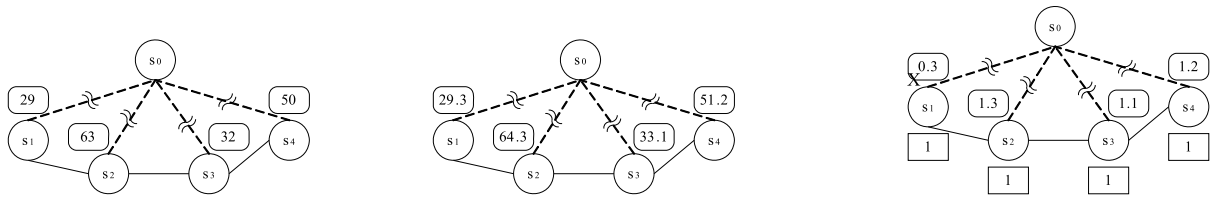


FIGURE 1. An example of a passive filtering scheme. Total user allowed filter size (error bound) is 4. Data readings are shown above the sensor nodes and filters are shown below the sensor nodes. Node s_0 is the base station. Only part of the network is shown, and the dashed lines with interruption represent multi-hop network connections. (a) Previously reported data readings. (b) Data readings of the current round. (c) Passive filter suppresses one data report at s_1.

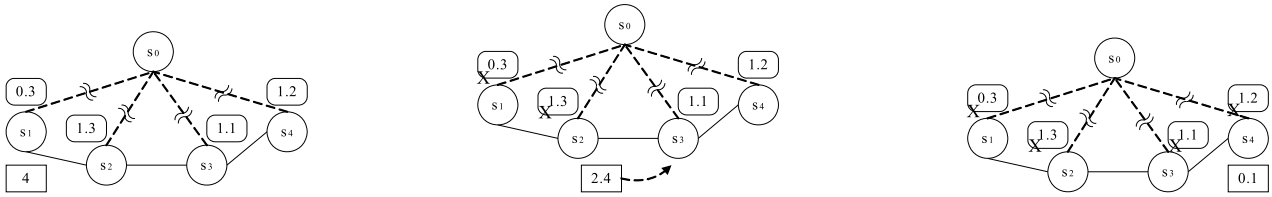


FIGURE 2. An example of the data sweeper scheme. Total user allowed filter size (error bound) is 4. Data readings are shown above the sensor nodes and filters are shown below the sensor nodes. Node s_0 is the base station. Only part of the network is shown, and the dashed lines with interruption represent multi-hop network connections. (a) Data sweeper with whole filter size. (b) Data sweeper moves and suppresses data reports. (c) All four data reports are suppressed.

by the user-specified error budget. Intuitively, if the data change from the last update report is smaller than the filter size, the current update is *suppressed*, i.e., not to report to the base station. To adapt to system dynamics, the sizes of all filters are periodically shrunk and the left-over budget is re-allocated to the node with the highest load. Recently, there are many follow up studies, see [5], [24], where hierarchical routing and energy issues are taken into consideration for computing the filter sizes for re-allocation. However, the filters in all these prior studies are *passive* in the sense that once installed, each filter sticks to a particular sensor node and only waits for local data updates for possible suppression. Since each filter suppresses the local updates only, an intrinsic problem is that the residual filter size after suppression, if any, cannot be used by other nodes and hence is wasted. Even worst, if a filter is violated, its filtering capacity is not utilized at all.

In this paper, we propose a novel *data sweeper* to address this problem. Unlike existing passive filters, a data sweeper, carrying the error budget, migrates in the network and *proactively* suppresses data updates while maintaining the user-defined error bound. Intuitively, the migration of a data sweeper learns the data change of each sensor node on the fly, which helps to fully utilize the filtering capacity. As a result, traffic reduction can be maximized. We illustrate this idea through a simple example.

An Example: *We compare the passive filtering scheme and the data sweeper by the scenario shown in Figs. 1 and 2, where s_0 is the base station. The previously reported sensor readings and the readings in the current data collection round are shown in Figs. 1(a) and 1(b), respectively. Suppose that the total user allowed error bound is 4. One possible filter allocation is shown in Fig. 1(c). We can see that in the passive filtering scheme, only one data update at node s_1 is suppressed, and three (remote) updates are incurred. In fact, the residual filter size at s_1 (i.e., 0.7) and the violated filters at $s_2 - s_4$ (i.e., 1) are totally wasted. In contrast, in the new scheme, a data sweeper is released with the whole filter size, as shown in Fig. 2(a). It travels the sensor network and suppresses the data updates on the fly; see Fig. 2(b). As a result, all four data updates are suppressed; see Fig. 2(c).*

One may think that the data sweeper is similar to the re-allocation of the filters for each data collection round. Their differences, however, are fundamental. First, in the passive filtering schemes, the re-computation of the filter sizes is based on statistical system information collected in the previous rounds. More importantly, this re-computation is done before dissemination of the adjusted filter sizes to their respective sensor nodes. Thus, though all previous efforts have proposed intelligent schemes in this respect, the online change of the data updates would make the filter re-allocation inaccurate. In contrast, the data sweeper makes suppression decisions on the fly so as to efficiently adapt to the actual data changes and better exploit the filtering capacity. Second, filter re-allocation for each collection round would be extremely costly since it involves the dissemination of the adjusted filter sizes to their respective sensor nodes. For this reason, the previous studies have suggested re-allocating filters only every 50 to 500 rounds [5], [24] to amortize this cost, making an even larger mismatch between the pre-allocated filter sizes and actual data changes in each round. Intrinsically, in the passive scheme, the probability that a filter can match a data update is small; leading to a waste of the filtering capacity in either the residual filter sizes or the violated filters.

As a result, a large number of data updates can be observed. This intuition will be formally analyzed in this paper.

To implement the data sweeper for sensor networks, we propose a lightweight data sweeper framework such that the migration and behavior of the data sweeper are separated from the routing of the data update reports. Thus, the framework can be easily integrated into the existing sensor network infrastructure. We then develop a comprehensive set of models and protocols to maximize the benefit of the data sweeper:

- A general error bound model which accommodates user precision requirements for various query types.
- A tour generator which guides the data sweeper to efficiently migrate in the sensor network.
- A sensor-sweeper interaction module based on the Lagrangian relaxation which maximizes update suppression.
- A data sweeper self-adjustment module which adapts the data sweeper to system dynamics.

We evaluate the effectiveness of the data sweeper through extensive simulations in ns-2 using both the synthetic data and the real-world traces obtained from the LEM project [28]. The results show that the data sweeper substantially outperforms the state-of-the-art data filtering schemes under various network configurations.

The rest of this paper proceeds as follows. We review the related work in Section II. An overview of the data sweeper framework is given in Section III. Section IV is devoted to the detailed design and optimization of the data sweeper framework. Discussions on accommodating various extensions are presented in Section V. Simulation results are reported in Section VI, followed by a final conclusion in Section VII.

II. RELATED WORK

Wireless sensor networks have been extensively studied in recent years; a survey can be found in [1]. Many sensor networks are designed for continuous monitoring applications. Examples include the sensor network deployed in Great Duck Island [14] to monitor the habitat of birds and ZebraNet in Africa [11] to monitor the behavior of wildlife.

In recent years, people have used wireless sensor networks more widely in civil applications rather than in the wild, such as structural bridges, energy conserving buildings [12], [19]. These leads to Internet-of-Things, where the sensor networks from different applications are interconnected. The amount of data generated are even greater. If all the data are to be transmitted and analyzed, this poses burden to the sensors on its energy reserve, the aggregators on transmission congestion and the analytic devices on the amount of data.

For data collection, a typical routing structure is to have the base station serve as an interconnection point between the user and the sensor network. In tree-structured models [7], [13], data collection is performed through a routing tree rooted at the base station. In hierarchical models [10], sensor nodes are grouped into clusters and data are collected by cluster heads, which in turn report to the base station. Multi-path [16] and hybrid routing structures [15] have also been suggested for energy balancing and error resilience purposes.

Energy efficiency is a key consideration in senor network designs. A pioneer work [7] has suggested in-network processing to reduce data traffic in sensor networks. By exploring the query's characteristics, in-network aggregation is used where an intermediate sensor node can compute a partial result based on the data received from its descendants. Such aggregate queries as MAX, MIN, AVG, SUM, and MEDIAN have been studied in [8], [13], and [21]. There are other sensor applications which are interested in the data distribution information and in favor of non-aggregate data. For example, a consistent change of the population distribution of wildlife can serve as an important indication of the change of the surrounding environment [9]. For these applications, in-network processing techniques have been proposed by exploring data correlation. For snap-shot data collection, network traffic can be compressed based on spatial correlation, e.g., using clustering [10], sampling [26], or overhearing techniques [22]. For continuous data collection, an orthogonal approach is to explore temporal correlation; which is the focus of this paper. A typical idea is to filter out data update reports with small changes between consecutive collection rounds.

A first study on data filtering for continuous queries is [18]. A filter is allocated to each sensor node where the total filter size is constrained by the user error bound. The filters periodically shrink and the server will re-allocate the left-over filter size to the sensor nodes based on a set of parameters, such as the number of update packets generated by the sensor node since the last filter re-allocation, the current filter size, and the data reporting cost. The work in [5] and [24] extends the filter allocation algorithms to multi-hop sensor networks with more refined filter allocation schemes. Integration of different in-network processing techniques to achieve better performance can be found in [5] and [22].

The above studies of data filtering have explored data semantics in their algorithm designs, by considering historical data updates and spatial/temporal data correlations. However, as pointed out in the Introduction, the filters in the network are passive and blind to each other; hence the utilization of them is not optimized. As an improvement, in this paper we propose an innovative data sweeper that proactively suppresses data updates. Other works related to query processing include [3], [23], and [29].

One work has suggested the effectiveness towards this direction [27]. In [27], a line topology is studied where a filter is allocated to the leaf node and move upstream. However, the routing for the data collection is a tree and the general topology divide the total filter into the leaf nodes of each of the tree branches (lines) in advance. The allocation of the filters is still passive and the filters at the leaf nodes are blind to each other. As a result, the performance gain sharply reduced. The key observation made in this paper is that we

separate the data sweeper and the routing of the data collection. This motivated us to design the fully proactive data sweeper with the following advantages. First, it has better performance over all the passive schemes. Second, some passive schemes (e.g., our preliminary study [27]) request key statistical parameters to be hard coded. We provide a self-adjust scheme for the data sweeper and we will show that the performance is not affected by the initial values of the parameters. Third, the data sweeper is not affected by the underlying data routing structure (e.g., tree, multi-path, etc) or the queries, and thus more generic and comprehensive. We also show how it can be integrated with data aggregation queries.

III. ARCHITECTURE OVERVIEW

We first clarify the application scenario. We assume that sensor networks are deployed to monitor the physical environments. The sensor must report the physical phenomena to the base station periodically, i.e., in rounds. The application allows certain errors for the sensor report. Between two rounds, the sensor network can conduct processing on the sensor readings. The objective of such processing (either the passive filter or data sweeper) is to minimize energy consumption as long as the allowable error is satisfied.

A data sweeper is a special control packet that carries query (precision) information and can be routed within the sensor network. Upon interacting with a sensor node, it applies the precision requirements (for example, residual filter size) to suppress the data update report of the node.

As mentioned, there are two design objectives for the data sweeper: 1) generic and flexible to accommodate different query types, precision requirements, and routing structures for data collection; and 2) energy efficient to maximize the overall traffic reduction. We thus design a lightweight data sweeper framework, to accommodate the proactive data sweeper. This framework sits in the application layer and interacts with the routing layer (see Fig. 3). The application layer (User Queries Register module) accepts continuous queries registered to the sensor network. During registration, a query specifies the following information: 1) the query type, e.g., simple aggregates or complex data distribution queries; 2) the query duration, e.g., five months; 3) the query execution period, e.g., every two hours; and 4) the precision threshold. The routing layer (Data Aggregation/Routing module) maintains the data collection structure to report the data.

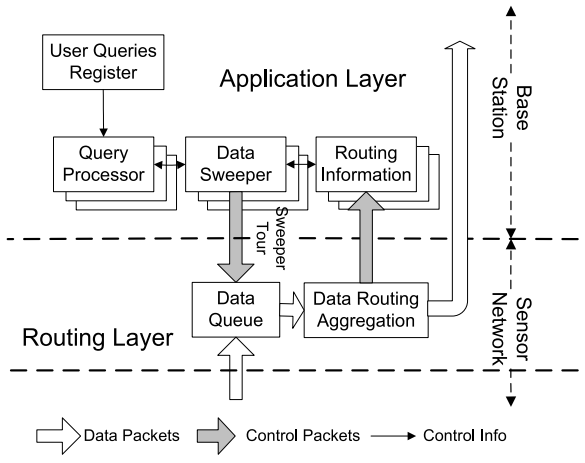


FIGURE 3. **Data Sweeper framework.**

In the first round of data collection, all sensor nodes report their readings to the base station. In each subsequent rounds, a sensor node reports its data only if it is not suppressed. Specifically, for each round, the data collection is divided into two phases. In Phase I, every sensor node senses a new reading and store the reading in its Data Queue. Meanwhile, the data sweeper is released from the base station to travel through the sensor network. When the data sweeper reaches a sensor node, it decides whether to suppress the data update based on the sensor-sweeper interaction algorithm (to be discussed in detail in Section IV.D). In our design the algorithm never allows a suppression that violates the user precision requirement. In Phase II, the data updates that are not suppressed are reported to the base station through the underlying data collection structure; in-network aggregation and other in-network processing techniques can be performed if necessary. In a data collection round, if the update report of a sensor node is suppressed, the base station will use the previously obtained reading from this node for query evaluation.

It is easy to see that, based on this framework, the user precision is guaranteed at any round during the continuous data collection. In addition, our framework is designed in a way such that the User Query module and Data Routing module are separated from the data sweeper framework and remain unchanged. For ease of exposition, in this paper we employ a tree-based routing structure for data collection. Note however our framework can accommodate any specific networking layer implementation and can be adapted to different routing structures including multi-path routing [16] and mixed tree-multi-path routing [15]. Similarly, we mainly focus on the queries that require non-aggregate data collection (e.g., data distribution queries) to illustrate the design. Aggregate queries such as SUM are separately discussed in Section V.

A. ANALYTICAL COMPARISON OF DATA SWEEPER & PASSIVE FILTERS

Before we go into the detailed design, we first take a look at the comparison of the performance of the data sweeper and the passive schemes from a high level point of view. Let the data change at each sensor node conforms to standard normal distribution $N(0, 1)$. Let E be the total error bound. Let N be the total number of sensors.

Consider such passive filtering scheme where the total error bound is divided equally to each sensor nodes. Let X_i denote the random variable of the data change at

sensor node i. Since, $X_i \sim N(0,1)$, the probability that the filter at node i is violated is $p_i = Pr[X_i > \frac{E}{N}] = 1 - Pr[X_i < \frac{E}{N}] = \frac{1}{2}(1 - \text{erf}(\frac{E}{\sqrt{2N}}))$. Define an indicator variable Y_i such that

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

Thus, the expectation of a sensor whose filter is violated is $E[Y_i] = p_i \times 1 + (1 - p_i) \times 0 = p_i = \frac{1}{2}(1 - \text{erf}(\frac{E}{\sqrt{2N}}))$.

For data sweeper, we assume that there is a data filtering scheme such that if the total data change of the system in a round is greater than the error bound, the sensor nodes whose data reports are not suppressed are selected randomly and uniformly by this scheme. In addition, if the total data change is less than the error bound, all the updates will be suppressed. Similarly, let X_i denote the random variable of the data change at sensor node i. Let $X = \sum_{i=1}^{N} X_i$. Since $X_i \sim N(0,1)$, we have $X \sim N(0, N)$. Define an indicator variable Y such that

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

Thus, the expectation of a sensor whose filter is violated is $E[Y] = \frac{1}{2}(1 - \text{erf}(\frac{E}{\sqrt{2N}}))$.

The expected number of link messages (the cost) of the system is the expected number of messages incurred by violating the filter of a sensor multiplied by the expectation of the filter to be violated. The expected number of messages incurred by violating the filter of each sensor depends on the underlying data routing structure. We use a tree as an example to estimate the cost since it is the current dominate data routing structure for sensor data collection. We emphasize that the tree is not special, however; other routing structure will not affect the performance comparison, as given the routing structure fixed, the expected number of messages incurred by a sensor is the same under different filtering schemes. For a standard full k-nary tree, the expected number of messages for a sensor node to report its data update is $C(k) = \sum_{i=1}^{\log_k(N+k-1)} \frac{N+k-1}{k^i}(\log_k(N+1) - 1 - i)$.

We also compare our recent work on mobile filter with the data sweeper. In the mobile filter scheme, the total error bound is divided into each branch. Let X_i denote the random variable of the data changes at sensor node i. Let the number of branches be w. The total number of sensors in each branch is $Z_j = \sum_{i=1}^{\frac{N}{w}} X_i$. Clearly, $Z_j \sim N(0, \frac{N}{w})$. Define an indicator variable Y_j such that

$$Y_j = \begin{cases} 1 & \text{if } Z_j > \frac{E}{w}; \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the expectation of a sensor whose filter is violated is $E[Y_j] = \frac{1}{2}(1 - \text{erf}(\frac{E}{\sqrt{2wN}}))$.

Overall, the expected number of messages of the system for these three schemes are summarized as follows:

$$\begin{cases} E[Y_i] \times C(k) & \text{passive scheme} \\ (E[Y] \times C(k) + N)/N & \text{data sweeper} \\ E[Y_j] \times C(k) & \text{mobile filter.} \end{cases}$$

Notice that for data sweeper migration, we add an additional cost of N. We also want to comment that mobile filter can also be considered as multiple data sweepers, where the total error bounds are divided and allocated to each mobile filter.

To illustrate the comparison clearly, we plot the numerical results of these three schemes in Fig. 4. We see that the passive scheme shows a much higher total transmission cost. The cost increases much steeper than the data sweeper. The mobile filter improves the performance, but did not fully utilize the benefit.

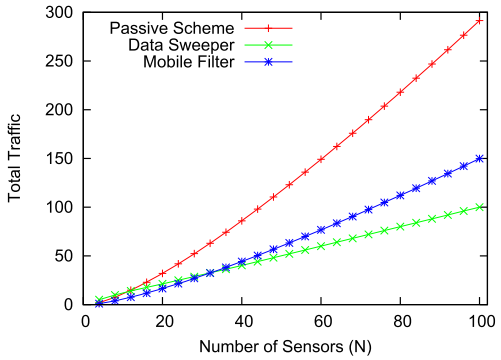


FIGURE 4. Transmission cost as a function of the number of nodes *N*.

B. DISCUSSION ON OVERALL DESIGN CHOICES

We now present some of our high level design choices. The most important thing is what must be included in the framework and what are the details that can be further developed according to different applications.

Our rule of thumb is that in our data sweeper framework, we do not consider the problems that are related to the data pattern. In other words, we do not assume prior knowledge of the data distribution and we also do not consider it is a must to include learning algorithms into our framework.

We use an example to explain this point. One may easily consider treating different data updates differently; for example, giving greater weights to a bigger update. For example, given an error bound of 4, and two updates are 3.99 and 0.02 respectively. As we cannot suppress both updates, one may consider it could be better to suppress the update of 0.02. An hidden assumption of this is, however, that the data pattern shows a graduate change (in the order of 0.01) than an abrupt change (in the order of 1). Without this assumption there is no reason to suppress the update of 3.99 as compared to 0.02.

Note that the data sweeper outperforms the passive filter independent to any data distributions.

IV. DATA SWEEPER: DESIGN AND OPTIMIZATION

A. DESIGN CHALLENGES AND PHILOSOPHY

The concrete issues to be addressed for the high performance of the data sweeper are: 1) the information that the data sweeper should carry so that it is lightweight; 2) a specific error model for the data sweeper to incorporate

queries with different types and precision requirements; 3) the sensor-sweeper interaction algorithm that maximizes traffic reduction.

Our first design decision is that the data sweeper will maintain only the information about the *residual* filter size (unused error bound) and some necessary statistical information. We consider this to be the key to keep our data sweeper constant in size[1] and thus scalable to different network sizes.

Secondly, to maximize traffic reduction, we design a sweeper routing algorithm (to determine the traveling tour of the data sweeper in the network) and a sensor-sweeper interaction algorithm (to decide whether to suppress a data update report or not; intuitively the data sweeper may not suppress a data update if it perceives that it is more beneficial to save the residual filter size for future data suppression). Instead of a joint optimization of these two algorithms, we advocate a separation design due to the following reasons:

1) MULTIPLE QUERIES

It is possible that the data sweeper carries filters for precision requirements of multiple queries. These queries may employ different underlying data routing structures (e.g., different trees, multi-path routing). In this case, a joint optimization of sweeper routing and sensor-sweeper interaction is difficult. In addition, new query registration may introduce complicated re-optimization.

2) ONLINE DATA UPDATES

If data updates are predictable, a joint optimization would maximize the performance gain. For example, the data sweeper may not visit the nodes without data changes (i.e., no sensor-sweeper interaction is needed). As in practice data updates are online, however, we foresee the benefit of a joint optimization is largely reduced.

3) OVERHEAD FOR THE DATA SWEEPER

A joint optimization may lead the sweeper routing tour dynamic based on the residual filter size; that is, the routing tours may differ given the data sweeper's current status. Thus, the data sweeper packet has to carry the tour information, which introduces an overhead that is proportional to the network size. This sharply violates our lightweight design philosophy.

Based on these considerations, we design the data sweeper such that it contains only the filter parameters. The sweeper tour will be constructed independent of specific queries. The sensor-sweeper interaction algorithm will be developed based on queries' error bounds and data collection structures, with the objective of maximizing overall traffic reduction. The data sweeper will also carry several statistical parameters of constant size so that the sensor-sweeper interaction algorithm can be adjusted each round according to system dynamics.

[1]The packet size is usually 60 bytes for Mica-2 motes [6].

B. QUERY AND ERROR BOUND MODEL

We first focus on the design for a single query; we shall discuss the extension to multiple queries in Section V. To guarantee the precision of query results, an error bound should be introduced for data collection. The Query Processor module in the data sweeper framework (Fig. 1) converts the precision requirement to a filter size maintained by the data sweeper.

In general, our data sweeper works for any error bound model in which the total error of the data collection is a function of the errors from individual sensor nodes, such as L_k distance, weighted L_k distance and KL-divergence. The data sweeper framework is open for different error bound models based on specific query requirements. To illustrate our design, in this paper, we mainly adopt L_1 distance as a measurement of the error bound. The L_1 distance is commonly used for quantifying the (dis)-similarity of data distributions [2], and is suitable for data distribution queries. The definition is as follows. Let the true readings of the sensors be $x_1, x_2, \ldots, x_n$; and let the data seen by the base station be $y_1, y_2, \ldots, y_n$.[2] The L_1 distance of these two sets of readings is $L_1 = \sum_i |y_i - x_i|$. Thus, if the user-specified precision threshold is E_u, the continuous data collection must ensure $L_1 < E_u$.

For simple aggregates such as SUM and AVG, we use a *linear additive error model* where $E = \sum_i (y_i - x_i)$. Thus, for the user-specified precision threshold E_u, the data collection should ensure $|E| < E_u$. This will be further elaborated when we discuss aggregate queries in Section V.

Note that our error model is a fixed bound. There are many variances in error modeling, such as probabilistic error models, weighted error models, etc. These models are heavily studied in the passive filter scheme. This paper does not try to be exhausted, and good summarization on error models can be found in [18], [25], and [31].

C. ROUTING TOUR OF DATA SWEEPER

The routing tour of the data sweeper is built during the initialization phase when the query is registered at the base station. The calculated next hop information will be maintained by each sensor node to forward the data sweeper. Our objective is to travel through every sensor node in the network.[3] This depends on the underlying topology of the sensor nodes. Below we give one possible scheme for a sensor network deployed in a grid structure and then generalize it to a random sensor network. We assume that all the sensor nodes in the network have the same communication range.

1) TOUR IN A GRID STRUCTURE

Let the sensor network form an $N \times M$ grid with each node at location a_{ij}, $i \in [0, N)$ and $j \in [0, M)$. We consider

[2]If the true reading of a sensor is not reported in the current collection round, its last reported data is seen by the base station.

[3]Note that when the data sweeper executes, not all sensor nodes are necessarily to be visited. The data sweeper may terminate earlier due to various reasons, e.g., use-up of the filter resource.

two cases: 1) N or M is an even number; and 2) N and M are both odd numbers. In this section, we discuss only the first case in the interest of space. The routing tour can be developed for the second case in a similar way.

Assume that each sensor node is able to communicate with the four adjacent neighbors around it, it can be verified that a complete tour exists.[4] The algorithm in Fig. 5 shows how to determine the next hop for each node.

```
Algorithm nextHopEven(a_ij)
1  if j ≠ 0, 1, M − 1
2      if i is even, nextHop = a_{i,j+1};
3      else nextHop = a_{i,j−1};
4  else if j = M − 1
5      if i is even, nextHop = a_{i+1,j};
6      else nextHop = a_{i,j−1}
7  else if j = 1
8      if i = N − 1, nextHop = a_{i,j−1}
9      else if i is even, nextHop = a_{i,j+1};
10     else nextHop = a_{i+1,j};
11 else if j = 0
12     if i ≠ 0, nextHop = a_{i−1,j};
13     else nextHop = a_{i,j+1};
```

FIGURE 5. Tour construction for a $N \times M$ grid where N is even number.

In our implementation, the sweeper routing tour is computed by the base station, which broadcasts it to each sensor node. We choose this strategy as the base station needs the tour information for important statistics estimation (see Section IV-E for detailed discussions). Nevertheless, the cost of the tour construction is only in the initialization phase, which would be amortized by the long-running continuous query.

2) TOUR IN A RANDOM SENSOR NETWORK

The routing tour of the data sweeper for a random sensor network in a rectangular region can be constructed as follows. Assume that the communication range is $\mathcal{R}$. First, we divide the sensor network into a virtual grid, with the size of each grid cell of $\frac{\mathcal{R}}{\sqrt{2}}$ such that any two nodes in a cell can communicate directly. A routing tour going through these cells can be constructed using the tour construction algorithm discussed in the last subsection. Second, for the sensor nodes within a grid cell, they form a complete connectivity graph since they all can hear from each other. Thus, a routing tour within the cell starts from a sensor node and jumps to any unvisited node. It is easy to prove that there will always be an unvisited node to choose unless all the sensor nodes have been visited.

We remark that the routing tour of the data sweeper is different from the routing structure for data collection, which is built in the networking layer. Furthermore, note that for a sensor node visited by the data sweeper along this tour, we do not guarantee that its data update report will be suppressed.

[4]For a grid with odd number of nodes, some of the nodes need to be able to communicate with the eight neighboring nodes.

This decision will be made by the sensor-sweeper interaction algorithm, which will be discussed next.

D. SENSOR-SWEEPER INTERACTION

When the data sweeper arrives at a sensor node, we need to determine whether to suppress its data update or not. Suppressing a data update will save the data traffic to report this update. It also consumes the filter size, which may restrict future data suppression. We study the trade-off in this subsection.

We first determine the gain for suppressing a data update. This is related to the underlying data collection structure, which is provided by the Routing Information module (Fig. 3). Given any data collection structure, a sensor node s_i is able to compute a gain for suppressing its data to the base station. We use G_i to denote this gain. For non-aggregate data collection and a tree-based data collection structure, G_i is equal to the depth of node s_i in the tree.

The objective of the data sweeper is to maximize the overall data reduction spanning the entire query period. Let x_{ij} and y_{ij} be the true reading and the last reported reading by node s_i at the jth data collection round. Let p_{ij} be an indicator variable, where $p_{ij} = 1$ if the update of s_i is suppressed in the jth round and $p_{ij} = 0$ otherwise. We are looking for an assignment of $\vec{p}^* = [p^*_{ij}]$ where we can maximize the overall data suppression while maintaining the error bound for each round of data collection. Thus, we have

$$\text{Maximize} \sum_{ij} p_{ij} G_i$$

$$\text{s.t.} \quad \forall j \sum_{i} p_{ij} |x_{ij} - y_{ij}| < E$$

As the sensor readings are known only online, we take a greedy strategy to solve this problem; that is, we aim to maximize the gain for each round:

$$P: \quad \text{Maximize} \sum_{i} p_i G_i$$

$$\text{s.t.} \sum_{i} p_i |x_i - y_i| < E \qquad (1)$$

We use Lagrangian Relaxation [20] to solve this problem. Multiply $\lambda > 0$ to constraint (1) and take the summation with the objective function, P can be re-written as:

$$P': L(\lambda) = \max \sum_{i} p_i (G_i - \lambda |x_i - y_i|) + \lambda E,$$

$$\lambda > 0, p_i \in \{0, 1\}$$

Claim 1: P' is an upper bound of P.

Proof: Let the optimal solution of P be $\vec{p}^* = [p^*_i]$. Thus, $L(\lambda) = \max \sum_i p_i (G_i - \lambda |x_i - y_i|) + \lambda E \geq \sum_i p^*_i (G_i - \lambda |x_i - y_i|) + \lambda E = \sum_i p^*_i G_i + \lambda (E - \sum_i p^*_i |x_i - y_i|) > \sum_i p^*_i G_i$, which is exactly P; the last inequality comes from the fact that $E > \sum_i p^*_i |x_i - y_i|$ as p^*_i is a feasible solution to P. ■

The dual problem is to find an optimal λ^* to minimize $L(\lambda)$:

$$P'': \quad \min_{\lambda \geq 0} \quad L(\lambda)$$

First, assume that we know the optimal λ^* for P''. The optimal solution $\vec{p}^*$ for $L(\lambda^*)$ in P' is: if $G_i > \lambda|x_i - y_i|$, $p_i^* = 1$, otherwise $p_i^* = 0$. Notice that $p_i^* = 1$ indicates that the data sweeper should suppress the update. Therefore, we should suppress only when $\frac{G_i}{|x_i - y_i|} > \lambda^*$. Thus, we develop the algorithm DataSuppressing as described in Fig. 6.

Algorithm DataSuppressing (E_r, T_s, T_f)
E_r: residual filter size, T_s: suppression threshold, T_f: forward threshold
1 **if** $|x_i - y_i| < E_r$ **and** $\frac{G_i}{|x_i - y_i|} > T_s$
2 mark x_i suppressed
3 $E_r = E_r - |x_i - y_i|$
4 **if** $E_r > T_f$
5 forward data sweeper to next hop
6 **else** mark the data sweeper stops at node i

FIGURE 6. **Algorithm: Data Suppression.**

In this algorithm, we use a suppression threshold T_s to represent the optimal λ^*. We have one additional forward threshold T_f. This T_f restricts the data sweeper from forwarding further. The intuition is that if the residual filter size is smaller than T_f, the data sweeper is unlikely to suppress more update reports in the remaining part of the sweeper tour.

E_r, T_s and T_f are carried by the data sweeper. Note that here we assume the optimal T_s is known. In the next section, we will discuss how these parameters are estimated.

E. STATISTICAL INFORMATION GATHERING AND ESTIMATION

Recall that T_s reflects the optimal λ^* for problem P''. A common technique to solve P'' (i.e., to obtain this optimal λ^*) is the subgradient method. We cannot use this method, however, due to two reasons. First, the subgradient method requires complete knowledge (of the sensor readings, etc.) so that an iterative search can be conducted for the optimal solution. This is against our lightweight design philosophy. Second, the sensor readings are known online. As a result, the optimal λ^* of the current collection round is not necessarily the optimal for the next round. This further makes the subgradient method unnecessary. In what follows, let $T_s(i)$ be the suppression threshold used in the ith round. We develop a heuristic to estimate $T_s(i+1)$ to be used in the next round.

We first take a look at the implication of T_s. If T_s is smaller than the optimal, it means that the filter size of the sweeper will be used up early during traveling and T_s should be increased. On the other hand, if T_s is larger than the optimal, there will be residual filter size left after traveling throughout the network and T_s should be decreased. We discuss these two cases in more detail below.

Case I: The data sweeper stops at sensor node s_k and T_s should be increased in the next round. Node s_k will report this information to the base station by piggybacking in its data report packet in the Phase II of data collection (see Section III). Let l denote the number of sensor nodes in the sweepers' routing tour that are left unvisited (the base station can figure out this information easily as it knows the sweeper tour and the location of s_k). We update $T_s(i) = \frac{T_s(i) \times N}{N - l}$; that is, we increase T_s by a factor proportional to the number of sensor nodes visited in the sweeper tour. Finally, we set $T_s(i+1)$ to be the average of the T_s's of the previous $\mathcal{H}$ rounds, i.e., $T_s(i+1) = \frac{\sum_{j=i-\mathcal{H}+1}^{i} T_s(j)}{\mathcal{H}}$.

Case II: The base station reclaims the data sweeper and T_s should be decreased. Let E_r be the residual filter size. Then we update $T_s(i) = \frac{T_s(i) \times (E - E_r)}{E}$; that is, we decrease T_s by a factor proportional to the E_r left. Finally, we set $T_s(i+1)$ to be the average of the T_s's of the previous $\mathcal{H}$ rounds, i.e., $T_s(i+1) = \frac{\sum_{j=i-\mathcal{H}+1}^{i} T_s(j)}{\mathcal{H}}$.

We next study T_f, the stop condition of the data sweeper. Intuitively, when the residual filter size is small, the data sweeper should stop traveling as it would only introduce more overhead to the system than the gain of suppressing update reports. Let $T_f(i)$ be the forward threshold of the ith round. We now develop an adaptive adjustment scheme for $T_f(i+1)$.

Let the height of the data collection tree be D. This is also the maximum gain of suppressing one data update report. Intuitively, we hope the data sweeper can suppress at least one data update report after traveling D steps. In other words, we would like to know the number of hops the data sweeper travels before it encounters a data update report with change smaller than T_f. To figure out this, we introduce an additional parameter L, which denotes the total number of updates smaller than $T_f(i)$ in the ith round. Specifically, L is initialized to 0; as the data sweeper travels along the tour, L increases for every data update (suppressed or not) that is smaller than $T_f(i)$. Still, let l denote the number of nodes left unvisited ($l = 0$ if the data sweeper returns). Therefore, $\frac{N-l}{L}$ denotes the number of hops before the data sweeper encounters a data change smaller than $T_f(i)$, and $\frac{N-l}{DL}$ represents the ratio of the overhead to the gain. We set $T_f(i)$ proportional to this ratio as $T_f(i) = \frac{T_f(i) \times (N-l)}{DL}$, and $T_f(i+1) = \frac{\sum_{j=i-\mathcal{H}+1}^{i} T_f(j)}{\mathcal{H}}$.

In the above scheme, $L = 0$ is possible if the initial value of T_f is poorly chosen. $L = 0$ indicates that either 1) T_f is too small so that no change is less than T_f, or 2) T_f is too large so that the data sweeper stops traveling in the very beginning. Thus, if the data sweeper stops at the first node, we set $T_f(i+1) = \frac{T_f(i)}{2}$, else we set $T_f(i+1) = T_f(i) \times 2$.

In summary, the data sweeper carries the statistical parameter L, and the algorithm parameters T_f and T_s for each query. The size of the data sweeper is constant and independent of the size of the sensor network. Note that T_f and T_s are adaptively adjusted online. As shown in our experiments (Section VI), both T_s and T_f are insensitive to their initial values.

V. FURTHER DISCUSSIONS

We have detailed the designs of the data sweeper. Our data sweeper and the data sweeper framework are general enough to accommodate different application requirements.

In particular, its integration with other in-network processing techniques is possible. In this section, we discuss various extensions to the basic data sweeper scheme.

1) MULTIPLE QUERIES

Instead of using separate data sweepers, multiple queries can share one data sweeper to save the migration overhead. For the queries that evaluate on the same data, e.g., MEDIAN and SUM of the temperature, they can even share a common filter size (the most restrictive filter size).

2) PARTIAL SWEEPER TOUR

In the framework, the sweeper routing tour is not necessarily to be a complete path involving all sensor nodes in the network. In some sensor networks, it is possible that the updates (or aggregated updates) of certain sensor nodes are very stable. In this case, it is preferred to put passive filters on these sensor nodes so as to reduce the migration overhead of the data sweeper.[5] Moreover, a complete path may not be easily found in some networks.

In these circumstances, one or more partial sweeper tours can be constructed, and a data sweeper is used for each sweeper tour. Nevertheless, the parameter adjustment algorithms of the data sweeper are largely unchanged.

3) INTEGRATION WITH OTHER IN-NETWORK PROCESSING TECHNIQUES

In Related Work, we discussed three possible in-network processing schemes: 1) in-network aggregation (e.g., for SUM query); 2) exploring spatial data correlation (e.g., clustering and sampling techniques); and 3) exploring temporal data correlation. Our data sweeper proposal falls in the third category. We now show how to integrate it with spatial data correlation and in-network aggregation techniques.

For spatial correlation, it can be integrated either with the Data Queue module or the Data Routing/Aggregation module in the routing layer (Fig. 1). Specifically, consider the sampling technique proposed in [26]. The sensor nodes can first apply the sampling scheme and then save all the data unsuppressed into the Data Queue module. Since [26] produces only approximate results, the total error bound E_u can be split into two sub-error bounds, E_s for the sampling technique and E_d for data sweeper, where $E_s + E_d \leq E_u$. Next consider the clustering technique proposed in [10]. The data sweeper can first suppress the data updates in the Phase I of our framework (Section III). In the Phase II of data collection, the Data Routing/Aggregation module can directly apply the clustering scheme over the unsuppressed data updates.

We use SUM as an example to illustrate the integration of the data sweeper with in-network aggregation. First, to bound the error of data collection, we can employ the linear additive model (see Section IV-A) instead of the L_1 model. In the linear additive model, negative data changes can be canceled out by positive data changes. The filter size of the data sweeper is initialized to 0. During the touring of the data sweeper, it is increased by a positive change or decreased by a negative change. The data update reports are suppressed as long as the absolute value of the current filter size is smaller than the given precision threshold E_u. Note that the linear additive model cannot be used for SUM in passive filtering schemes. This is because each sensor node has no knowledge of how other nodes use their filters, which disables the cancel-out effect and greatly wastes the overall filtering capability. This again reflects the intrinsic advantage of our data sweeper. Second, the data sweeper always completes the routing tour (by fixing $T_f = 0$). In contrast to the L_1 distance model, where the filter size monotonically decreases during the touring, with the linear additive model, the filter size may increase as well. Thus, we anticipate the data sweeper to suppress more update reports by completing the touring.

[5] If the data changes are completely predictable, no filter needs to be used.

4) THE DELAY OF DATA SWEEPER VISITING TOUR

We would like to point out that data sweeper introduce a delay due to the sweeper tour. As, the data sweeper needs to visit all sensor nodes, such traveling time is $O(N)$. Even if we apply k multiple sweepers, there is still a delay of $O(\frac{N}{k})$. The passive filter schemes do not have such delay.

Nevertheless, we comment that we are working on the sensor applications where we periodically update the sensor readings in rounds. The time between rounds is several orders longer than this sweeper traveling tour. This is because the natural of sensor network is to monitor the physical worlds. For example, in our real trace, we are working on the sensor network of Live from Earth and Mars project of the temperature, dew points etc. The time for physical environment to change (in the orders of minutes, hours or even days) dominates the time of the data sweeper tour.

VI. PERFORMANCE EVALUATION

A. SIMULATION SETUP

We have implemented our data sweeper scheme using ns-2 [17]. The initial energy reserve for each sensor node is set to 1 Joule. The energy consumption of sending and receiving a data packet is 24.75mW and 13.5mW [30], and we neglect the power consumption of the sensor nodes in sleeping state.

We test three 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 other two are real-world traces obtained from the Live from Earth and Mars (LEM) project [28] at the University of Washington. We used the dewpoint trace and temperature 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 3000 data points of these traces in Fig. 7. It can be easily seen that the temperature trace has a larger variation than the dewpoint trace. We have evaluated our algorithm against other traces from the LEM project, and similar performance trends are obtained.

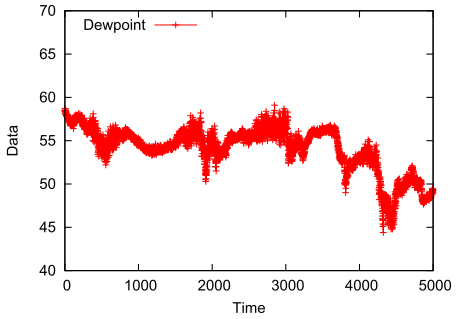


FIGURE 7. Dewpoint and temperature traces from LEM project.

We simulate various grid topologies for the sensor network. The default topology is a 7×7 grid. The base station is by default set to the center node of the network. The distance between two neighboring sensor nodes is set to 2m and the transmission power on the physical layer is set to 0.5×10^{-6}dBm. We implement the TAG [13] routing tree as the data collection structure in the routing layer. We evaluate a single query only; note that the performance advantage would be more significant for multiple queries since they can share the migration overhead incurred by the data sweeper. The query execution period is set to 10 minutes. For the synthetic data trace, the default error bound is set at 25, i.e., about 10% of the expected total change, which is 245. For the dewpoint and temperature traces, the error bounds are both set at 5. Since the temperature trace has a larger variation, this means that the temperature trace has a tighter precision requirement.

For our data sweeper scheme, the initial values of T_s and T_f are set to 0.2 and 1, respectively. The default value of $\mathcal{H}$ is set to 10. Since our scheme provide precision guarantee for data collection, we use lifetime as the major measurement metric for our system. The system lifetime is set to that of the first depleted sensor node.

We compare our data sweeper scheme with a state-of-art passive filtering technique [24], which have shown better performance as against previous designs [5], [18]. In this passive filtering technique, all sensor nodes are allocated with a uniform filter size initially. The base station periodically re-computes the filter size for each sensor node based on a set of parameters, including the node's residual energy, communication cost, and data changing pattern. A larger filter size will be given to the sensor nodes with a low residual energy, a high communication cost, and a high update rate. The new filter size is broadcasted every 50 rounds, which captures the balance between the overhead of this re-allocation and the adaptation to system dynamics [24]. We also compare with the mobile filter scheme [27]. In mobile filter scheme the total filter is allocated to the leaf nodes. The filters can be piggy-backed with the data update reports. Thus, the performance is better than the passive scheme [27]. Notice, however, this improvement comes with the inflexibility that it is only workable with a tree structure for routing the data update reports. Both our data sweeper and the passive filtering scheme are not affected by the underlying routing structure. Nevertheless, we employ the tree structure in our simulation so that the comparison can be made.

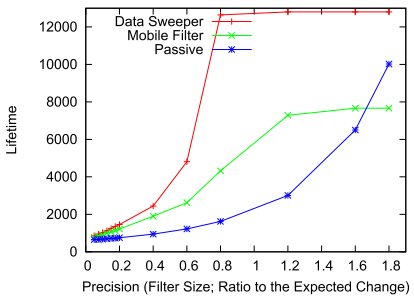


FIGURE 8. Lifetime as a function of precision settings for synthetic data.

B. SIMULATION RESULTS

We first evaluate the effect of precision settings (total filter size) of E. We show the network lifetime results for the synthetic data trace in Fig. 8, where the x-axis denotes the ratio of the filter size setting to the expected data change (i.e., ~ 250). We first observe that the data sweeper always performs better than the passive filtering scheme. Second, the lifetime of data sweeper improves quickly after the ratio of the filter size to the expected data change is above 0.4. After the ratio reaches 0.8, its lifetime flattens out. This is because the data sweeper can suppress almost all data updates; and the system is constrained only by the migration overhead incurred by the data sweeper. The passive filtering scheme, on the other hand, improves its lifetime slowly before the ratio is 1.2. Even when the ratio is 1.8, the performance of passive filtering is still lower than that of the data sweeper. This is explained as follows. First, theoretically speaking, when the ratio is larger than 1, in expectation the total filter size is able to suppress all the data updates, which is the case of the data sweeper. In the passive filtering scheme, however, as the total filter size is divided into smaller ones fixed to the sensor nodes, the data update variation still incurs many data updates unsuppressed even when the ratio is large. Second, the overhead incurred by the data sweeper is one packet sending and one packet receiving for each node in the sensor network; the energy consumption is balanced among all nodes. In contrast, in the passive filtering scheme, the sensor nodes near to the base station are hotspots. As long as there are non-suppressed update reports, these nodes will be involved to relay update reports, thereby deteriorating the network lifetime.

As for mobile filter, the performance is generally better than the passive filtering scheme; this confirms the results in [27]. Our data sweeper prevails as the mobile filter still faces the problem where the total filter size has to be divided and allocated to the leaf nodes. In addition, the mobile filter has larger overhead on the hotspots near the base station, making its performance worse than the passive

filtering scheme when the total filter size is very big; this is also observed in [27].

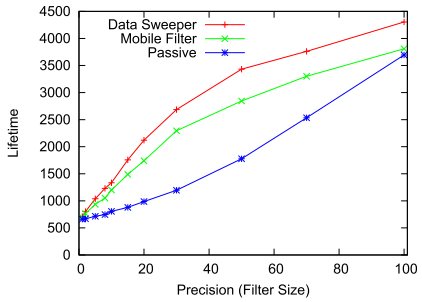


FIGURE 9. Lifetime as a function of precision settings for dewpoint trace.

For real-world traces, we show the result in Fig. 9, where the filter size is changed from 0 to 100.[6] As can be seen, the performance of data sweeper improves fast when the filter size is above 20. The passive filtering scheme approaches the data sweeper as the filter size goes towards 100. Based on the study in the synthetic data trace, this indicates that at a filter size of around 100, the system reaches a stage where almost all data updates are suppressed. We set the same filter sizes to evaluate the performance for the temperature trace. In Fig. 10, we can see that there is still a large performance gap between the passive filtering and data sweeper when the filter size is 100. This is because the temperature trace has a larger data variation as shown in Fig. 7.

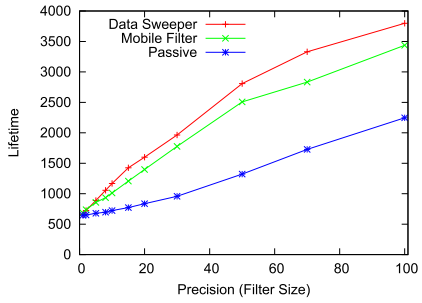


FIGURE 10. Lifetime as a function of precision settings for temperature trace.

In real applications, we believe that the results may not be meaningful for very large filter sizes (i.e., very low precision) where most updates are suppressed. Therefore, in the rest of this paper, we will focus on small and moderate filter sizes.

We show the effect of $\mathcal{H}$, the number of previous rounds for the calculation of T_s and T_f, in Figs. 11, 12, and 13. We can see that, although small, a larger $\mathcal{H}$ generally leads to a better estimation of T_s and T_f and improves the system lifetime. As $\mathcal{H}$ only incurs overhead of small memory space in the base station, we set $\mathcal{H}$ to be 10 for the rest of the experiments.

[6]We do no plot the filter size in terms of the ratio to the average data update as the updates do not follow a regular pattern in real traces.

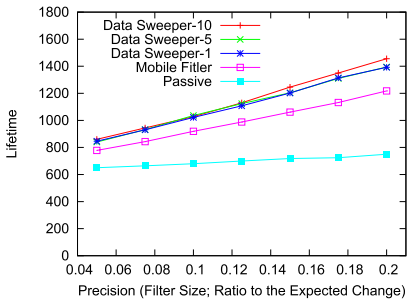


FIGURE 11. Lifetime as a function of precision settings for synthetic data.

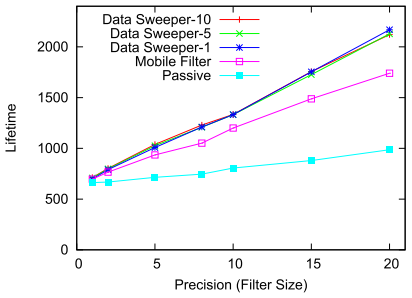


FIGURE 12. Lifetime as a function of precision settings for dewpoint trace.

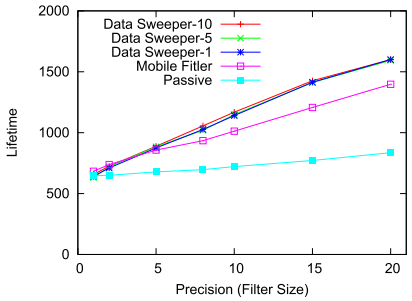


FIGURE 13. Lifetime as a function of precision settings for temperature trace.

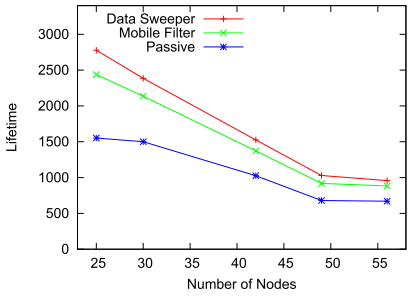


FIGURE 14. Lifetime as a function of the number of nodes for synthetic data.

We next study the effect of the network size. Fig. 14 shows the performance for grid topologies with different number of nodes. It can be seen that the network lifetime decreases when the network size increases. This is because with

a fixed precision requirement, a larger network allocates less filters to each node on average. The data sweeper performs consistently better than the passive filtering scheme by at least 20% - 30% to as much as 80%. It also outperforms mobile filter scheme for a margin of a 5% to 20%. Similar trends are observed for the dewpoint and temperature traces in Figs. 15 and 16.

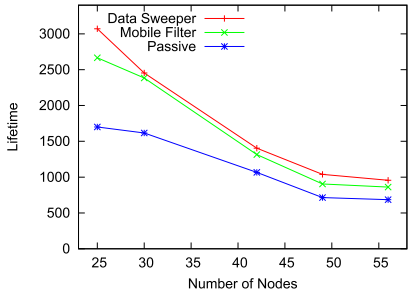


FIGURE 15. Lifetime as a function of the number of nodes for dewpoint trace.

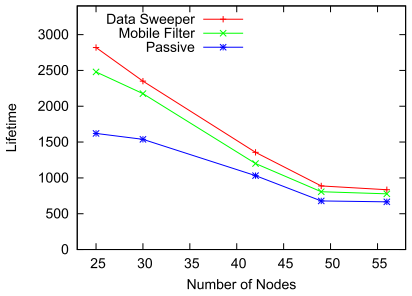


FIGURE 16. Lifetime as a function of the number of nodes for temperature trace.

We next examine the setting of the parameters T_s and T_f of our data sweeper and evaluate our statistical parameter adjustment schemes. In Fig. 17, the x-axis represents T_s, and D-0.2 and D-1 denote that T_f is initially set to 0.2 and 1. We also show the results where the values of T_s and T_f are both fixed throughout the simulation (denoted by D-0.2-Fix and D-1-Fix). First, we can see that T_s has a higher impact than T_f as different T_s's result in larger fluctuation in lifetime. This is because T_f is used mainly for control the overhead

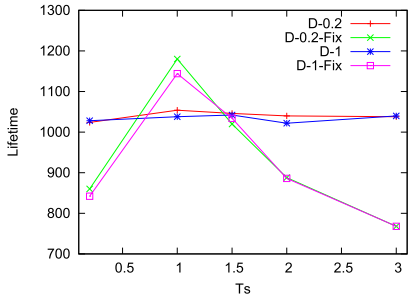


FIGURE 17. Lifetime as a function of T_S for synthetic data.

of forwarding the data sweeper, whereas T_s has directly impact on data suppression decisions. Obviously the data traffic dominates system lifetime. Second, setting $T_s = 1$ and $T_f = 0.2$ fixed achieves the best performance among all settings tested, i.e., 1200 rounds. These values, however, depend on the trace data and are difficult to know in advance. Our statistical parameter adjustment scheme performs close to the best performance and is insensitive to the initial settings of T_f and T_s, i.e., for all the initial values of T_s and T_f tested, our scheme consistently has a lifetime of 1050 rounds. Similar effect is observed in other traces (See Figs. 18 and 19). These show the very desirable self-adjustable feature of our scheme.

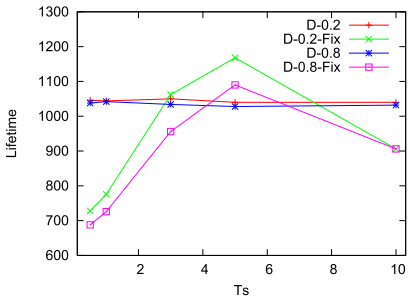


FIGURE 18. Lifetime as a function of T_S for dewpoint trace.

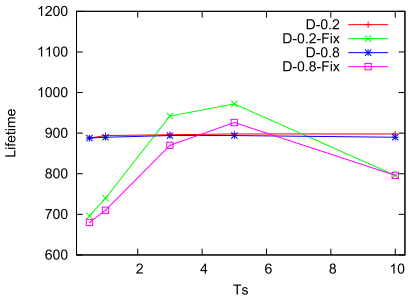


FIGURE 19. Lifetime as a function of T_S for temperature trace.

We next study the impact of different underlying topologies. We place the base station at different positions of the network. In all topologies, the TAG tree is employed as the data collection structure. The results are shown in Figs. 20, 21, and 22. The network lifetime is the longest when the base station is at the center of the grid. This is not surprising, as the lifetime of a sensor network is usually constrained by the sensor nodes near to the base station. Setting the base station at the corner of the network will make the system more stressed. Nevertheless, we can see that the relative performance of passive filtering and data sweeper is hardly affectedly by the underlying topology.

C. RESULTS FOR AGGREGATION

In this section, we further study the effect of the data sweeper with in-network aggregation. We perform data collection for SUM queries. To have a fair comparison, the passive filtering

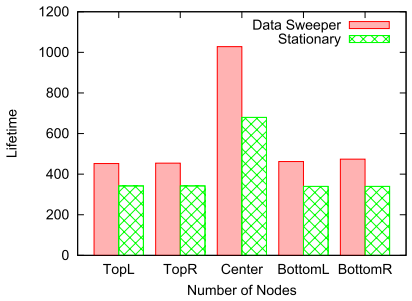


FIGURE 20. Lifetime for different underlying routing topologies for synthetic data.

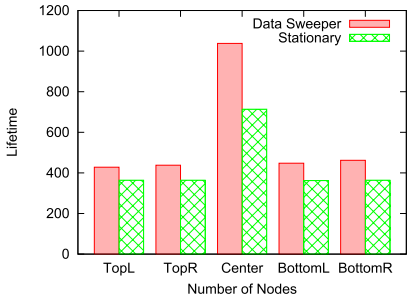


FIGURE 21. Lifetime for different underlying routing topologies for dewpoint trace.

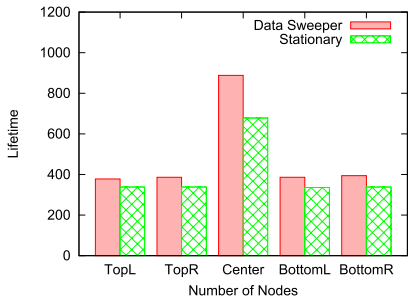


FIGURE 22. Lifetime for different underlying routing topologies for temperature trace.

scheme is also integrated with in-network aggregation [24]. As the mobile filter scheme does not apply for aggregation, we omit the comparison with it.

In Fig. 23, we plot the results for both the default 7×7 grid and a 9×9 grid. When the filter size is very small (with a ratio of 0.04 to the expected update), the passive filtering scheme performs better for the default grid. In this case, most data packets are not suppressed by the filter, but by in-network aggregation. Therefore, the overhead of the data sweeper is perceptible. As the filter size increases, the improvement of the data sweeper becomes clearer. As discussed in the last section, a larger network size results in a worse performance with the same filter size setting. However, it is surprising to see that the data sweeper achieves a better performance for the 9×9 grid than the 7×7 grid. By looking into the simulation, we find that a significant portion of the benefit for

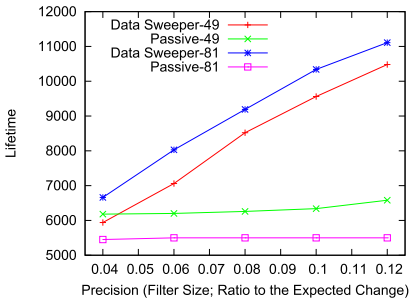


FIGURE 23. Lifetime as a function of precision settings for synthetic data with aggregation.

the data sweeper comes from the fact that it is able to cancel out the positive and negative data updates during in-network processing. Given the same precision ratio, a larger network has a larger total filter size, and offers more such cancel-out opportunities.

For the dewpoint trace, as shown in Fig. 24, still, the data sweeper outperforms the passive filtering scheme for all large filter sizes. In contrast to the synthetic data trace, the data sweeper has a better performance for the 7×7 grid than the 9×9 grid. This is because we use fixed total filter sizes. As the filter size increases, both schemes improve and the data sweeper has a larger improvement than the passive filtering. This again implies the scalability of our data sweeper scheme. The result for the temperature trace is shown in Fig. 25. The gain of the data sweeper is smaller, as the temperature

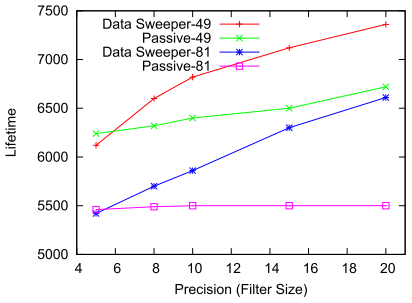


FIGURE 24. Lifetime as a function of precision settings for dewpoint trace with aggregation.

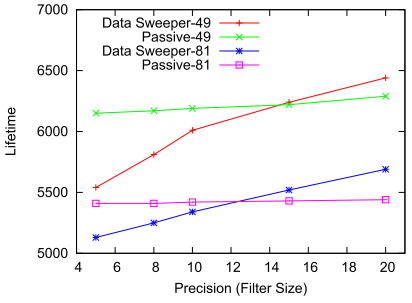


FIGURE 25. Lifetime as a function of precision settings for temperature trace with aggregation.

trace has a larger variation as shown in Fig. 7. From these results, for aggregate queries with very small filter sizes, it could be an interesting future work to study a hybrid scheme with the data sweeper travels a partial tour and the passive filters allocated on bottleneck nodes.

VII. CONCLUSION

Existing in-network processing techniques for continuous sensor data collection, such as filtering, work passively with data collection, i.e., the filter sticks to a sensor node and suppresses the data updates on this node only. In this paper, we suggested a fundamentally novel scheme called data sweeper that proactively migrates and suppresses traffic in the sensor network. This scheme, implementing through a lightweight data sweeper framework, offers two significant benefits: 1) The data sweeper migrates in the network and has a better knowledge of the data updates on the sensor nodes. Thus, it is able to suppress significantly more data traffic and balances the energy consumption; 2) The data sweeper works asynchronously with data collection; making this framework generic and flexible, and thus well suits diverse query types and data collection structures.

We demonstrated a concrete design for shifting filtering operations to the data sweeper framework, and developed a comprehensive set of algorithms for sweeper touring, sensor-sweeper interaction, and sweeper self-adjustment. Through both analysis and experiments, we showed that great performance benefit is achieved.

REFERENCES

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

[2] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White, "Testing that distributions are close," in *Proc. IEEE FOCS*, Redondo Beach, CA, USA, Nov. 2000, pp. 259–269.

[3] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Energy and bandwidth-efficient wireless sensor networks for monitoring high-frequency events," in *Proc. 10th IEEE SECON*, New Orleans, LA, USA, Jun. 2013, pp. 194–202.

[4] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate data collection in sensor networks using probabilistic models," in *Proc. 22nd IEEE ICDE*, Atlanta, GA, USA, Apr. 2006, p. 48.

[5] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical in-network data aggregation with quality guarantees," in *Proc. EDBT*, Heraklion, Greece, Mar. 2004, pp. 658–675.

[6] C. T. Ee *et al.*, "A modular network layer for sensorsets," in *Proc. 7th USENIX OSDI*, Seattle, WA, USA, Nov. 2006, pp. 249–262.

[7] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proc. 5th ACM MOBICOM*, Seattle, WA, USA, Aug. 1999, pp. 263–270.

[8] M. B. Greenwald and S. Khanna, "Power-conserving computation of order-statistics over sensor networks," in *Proc. ACM PODS*, Paris, France, Jun. 2004, pp. 275–285.

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

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd HICSS*, Wailea Maui, HI, USA, Jan. 2000, pp. 3005–3014.

[11] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proc. ACM ASPLOS*, San Jose, CA, USA, Oct. 2002, pp. 96–107.

[12] X. Liu, J. Cao, S. Tang, and P. Guo, "A generalized coverage-preserving scheduling in WSNs: A case study in structural health monitoring," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr./May 2014, pp. 718–726.

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

[14] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. ACM WSNA*, Atlanta, GA, USA, Sep. 2002, pp. 88–97.

[15] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and deltas: Efficient and robust aggregation in sensor network streams," in *Proc. ACM SIGMOD*, Baltimore, MD, USA, Jun. 2005, pp. 287–298.

[16] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proc. ACM SENSYS*, Baltimore, MD, USA, Nov. 2004, pp. 250–262.

[17] *The Network Simulator ns-2*. [Online]. Available: http://www.isi.edu/nsnam/ns/

[18] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proc. ACM SIGMOD*, San Diego, CA, USA, Jun. 2003, pp. 563–574.

[19] D. Pan, D. Wang, J. Cao, Y. Peng, and X. Peng, "Minimizing building electricity costs in a dynamic power market: Algorithms and impact on energy conservation," in *Proc. 34th IEEE RTSS*, Vancouver, BC, Canada, Dec. 2013, pp. 107–117.

[20] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. New York, NY, USA: Halsted, Apr. 1993.

[21] N. Shrivastava, C. Buragohain, S. Suri, and D. Agrawal, "Medians and beyond: New aggregation techniques for sensor networks," in *Proc. ACM SENSYS*, Baltimore, MD, USA, Nov. 2004, pp. 239–249.

[22] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: On energy-efficient continuous monitoring in sensor networks," in *Proc. ACM SIGMOD*, Chicago, IL, USA, Jun. 2006, pp. 157–168.

[23] S. Tang and J. Wu, "Qute: Quality-of-monitoring aware sensing and routing strategy in wireless sensor networks," in *Proc. MobiHoc*, 2013, pp. 119–126.

[24] X. Tang and J. Xu, "Extending network lifetime for precision-constrained data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006, pp. 1–12.

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

[26] D. Wang, Y. Long, and F. Ergun, "A layered architecture for delay sensitive sensor networks," in *Proc. IEEE SECON*, Santa Clara, CA, USA, Sep. 2005, pp. 24–34.

[27] D. Wang, J. Xu, J. Liu, and F. Wang, "Mobile filtering for error-bounded data collection in sensor networks," in *Proc. IEEE ICDCS*, Beijing, China, Jun. 2008, pp. 530–537.

[28] (2006). *Live From Earth and Mars (LEM) Project*. [Online]. Available: http://www-k12.atmos.washington.edu/k12/grayskies

[29] X. Xu, X.-Y. Li, P.-J. Wan, and S. Tang, "Efficient scheduling for periodic aggregation queries in multihop sensor networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 690–698, Jun. 2012.

[30] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. IEEE INFOCOM*, New York, NY, USA, Jun. 2002, pp. 1567–1576.

[31] Y. Zhang, "Evaluating continuous probabilistic queries over constantly-evolving data," M.S. thesis, Dept. Comput. Sci., Univ. Hong Kong, Hong Kong, 2010.

DAN WANG received the B.Sc. degree from Peking University, Beijing, the M.Sc. degree from Case Western Reserve University, Cleveland, OH, and the Ph.D. degree from Simon Fraser University, Vancouver, Canada, all in computer science. He is currently an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interest includes wireless sensor networks, internet routing, and applications.

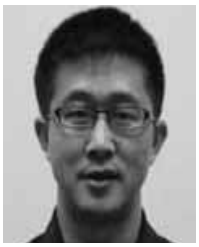
JIANGCHUAN LIU was an Assistant Professor with the Chinese University of Hong Kong, from 2003 to 2004. He is currently a Full Professor with the School of Computing Science, Simon Fraser University, BC, Canada. He was also an NSERC E. W. R. Steacie Memorial Fellow, and an EMC-Endowed Visiting Chair Professor with Tsinghua University, Beijing, China, from 2013 to 2016.

He received the B.Eng. (*c*um laude) degree in computer science from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, in 2003. He was a co-recipient of the ACM TOMCCAP Nicolas D. Georganas Best Paper Award in 2013, the ACM Multimedia Best Paper Award in 2012, the IEEE Globecom 2011 Best Paper Award, and the IEEE Communications Society Best Paper Award on Multimedia Communications in 2009.

His research interests include multimedia systems and networks, cloud computing, social networking, online gaming, big data computing, crowdsourcing, wireless sensor networks, and peer-to-peer networks. He has served on the Editorial Boards of the IEEE TRANSACTIONS ON BIG DATA, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, the IEEE ACCESS, the IEEE INTERNET OF THINGS JOURNAL, *Computer Communications* (Elsevier), and *Wireless Communications and Mobile Computing* (Wiley).

JIANLIANG XU received the B.Eng. degree in computer science and engineering from Zhejiang University, Hangzhou, China, and the Ph.D. degree in computer science from the Hong Kong University of Science and Technology. He held visiting positions with Pennsylvania State University, State College, PA, USA, and Fudan University, Shanghai, China. He is currently a Professor with the Department of Computer Science, Hong Kong Baptist University (HKBU). He has authored over 150 technical papers in these areas. His research interests include data management, mobile/pervasive computing, and networked and distributed systems. He was a recipient of the IEEE ICDE Outstanding Reviewer Award in 2010, and the HKBU Faculty Performance Award for Outstanding Young Researcher in 2012. He has served on the Editorial Boards of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING and the *International Journal of Distributed Sensor Networks*. He was the Vice Chairman of the Hong Kong Chapter of Association for Computing Machinery.

HONGBO JIANG received the B.S. and M.S. degrees from the Huazhong University of Science and Technology, China, and the Ph.D. degree from Case Western Reserve University, in 2008. He joined the faculty of Huazhong University of Science and Technology, where he is currently a Full Professor. His research concerns computer networking, in particular, algorithms and protocols for wireless networks and mobile computing.



CHONGGANG WANG received the Ph.D. degree from the Beijing University of Posts and Telecommunications, China, in 2002. He is currently a member of the Technical Staff of InterDigital Communications with focuses on Internet of Things (IoT) research and development activities, including technology development and standardization. His current research interest includes IoT, mobile communication and computing, and big data management and analytics. He is a Founding Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, and on the Editorial Board of several journals, including the IEEE TRANSACTIONS ON BIG DATA and the IEEE ACCESS. He is also the IEEE ComSoc Distinguished Lecturer from 2015 to 2016.