Top-k Monitoring in Wireless Sensor Networks

Minji Wu, *Student Member*, *IEEE*, Jianliang Xu, *Member*, *IEEE*, Xueyan Tang, *Member*, *IEEE*, and Wang-Chien Lee, *Member*, *IEEE*

Abstract—Top-k monitoring is important to many wireless sensor applications. This paper exploits the semantics of top-k query and proposes an energy-efficient monitoring approach called *FILA*. The basic idea is to install a filter at each sensor node to suppress unnecessary sensor updates. Filter setting and query reevaluation upon updates are two fundamental issues to the correctness and efficiency of the FILA approach. We develop a query reevaluation algorithm that is capable of handling concurrent sensor updates. In particular, we present optimization techniques to reduce the probing cost. We design a skewed filter setting scheme, which aims to balance energy consumption and prolong network lifetime. Moreover, two filter update strategies, namely, eager and lazy, are proposed to favor different application scenarios. We also extend the algorithms to several variants of top-k query, that is, order-insensitive, approximate, and value monitoring. The performance of the proposed FILA approach is extensively evaluated using real data traces. The results show that FILA substantially outperforms the existing TAG-based approach and range caching approach in terms of both network lifetime and energy consumption under various network configurations.

Index Terms—Sensor network, data management, energy efficiency, top-k, continuous query.

1 Introduction

Recent advances in signal processing, microelectronics, and wireless communications have enabled the deployment of large-scale sensor networks for many applications such as habitat and environment monitoring [28]. A wireless sensor network typically consists of a base station and a group of sensor nodes (see Fig. 1). The base station serves as a gateway for the sensor network to exchange data with external users. The sensor nodes, on the other hand, are responsible for sensing and collecting data from their local environments. They are also capable of processing sensed data and communicating with their neighbors and the base station.

Monitoring aggregate forms of sensed data is important to many sensor applications and has drawn a lot of research attention [6], [7], [16], [19], [29], [30]. Among those aggregates, a top-k query requests the list of k sensor nodes with the highest (or lowest) readings. For example:

- **Environmental Monitoring**. Consider an environment-monitoring sensor network. A top-k query is issued to find out the nodes and their corresponding areas with the highest pollution indexes for the purpose of pollution control or research study.
- **Network Management**. Power supply is critical to the operation of a wireless sensor network. Thus, a top-k query may be issued to continuously monitor the sensor nodes with the least residual energy so that these sensor nodes can be instructed to adapt themselves (for example, reducing sampling rates) to extend network lifetime.

A naive implementation of monitoring top-k query is to use a centralized approach in which all sensor readings are periodically collected by the base station, which then computes the top-k result set. To reduce network traffic in data collection, an *in-network aggregation* technique, known as *TAG*, has been proposed [16]. In this approach, a routing tree rooted at the base station is first established, and the data are then aggregated and collected along the routing tree to the base station. Consider an example shown in Fig. 2a, where sensor nodes A, B, and C form a routing tree. The readings of these sensor nodes at three successive sampling instances t_1, t_2, and t_3 are shown in the tables in Fig. 2a. Suppose we are monitoring a top-1 query. Employing TAG, at each sampling instance, nodes B and C send their current readings to the parent (that is, node A), which compares the data received with its own reading and sends the highest reading to the base station. In this example, a total of nine messages are sent. Node B is involved in the message exchange at each sampling instance though the top-1 result is always node C. Therefore, this approach incurs unnecessary updates in the network and is not energy efficient.

In this paper, we exploit the semantics of top-k query and propose a novel filter-based monitoring approach called *FILA*. The basic idea is to install a filter at each sensor node to suppress unnecessary sensor updates. The base station also keeps a copy of the filter setting to maintain an (error bounded) *approximate view* of each node's reading. A sensor node updates its reading with the base station only when the reading passes the filter. The correctness of the top-k result is ensured if all sensor nodes perform updates according to their filters. Fig. 2b shows an

- *M. Wu and J. Xu are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, KLN, Hong Kong. E-mail: {alexwu, xujl}@comp.hkbu.edu.hk.*
- *X. Tang is with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. E-mail: asxytang@ntu.edu.sg.*
- *W.-C. Lee is with the Department of Computer Science and Engineering, The Penn State University, University Park, PA 16802. E-mail: wlee@cse.psu.edu.*

Manuscript received 3 Mar. 2006; revised 13 Oct. 2006; accepted 11 Jan. 2007; published online 7 Mar. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0132-0306.
Digital Object Identifier no. 10.1109/TKDE.2007.1038.

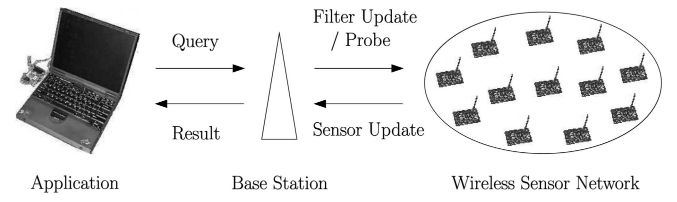


Fig. 1. The system architecture.

example, where the base station has collected the initial sensor readings and installed three filters [20, 39], [39, 47], and [47, 80] at sensor nodes A, B, and C, respectively. At sampling instances t_1 and t_2, no update is reported, since all updates are filtered out by the nodes' respective filters. At instance t_3, the updated reading of node B (that is, 48) passes its filter [39, 47]. Hence, node B sends the reading 48 to the base station via node A (Step 1). Since 48 lies in the filtering window of node C (that is, [47, 80]), the top-1 result becomes undecided at the base station, as either node B or node C can have the highest reading. Therefore, the base station probes node C for its current reading to reevaluate the top-1 result (Steps 2 and 3). Thus, a total of four update messages and one probe message are incurred in this approach.[1] Compared with the aforementioned TAG-based approach, five update messages are saved at the cost of one probe message. Clearly, this approach achieves a better performance than the TAG-based approach.

However, in order to make FILA work efficiently, two fundamental issues have to be addressed:

- What are the requirements of the filter settings such that the top-k result set is correctly returned if all nodes perform updates according to their filters? Given the correctness requirements, how can we set the filter for each sensor node to balance their energy consumption and maximize network lifetime? In the above example, if nodes B and C have the filters set to [39, 50] and [50, 80], respectively, then no update needs to be reported at any of the three sampling instances.
- Upon receiving updates from the sensor nodes, how does the base station reevaluate the top-k result, and how can we adjust the affected filters? In particular, it is likely that multiple nodes update during the same sampling interval. Thus, query reevaluation needs to handle concurrent updates.

These issues are essential to the correctness and efficiency of the FILA approach and are investigated in this paper. Our contributions are summarized as follows:

- We investigate top-k monitoring in wireless sensor networks, which distinguish themselves from traditional distributed networks in system architecture and performance concern. On one hand, we can take advantage of the architecture of sensor network (for example, hierarchical multihop routing) to improve performance. On the other hand, sensor nodes are battery powered. When a certain portion of the nodes run out of their battery power and lose their coverage, the whole network would be down. Thus, balancing the energy consumption of the sensor nodes to prolong network lifetime becomes a primary performance objective in sensor networks (as opposed to reducing network traffic in traditional distributed networks).
- We exploit the semantics of top-k query and propose a filter-based approach called FILA for monitoring top-k query (and its variants) in wireless sensor networks.
- We develop a query reevaluation algorithm that is capable of handling concurrent updates for FILA. In particular, we present optimization techniques to reduce the probing cost.
- We design a skewed filter setting scheme, which aims to balance energy consumption and prolong network lifetime. Moreover, two filter update strategies (that is, *eager* and *lazy*) are proposed to favor different application scenarios.
- Extensive experiments are conducted to evaluate the performance of the proposed FILA approach by using real data traces. The results provide a number of insightful observations and show that FILA substantially outperforms TAG [16] and range caching [25] in terms of both energy consumption and network lifetime under various network configurations.

1. For simplicity, the overhead for initial data collection and filter setting is not shown here, but is counted in our experiments.

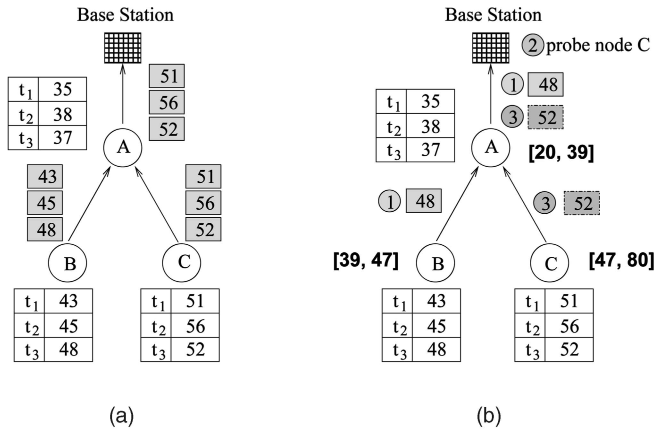


Fig. 2. An example of top-k monitoring. (a) TAG. (b) FILA.

The remainder of this paper proceeds as follows: Section 2 reviews the related work on top-k query processing in distributed environments. Section 3 presents our proposed approach, FILA, and discusses how we can set and maintain the filter for each sensor node and how we can reevaluate the top-k query result when updates occur. We extend FILA to handle approximate top-k monitoring, order-insensitive top-k monitoring, and top-k value monitoring in Section 4. The FILA approach is experimentally evaluated in Section 5. Finally, we conclude this paper and present some future research plans in Section 6.

2 Related Work

Evaluating top-k queries in distributed networks has been extensively studied in the literature (for example, [4], [5],

[10], [11], [18], [33], [41]). A typical assumption is that the ranking score of an object should be aggregated from a number of attribute values stored at distributed data sources (formally called *vertically partitioned data sets*). The best known algorithm is the threshold algorithm (TA) [10], [11], [21]. Although TA requires data sources to support sorted accesses, Marian et al. [18] proposed the Upper algorithm for sources that support random accesses only. Cao and Wang [4] developed a three-phase uniform threshold (TPUT) algorithm, which significantly reduces remote accesses in large networks. In [33], Theobald et al. further extended TA by introducing a family of approximate variables based on probabilistic arguments to reduce runtime costs. Michel et al. [20] proposed a flexible framework for distributed top-k algorithms, which allows for trade-off efficiency against result quality and bandwidth saving against the number of communication phases. While the above approaches assumed a single-hop communication network, Zeinalipour-Yazti et al. [42] proposed a distributed threshold join algorithm (TJA) to exploit in-network aggregation for multihop sensor networks. More recently, Silberstein et al. [26] developed a sampling-based approach to evaluate approximate top-k queries in sensor networks. Top-k processing algorithms have also been developed for peer-to-peer networks [2] and private databases [36]. However, all these studies have focused on *snapshot* top-k queries, whereas we are interested in monitoring continuous top-k queries in this paper. As pointed out in [1], although continuous monitoring could be simulated by repeatedly executing a snapshot query, many snapshot queries would be executed in vain if the answer remains unchanged, thereby being cost inefficient. Moreover, it is difficult to determine the optimal frequency of repeated query executions.

Babcock and Olston [1] did an inspiring work on monitoring continuous top-k queries over distributed data sources. Their idea is to add an adjustment factor to each source to ensure that the local top-k list aligns to the global top-k list maintained at the coordinator. Focusing on vertically partitioned data sets, their algorithm maintains an invariant that the adjustment factors allocated to different sources for each data object sum to zero. When the algorithm is applied to nonpartitioned data (our problem setting), each object is allocated with an adjustment factor of 0, which degenerates to the TAG approach discussed in Section 1. Thus, it is not effective to our problem. Moreover, their work was limited to order-insensitive top-k monitoring; the more challenging order-sensitive top-k monitoring problem was not studied. Our work also bears some similarity to [24], which installs filters for monitoring average and sum aggregates over distributed data streams. However, the filter usages are different: Olston et al. [24] uses filters to bound error in query result, whereas we leverage filters to maintain top-k ordering. In addition, both [1] and [24] aimed at reducing update traffic in a single-hop network. In contrast, our main performance objective is to extend network lifetime for a multihop sensor network.

Monitoring of aggregation functions (such as average, sum, count, min, and max) in sensor networks has been investigated in the past few years. However, the main focus has been on how to establish the routing architecture and how to apply in-network aggregation techniques to reduce network traffic [6], [14], [16], [30], [38]. Approximate monitoring schemes have been proposed for average and sum aggregates in [7], [29], [32]. In [39], a clustered aggregation (CAG) technique was proposed for approximate query processing. Compared with TAG, CAG can reduce the message transmissions (particularly for lower level nodes) for data collection. However, CAG does not alleviate the hot spots, since the cluster heads need to report all their readings to the base station, and these messages have to be relayed by the hot-spot nodes (that is, the root's direct children). Different from [39], we exploit the semantics of top-k query and propose a new method to prolong network lifetime. In a separate study, Silberstein et al. [27] explored techniques for monitoring extreme values, with the objective of minimizing network traffic. Data storage and query processing for snapshot queries in sensor networks have also been studied (for example, [3], [8], [9], [12], [15], [17], [37]). These works focused on applications different from ours.

Another related work is *range caching*, proposed by Olston et al., for approximate query processing [25]. The base station caches a value range (bounded by an *approximation range*) for the value at each source node. A source updates with the base station only when the new value is beyond the approximation range of the previously reported value. For query processing, the base station first computes a tentative result based on cached value ranges. If the tentative result is not sufficiently precise, then the base station refreshes some necessary nodes and recomputes the query result based on refreshed values. The range caching approach has been adapted for answering top-k queries in sensor networks [35]. However, as range caching does not explore the semantics of top-k query, its performance is not as good as FILA, as will be shown in Section 5.

3 TOP-k MONITORING

We first describe the system model and give a formal problem definition in Section 3.1. Then, Section 3.2 provides an overview of the proposed FILA monitoring approach. The query reevaluation, filter setting, and filter update algorithms are discussed in detail in Sections 3.3, 3.4, and 3.5, respectively.

3.1 System Model and Problem Definition

We consider a wireless sensor network, as depicted in Fig. 1. It is assumed that the base station has a continuous power supply. In contrast, the sensor nodes are powered by battery. The radio coverage of a sensor node is constrained to a local area. When the base station is beyond a sensor node's radio coverage, a routing infrastructure (for example, a TAG tree [16]) is used to route data to the base station.

Each sensor node i measures the local physical phenomenon v_i (for example, pollution index, temperature, or residual energy) at a fixed sampling rate. Without loss of generality, we consider a top-k monitoring query that continuously requests the (ordered) list of sensor nodes $\mathcal{R}$ with the highest readings, that is

$$\mathcal{R} = < n_1, n_2, \ldots, n_k >,$$

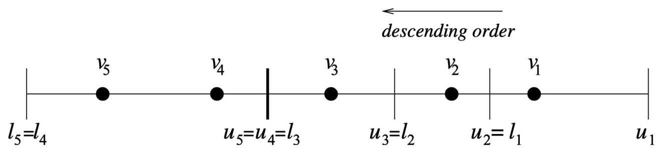


Fig. 3. Filter settings for top-3 monitoring.

where

$$\forall i < j, v_{n_i} \geq v_{n_j}$$

and

$$\forall l \neq n_i (i = 1, 2, \ldots, k), v_l \leq v_{n_k}.$$

The monitoring result is maintained by the base station and supplied to external users. To produce query results continuously, the monitoring algorithm controls when and how sensor readings should be collected to the base station.

3.2 FILA Overview

Initially, the base station collects the readings from all sensor nodes. It sorts the sensor readings and obtains the initial top-k result set. Then, the base station computes a filter (represented by a window $[l_i, u_i]$) for each sensor node i and sends it to the node for installation. At the next sampling instance, if the new reading of sensor node i is within $[l_i, u_i]$, then no update is sent to the base station. Otherwise, if the new reading changes beyond the filtering window and passes the filter, then an update is sent to the base station. The base station will then reevaluate the top-k result and adjust the filter setting(s) for relevant sensor node(s) if necessary. The query reevaluation algorithm will be discussed in detail in Section 3.3.

The purpose of using filters is to filter out unnecessary sensor updates and thereby suppress the traffic in the network. The correctness of the top-k result must be guaranteed, provided that all sensor nodes perform updates according to their filters. Thus, the filter settings have to be carefully planned in a coordinated manner. Denote the current reading of node i as v_i. Without loss of generality, we number the sensor nodes in decreasing order of their current readings, that is, $v_1 \geq v_2 \geq \cdots \geq v_N$, where N is the number of sensor nodes under monitoring. Intuitively, for the correctness of monitoring, the filters assigned to the nodes in the top-k result set should cover their current readings but do not overlap with each other. On the other hand, the nodes in the nontop-k set could share the same filter setting. Thus, we consider the filter settings for the top-$(k+1)$ nodes only. A *feasible* filter setting $[l_1, u_1], [l_2, u_2], \ldots, [l_k, u_k], [l_{k+1}, u_{k+1}]$ must satisfy

$$\begin{cases} v_1 \leq u_1; \\ v_{i+1} \leq u_{i+1} \leq l_i \leq v_i, \quad (1 \leq i \leq k); \\ l_{k+1} \leq v_N. \end{cases} \tag{1}$$

We propose to set u_{i+1} equal to l_i in order to maximize the filtering capability and, hence, reduce the sensor nodes' energy consumption in sending updates. Fig. 3 shows a feasible filter setting for top-3 monitoring, where nodes 4 and 5 share a filter setting, and u_{i+1} is set equal to l_i for $1 \leq i \leq 3$. As can be seen, *a filter setting is a (constrained) partitioning of the data space*. The filter setting algorithms will be presented in Section 3.4.

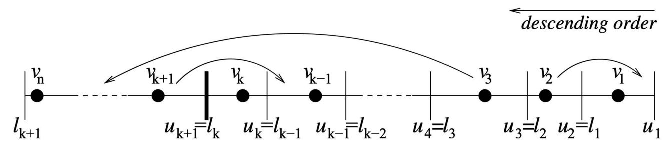


Fig. 4. Three categories of sensor-initiated updates.

We note that, in addition to keeping track of the ordered list of top-k sensor nodes, our FILA approach also continuously returns for each node an approximate reading bounded by the filtering window. This approximate reading is useful and sufficient to many applications because maintaining the exact sensor readings at the base station is costly. We will measure the level of approximation by using trace-driven simulation in Section 5.3.

3.3 Query Reevaluation

In this section, we discuss the query reevaluation algorithm. Under the proposed FILA monitoring approach, a sensor node sends an update to the base station only when the reading passes its filter. We shall call it *a sensor-initiated update*. If the updated reading overlaps with the filtering window of any other sensor node, then the top-k result becomes undecided at the base station. Thus, the base station will have to probe some sensor node(s) to reevaluate the top-k result.

We call the lower bound of the top-kth node's filter (that is, l_k) the *critical bound*. We classify sensor-initiated updates into three types: 1) *Internal update*. An update originated from a top-k node jumps into the filtering window of another top-k node (for example, v_2 jumps into $[l_1, u_1]$, as shown in Fig. 4). 2) *Join update*. An update from a nontop-k node jumps over the critical bound and falls into the filtering window of a top-k node (for example, v_{k+1} jumps into $[l_{k-1}, u_{k-1}]$, as shown in Fig. 4). 3) *Leave update*. An update from a top-k node jumps over the critical bound and falls into the filtering window of nontop-k nodes (for example, v_3 jumps into $[l_{k+1}, u_{k+1}]$, as shown in Fig. 4).

Consider a simple case where only one sensor-initiated update is received by the base station at a sampling instance. If it is an internal or a join update, then only the relevant top-k node whose filter covers the updated sensor reading needs to be probed to reevaluate the top-k result. For example, if v_2 jumps into $[l_1, u_1]$, then node 1 is probed; if v_{k+1} jumps into $[l_{k-1}, u_{k-1}]$, then node $k-1$ is probed. On the other hand, if the update is a leave update from a node i, then we may have to probe all nontop-k nodes to look for the new top-kth node. This can introduce high energy consumption. To minimize the cost, we propose to include in the probe message the newly reported reading v_i' from the leave update. Only the sensor nodes whose current readings are higher than v_i' respond to the probe, since only these nodes have a chance to beat node i and join the new top-k set.

We now extend the discussion to general cases, in which multiple sensor-initiated updates may be received by the base station at a sampling instance. We denote the set of internal updates as $\mathcal{T}_{internal}$, the set of join updates as $\mathcal{T}_{join}$,

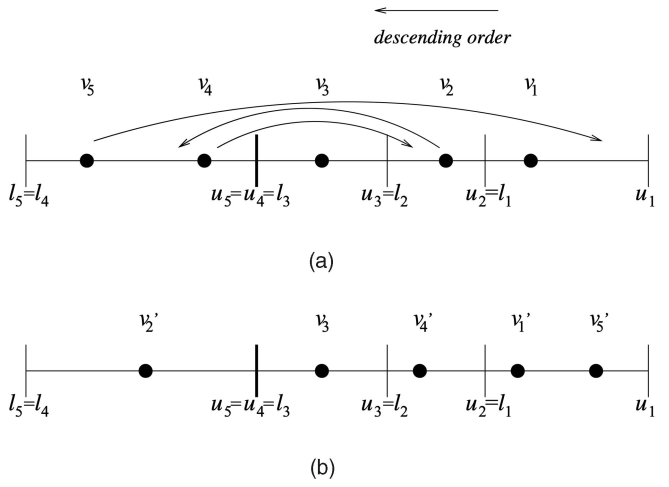


Fig. 5. An example of concurrent sensor-initiated updates in top-3 monitoring. (a) Sensor-initiated updates from nodes 2, 4, and 5. (b) Sensor readings maintained at the base station after probe (the newly updated readings of nodes 2, 4, and 5 are v'_2, v'_4, and v'_5, respectively, and the probed reading of node 1 is v'_1).

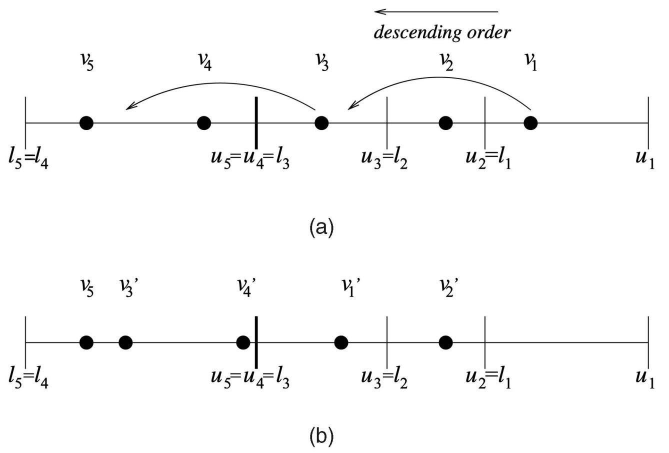


Fig. 6. Another example of concurrent sensor-initiated updates in top-3 monitoring. (a) Sensor-initiated updates from nodes 1 and 3. (b) Sensor readings maintained at the base station after probe (the newly updated readings of nodes 1 and 3 are v'_1 and v'_3, respectively, and the probed reading of node 4 is v'_4).

and the set of leave updates as $\mathcal{T}_{leave}$. Suppose the old top-k set is $\mathcal{T}$, where $|\mathcal{T}| = k$. Consider the set of sensor nodes $\mathcal{T}' = \mathcal{T} - \mathcal{T}_{leave} + \mathcal{T}_{join}$. The current reading of any node in $\mathcal{T}'$ must be not less than the critical bound l_k. On the other hand, the current reading of any node that is not in $\mathcal{T}'$ must not exceed l_k. Therefore, if $|\mathcal{T}'| = |\mathcal{T}| - |\mathcal{T}_{leave}| + |\mathcal{T}_{join}| \geq k$ (that is, $|\mathcal{T}_{join}| \geq |\mathcal{T}_{leave}|$), then the new top-$k$ set must be a subset of $\mathcal{T}'$. In this case, we do not need to probe any node that is not in $\mathcal{T}'$. Otherwise, if $|\mathcal{T}'| < k$, then the nodes that are not in $\mathcal{T}'$ have to be probed.

The query reevaluation algorithm (Algorithm 1) works as follows: Note that each node in $\mathcal{T}'$ either has a deterministic value at the base station (that is, the newly reported reading in the sensor-initiated update) or falls in a value range bounded by $[l_i, u_i]$ for some $1 \leq i \leq k$. Thus, it is easy to compute the number of sensor readings falling in each of the windows $[u_1, +\infty], [l_1, u_1], \ldots, [l_i, u_i], \ldots, [l_k, u_k]$. We shall denote them as $c_0, c_1, \ldots, c_i, \ldots, c_k$.

If $\sum_{i=0}^{k} c_i = |\mathcal{T}'| \geq k$, then there must exist an integer m $(1 \leq m \leq k)$ such that $\sum_{i=0}^{m-1} c_i < k$ and $\sum_{i=0}^{m} c_i \geq k$. To construct the new top-k set, we only need to compute the order of all sensor readings falling in

$$[u_1, +\infty], [l_1, u_1], \ldots, [l_m, u_m].$$

Specifically, for each node i $(1 \leq i \leq m)$, if node i did not generate a sensor-initiated update, and some sensor-initiated updates fall in $[l_i, u_i]$, then node i needs to be probed to compute the relative order of its reading with respect to the newly updated readings in $[l_i, u_i]$. On computing the order of all sensor readings in $[u_1, +\infty], [l_1, u_1], \ldots, [l_m, u_m]$, the top-$k$ nodes are selected as the new top-k set. In the example in Fig. 5, only node 1 is probed to compute its relative order to node 5, and the new top-3 set is $< 5, 1, 4 >$.

If $\sum_{i=0}^{k} c_i = |\mathcal{T}'| < k$, then we first compute the order of all sensor readings in $\mathcal{T}'$ in a similar way as above. Besides $\mathcal{T}'$, we also need to compute the top $k - |\mathcal{T}'| = |\mathcal{T}_{leave}| - |\mathcal{T}_{join}|$ nodes that are not in $\mathcal{T}'$ in order to construct the new top-k set. A simple method is to probe all the nodes that are not in $\mathcal{T}'$. However, this may introduce excessive network traffic and energy consumption. In the following, we propose two optimizations to reduce the probing cost.

The first takes advantage of the hierarchical multihop routing architecture in wireless sensor networks. When the probed readings are propagated up the routing tree toward the base station, in-network aggregation can be performed at the intermediate nodes. Since we are interested in the top $|\mathcal{T}_{leave}| - |\mathcal{T}_{join}|$ nodes that are not in $\mathcal{T}'$, each intermediate node needs to send upstream only the highest $|\mathcal{T}_{leave}| - |\mathcal{T}_{join}|$ readings received from its descendants.

The second optimization follows the semantics of top-k queries. Note that the base station has the newly updated readings of the nodes in $\mathcal{T}_{leave}$ that are not in $\mathcal{T}'$. Thus, only the nontop-k nodes whose current readings are higher than the $(|\mathcal{T}_{leave}| - |\mathcal{T}_{join}|)$th reading in $\mathcal{T}_{leave}$ (denoted by u) have a chance to join the new top-k set. Instead of probing all nodes that are not in $\mathcal{T}'$, we include the value u as a probe threshold in the probe message, and only the nodes with readings higher than u will respond to the probe. In the example in Fig. 6, the newly updated reading v'_3 of node 3 is included in the probe message. As a result, only node 4 reports its current reading to the base station in response to the probe. The new top-3 set is $< 2, 1, 4 >$.

Algorithm 1 Query Reevaluation Algorithm for Each Update Instance (Performed at the Base Station)

```
1: $to\text{-}probe = \phi$
2: $top\text{-}k$ = the nodes with updated values falling in $[u_1, +\infty]$
3: for $i = 1$; $i <= k$; $i++$ do
4:     let $[l_i, u_i]$ be the filtering window of the top-$i$th node
       in the old top-$k$ set
5:     $top\text{-}k = top\text{-}k \cup$ the nodes with updated values
       falling in $[l_i, u_i]$
6:     if the top-$i$th node does not have a sensor-initiated
       update then
7:         $top\text{-}k = top\text{-}k \cup$ top-$i$th node
8:         if there is a new updated value falling in $[l_i, u_i]$
           then
9:             $to\text{-}probe = to\text{-}probe \cup$ top-$i$th node
```

```
10:         end if
11:     end if
12:     if $|top\text{-}k| \geq k$ then
13:         break
14:     end if
15: end for
16: if $|top\text{-}k| < k$ then
17:     to-probe = to-probe $\cup$ all the nontop-$k$ nodes that have
        a reading higher than the $(k - |top\text{-}k|)$th highest
        reading in the leave updates
18: end if
19: probe the set of nodes in to-probe
20: recompute the top-$k$ result
```

3.4 Filter Setting

On reevaluating the top-*k* result set after receiving the sensor-initiated updates, the filter settings of sensor nodes are recomputed. To maximize the filtering capability, the upper bound of the top-1 node's filter (that is, u_1) and the lower bound of nontop-*k* node's filter (that is, l_{k+1}) are set to $+\infty$ and $-\infty$, respectively. We now consider the settings for all nodes except u_1 and l_{k+1}. Recall that u_{i+1} is set equal to l_i to optimize performance (Section 3.2). The intuitive way is to set them at the midpoint of two sensor readings, that is

$$u_{i+1} = l_i = \frac{v_i + v_{i+1}}{2} \quad (1 \leq i \leq k). \tag{2}$$

This is obviously a feasible filter setting satisfying (1), and we call it *uniform* filter setting. It is simple and favorable when the readings of all sensor nodes follow a similar changing pattern. However, the uniform setting fails to consider the possible diversity in the changing patterns of sensor readings. If the readings of node $i+1$ change much faster than those of node i, then node $i+1$ is more likely to initiate a sensor update than node i, under the uniform filter setting. In the following, we develop a *skewed* filter setting algorithm by taking into account the changing patterns of sensor readings. Our aim is to balance the energy consumption of the "neighboring" nodes in the top-*k* result set and thus improve network lifetime.

Suppose the average time for the reading of node i to change beyond δ is $f_i(\delta)$. Then, the rate of sensor-initiated updates by node i is given by $\frac{1}{f_i(\delta)}$. In order to balance the energy consumption (that is, the update rates) of nodes i and $i+1$, u_{i+1} and l_i should be set such that

$$\frac{1}{f_i(v_i - l_i)} = \frac{1}{f_{i+1}(u_{i+1} - v_{i+1})}. \tag{3}$$

In practice, it is difficult to know in advance how the sensor readings evolve dynamically, as well as the exact form of $f_i(\delta)$. One approach is to use the historical sensor readings to predict $f_i(\delta)$. However, this approach is costly, as the base station has to collect all sensor readings. Here, we propose a practical low-cost approach by assuming that the reading changes follow a well-known random walk model [25]. Under the random walk model, the reading changes in steps. For simplicity, we assume that the reading increases or decreases by an amount d at each step. Denote the interstep interval by l. The average time for the reading to change beyond δ can be expressed as (see the Appendix for the derivation)

$$f(\delta) = \left(\frac{\delta}{d}\right)^2 \cdot l. \tag{4}$$

We let every node measure the average delta change d_i of their sensor readings at a fixed rate. When the sensor node reports a new reading to the base station, it piggybacks the measured value of d_i. Let L be the time interval chosen to measure the average delta change. Then, $f_i(\delta)$ can be approximated by

$$f_i(\delta) = \left(\frac{\delta}{d_i}\right)^2 \cdot L. \tag{5}$$

By substituting $f_i(\delta)$ by (5) into (3), we obtain

$$\left(\frac{d_i}{v_i - l_i}\right)^2 / L = \left(\frac{d_{i+1}}{u_{i+1} - v_{i+1}}\right)^2 / L.$$

By solving this equation, we get

$$\frac{u_{i+1} - v_{i+1}}{v_i - l_i} = \frac{d_{i+1}}{d_i}.$$

By letting $u_{i+1} = l_i$, we have

$$u_{i+1} = l_i = v_{i+1} + \frac{d_{i+1}}{d_i + d_{i+1}} \cdot (v_i - v_{i+1}), \quad (1 \leq i \leq k) \tag{6}$$

for skewed filter setting.

3.5 Filter Update

On recomputing the filter settings of sensor nodes, the new filtering windows are to be sent to the nodes for installation. We propose two approaches for updating the filter of each node i:

- **Eager Filter Update**. If a new filtering window $[l'_i, u'_i]$ is different from the old one $[l_i, u_i]$, then the new filter $[l'_i, u'_i]$ is immediately sent to node i to replace $[l_i, u_i]$.
- **Lazy Filter Update**. If a new filtering window $[l'_i, u'_i]$ fully contains the old one $[l_i, u_i]$, that is, $[l_i, u_i] \subset [l'_i, u'_i]$, then the base station delays the filter update until node i's reading violates the old filter $[l_i, u_i]$. During this period, node i continues using $[l_i, u_i]$ to filter out local sensor updates. It is easy to verify that the top-*k* order is still maintained with such a *conservative* filter setting. When node i's new reading v'_i is outside $[l_i, u_i]$, a sensor-initiated update is sent to the base station. If v'_i is within $[l'_i, u'_i]$, then the top-*k* set remains unchanged, and the new filtering window $[l'_i, u'_i]$ will be passed to node i. In this case, lazy filter update incurs one more sensor-initiated update compared to eager filter update. Otherwise, if v'_i is out of $[l'_i, u'_i]$, then it is treated as a normal sensor-initiated update. The query reevaluation algorithm is executed to update the top-*k* set, and the filter settings will also be recomputed. In this case, the lazy filter update saves a filter update message compared to the eager filter update. Thus, the lazy

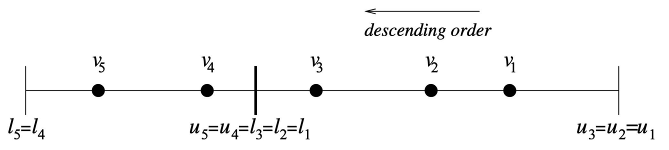


Fig. 7. Filter settings for order-insensitive top-k monitoring.

filter update trades more sensor-initiated updates for fewer filter update messages.

The relative performance of the eager and lazy approaches depends on the changing pattern of sensor readings. We will investigate their performance by using trace-driven simulation in Section 5.2.

4 Extensions

So far, we have focused on order-sensitive exact top-k set monitoring. However, the ordering information in the top-k set may not be needed for some applications. A certain degree of approximation in top-k ordering may also be tolerable for some applications to trade result quality for energy efficiency. Moreover, some applications may be interested in knowing, besides the top-k set, the top-k values with bounded precision. Motivated by these observations, in this section, we extend FILA to handle order-insensitive top-k monitoring, approximate top-k monitoring, and top-k value monitoring.

4.1 Order-Insensitive Top-*k* Monitoring

The order-sensitive algorithm discussed in Section 3 is also applicable to order-insensitive top-k monitoring. However, since order-insensitive top-k monitoring does not care about the exact order of sensor nodes in the top-k set, internal updates do not have to be reported by the sensor nodes. Therefore, only a critical bound needs to be set between the top-k nodes and the nontop-k nodes, as shown in Fig. 7. When join and leave updates are reported by sensor nodes, the query reevaluation algorithm is similar to what was discussed in Section 3.3.

4.2 Approximate Top-*k* Monitoring

We now consider approximate top-k monitoring, assuming that a certain *degree of approximation* in the result is acceptable. That is, given an approximation degree ϵ, we do not care about the order of two sensor nodes in the result set if their sensor readings are within a difference of ϵ. Formally, an approximate top-k monitoring query continuously requests the list of sensor nodes:

$$\mathcal{R} =< n_1, n_2, \dots, n_k >$$

such that

$$\forall i < j, v_{n_i} \geq v_{n_j} - \epsilon$$

and

$$\forall l \neq n_i (i = 1, 2, \dots, k), v_l \leq min\{v_{n_1}, v_{n_2}, \dots, v_{n_k}\} + \epsilon.$$

A *feasible* filter setting scheme for approximate top-k monitoring, thus, must satisfy

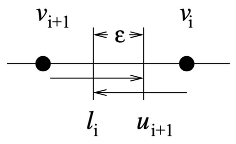


Fig. 8. Filter settings for approximate top-k monitoring.

$$\begin{cases} v_1 \leq u_1; \\ v_{i+1} \leq u_{i+1},\ u_{i+1} \leq l_i + \epsilon,\ l_i \leq v_i; \quad (1 \leq i \leq k). \\ l_{k+1} \leq v_N. \end{cases}$$

This means that an overlap of ϵ is allowed between two neighboring filters (see Fig. 8). Setting $\epsilon = 0$ degenerates approximate top-k monitoring to exact top-k monitoring.

The filter settings should be revised accordingly. Under the *uniform* filter setting, for each $1 \leq i \leq k$

$$\begin{cases} u_{i+1} = \frac{v_i + v_{i+1} + \epsilon}{2}; \\ l_i = \frac{v_i + v_{i+1} - \epsilon}{2}. \end{cases}$$

Under the *skewed* filter setting, for each $1 \leq i \leq k$,

$$\begin{cases} u_{i+1} = v_{i+1} + \frac{d_{i+1}}{d_i + d_{i+1}} \cdot (v_i - v_{i+1} + \epsilon); \\ l_i = v_i - \frac{d_i}{d_i + d_{i+1}} \cdot (v_i - v_{i+1} + \epsilon). \end{cases}$$

4.3 Top-*k* Value Monitoring

Top-k value monitoring returns not only the set of top-k nodes, but also their precision-bounded readings. The extension to top-k value monitoring is straightforward. Besides the filter maintained for monitoring top-k result set (called *order filter*), we maintain another tighter filter for each node in the top-k result set (called *value filter*) for value reporting purposes. Consider a top-k node i with an order filter $[l_i, u_i]$. Assuming that its last reported value is v_i, and the precision constraint required in value reporting is p, the value filter is then set to $[max(l_i, v_i - p), min(u_i, v_i + p)]$. When a new reading of the node violates either the order filter or the value filter, the reading is sent to the base station. In case only the value filter is violated, the base station updates the reading maintained for the node and its corresponding value filter. Otherwise, if the order filter is also violated, then the top-k result set is reevaluated following the algorithms discussed in Section 3.

5 Performance Evaluation

5.1 Simulation Setup

We have developed a simulator based on ns-2 (version 2.26) [22] and the US Naval Research Laboratory (NRL)'s sensor network extension [23] to evaluate the proposed FILA approach. The simulator includes the detailed models of the media access control (MAC) and physical layers for wireless sensor networks. The sensor nodes can operate in one of three modes: sending message, receiving message, and sleeping. These modes differ in energy consumption. The energy consumption for sending a message is determined by a cost function $s \cdot (\alpha + \beta \cdot d^q)$, where s is the message size, α is a distance-independent term, β is the coefficient for a distance-dependent term, q is the component for the distance-dependent term, and d is the distance of message transmission. As in [13] and [32],

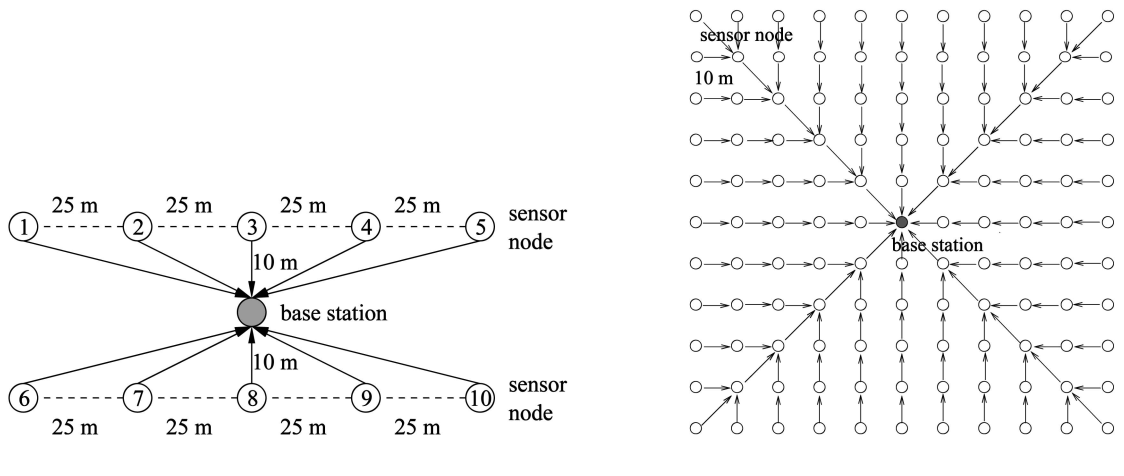


Fig. 9. Network layouts. (a) Single-hop network. (b) Multihop network.

we set $\alpha = 50\ \mathrm{nJ/b}$, $\beta = 100\ \mathrm{pJ/b/m^2}$, and $q = 2$ in the simulation. The energy consumption for receiving a message is given by $s \cdot \gamma$, where γ is set at 50 nJ/b. The power consumption in the sleeping mode is set at 0.016 mW. For simplicity, the energy overhead of mode switching is ignored. We assume that a sensor identity and a sensor reading both take 4 bytes, and a filtering window is characterized by 8 bytes. The initial energy budget at each sensor node was set at 0.01 Joule.

We simulated a single-hop network of 10 sensor nodes and a multihop network of 120 sensor nodes. Their layouts are shown in Figs. 9a and 9b, respectively. We assume that the TAG routing tree [16] is employed for the base station to communicate with the sensor nodes. The sensor readings are simulated using the real traces provided by the Live from Earth and Mars (LEM) project [31] at the University of Washington. We used the temperature (TEMP) and dew point (DEW) traces logged by the station at the University of Washington from August 2004 to August 2005 in our experiments. Each trace consists of more than 500,000 sensor readings. We extracted many subtraces starting at different dates. Each subtrace contained 20,000 readings. The subtraces were used to simulate the physical phenomena in the immediate surroundings of different sensor nodes. The subtraces starting at successive dates are similar. To simulate the spatial correlation of sensor readings, the subtraces starting at successive dates were assigned to neighboring nodes in the simulated network. Fig. 10 shows some representative segments of the TEMP and DEW data traces, where the interval between two successive readings was assumed to be one time unit (that is, 10 sec).

We modified the sensor sampling interval to simulate two different workloads. In the homogeneous (HM) setting, the sampling interval for all sensors is set at one time unit; in the heterogeneous (HT) setting, the sampling interval for half of the sensors is set at one time unit, and that for the other half is set at five time units. As a result, the readings of the sensors in the former half change more rapidly than those in the latter half. The default values of k in top-k queries are set at 3 and 10 for the single-hop and multihop network configurations, respectively.

In the following, we first compare the two filter update strategies (that is, eager and lazy) of the proposed FILA approach. We then evaluate FILA (with two different filter setting schemes, that is, uniform and skewed) against the TAG-based periodic aggregation approach (which was illustrated in Section 1) and the range caching approach (which was explained in Section 2). For FILA with skewed filter setting (Section 3.4), the time interval of measuring the average delta change is set to one time unit. The following metrics are employed for performance comparison:

- **Network Lifetime**. As in the previous work [32], [40], the network lifetime is defined as the time duration before the first sensor node runs out of power. It serves as the primary metric in the performance evaluation.
- **Average Energy Consumption**. It is defined as the average amount of energy consumed by a sensor node per time unit.
- **Monitoring Accuracy**. This is defined as the mean accuracy of monitored results against the real results. Specifically, let $\mathcal{M}(t)$ denote the monitored result set at time t and $\mathcal{R}(t)$ denote the real result set. The monitoring accuracy is thus defined as $\frac{1}{t_e - t_b} \int_{t_b}^{t_e} a(t)dt$,

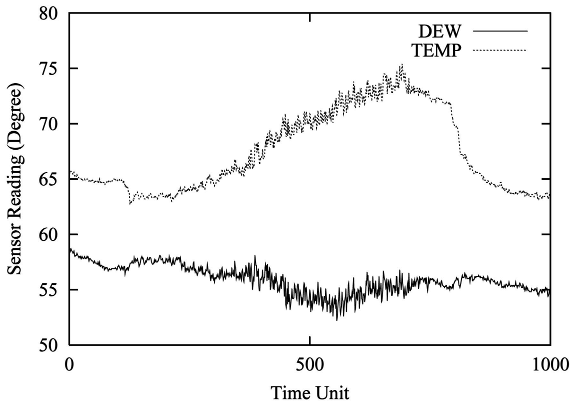


Fig. 10. Sample real data traces.

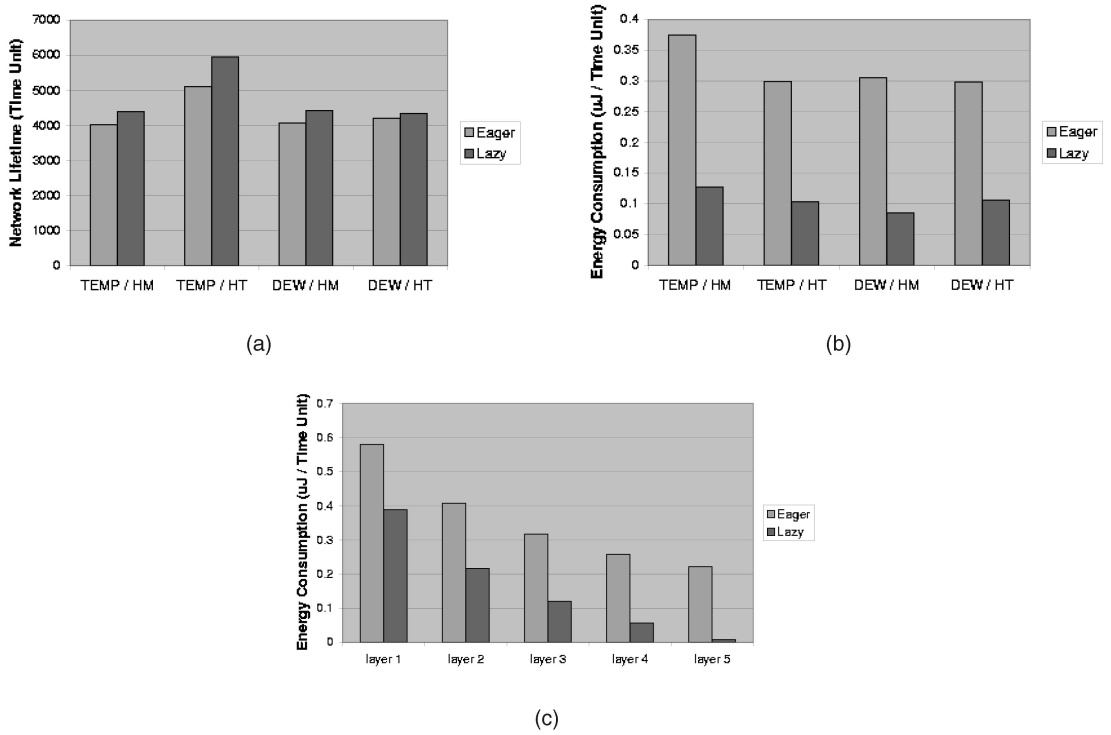


Fig. 11. Eager versus lazy update (multihop, $k = 10$). (a) Network lifetime. (b) Average energy consumption. (c) Energy consumption by layer (TEMP/HT).

where $[t_b, t_e]$ is the monitoring period, $a(t) = 1$ if $\mathcal{M}(t) = \mathcal{R}(t)$, and $a(t) = 0$ otherwise.

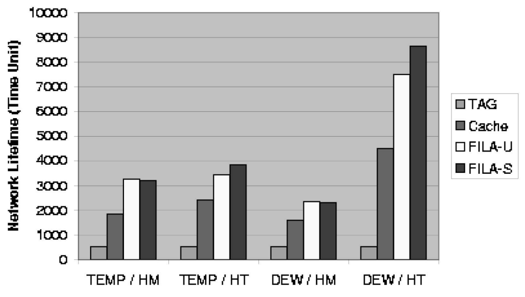
5.2 Eager versus Lazy Filter Update

The first set of experiments compares the eager and lazy filter update strategies discussed in Section 3.3. The uniform filter setting is employed. As shown in Fig. 11a, the lazy approach achieves a longer network lifetime for both the TEMP and DEW traces in the multihop network. Moreover, the lazy approach performs much better in terms of average energy consumption, as plotted in Fig. 11b. The energy saving is up to 64 percent. Based on the discussions in Section 3.5, this implies that most sensor reading changes have a magnitude wider than the new filtering windows. The lazy approach helps save the filter update messages, which contributes to a significant portion of the overall traffic (more than 50 percent for the eager approach, as observed in the experiments). In addition, we plot in Fig. 11c the average energy consumption of the nodes in different layers of the routing tree. As can be seen, the relative performance improvement for layer-1 nodes (that is, the root's direct children) is much lower than that for layer-5 nodes. Since the network lifetime is determined by the hot-spot layer-1 nodes, this explains why the lazy approach shows a smaller performance gain over the eager approach in network lifetime than in energy consumption. Overall, the lazy approach is considered having a better performance than the eager approach for the traces tested. Similar performance trends are obtained for the single-hop network. In the following experiments, we use the lazy approach as the default filter update strategy in FILA.

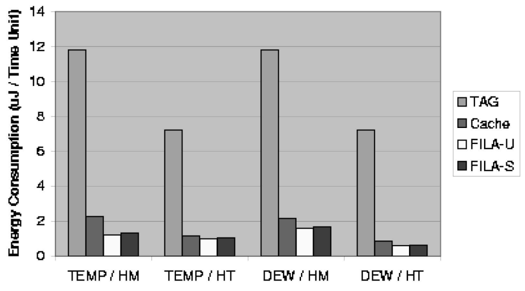
5.3 Performance Comparison against TAG and Range Caching

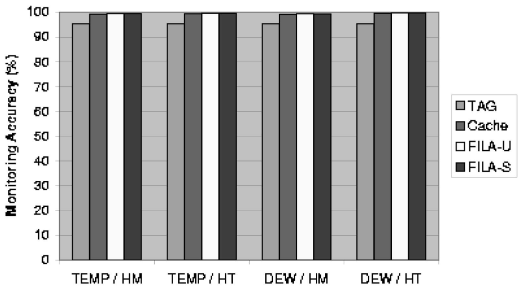
In this section, we compare the performance of FILA against TAG and range caching (or *Cache* for brevity). We denote FILA with uniform filter setting as *FILA-U* and FILA with skewed filter setting as *FILA-S*. For Cache, at every query evaluation instance, if any two candidate top-k nodes have their cached value ranges overlap, then refreshment is needed to resolve the query result [35]. In each experiment, we varied the approximate range setting for Cache from 0.1 to 3.2, and the result reported below is obtained from the best setting.

Figs. 12 and 13 show the results for the single-hop and multihop networks, respectively. Several observations are obtained. First, both FILA-U and FILA-S significantly improve the network lifetime over TAG and Cache and achieve a much lower average energy consumption. This result is consistent across all data traces examined, indicating a significant performance advantage of our proposed approach. Second, when comparing FILA-U and FILA-S, FILA-S is slightly worse than FILA-U in terms of the average energy consumption. Nevertheless, FILA-S achieves 6-15 percent longer network lifetime for the HT sampling scenarios by differentiating the filter widths of different nodes based on their changing patterns in sensor reading. On the other hand, for the HM scenarios, all sensors have similar changing patterns in reading. As a result, FILA-S does not improve the network lifetime against FILA-U. Third, FILA-U and FILA-S get a higher monitoring accuracy than TAG. This is because in TAG, all sensor nodes update at the sampling instance and, hence, a longer update propagation delay is incurred due to message collisions and retransmissions. With much less



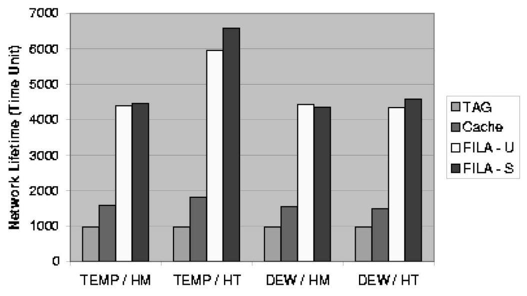
(a)



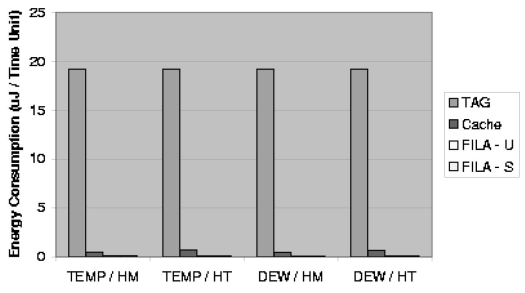


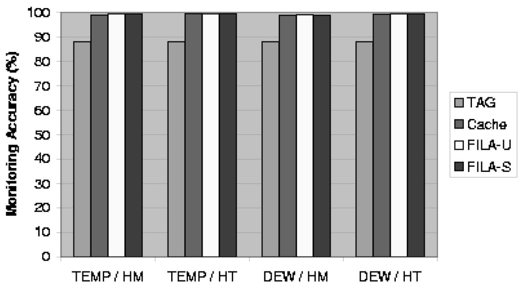
(b) (c)

Fig. 12. Performance comparison with TAG and Cache (single hop, $k = 3$). (a) Network lifetime. (b) Average energy consumption. (c) Monitoring accuracy.



(a)





(b) (c)

Fig. 13. Performance comparison with TAG and Cache (multihop, $k = 10$). (a) Network lifetime. (b) Average energy consumption. (c) Monitoring accuracy.

update traffic, FILA-U and FILA-S have a faster query reevaluation time. Fourth, Cache has a relatively better performance in the single-hop network than in the multihop network. This reason is explained as follows: Since there are more nodes in the multihop network, the chance of two nodes having their cached value ranges overlap is higher and, hence, more refreshments are incurred. Moreover, the hot-spot nodes in the multihop network consume energy not only for reporting their own updates but also for relaying the updates of other lower level nodes. Finally, we observed in the experiments that the average width of top-k nodes' filtering windows (except that for the top-1 node) is

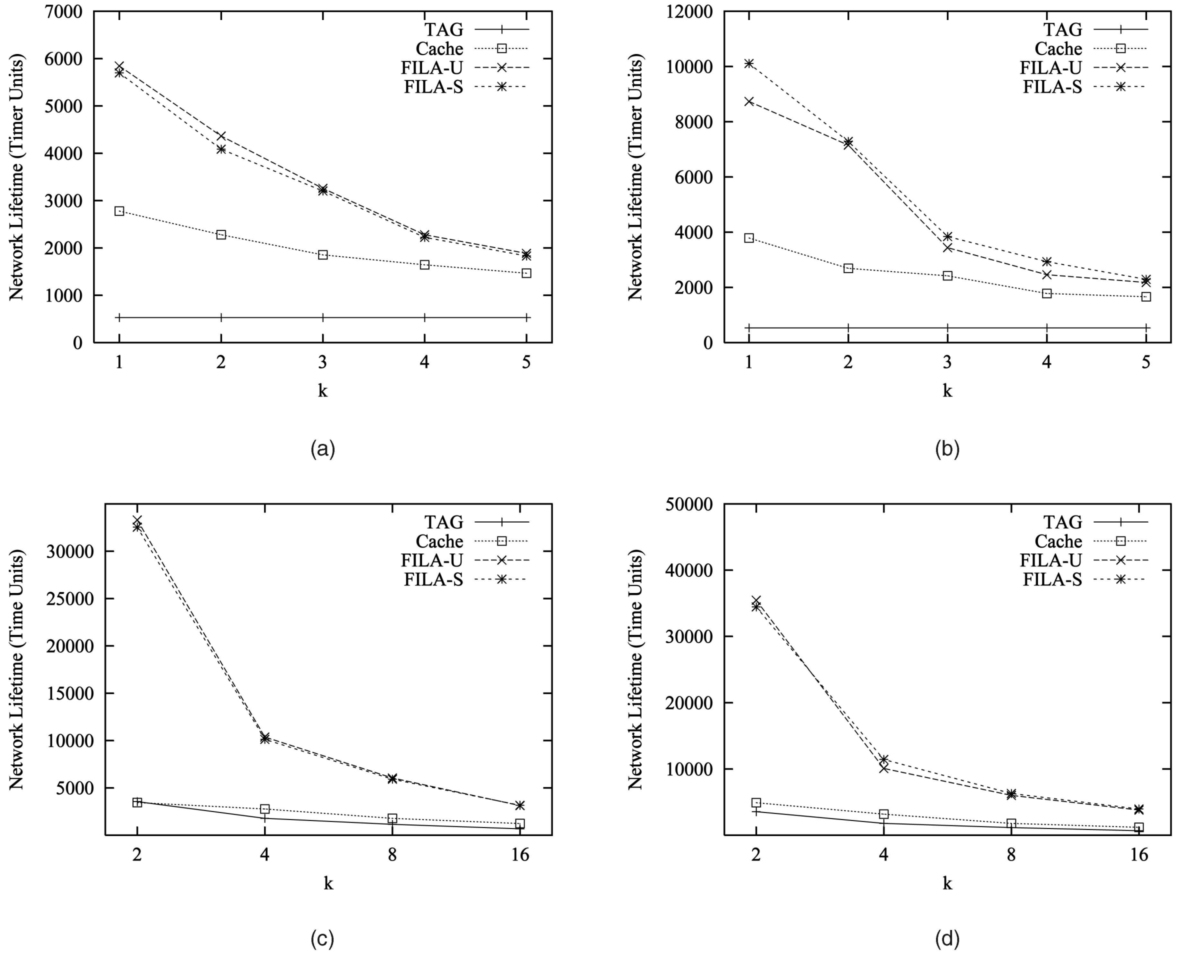


Fig. 14. Lifetime as a function of k. (a) TEMP/HM, single hop. (b) TEMP/HT, single-hop. (c) TEMP/HM, multihop. (d) TEMP/HT, multihop.

0.26 degree for the TEMP trace. This confirms our argument in Section 3.2 that FILA not only returns the top-k result set but also provides a tightly bounded approximation for each of the top-k sensor readings.

Fig. 14 shows the network lifetime as a function of k for the TEMP trace. Similar performance trends are obtained for the DEW trace. As expected, the network lifetime decreases with increasing k for all schemes except for TAG in the single-hop network.[2] In all cases examined, again, FILA-U and FILA-S significantly outperform TAG and Cache. As TAG shows a poor performance, for clarity, we compare FILA against Cache only in the rest of the experiments.

5.4 Approximate Top-*k* Monitoring

This section investigates the performance of approximate top-k monitoring. We vary the approximation degree ϵ from 0.0 to 0.5 for the TEMP trace. The values of k are set at 3 and 10 for the single-hop and multihop networks, respectively. To have a fair comparison, in Cache, a refreshment is needed only when the value ranges of two candidate top-k nodes have an overlap larger than the approximation degree. Thus, Cache can also take advantage of error tolerance to improve the performance.

2. In a single-hop network, the rate and sizes of update messages under TAG are independent of k.

As shown in Fig. 15, FILA-U and FILA-S consistently outperform Cache in terms of network lifetime. Similar to the observations made for exact top-k monitoring, FILA-S gets a similar or slightly worse performance than FILA-U for the HM sampling scenario but performs better (with 6.2 percent to 11.5 percent of improvement) for the HT scenario.

It is also interesting to observe that, for all traces, the network lifetime can be noticeably extended with a small degree of approximation allowed in the result. For example, with an approximation degree of 0.5, using FILA-U or FILA-S, the network lifetime is prolonged by more than two times in most cases compared to the exact monitoring.

5.5 Top-*k* Value Monitoring

Now, we evaluate the performance of top-k value monitoring that maintains the set of top-k nodes as well as their precision-bounded readings. We vary the precision requirement p for value reporting from 0.1 to 3.2 for the single-hop network and 0.05 to 1.6 for the multihop network. Fig. 16 shows the results for the TEMP/HM trace, where FILA-U (Set) stands for the scheme in which only the top-k result set is maintained (that is, with an infinitely large value of p), and FILA-U (Value) maintains also precision-bounded top-k readings (discussed in Section 4.3). FILA-U (Set) represents the performance upper bound that FILA-U (Value) could achieve.

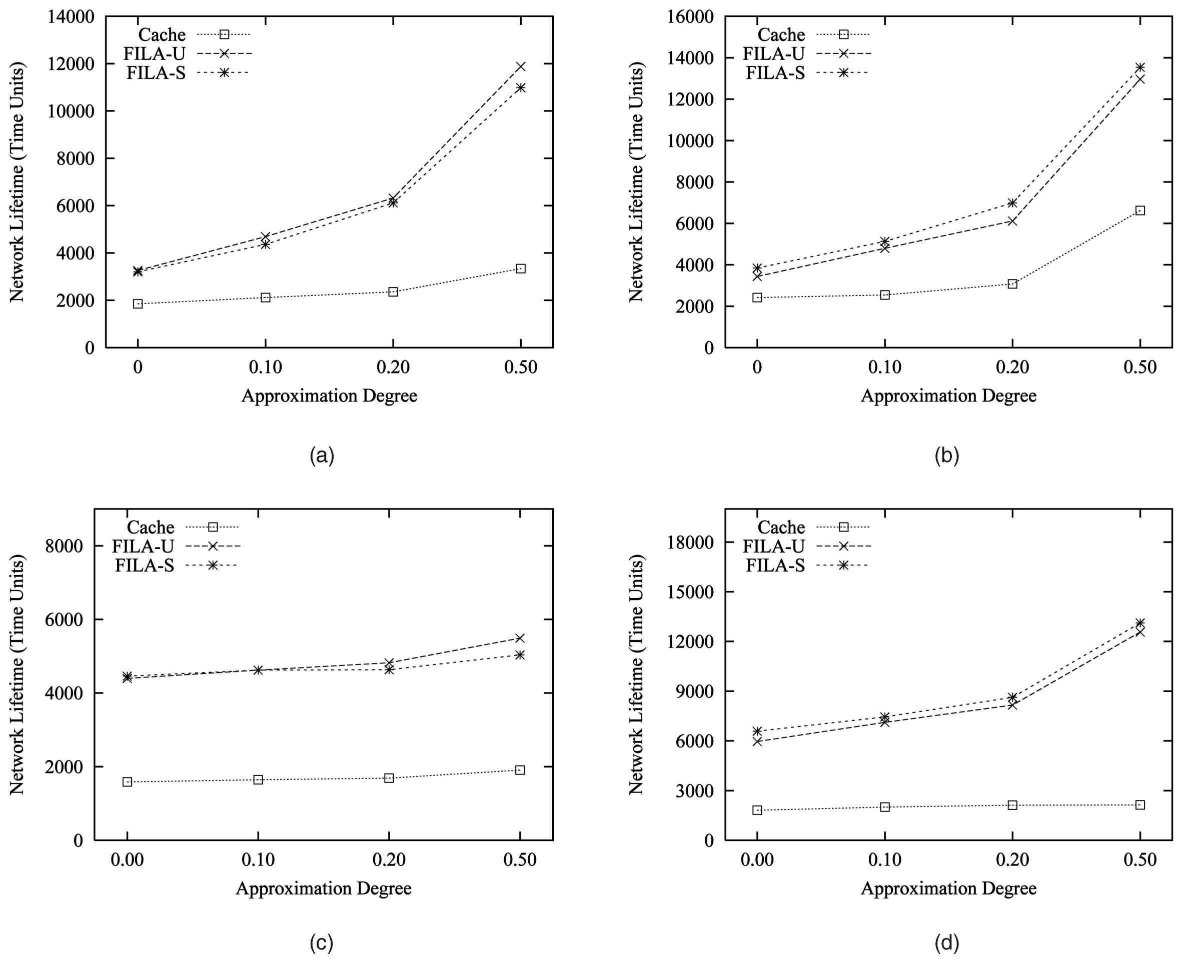


Fig. 15. Approximate top-k monitoring. (a) TEMP/HM, single hop, $k = 3$. (b) TEMP/HT, single hop, $k = 3$. (c) TEMP/HM, multihop, $k = 10$. (d) TEMP/HT, multihop, $k = 10$.

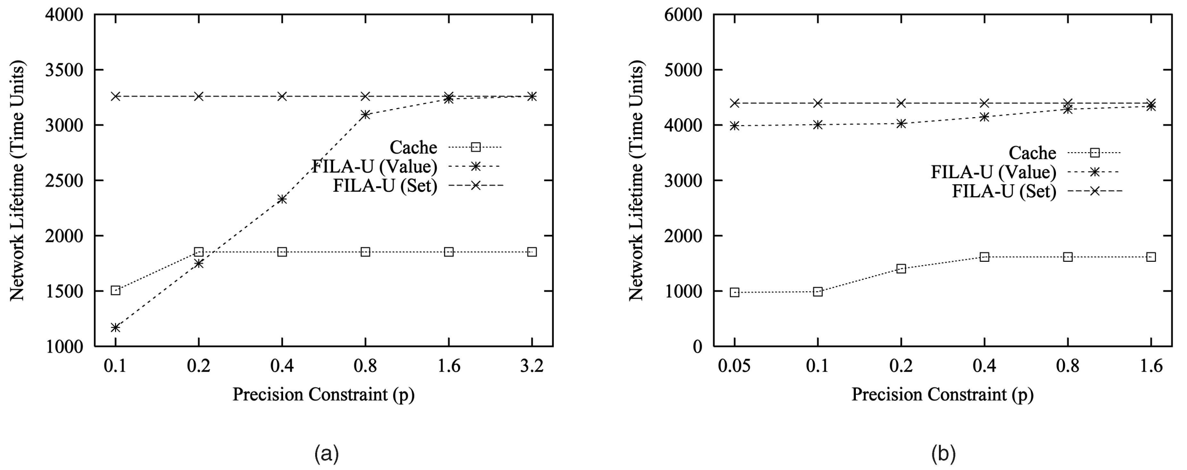
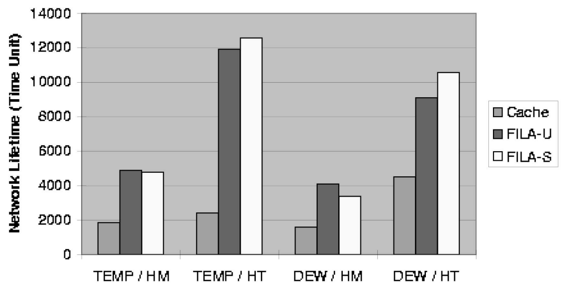


Fig. 16. Top-k value monitoring (TEMP/HM). (a) Single hop, $k = 3$. (b) Multihop, $k = 10$.

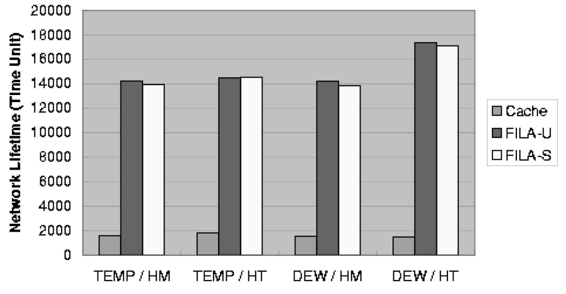
It is observed that, with increasing p, FILA-U (Value) improves the network lifetime gracefully. After reaching a certain point (for example, 0.8 for the multihop network), the lifetime of FILA-U (Value) approaches that of FILA-U (Set) and flattens out. This is because, in these cases, the filter used for maintaining a precision-bounded top-k reading is wider than that of the corresponding order filter and, hence, the value filter is set almost the same as the order filter. This result again confirms, from another angle, that our proposed FILA approach, even without the value monitoring requirement, is able to maintain a tightly bounded approximation for each top-k reading.

5.6 Order-Insensitive Top-*k* Monitoring

Finally, this section evaluates the performance of order-insensitive top-k monitoring. Fig. 17 shows the results



(a)



(b)

Fig. 17. Order-insensitive monitoring. (a) Single hop, $k = 3$. (b) Multihop, $k = 10$.

under the default system settings. For order-insensitive top-k monitoring, only the critical bound is maintained as the filter by all sensors; hence, the filter setting does not have a very high impact on the performance. As a result, FILA-U and FILA-S perform similarly in most cases. Compared with Cache, once again, FILA runs a much longer network lifetime by making use of the effective filters.

6 Conclusions

This paper has performed a comprehensive study on monitoring top-k query in wireless sensor networks. Different from existing work focusing on in-network data aggregation techniques, we exploited the semantics of top-k query and proposed a novel energy-efficient monitoring approach called FILA. We presented detailed algorithms to address two critical issues arising in the FILA approach, that is, filter setting and query reevaluation. Two filter setting algorithms (that is, uniform and skewed) and two filter update strategies (that is, eager and lazy) have been proposed. We have also extended the algorithms to order-insensitive top-k monitoring, approximate top-k monitoring, and top-k value monitoring.

A series of simulation experiments have been conducted to evaluate the performance of the proposed FILA approach by using real traces. The results show that:

1. FILA consistently outperforms the existing TAG-based approach and Cache approach in terms of both energy consumption and network lifetime, under various network configurations,
2. in addition to returning the top-k result set, FILA can also provide a tightly bounded approximation for each of top-k sensor readings,
3. the lazy filter update approach obtains a better overall performance than the eager approach for the traces examined, and
4. the uniform filter setting performs slightly better than the skewed filter setting for the HM sampling scenario, whereas the skewed filter setting is better for the HT sampling scenario.

As for future work, we plan to extend the proposed monitoring approach to other aggregate functions such as kNN, average, and sum. We are going to build a prototype based on Motes and measure the performance in real environments. We are also interested in monitoring spatial queries in object-tracking sensor networks.

Appendix

Analysis of Random Walk Model

Consider a one-dimensional random walk model, where at each step, an object moves distance d along a straight line in one of the two directions with equal probabilities. Let l be the interstep interval. Assume that the object starts a random walk from point O. Then, at any time, the distance between the object location and O is a multiple of d.

Suppose at some timepoint, an object is located $k \cdot d$ away from O, where k is an integer. If $k \neq 0$ (that is, the object is not at O), then the object would move toward and away from O, with equal probabilities in the next step. Thus, its distance to O after the next step is $(k-1) \cdot d$, with probability 0.5, and $(k+1) \cdot d$, with probability 0.5. If $k = 0$ (that is, the object is at O), then the object can only move away from O in the next step. Thus, its distance to O after the next step must be d.

We model the object location during random walk as a probability vector $[p_0, p_1, p_2, \ldots]$, where p_i is the probability that the object is located $i \cdot d$ away from O. Starting from the vector $[1, 0, 0, \ldots]$ (that is, the object starts a random walk from point O), the vector after each move can be computed iteratively with the above transition probabilities. It is then easy to calculate the probability of the object first moving beyond a given distance $\delta = x \cdot d$ from O at each step. This way, the average number of steps t that the object takes to first move beyond δ from O can be derived numerically.

Fig. 18 shows the simulation results, where the x-axis represents the normalized distance $x = \delta/d$ from the starting point O, and y-axis represents the ratio of t to x^2. As seen in Fig. 18, when δ is beyond a few times d, the ratio

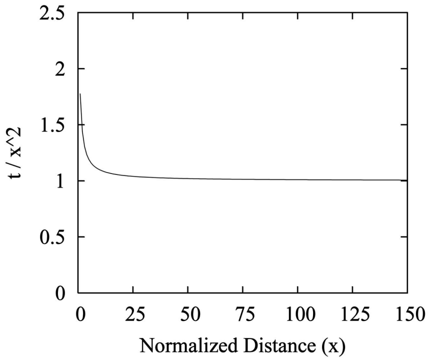


Fig. 18. Simulation results of random walk.

approaches a constant 1. Therefore, t can be approximated by $x^2 = (\delta/d)^2$. Let $f(\delta)$ denote the average time taken by the object to move beyond δ from O. Then, we have $f(\delta) = t \cdot l = (\delta/d)^2 \cdot l$.

ACKNOWLEDGMENTS

The authors would like to thank the editor and the anonymous reviewers for their valuable suggestions, which improved the quality of this paper. This work was partially supported by grants from the Research Grants Council of the Hong Kong SAR, China (Project nos. HKBU 211505, HKBU211206, and FRG/05-06/II-65). Xueyan Tang's work was supported in part by the Nanyang Technological University Grant RG47/06. Wang-Chien Lee's work was supported in part by the National Science Foundation Grants CNS-0626709, IIS-0534343, and IIS-0328881. A preliminary report of this work appeared in the *Proceedings of the International Conference on Data Engineering* as a poster paper [34].

REFERENCES

[1] B. Babcock and C. Olston, "Distributed Top-k Monitoring," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03),* pp. 28-39, June 2003.

[2] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '05),* Apr. 2005.

[3] P. Bonnet, J.E. Gerhke, and P. Seshadri, "Towards Sensor Database Systems," *Proc. Int'l Conf. Mobile Data Management (MDM),* Jan. 2001.

[4] P. Cao and Z. Wang, "Efficient Top-k Query Calculation in Distributed Networks," *Proc. 23rd Ann. ACM SIGACT-SIGOPS Symp. Principles on Distributed Computing (PODC '04),* July 2004.

[5] R. Cheng, B. Kao, S. Prabhakar, A. Kwan, and Y.-C. Tu, "Adaptive Stream Filters for Entity-Based Queries with Non-Value Tolerance," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05),* 2005.

[6] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate Aggregation Techniques for Sensor Databases," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '04),* Mar. 2004.

[7] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical In-Network Data Aggregation with Quality Guarantees," *Proc. Int'l Conf. Extending Database Technology (EDBT '04),* Mar. 2004.

[8] A. Deshpande, C. Guestrin, S. Madden, J.M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04),* 2004.

[9] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy, "Rethinking Data Management for Storage-Centric Sensor Networks," *Proc. Third Biennial Conf. Innovative Data Systems Research (CIDR '07),* Jan. 2007.

[10] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," *Proc. ACM Symp. Principles of Database Systems (PODS '01),* Aug. 2001.

[11] U. Güntzer, W.-T. Balke, and W. Kieβling, "Optimizing Multi-Feature Queries for Image Databases," *Proc. 26th Int'l Conf. Very Large Data Bases (VLDB '00),* 2000.

[12] Q. Han, S. Mehrotra, and N. Venkatasubramanian, "Energy-Efficient Data Collection in Distributed Sensor Environments," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '04),* Mar. 2004.

[13] W. Heinzelman, "Application-Specific Protocol Architectures for Wireless Networks," PhD dissertation, Massachusetts Inst. of Technology, 2000.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom '00,* Aug. 2000.

[15] Y. Kotidis, "Snapshot Queries: Towards Data-Centric Sensor Networks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '05),* Apr. 2005.

[16] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad Hoc Sensor Networks," *Proc. Usenix Fifth Symp. Operating Systems Design and Implementation (OSDI '02),* pp. 131-146, Dec. 2002.

[17] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "The Design of an Acquisitional Query Processor for Sensor Networks," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03),* June 2003.

[18] A. Marian, L. Gravano, and N. Bruno, "Evaluating Top-k Queries over Web-Accessible Databases," *ACM Trans. Database Systems,* vol. 29, no. 2, pp. 319-362, 2004.

[19] K. Mouratidis, D. Papadias, S. Bakiras, and Y. Tao, "A Threshold-Based Algorithm for Continuous Monitoring of k Nearest Neighbors," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 11, pp. 1451-1464, Nov. 2005.

[20] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: A Framework for Distributed Top-k Query Algorithms," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05),* 2005.

[21] S. Nepal and M.V. Ramakrishna, "Query Processing Issues in Image (Multimedia) Databases," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '99),* 1999.

[22] The Network Simulator—ns-2, http://www.isi.edu/nsnam/ns/, 2006.

[23] NRL's Sensor Network Extension to ns-2, http://nrlsensorsim.pf.itd.nrl.navy.mil/, 2006.

[24] C. Olston, J. Jiang, and J. Widom, "Adaptive Filters for Continuous Queries over Distributed Data Streams," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03),* pp. 563-574, June 2003.

[25] C. Olston, B.T. Loo, and J. Widom, "Adaptive Precision Setting for Cached Approximate Values," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '01),* pp. 355-366, 2001.

[26] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang, "A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks," *Proc. Int'l Conf. Data Eng. (ICDE '06),* Apr. 2006.

[27] A. Silberstein, K. Munagala, and J. Yang, "Energy-Efficient Monitoring of Extreme Values in Sensor Networks," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06),* June 2006.

[28] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat Monitoring with Sensor Networks," *Comm. ACM,* vol. 47, no. 6, pp. 34-40, June 2004.

[29] M.A. Sharaf, J. Beaver, A. Labrinidis, and P.K. Chrysanthis, "Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks," *VLDB J.,* vol. 13, no. 4, pp. 374-403, Dec. 2004.

[30] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and Beyond: New Aggregation Techniques for Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys '04),* 2004.

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

[32] X. Tang and J. Xu, "Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* Apr. 2006.

[33] M. Theobald, G. Weikum, and R. Schenkel, "Top-k Query Evaluation with Probabilistic Guarantees," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB '04),* Aug. 2004.

[34] M. Wu, J. Xu, X. Tang, and W.-C. Lee, "Monitoring Top-k Query in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '06),* Apr. 2006.

[35] M. Wu, J. Xu, and X. Tang, "Processing Precision-Constrained Queries in Wireless Sensor Networks," *Proc. Int'l Conf. Mobile Data Management (MDM '06),* May 2006.

[36] L. Xiong, S. Chitti, and L. Liu, "Top-k Queries across Multiple Private Databases," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '05),* June 2005.

[37] J. Xu, X. Tang, and W.-C. Lee, "EASE: An Energy-Efficient In-Network Storage Scheme for Object Tracking in Sensor Networks," *Proc. IEEE Conf. Sensor and Ad Hoc Comm. and Networks (SECON '05),* Sept. 2005.

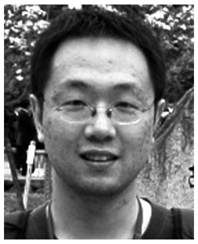
[38] Y. Yao and J.E. Gehrke, "Query Processing in Sensor Networks," *Proc. First Biennial Conf. Innovative Data Systems Research (CIDR '03),* Jan. 2003.

[39] S. Yoon and C. Shahabi, "The Clustered AGgregation (CAG) Technique Leveraging Spatial and Temporal Correlations in Wireless Sensor Networks," *ACM Trans. Sensor Networks,* 2006.

[40] O. Younis and S. Fahmy, "Distributed Clustering for Ad Hoc Sensor Networks: A Hybrid Energy-Efficient Approach," *Proc. IEEE INFOCOM,* Mar. 2004.

[41] C.T. Yu, G. Philip, and W. Meng, "Distributed Top-n Query Processing with Possibly Uncooperative Local Systems," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03),* 2003.

[42] D. Zeinalipour-Yazti, Z. Vagena, D. Gunopulos, V. Kalogeraki, V. Tsotras, M. Vlachos, N. Koudas, and D. Srivastava, "The Threshold Join Algorithm for Top-k Queries in Distributed Sensor Networks," *Proc. Int'l Workshop Data Management for Sensor Networks (DMSN '05),* Sept. 2005.



Minji Wu received the BEng degree in information engineering from the Nanjing University of Posts and Telecommunications. He is an MPhil student in the Department of Computer Science, Hong Kong Baptist University. His research interests include query processing, wireless sensor networks, and Web data management. He is a student member of the IEEE.

Jianliang Xu received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China, in 1998 and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2002. He is currently an assistant professor in the Department of Computer Science, Hong Kong Baptist University. He is an editor of the book *Web Content Delivery* published by Springer. He has also served as a session chair and a program committee member for many international conferences. His research interests include mobile and pervasive computing, wireless sensor networks, and distributed systems, with an emphasis on data management. He has published more than 50 technical papers in these areas, most of which appeared in prestigious journals and conferences including the ACM International Conference on Management of Data (SIGMOD), the International Conference on Mobile Systems, Applications, and Services (MobiSys), the IEEE International Conference on Data Engineering (ICDE), the IEEE INFOCOM, the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on Parallel and Distributed Systems*, and the *VLDB Journal*. He is a member of the IEEE.

Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, in 1998 and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an assistant professor in the School of Computer Engineering, Nanyang Technological University, Singapore. He has served as a program committee member for a number of international conferences. He is also an editor of the book *Web Content Delivery* published by Springer. His research interests include mobile and pervasive computing, wireless sensor networks, Web and Internet, and distributed systems, particularly the data management aspects in these areas. He has published more than 30 papers in prestigious journals and conferences. He is a member of the IEEE.

Wang-Chien Lee received the BS degree from the Information Science Department, National Chiao Tung University, Taiwan, the MS degree from the Computer Science Department, Indiana University, and the PhD degree from the Computer and Information Science Department, Ohio State University. He is an associate professor in the Department of Computer Science and Engineering, Pennsylvania State University. Prior to joining Pennsylvania State University, he was a principal member of the technical staff at Verizon/GTE Laboratories. He leads the Pervasive Data Access (PDA) Research Group at the Pennsylvania State University which performs cross-area research in database systems, pervasive/mobile computing, and networking. He has served as a guest editor for several journal special issues on mobile database-related topics, including the *IEEE Transaction on Computer*, the *IEEE Personal Communications Magazine*, the *ACM Mobile Networks and Applications* (*MONET*), and the *ACM Wireless Networks* (*WINET*). He was the founding program committee cochair of the International Conference on Mobile Data Management (MDM). He is particularly interested in developing data management techniques (including accessing, indexing, caching, aggregation, dissemination, and query processing) for supporting complex queries in a wide spectrum of networking and mobile environments such as peer-to-peer networks, mobile ad hoc networks, wireless sensor networks, and wireless broadcast systems. Meanwhile, he has worked on XML, security, information integration/retrieval, and object-oriented databases. His research has been supported by the US National Science Foundation (NSF) and industry grants. Most of his research results have been published in prestigious journals and conferences in the fields of databases, mobile computing, and networking. He is a member of the IEEE and the ACM.