Wireless Netw (2014) 20:591–610
DOI 10.1007/s11276-013-0625-6

Energy-efficient top-*k* query evaluation and maintenance in wireless sensor networks

Baichen Chen · Weifa Liang · Jeffrey Xu Yu

Published online: 7 August 2013

Abstract Top-*k* query in a wireless sensor network is to identify *k* sensors with the highest sensor readings. Since sensors usually are powered by energy-limited batteries, a fundamental problem associated with top-*k* query evaluation in such a network is to maximize network lifetime, which poses great challenges due to the unique characteristics of sensor networks. In this paper, we first propose a novel filter-based algorithm for top-*k* query evaluation, which is able to filter out a fractional amount of data from network-wide transmission. We then develop an online algorithm for answering time-dependent top-*k* queries with different values of *k* through the dynamic maintenance of a materialized view that consists of historical top-*k* results. We finally conduct extensive experiments by simulations to evaluate the performance of the proposed algorithms using real sensing data sets. Experimental results show that the proposed algorithms outperform a well known existing algorithm significantly, and the network lifetime delivered by the proposed optimal quantile algorithm is at least 142 % times longer than that by an existing algorithm.

B. Chen · W. Liang (✉)
Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia
e-mail: wliang@cs.anu.edu.au

B. Chen
e-mail: baichen.chen@anu.edu.au

J. X. Yu
Department of System Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
e-mail: yu@se.cuhk.edu.hk

Keywords Top-*k* queries · Energy optimization · Distributed filtering algorithms · Routing trees · Network lifetime · Query optimization · Online top-*k* queries

1 Introduction

Top-*k* query is a fundamental operation in modern database systems, motivated by many applications such as Web service, multimedia search and monitoring natural phenomenas. Technological advances have enabled the deployment of large-scale sensor networks for environmental monitoring and security surveillances to become possible. However, efficient processing of the top-*k* query in energy-constrained sensor networks poses great challenges due to the unique characteristics of sensors and a large volume of sensing data generated by sensor networks [5, 23].

Unlike previous studies on top-*k* query evaluation in traditional databases [8, 27], in this paper we focus on top-*k* query evaluation in wireless sensor networks (WSNs). A WSN supporting top-*k* queries can be used not only to monitor the data generated by the sensors in no time but also to perform further data analysis for decision making purpose. For example, meteorologists make use of WSNs to sense meteorological attributes such as temperature, humidity, and rainfall in a region of interest. A top-*k* query in such a network can identify the *k* places suffering high temperatures for statistic analysis purpose. Another scenario is that ornithologists who study various bird species in a specific forest are interested to know where birds are most likely to gather in the forest region by placing bird feeders at various locations and using sensors to count the number of different types of birds at each feeder [26]. The query result can assist the ornithologists to determine the locations that are most attracting birds. For example, a top-k query can inquire which feeders attract the maximum number of birds. Thus, the ornithologists can observe bird behaviors at a few places where the most attractive feeders are located.

1.1 Related work

In general, query processing in WSNs is essentially different from the processing of queries in traditional databases due to the unique characteristics imposed on tiny sensors, these include their slow processing capabilities, limited storages, and energy-limited batteries, etc. [12, 17]. Specifically, the differences between the tiny sensors in sensor networks and the powerful servers in traditional databases lie in the following several aspects. Firstly, to evaluate a query in sensor networks, the energy consumption of sensors rather than the query response time will be the main optimization objective, since battery-powered sensors will become inoperative quickly if a large quantity of sensed data needs to be transmitted to the base station through multi-hop relays, while the lifetime of a sensor network is closely tied to the energy consumption rate of its sensors. To prolong the network lifetime, a desirable evaluation algorithm should be able to optimize the energy consumed by answering each top-k query. Secondly, sensors sense their vicinities periodically and will generate large volumes of continuous streaming data. A wireless sensor network containing n sensors is usually viewed as a distributed streaming system with n streaming data [7]. However, this special distributed streaming system is different from the traditional distributed streaming system. In sensor networks each sensor transmits its data to the base station through multiple relays rather than a one-hop relay, and the sensors involved in data transmission relay consume their communication energy. This implies that it is more expensive to obtain sensed data from the sensors that are far away from the base station than those near to it. Finally, as a major optimization objective of query processing in sensor networks, network lifetime is determined not only by the total energy consumption of all sensors but also by the maximum energy consumption among the sensors. The sensors near to the base station consume much more energy than the others, because they get to relay the data for others and will consequently exhaust their batteries first. Once they run out of energy, the rest of the sensors will be disconnected from the base station. As a result, the network will no longer function any more even if the total energy consumed per query is reasonably small. This implies that only minimizing the total energy consumption is insufficient, minimizing the maximum energy consumption among the sensors is crucial to prolong the network lifetime. Although there are many different definitions about the network lifetime, they are not essentially different from each other. For the sake of convenience, in this paper we adopt a widely accepted one by Chang and Tassuulas [3]. That is, the network lifetime is defined as the first sensor failure time due to its energy expiration.

Previous studies on top-k query evaluation in distributed systems focused mainly on the distributed top-k query evaluation problem, which is to locate k objects with the highest scores, assuming that each object is distributed in multiple nodes, whereas the object in each node has a partial score. The overall score of an object is the combined score of all of its partial scores by a scoring function [7]. We here deal with another kind of top-k query that is identical to those discussed in [10, 26]. That is, we assume that each sensor senses a numerical value from its vicinity. The sensing value, its generation time, and the ID of its generator (sensor) put together to form a *point*. Thus, a point p can be represented by a tuple $<p.sid, p.time, p.val>$, where $p.sid$ is the ID of sensor generating point p, $p.time$ is the time stamp of p, and $p.val$ is the sensed reading value. A *top-k query in WSNs* is to return the k points with the k highest values in the sensor network.

Although top-k query evaluation in both centralized and distributed databases has been extensively studied [7, 14–16, 28, 31], the proposed techniques and algorithms are not applicable to the same problem in WSNs. Query optimization in WSNs has also been considered in the past several years. For example, algorithms in [1, 2, 21, 22] have been proposed for simple aggregation queries, while other studies dealt with more complicated queries including order-statistics [4], skyline query [24], and top-k query [9–11, 13, 26]. Particularly, Silberstein et al. [26] considered the top-k query evaluation problem by providing a centralized, approximate solution with a high probability, based on a prediction model built on the samples of previous top-k results. They demonstrated how to improve the accuracy of the top-k results under a given energy constraint by formulating the problem as a linear programming problem and developing a series of top-k query planning algorithms. Wu et al. [9, 10] exploited the semantics of top-k query and proposed a filter-based maintenance algorithm (FILA) for continuously maintaining the current top-k results (points) by assigning each sensor a dedicated filter. Given the current top-k points, each sensor is assigned an interval of values which serves as the filter of the sensor to suppress unlikely top-k points. Meanwhile, the base station also maintains a copy of the filter at each sensor. The energy savings of their solution is assumed that the probing broadcasting from the base station can be received by all the sensors and the reception energy consumption of sensors will not be taken into account. However, in reality the reception energy consumption of each sensor cannot be ignored as the reception energy consumption of most commercial sensors is around one third of the transmission energy consumption. In addition, they assumed that the value of k is fixed and each of the top-k points has an infinite lifetime. Note that the work by Wu et al. deals with the maintenance of the top-k query results, they did not consider the top-k query evaluation. On the other hand, the proposed algorithms by both Silberstein et al. [26] and Wu et al. [9, 10] are the centralized algorithms, they may not be suitable for real distributive sensor networks. Liang et al. considered the evaluation of time-constrained top-k query by focusing on the tradeoff between the energy consumption and the end-to-end data delivery delays. Malhotra et al. [32] addressed the top-k query maintenance problem with the similar assumption imposed on the paper [10]. Chen et al. [33] studied top-k query evaluation under unreliable communication models by proposing a robust algorithm. Cheng et al. [6] considered processing of continuous historical top-k results in wireless sensor networks by proposing a simple top-k extraction algorithm. Liang et al. [27] considered the time-constrained top-k query processing.

1.2 Contributions

In this paper, our main contributions for the top-k query problem in sensor networks are as follows. We first devise a novel, distributed filter-based algorithm for top-k query evaluation, which enables to filter out as much unnecessary data as possible within the network from transmission, thereby reducing the energy consumption of sensors significantly. We then develop an online algorithm for answering various time-dependent top-k queries with different values of k through maintaining a materialized view that consists of the previous top-k results. We finally conduct extensive experiments by simulation to evaluate the performance of the proposed algorithms on real sensing datasets. The experimental results showed that the proposed algorithms outperform a popular existing algorithm significantly, and the network lifetime delivered by the proposed optimal quantile algorithm is at least 142 % times longer than that by the existing algorithm.

The remainder of this paper is organized as follows. In Sect. 2, the wireless sensor network model and the problem definition are introduced. An existing top-k query evaluation algorithm is briefly mentioned for the benchmarking purpose. In Sect. 3 a novel, distributed filtering algorithm for top-k query evaluation on snapshot datasets is devised. In Sect. 4 an online algorithm for answering various time-dependent top-k queries with different values of k on streaming datasets is developed. In Sect. 5, extensive experiments by simulations on real sensing datasets are conducted to evaluate the performance of proposed algorithms. The conclusions are given in Sect. 6

2 Preliminaries

2.1 System model

We consider a wireless sensor network consisting of n stationary sensors randomly deployed in a region of interest. There is a base station with unlimited energy supply serving as the gateway between the sensor network and users. Users issue their queries to and get the answers from the network through the base station. The battery-powered sensors are responsible for sensing and collecting sensing data from their vicinities. They are also capable of processing sensed data and transmitting aggregate results to their neighbors. Each sensor has a fixed, identical transmission range. Two sensors are neighbors if they are within the transmission range of each other. To transmit a message consisting of l bytes of data from a sensor to its neighbor, the amount of transmission energy consumed at the sender is $\rho_t + r_t{*}l$ while the amount of reception energy consumed at the receiver is $\rho_r + r_e{*}l$, where ρ_t and ρ_r are the sums of energy overhead on handshaking between the two sensors and transmitting and receiving the message header, r_t and r_e are the amounts of $\underline{t}$ransmission and r$\underline{e}$ception energy per byte. We assume that the energy consumptions on sensing and computation are not taken into account, because in practice they are several orders of magnitude less than that on wireless communication. For example, the authors in [2, 17] claimed that the transmission of 1-bit data consumes as much energy as executing 1,000 CPU instructions. Therefore, unless otherwise specified, we only compare the communication energy consumption of different algorithms by ignoring sensing and computational energy consumptions in the later performance evaluation. In addition, we must mention that in this paper we focus on devising higher-level routing protocols, rather than dealing with lower-level network layer protocols such as physical and MAC layers for transmission interference, for top-k query evaluation and maintenance in wireless sensor networks. The proposed algorithms do not need to be modified and will work for different lower-level network protocols such as TDMA/CDMA MAC layer protocols by reducing data re-transmission and interference problems.

For the sake of convenience, in the rest of the paper we assume that the sensor network is a tree $\mathcal{T}$ rooted at the base station and spanning all sensors [2]. Otherwise, a routing tree in the network rooted at the base station and spanning all sensors can be found by applying any spanning tree algorithm [1]. As each sensor generates sensing data from its vicinity continuously, the data generated by each sensor is a boundless streaming data. It is usually impossible to store all of the generated data at sensors due to their limited storage capacities. Hence we assume that time is slotted into equal *time slots* and each time slot is referred to as one *time step*.

2.2 Problem definition

We consider top-k query processing in a sliding time window with *window width* (or length) w time steps. Assuming that each reading (or a reading vector) is represented by a point p that consists of a number of attributes such as the sensor's ID $p.sid$, the data generation time $p.time$, and the data value(s) $p.val$. Here w is the *lifespan* of any point p which is the duration from its generation time $p.time$ to its expiration time $p.time + w$. We say a point p is *valid* at time step t if $p.time \geq t - w$; otherwise, p is *invalid*. We further assume that each of the attributes $p.sid$, $p.time$ and $p.val$ of a point p is represented by 4 bytes, then point p is represented by 12 bytes in total.

Denote by $P_t(v_i)$ the set of valid points at a sensor v_i and $P_t = \bigcup_{i=1}^{n} P_t(v_i)$ the set of valid points in a sensor network at time step t. Let $P_t(v_i)[t_1,t_2]$ and $P_t[t_1,t_2]$ be the subsets of $P_t(v_i)$ and P_t in which the points are generated between time steps t_1 and t_2, respectively. Denote by $Update_t(v_i)$ and $Expired_t(v_i)$ the sets of points at sensor v_i generated and expired at time step t, respectively. At each time step t, each sensor updates the set of points stored at it, i.e., $P_t(v_i) = P_{t\text{-}1}(v_i) \cup Update_t(v_i) - Expired_t(v_i)$. Therefore, $P_t = P_{t-1} \cup \bigcup_{i=1}^{n} Update_t(v_i) - \bigcup_{i=1}^{n} Expired_t(v_i)$. Because each point can only survive w time steps and all points generated before time step $t - w + 1$ must expire at time step $t, P_t = \bigcup_{i=1}^{n} \bigcup_{x=t-w+1}^{t} Update_x(v_i)$. In other words, all valid points in P_t are generated at or after time step $t - w + 1$. A *sliding window* consists of w consecutive time steps. At time step t, the sliding window contains all the points generated from $t - w + 1$ to t. Whenever the next time step $t + 1$ starts, the points generated at time step $t - w + 1$ will expire and the points generated from time step $t - w + 2$ to time step $t + 1$ is still in P_{t+1}.

A top-k query issued at time step t is represented by top-(k_t, [t_s, t_e]) with $t_s \leq t_e \leq t$, and the *top-k_t query evaluation in a sliding window* for a top-(k_t, [t_s, t_e]) query is to inquire k_t points in the network having the k_t highest values in $P_t[t_s, t_e]$, where $t - w + 1 \leq t_s \leq t_e \leq t$. If there are more than k_t points with the highest values, arbitrary k_t of them are selected as the result. *Online top-k query processing* returns the top-k results for each top-k query with different ks and time intervals that is issued at different time steps.

2.3 Algorithm `Naive-`k

We here briefly review a well known algorithm `Naive-`k for top-k query evaluation by Silberstein et al. [26], which will be used for benchmark purpose.

Algorithm `Naive-`k computes the answer bottom-up in one pass over the network, based on the routing tree structure. In algorithm `Naive-`k, each leaf sensor forwards its own points to its parent. If an internal sensor contains k' ($<k$) points (including its own), then it forwards all the points to its parent; otherwise, it first identifies the top-k points stored at it and then sends the top-k points to its parent. In the end, the base station obtains the top-k points from the set of the received points and its own points, which is the result of the top-k query.

Although algorithm `Naive-`k is very simple, it is an efficient algorithm for top-k query evaluation that we could find. Other algorithms like the probabilistic algorithms in [26] are not based on the same assumption as ours, and the solution returned by these algorithms may not be truly top-k results, rather than the top-k results with high probability. Also, the proposed probabilistic algorithms are centralized algorithms, which may not have an efficient distributed implementation. In this paper, we focus on developing efficient distributed algorithms for the top-k query evaluation and maintenance.

2.4 An α-quantile

For a given set S of points, assume that the points in S are sorted in decreasing order of their values. The *α-quantile* of S is a point whose rank is $\lceil \alpha * |S| \rceil$ and the value of the α-quantile is referred to as *α-quantile value*, where α is constant with $0 < \alpha < 1$. For example, given a set $S = \{p_1, \ldots, p_{|S|}\}$ and assume that $p_i.val \geq p_{i+1}.val$ for all i with $1 \leq i \leq |S|$, the α-quantile of S is point $p_{\lceil \alpha * |S| \rceil}$ and $p_{\lceil \alpha * |S| \rceil}.val$ is the α-quantile value of S.

3 Algorithm for top-k query evaluation

In this section, we propose a novel filter-based algorithm for top-k query evaluation on a snapshot dataset. The idea behind the proposed algorithm is as follows. Firstly, each sensor sorts its points in decreasing order of their values and sends the α-quantile value of the points to its parent. Secondly, the parent chooses one of the received α-quantile values from its children as the children filter and broadcasts the filter back to its children. Finally, the children send its points whose values are no less than the filter to their parent. The chosen filter by the parent is called the *α-quantile filter*.

In the following we first identify the quantile filter, followed by analyzing the filtering ability of the filter. We then analyze the energy savings brought by the quantile filter and the energy overhead on installing the filter to decide whether a sensor should be installed the filter, and present the α-quantile filter-based algorithm for top-k query evaluation. We finally show that the proposed algorithm is also applicable to wireless sensor networks in which sensors have duty-cycles for further energy savings.

3.1 Optimal quantile filters

Assume that a top-(k, [t_s, t_e]) query has been issued at time step t. Without loss of generality, we assume that all the points are the valid points generated between time steps t_s and t_e. Consider a sensor v in the routing tree $\mathcal{T}$ that has d_v children, they are $u_1, u_2, \ldots, u_{d_v}$. Let $L(u_i) = \{p(i)_1, \ldots, p(i)_{l(i)}\}$ be the set of points at sensor u_i with $p(i)_{j1}.val \geq p(i)_{j2}.val$ if $j_1 < j_2$, $1 \leq j_1 \leq j_2 \leq l(i)$, and $l(i)$ is the number of points at u_i if $l(i) < k$; otherwise, $l(i) = k$. In other words, there are at most k points in $L(u_i)$. $S(v) = \bigcup_{i=1}^{d_v} L(u_i)$ is the set of potential top-k points in the subtree rooted at v, and thus $|S(v)| \leq k{*}d_v$. The α-quantile of $L(u_i)$ is the point $p(i)_{\lceil \alpha * l(i) \rceil}$, and $p(i)_{\lceil \alpha * l(i) \rceil}.val$ is the α-quantile value of $L(u_i)$, which is also referred to as the α-quantile value of sensor u_i. Identifying a quantile filter that can filter out as many unlikely top-k points in $S(v)$ as possible is described below.

Each child u_i sends a pair of values, $p(i)_{\lceil \alpha * l(i) \rceil}.val$ and $l(i)$, to its parent v, $1 \leq i \leq d_v$. Sensor v that has received the d_v α-quantile values will sort the values in decreasing order. Let $q_{i_1}, q_{i_2} \ldots, q_{i_{d_v}}$ be the sorted value sequence, where $q_{i_j} (= p(i_j)_{\lceil \alpha * l(i_j) \rceil}.val)$ is the value sent by child u_{ij}. The mth largest α-quantile value q_{im} in the sequence is chosen as *the quantile filter* by sensor v, where the integer m is found such that $\sum_{j=1}^{m-1} \lceil \alpha * l(i_j) \rceil < k$ and $\sum_{j=1}^{m} \lceil \alpha * l(i_j) \rceil \geq k$. It is guaranteed that there are at least k points in $S(v)$ whose values are no less than the quantile filter $q_{i_m} = p(i_m)_{\lceil \alpha * l(i_m) \rceil}.val$. Sensor v then broadcasts the found quantile filter to each child u_i, and u_i sends the points in $L(u_i)$ whose values are no less than the quantile filter to its parent v, $1 \leq i \leq d_v$. Denote by $Send(u_i)$ the set of points sent by u_i. Having received $Send(u_i)$ from each child u_i, sensor v identifies the k points with the highest values in $\bigcup_{i=1}^{d_v} Send(u_i) \cup P(v)$ as its top-k points.

We here use examples to illustrate the procedure of finding the quantile filter. Assume that sensor v has six children $u_1, \ldots, u_6$, and the values of points at each sensor are shown in Fig. 1, where the numbers in each column are the values of points in the corresponding node of the column.

Assume that a top-5 query is issued at the base station and broadcast to v and its children. Figure 1(a) and (b) show different quantile filters identified when $\alpha = 0.4$ and $\alpha = 0.5$, respectively. In Fig. 1(a), the sorted 0.4-quantile values of the six children of v are $q_{i1(=1)} = 22$, $q_{i2(=2)} = 12$,

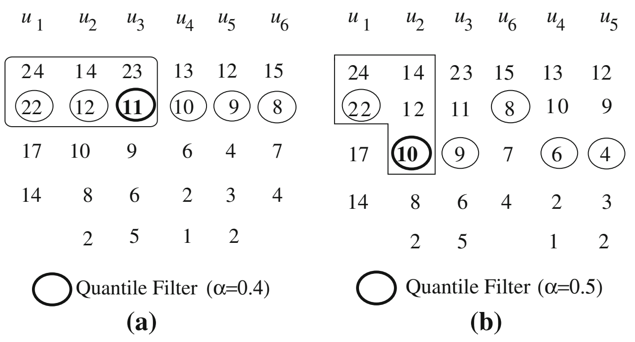


Fig. 1 The examples of quantile filters for top-5 query

$q_{i3(=3)} = 11$, $q_{i4(=4)} = 10$, $q_{i5(=5)} = 9$, and $q_{i6(=6)} = 8$, while the sorted 0.5-quantile values of the six children of v are $q_{i1(=1)} = 22$, $q_{i2(=2)} = 10$, $q_{i3(=3)} = 9$, $q_{i4(=6)} = 8$, $q_{i5(=4)} = 6$, and $q_{i6(=5)} = 4$,shown in Fig. 1(b). Because $\lceil 0.4 * l(1) \rceil + \lceil 0.4 * l(2) \rceil + \lceil 0.4 * l(3) \rceil = 6 \geq k$ and $\lceil 0.5 * l(1) \rceil + \lceil 0.5 * l(2) \rceil = 5 \geq k$, $q_{i3=3} = 11$ and $q_{i2=2} = 10$ are chosen as the quantile filters in Fig. 1(a) and (b), respectively. Notice that the values of the points in the top left corner regions (circled by the lines) are no less than the quantile filter, because there are at least k points whose values are no less than the quantile filter. The points with values smaller than the quantile filter will not be part of the final top-k results, and therefore are safely filtered out. In comparison with algorithm `Naive-k` in which the number of received points at v is $|S(v)|$, the quantile filter can filter out many points from $S(v)$ from the network-wide transmission. Consequently, this leads to less energy consumption by transmitting fewer points within the network.

As demonstrated by the example, although the use of quantile filters can prune some points in $S(v)$ from transmission, the number of points filtered out depends on the choice of the value of α. To prune as many points as possible from $S(v)$, we need to identify an optimal value of α in order to reduce the transmission and reception energy consumptions of sensors significantly. Recall that sensor v has d_v children $u_1, \ldots, u_{d_v}$. Define *the shedding ratio* of a quantile filter as the number of points filtered out by the filter to the number of points in $S(v)$. Assume that the points in the d_v children of v are arranged into d_v columns, where $L(u_{ij})$ occupies column i_j if the α-quantile value of sensor u_{ij} is ranked at the i_jth position in the sorted value sequence, as shown in Fig. 2. We sort the α-quantile values of the children of v in decreasing order and let $q_{i_1}, \ldots, q_{i_{d_v}}$ be the sorted value sequence, i.e., $q_{i_x} = p(i_x)_{\lceil \alpha * l(i_x) \rceil}.val$ and $q_{ix} \geq q_{ix+1}$. For convenience, the elements in Fig. 2 represent the values of corresponding points. Denote by $c_{x,y}$ the element located at the xth column and the yth row. All the elements in the xth column are from child u_{ix}, which have been sorted in decreasing order, i.e., $c_{x,y} = p(i_x)_y.val$ and $c_{x,y} \geq c_{x,y+1}$, $1 \leq x \leq d_v$, $1 \leq y \leq l(i_x)$. Suppose that

$q_{i_m} = c_{m,\lceil \alpha * l(i_m) \rceil}$ is chosen as the quantile filter, where $m = \min\{m' | \sum_{j=1}^{m'} \lceil \alpha * l(i_j) \rceil \geq k, 1 \leq m' \leq d_v\}$.

We observe from Fig. 2 that the elements in $S(v)$ are divided into four disjoint subsets by element q_{im} and the α-quantile value, i.e., $S_1(v)$, $S_2(v)$, $S_3(v)$, and $S_4(v)$. The values of points in $S_1(v)$ are no less than q_{im}, while the values of points in $S_4(v)$ are smaller than q_{im}. Because $|S_1(v)| \geq k$, all the points in $S_4(v)$ cannot be the top-k points. In comparison with algorithm Naive-k, at least $|S_4(v)|$ points are filtered out from $S(v)$ if the α-quantile filter is installed at the children. The ratio $\frac{|S_4(v)|}{|S(v)|}$ is thus regarded as *the shedding ratio* of the proposed quantile filter, where $|S_4(v)| = |S(v)| - |S_1(v)| - |S_2(v)| - |S_3(v)|$. The rest is to find an optimal α to maximize $|S_4(v)|$ by the following lemma.

Lemma 1 $|S_4(v)| \geq (1-\alpha)|S(v)| - d_v - \frac{1-\alpha}{\alpha}(k-1)$.

Proof It is obvious that the sizes of $S_1(v)$ and $S_2(v)$ are $|S_1(v)| = \sum_{j=1}^{m} \lceil \alpha * l(i_j) \rceil$ and $|S_2(v)| = \sum_{j=m+1}^{d_v} \lceil \alpha * l(i_j) \rceil$, respectively. Therefore, $|S_1(v)| + |S_2(v)| = \sum_{j=1}^{d_v} \lceil \alpha * l(i_j) \rceil$.

Since $\sum_{j=1}^{d_v} \lceil \alpha * l(i_j) \rceil \leq \sum_{j=1}^{d_v} (\alpha * l(i_j) + 1) = \alpha|S(v)| + d_v$, we have

$$|S_1(v)| + |S_2(v)| \leq \alpha|S(v)| + d_v. \tag{1}$$

Denote by $|S_1(v)'| = \sum_{j=1}^{m-1} \lceil \alpha * l(i_j) \rceil$. If $|S_1(v)'| \geq k$, $q_{i_{m-1}} = p(i_{m-1})_{\lceil \alpha l(i_{m-1}) \rceil}.val$ instead of $q_{i_m} = p(i_m)_{\lceil \alpha * l(i_m) \rceil}.val$ will be chosen as the quantile filter. Thus, $|S_1(v)'| \leq k - 1$.

The size of $S_3(v)$ is estimated as follows:

$$\begin{aligned}|S_3(v)| &= \sum_{j=1}^{m-1} \lfloor (1-\alpha) * l(i_j) \rfloor \leq \sum_{j=1}^{m-1} \frac{1-\alpha}{\alpha} \lceil \alpha * l(i_j) \rceil \\ &= \frac{1-\alpha}{\alpha} |S_1(v)'| \leq \frac{1-\alpha}{\alpha}(k-1),\end{aligned} \tag{2}$$

where $|S_1(v)'| \leq k - 1$.

$|S_4(v)| = |S(v)| - |S_1(v)| - |S_2(v)| - |S_3(v)|$. From inequalities (1) and (2), we have

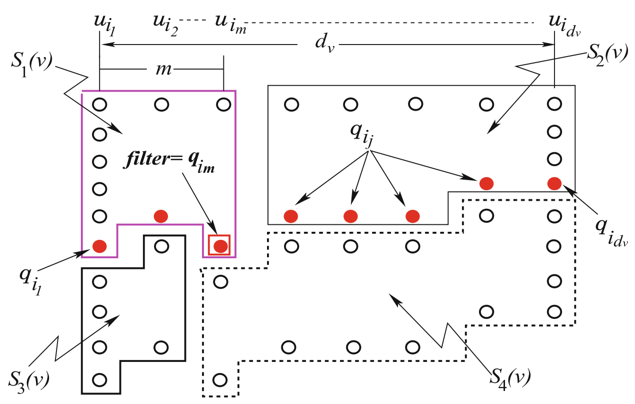


Fig. 2 The partition of set $S(v)$ by the quantile filter

$$\begin{aligned}|S_4(v)| &\geq |S(v)| - (\alpha * |S(v)| + d_v) - \frac{1-\alpha}{\alpha}(k-1) \geq \\ &(1-\alpha)|S(v)| - d_v - \frac{1-\alpha}{\alpha}(k-1).\end{aligned} \tag{3}$$

Therefore, the shedding ratio of the α-quantile filter is

$$\frac{|S_4(v)|}{|S(v)|} \geq 1 - \alpha - \frac{d_v}{|S(v)|} - \frac{(1-\alpha)(k-1)}{\alpha|S(v)|}. \tag{4}$$

By inequality (4), the shedding ratio of the quantile filter is determined by α. To maximize the value of $(1 - \alpha - \frac{d_v}{|S(v)|} - \frac{(1-\alpha)(k-1)}{\alpha|S(v)|})$ in inequality (4), we have

$$\frac{\partial((1-\alpha)|S(v)| - d_v - \frac{1-\alpha}{\alpha}(k-1))}{\partial\alpha} = 0, \tag{5}$$

i.e.,

$$-|S(v)| + \frac{1}{\alpha^2}(k-1) = 0. \tag{6}$$

Therefore, the shedding ratio of the quantile filter is maximized when $\alpha = \sqrt{\frac{k-1}{|S(v)|}}$, and the filter is referred to as the *optimal quantile filter*. The shedding ratio of the *optimal quantile filter* $\eta(v)$ thus is

$$\eta(v) = 1 - \sqrt{\frac{k-1}{|S(v)|}} - \frac{d_v}{|S(v)|} - \left(\sqrt{\frac{|S(v)|}{k-1}} - 1\right)\left(\frac{k-1}{|S(v)|}\right). \tag{7}$$

3.2 Installation of quantile filters

Although the installation of quantile filters at the children of v may eliminate unlikely top-k points from transmission, it does incur extra energy overhead on finding the quantile filter and the filter broadcasting. In the following we analyze in which case the filter should be installed, and we only install the filter at the sensors whose energy savings outweigh their energy overheads on the filter installation.

For a given sensor v, if no filter is installed at any of its children u_i, u_i sends its top-$l(i)$ points to v as algorithm Naive-k does, $1 \leq i \leq d_v$. As a result, the number of points transmitted to v is $|S(v)|$ and the total energy consumption by transmitting and receiving all $|S(v)|$ points is

$$E_{naive}(S(v)) = \rho_t d_v + |S(v)|12r_t + \rho_r d_v + |S(v)|12r_e. \tag{8}$$

Recall that each point is represented by 12 bytes. Within equation (8), $\rho_t d_v + |S(v)|12r_t$ is the total energy consumption of the d_v children transmitting the $|S(v)|$ points to sensor v, while $\rho_r d_v + |S(v)|12r_e$ is the total energy consumption of v by receiving the $|S(v)|$ points.

If the quantile filters are installed at the children of v, the shedding ratio induced by them is at least $\eta(v)$, and the total energy consumption is

$$E_{filter}(S(v)) = d_v(\rho_t + 8r_t + \rho_r + 8r_e) + \rho_t + 4r_t + d_v(\rho_r + 4r_e) + d_v(\rho_t + \rho_r) + (1-\eta(v))|S(v)|12(r_t + r_e), \quad (9)$$

where $d_v(\rho_t + 8r_t + \rho_r + 8r_e)$ is the sum of the amounts of energy consumed of the children of v by sending the α-quantile values and $l(i)$s to sensor v as well as the reception energy by sensor v, while $\rho_t + 4r_t + d_v(\rho_r + 4r_e)$ is the energy consumption of broadcasting the quantile filter. $d_v(\rho_t + \rho_r) + (1 - \eta(v))|S(v)|12(r_t + r_e)$ is the sum of energy consumption that each child sends the points passing through the quantile filter to sensor v. Note that for any leaf sensor v, $E_{naive}(S(v)) = 0$ and $E_{filter}(S(v)) = 0$ because $|S(v)| = 0$.

We now investigate the extra energy overhead on the filter finding and the energy saving of a child of sensor v by installing the filter. If there is no filter installed at a child u_i, u_i transmits its top-$l(i)$ points to sensor v and its energy consumption is

$$E_{naive}(L(u_i)) = \rho_t + l(i) * 12r_t; \quad (10)$$

otherwise, the energy consumption of u_i is

$$E_{filter}(L(u_i)) = 2\rho_t + 8r_t + \rho_r + 4r_e + (1-\eta(v))l(i)12r_t. \quad (11)$$

In equation (11), term $(1 - \eta(v))l(i)$ is the expected number of the points passing through by the quantile filter at sensor u_i. To ensure that the installation of the quantile filter is beneficial, we have $E_{filter}(S(v)) < E_{naive}(S(v))$ and $E_{filter}(L(u_i)) < E_{naive}(L(u_i))$. Meanwhile, the shedding ratio of the optimal quantile filter will be no less than 0 only when $|S(v)| \geq \frac{d_v}{1-\alpha} + \frac{k-1}{\alpha}$ from inequality (4). Combining with equations (8), (9), (10), and (11), we have

$$|S(v)| > \theta(v) = \max\left\{\frac{5d_v}{3(1-\alpha)} + \frac{(d_v+1)\rho_t + 2d_v\rho_r + 4(r_t + d_v r_e)}{8(r_t + r_e)(1-\alpha)} + \frac{(1-\alpha)(k-1)}{\alpha}, \frac{d_v}{1-\alpha} + \frac{k-1}{\alpha}\right\}, \quad (12)$$

where $\alpha = \sqrt{\frac{k-1}{|S(v)|}}, 0<\alpha<1$, and

$$l(i) > \frac{\rho_t + 8r_t + \rho_r + 4r_e}{\eta(v)12r_t}. \quad (13)$$

The values of $\theta(v)$ in inequality (12) and the shedding ratio $\eta(v)$ in inequality (13) are determined by $|S(v)|$. Note that $|S(v)| = \sum_{i=1}^{d_v} l(i)$, where $l(i) = \min\{k, |L(u_i)|\}$. In practice, the value of $l(i)$ is not known by v beforehand. A naive method for this is that each child u_i sends its $l(i)$ to v first and receives an optimal α from sensor v later. However, this method consumes additional energy. Instead, an approximation $|S(v)'|$ of $|S(v)|$ is used to find the optimal $\alpha' = \sqrt{\frac{k-1}{|S(v)'|}}$. We then determine whether the quantile filter should be installed at the children of v, where $|S(v)'|$ is the number of points received by sensor v when algorithm `Naive-`*k* is applied for top-k query evaluation. Denote by $desc(v)$ the set of descendant sensors of sensor v in the routing tree. As mentioned, at each round each sensor generates a reading value (or a point), thus, sensor v may contain $|desc(v)|$ points with each from one of its descendant sensors. For each sensor v, $|S(v)'| = \sum_{i=1}^{d_v} \min\{k, |desc(u_i)|\}$, where u_i is a child of v. In other words, if a child u_i contains more than k descendant sensors, it transmits its top-k points, or all its points to v otherwise. Obviously, the value of $|S(v)'|$ of sensor v i a routing tree can be obtained when the tree was built. In the following we show that $|S(v)| = |S(v)'|$, using the following theorem.

Theorem 1 *Given a sensor network* $\mathcal{T}(V,E)$, $|S(v)| = |S(v)'|$ *for each sensor* $v \in V$ *if the proposed optimal quantile filter strategy is adopted.*

Proof If v is a leaf sensor, $|S(v)| = |S(v)'| = 0$; otherwise, assume that $u_1, u_2, \ldots, u_{d_v}$ are the children of v. We partition the children of v into three subsets, U_1, U_2, and U_3, where $U_1 \cup U_2$ is the set of children whose descendant sensors are not installed the filters, $|desc(u_i)| < k$ if $u_i \in U_1$, and $|desc(u_i)| \geq k$ if $u_i \in U_2$. For each sensor $u_i \in U_1 \cup U_2$, $|S(u_i)| = |S(u_i)'|$ because there is no filter installed at any descendant of u_i and thus algorithm `Naive-`*k* is applied on the subtree rooted at u_i. If u_i in U_1, $|S(u_i)| = |S(u_i)'| = |desc(u_i)|$; otherwise, u_i receives at least k points, and $|L(u_i)| = k$. U_3 is the set of sensors that at least one descendant sensor is installed a filter. Assume that $u' \in desc(u_i)$ is installed a quantile filter and $u_i \in U_3$. It is clear that $|desc(u_i)| > |desc(u')| > |S(u')| > \theta(u') > k$. Thus, u_i receives at least k points from its descendant sensors. Consequently, $|L(u_i)| = k$. For each sensor $v \in V$, we have $|S(v)'| = \Sigma_{u\in U_1}(|desc(u)| + |U_2| * k + |U_3| * k$ and $|S(v)| = \Sigma_{u\in U_1} |desc(u)| + \Sigma_{u\in U_2}|L(u)| + \Sigma_{u\in U_3}|L(u)| = \Sigma_{u\in U_1}(|desc(u)| + |U_2| * k + |U_3| * k = |S(v)'|$.

3.3 Distributed quantile filter-based algorithm

Recall that sensor v has d_v children $u_1, \ldots, u_{dv}$. Each child u_i sends the number of its descendants $|desc(u_i)|$ to its parent v. Node v then broadcasts the d_v values as a set $\{|desc(u_i)|, \ldots, |desc(u_{d_v})|\}$ to its children once. Having received a top-k query, each child u_i can obtain $|S(v)'|$ (= $|S(v)|$) and $\alpha = \sqrt{\frac{k-1}{|S(v)|}}$. For a child u_i, if either $0 < \alpha < 1$ or inequality (12) or (13) is not satisfied, u_i sends its top-$l(i)$ points to sensor v directly, and no filter is installed at u_i; otherwise, a quantile filter will be installed at sensor u_i as follows. u_i first sends the α-quantile value of $L(u_i)$ and $l(i)$ to parent sensor v. If $|S(v)| = d_v{*}k$, there is no need for u_i to transmit $l(i)$ because $l(i) = k$. Sensor v then sorts the received d_v values and broadcasts the mth largest value as the quantile filter to its children, where $m = \min\{w| \sum_{t=1}^{w} \lceil \alpha l(i_t) \rceil \geq k\}, 1 \leq w \leq d_v$. Only child u_i with $l(i) > \frac{\rho_t+8r_t+\rho_r+4r_e}{\eta(v)12r_t}$ needs to receive the filter and then sends the points whose values are no less than the filter to parent v. The pseudo-code of the optimal Quantile Filter-based algorithm (QF for short) at each sensor is shown in Algorithm 1 and Algorithm 2. Notice that each internal sensor in the tree serves as parent and child roles alternatively, while a leaf sensor only serves as a child only.

We here use an example to illustrates the execution steps of algorithm QF, shown in Fig. 3(a) and (b). Sensor v has 6 children, $u_1, \ldots, u_6$, and each sensor contains several points as shown in Fig. 3(a). A top-10 query is issued at the base station and broadcast to the entire network. Since $k = 10$ and $|S(v)| = 40, \alpha = \sqrt{\frac{k-1}{|S(v)|}} = \sqrt{\frac{9}{40}} = 0.47$. Suppose that $\frac{\rho_t+8r_t+\rho_r+4r_e}{\eta(v)12r_t} = 3$. Consequently, sensor u_4 with $l(4) = 3$ will send all of its points to sensor v. The other sensors send their α-quantile values and $l(i)$s to sensor v with $\alpha = 0.47$, i.e., the pairs of the values sent to v are (6,5) from u_1, (16, 9) from u_2, (21, 9) from u_3, (11, 9) from u_5, and (19,5) from u_6. The circled numbers in Fig. 3(a) represent the values sent to sensor v in the first phase. The six quantile values received by sensor v are sorted in decreasing order, and the sorted sequence is $q_{i1(=4)} = 25$, $q_{i2(=3)} = 21$, $q_{i3(=6)} = 19$, $q_{i4(=2)} = 16$, $q_{i5(=5)} = 11$, $q_{i6(=1)} = 6$, as shown in Fig. 3(b). The quantile filter $q_{i3=6} = 19$ is chosen, because $\lceil 0.47 * l(4) \rceil + \lceil 0.47 * l(3) \rceil + \lceil 0.47 * l(6) \rceil = 10 \geq k = 10$. Sensor v broadcasts q_{i3} to its children. All children except u_4 send their points whose values are no less than q_{i3} to their parent v. In Fig. 3(b), all the points above the broken line are sent to v in the second phase. In the first phase, the children of v send 3 points (from u_4) and 5 pairs of values (from the other children) to sensor v, while in

Algorithm 1: Child_Sensor $(u_i, k, v, L(u_i))$

```
begin
    compute $|S(v)'|$ and $\alpha = \sqrt{\frac{k-1}{|S(v)'|}}$;
    $\theta(v) \leftarrow \frac{5d_v}{3(1-\alpha)} + \frac{(d_v+1)\rho_t+2d_v\rho_r+4(r_t+d_v r_e)}{8(r_t+r_e)(1-\alpha)} + \frac{(1-\alpha)(k-1)}{\alpha}$;
    if $(|S(v)'| < \max\{\theta(v), \frac{d_v}{1-\alpha} + \frac{k-1}{\alpha}\})$ or $(l(i) \leq \frac{\rho_t+8r_t+\rho_r+4r_e}{\eta(v)12r_t})$ or $(\alpha \geq 1)$ or $(\alpha \leq 0)$ then
        transmit top-$l(i)$ points to $v$;
    else
        send a pair $(p(i)_{\lceil\alpha * l(i)\rceil}.val, l(i))$ to $v$, where $p(i)_{\lceil\alpha l(i)\rceil} \in L(u_i)$ and $l(i) = |L(u_i)|$;
        if receive the quantile filter from $v$ then
            transmit the points passing through the quantile filter to $v$;
end
```

Algorithm 2: Parent_Sensor $(v, k, P(v))$

```
begin
    compute $|S(v)'|$ and $\alpha = \sqrt{\frac{k-1}{|S(v)'|}}$;
    $\theta(v) \leftarrow \frac{5d_v}{3(1-\alpha)} + \frac{(d_v+1)\rho_t+2d_v\rho_r+4(r_t+d_v r_e)}{8(r_t+r_e)(1-\alpha)} + \frac{(1-\alpha)(k-1)}{\alpha}$;
    receive the values from its children;
    if $|S(v)'| > \max\{\theta(v), \frac{d_v}{1-\alpha} + \frac{k-1}{\alpha}\}$ and $0 < \alpha < 1$ then
        the quantile filter $q_{i_m}$ is chosen, where $m \leftarrow \min\{m' \mid \sum_{i=1}^{m'} \lceil \alpha * l(i_{m'}) \rceil \geq k\}$;
        broadcast the quantile filter to children;
    if receive all the points from children then
        compute top-$k$ points from the received points and $v$'s own point;
end
```

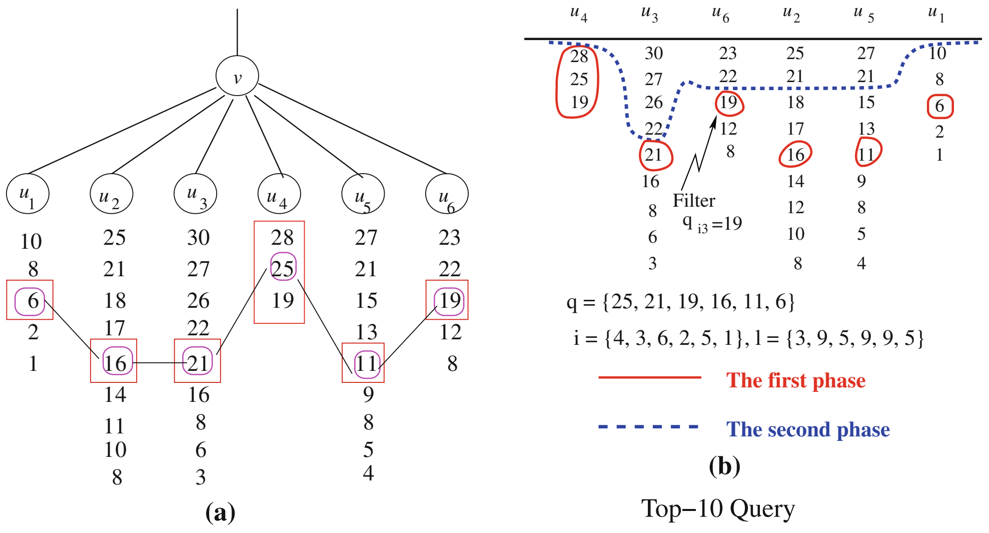


Fig. 3 The execution of the first phase of algorithm `QF`

the second phase, only the quantile filter is first broadcast by v to each child and 12 points are then sent to sensor v. If algorithm `Naive`-k is applied, all 40 points from the children are transmitted to sensor v. This example shows that algorithm `QF` transmits fewer points than algorithm `Naive`-k does, thereby reducing the energy consumption from transmission and prolonging network lifetime significantly.

It must be mentioned that there involve several message transfers between children nodes and their parent nodes in order to identify optimal filters. Although these messages will not be relayed to the base station as sensing data in the end, they are treated as the "regular messages", not treated as the "control" messages. The energy consumptions associated with these messages have also been taken into account as part of top-k query evaluation.

3.4 Analysis

We now analyze the filtering capability of algorithm `QF` by comparing the number of points transmitted by algorithms `QF` and `Naive`-k. Recall that $u_1, \ldots, u_{d_v}$ are the children of sensor v. If there is not any quantile filter installed at the children of v, $|S(v)|$ points are sent to v; otherwise, the points in $S_4(v)$ are filtered out, and the number of points sent to v is $(1 - \eta(v))|S(v)| + d_v$, where d_v is the number of pairs of values $\{p(i)_{\lceil \alpha l(i) \rceil}, l(i)\}$ sent by the children of v for quantile filter finding, $1 \le i \le d_v$. We partition the set of sensors V into two subsets V_{naive} and V_{filter}, where V_{naive} is the set of sensors whose children are not installed quantile filters, while V_{filter} is the set of sensors whose children are installed quantile filters. Let ζ be the ratio of the number of points transmitted within the network by algorithm `QF` to that by algorithm `Naive`-k, which is

$$\zeta = \frac{\Sigma_{v \in V_{naive}} |S(v)| + \Sigma_{v \in V_{filter}} ((1 - \eta(v))|S(v)| + d_v)}{\Sigma_{v \in V_{naive}} |S(v)| + \Sigma_{v \in V_{filter}} |S(v)|}. \qquad (14)$$

As different routing trees result in different sets V_{naive} and V_{filter}, the analysis of the bound of ζ is generally difficult. We thus analyze it on a special routing tree, a complete d-ary tree. Assume that the complete d-ary tree has h layers and thus includes $\frac{d^h - 1}{d-1}$ sensors. The base station is the root of the routing tree, which is located in layer 1, and the sensors located at the smaller layers are closer to the base station. Each sensor at the ith layer has $\frac{d^{h-i}-1}{d-1}$ descendant sensors. Assume that h_1 is the minimum layer at which the sensors have less than k descendants, i.e., $\frac{d^{h-h_1+1}-1}{d-1} \ge k$ and $\frac{d^{h-h_1}-1}{d-1} < k$. Thus, $h_1 = \lfloor h - \log_d(k(d-1)+1) \rfloor$. In realistic sensor networks, the average number of neighbors of each sensor is constant. Thus, the routing tree can be treated as a complete d-ary tree on average. We have the following theorem.

Theorem 2 *Assume that $\mathcal{T}(V, E)$ is a routing tree in a sensor network rooted at the base station and spanning all sensors. If $\mathcal{T}$ is a complete d-ary tree, algorithms* `QF` *and* `Naive`-*k are applied for top-k query evaluation on $\mathcal{T}$. Define $\zeta(d, h, k)$ the ratio of the number of points transmitted by algorithm* `QF` *to that by algorithm* `Naive`-*k within the network. Then, the bound of $\zeta(d, h, k)$ is $\zeta(d, h, k) = 1$ if $\theta(v) \ge d{*}k$ or $\frac{\rho_t + 8r_t + \rho_r + 4r_e}{\eta(v)12r_t} \ge k$; $\zeta(d, h, k) \le \frac{(d^h - d^{h_1})(h-h_1) + (k(1-\eta)+1)(d^{h_1}-1)}{(d^h - d^{h_1})(h-h_1) + k(d^{h_1}-1)}$ otherwise, where $\theta(v)$ is the condition of installing the quantile filter from inequality (12), h is the number of layers of the routing tree,*

$h_1 = \lfloor h - \log_d(k(d-1)+1) \rfloor$, *and* $\eta = \frac{d+2}{d} - \left(\frac{\sqrt{k}}{\sqrt{d(k-1)}} + \frac{\sqrt{k-1}}{\sqrt{dk}} + \frac{\sqrt{k-1}}{d\sqrt{dk}}\right)$.

Proof The definition of $\zeta(d,h,k)$ follows that of ζ and it is used to represent a special ζ on a complete d-ary routing tree with h layers. If algorithm `Naive-`*k* is applied to a top-*k* query evaluation, $|S(v)| = d * \frac{d^{h-i}-1}{d-1}$ for a sensor v at the ith layer where $i \geq h_1$, because $\frac{d^{h-i}-1}{d-1} < k$, while $|S(v)| = d*k$ for a sensor v at the layer smaller than h_1. Furthermore, there are d^{i-1} sensors at the ith layer of a complete d-ary tree. Therefore, the number of transmitted points by algorithm `Naive-`*k* is

$$\Sigma_{v\in V}|S(v)| = \Sigma_{i=h_1}^{h-1} d^{i-1} * \frac{d^{h-i+1}-1}{d-1} + \Sigma_{i=1}^{h_1-1}(d^i * k) = \Sigma_{i=h_1}^{h-1}\frac{d^h - d^{i-1}}{d-1} + \frac{(d^{h_1}-1)k}{d-1}. \quad (15)$$

Recall that $|S(v)| > \theta(v)$ is used to determine whether the children of sensor v are installed the quantile filters. According to the value of $\theta(v)$, we have the following cases.

Case 1 If $\theta(v) \geq d*k$ or $\frac{\rho_t + 8r_t + \rho_r + 4r_e}{\eta(v)12r_t} \geq k$, then no sensor will be installed the filter because each sensor v in a complete d-ary tree has at most k points in $L(v)$ and $d*k$ points in $S(v)$. Thus, $\zeta(d,h,k) = 1$.

Case 2 If $\theta(v) < d*k$, the quantile filters are installed at the children of each sensor v located at the layer smaller than h_1. For such a sensor v, $|S(v)| = dk$, and consequently $\eta(v) = \frac{d+2}{d} - (\frac{\sqrt{k}}{\sqrt{d(k-1)}} + \frac{\sqrt{k-1}}{\sqrt{dk}} + \frac{\sqrt{k-1}}{d\sqrt{dk}})$. The shedding ratio of all the installed quantile filters are the same, and η is used to represent the value of this shedding ratio. Thus, the number of points sent by algorithm `QF` is

$$\begin{aligned}\Sigma_{v\in V_{naive}}|S(v)| &+ \Sigma_{v\in V_{filter}}((1-\eta(v))|S(v)| + d)\\ &= \Sigma_{i=h_1}^{h-1}\frac{d^h - d^i}{d-1} + \Sigma_{i=1}^{h_1-1}(d^{i-1}((1-\eta)dk + d)) \quad (16)\\ &= \Sigma_{i=h_1}^{h-1}\frac{d^h - d^i}{d-1} + \frac{((1-\eta)k+1)(d^{h_1}-1)}{d-1}.\end{aligned}$$

Denote by

$$w_1 = \frac{\Sigma_{v\in V_{filter}}((1-\eta(v))|S(v)| + d)}{\Sigma_{v\in V_{filter}}(|S(v)|)} = \frac{((1-\eta)k+1)(d^{h_1}-1)}{(d^{h_1}-1)k} = 1 - \eta + \frac{1}{k},$$

and

$$w_2 = \frac{\Sigma_{v\in V_{filter}}|S(v)|}{\Sigma_{v\in V_{naive}}|S(v)|} = \frac{(d^{h_1}-1)k}{\Sigma_{i=h_1}^{h-1}(d^h - d^i)} \geq \frac{(d^{h_1}-1)k}{\Sigma_{i=h_1}^{h-1}(d^h - d^{h_1})} = \frac{(d^{h_1}-1)k}{(d^h - d^{h_1})(h-h_1)}. \quad (17)$$

It is obvious that $0 < w_1 \leq 1$ and $w_2 > 0$. Therefore,

$$\begin{aligned}\zeta(d,h,k) &= \frac{\Sigma_{v\in V_{naive}}|S(v)| + \Sigma_{v\in V_{filter}}((1-\eta(v))|S(v)| + d)}{\Sigma_{v\in V_{naive}}|S(v)| + \Sigma_{v\in V_{filter}|S(v)|}}\\ &= \frac{\Sigma_{v\in V_{naive}}|S(v)| + \Sigma_{v\in V_{naive}}|S(v)|w_1w_2}{\Sigma_{v\in V_{naive}}|S(v)| + \Sigma_{v\in V_{naive}}|S(v)|w_2} = \frac{1+w_1w_2}{1+w_2}\\ &\leq \frac{(d^h - d^{h_1})(h-h_1) + (k(1-\eta)+1)(d^{h_1}-1)}{(d^h - d^{h_1})(h-h_1) + k(d^{h_1}-1)},\end{aligned} \quad (18)$$

because the value of $\frac{1+w_1w_2}{1+w_2}$ decreases, with the increase of w_2, where $h_1 = \lfloor h - \log_d(k(d-1)+1)\rfloor$, and $\eta = \frac{d+2}{d} - \left(\frac{\sqrt{k}}{\sqrt{d(k-1)}} + \frac{\sqrt{k-1}}{\sqrt{dk}} + \frac{\sqrt{k-1}}{d\sqrt{dk}}\right)$.

3.5 Extensions to duty-cycling sensor networks

In real sensor networks, to further save sensor energy consumption, they force sensors to perform duty-cycling, i.e., each sensor has its duty-cycle. For example, within each time interval, a sensor spends 10 % of its time in active status while 90 % of its time in sleep status. In the following we show that the proposed distributed quantile filter-based algorithm can be applicable for this scenarios with a minor modification. That is, for each node (either as a parent node or as a child node) in the routing tree, when it is required to perform the local calculation or data transmission (receiving data from its children or sending data to its parent), it acts precisely as described in the original algorithm if the sensor is in active status; otherwise (it happens in sleep status), it can do nothing until it wakes up in the next duty-cycle, and then performs necessary actions. Consequently, the top-*k* results will be obtained at the base station. It must be mentioned that the top-*k* results with duty-cycling is very likely to be different from the ones without duty-cycling, and the reason behind is explained as follows.

Let r_g be the data generation rate and T the monitoring time interval. If there is no duty-cycling on sensors, the cardinality of the set of sensing data generated by all sensors during this period is $D_{no-duty}(T) = r_g \cdot T \cdot n$ and let $S_{no\text{-}duty}$ be the top-*k* results on this set. However, if adopting duty-cycling, let γ be the duty cycle rate with $0 < \gamma < 1$, then the cardinality of the set of data generated by all sensors during this period is $D_{duty}(T) = r_g \cdot T \cdot n \cdot \gamma$ and let S_{duty} the top-*k* results in the generated set. Clearly, the data set with duty-cycling is a proper subset of the data set without duty-cycling, since higher (sensing) readings from some sensors may be missed when they are in sleep statuses. In the worst case, none of the sensors with the top-*k* highest reading values were active, then $S_{duty} \cap S_{no\text{-}duty} = \emptyset$, the duty-cycling results are different from the non-duty-cycling result.

4 Online top-*k* query processing

So far, the proposed algorithm QF can be used to answer a top-(k_t, [t_s, t_e]) query on set $P_t[t_s, t_e]$. To answer multiple top-*k* queries with different values of *k* and various time intervals, issued at different time steps, a naive approach is to evaluate these queries one by one, by employing the proposed evaluation algorithm. However, this approach will consume excessive amount of energy because it is very likely that most of top-*k* points in previous top-*k* results are still the top-*k* points for the current top-*k* query. This is due to slow data updates (generation and expiration of points) in the network. In this section, we propose an online algorithm for evaluating top-*k* queries incrementally. Unlike previous studies on centralized databases that assumed there is only either an insertion or a deletion (an expiration) to the data base each time, or existing studies on top-*k* query in sensor networks by assuming a fixed *k* and the infinite lifespan of points ($w = \infty$) [10], In the following we propose an energy-efficient algorithm that deals with top-*k* query evaluation under multiple independent updates and supports multiple top-*k* queries with various *k*s and time intervals, through maintaining a materialized view built upon previous top-*k* results.

Intuitively, given a query top-(k_t, [t_s, t_e]), the base station maintains a materialized view that contains some previous top-*k* points, in which the points are sorted in decreasing order of their values. If there are at least k_t points in the view generated within time interval [t_s, t_e], the base station broadcasts the value of the k_tth point in the materialized view as the filter along with the query itself to the entire sensor network. Each sensor filters out the points whose values are smaller than the received filter. The proposed evaluation algorithm in the previous section is then applied to those remaining points at each sensors, and the query result then follows. Beyond this conceptually simple idea, several non-trivial issues need to be addressed. For example, how to maintain the materialized view such that it contains the k_t points generated in the unpredictable time interval of the next query with a high probability; how to decide the size of the materialized view, because it becomes prohibitive to store all historical top-*k* points in the view. In the following we first describe how to maintain the materialized view based on a probability model. We then show to answer various time-dependent top-*k* queries with different values of *k* using the materialized view.

4.1 Expected number of points in the materialized view

The materialized view at the base station consists of points which are the previous top-*k* results. The key to maintaining the view is to calculate the expected number of points generated within the time interval of the next query, referred to as $E(t_s, t_e)$. The materialized view will be dynamically updated by inserting new points from the current query result into and removing some old points from the view. Notice that the computation is based on a probability model because the exact time interval of the next query is not known beforehand. We describe the computation of $E(t_s, t_e)$ as follows.

Let t be the current time step. Denote by $V_t = \bigcup_{t_i=t-w+1}^{t} V_t(t_i)$ the set of points in the materialized view at time step *t*, where $V_t(t_i) = \{p \mid p.time = t_i\}$ is the set of points generated at time step t_i, $t - w + 1 \le t_i \le t$. Let $Pr(c_1|c_2)$ be the probability of event c_1 under condition c_2. For a point *p* with $p.time = t_i$, the probability of a point *p* generated within time interval [t_s, t_e] is

$$\begin{aligned} Pr(t_i \in [t_s, t_e]) &= Pr(t_s \le t_i \le t_e | t_e > t_s) \\ &= \frac{t_i - (t - w + 1) + 1}{w} \times \frac{t - t_i + 1}{t - t_s + 1} \\ &\ge \frac{t_i - t + w}{w} \times \frac{t - t_i + 1}{w}. \end{aligned} \tag{19}$$

The expected number of points generated in the view within interval [t_s, t_e] is

$$\begin{aligned} E(t_s, t_e) &= \sum_{t_i=t-w+2}^{t} (|V_t(t_i)| \times Pr(t_i \in [t_s, t_e])) \\ &\ge \sum_{t_i=t-w+2}^{t} \left(|V_t(t_i)| \times \frac{t_i - t + w}{w} \times \frac{t - t_i + 1}{w}\right). \end{aligned} \tag{20}$$

We refer to $E_{min} = \sum_{t_i=t-w+2}^{t} \left(|V_t(t_i)| \times \frac{t_i-t+w}{w} \times \frac{t-t_i+1}{w}\right)$ as *the minimum expected number of points* generated within the time interval of the next query. We only consider the points generated from time step ($t - w + 2$) because the points generated prior to that time step have been expired already. Notice that the above analysis is based on the assumption that t_s and t_e are randomly given, i.e., the probabilities of t_s and t_e being different at any time step are equal. If the probability distribution functions of t_s and t_e are given beforehand, the value of $E(t_s, t_e)$ can be calculated using the given probability functions. Let k_{max} be the maximum value of *k* in all issued top-*k* queries so far. If $E_{min} \ge k_{max}$, we only execute the *replacement* operations; otherwise, we execute the *insertion* operations until E_{min} of the updated view is not smaller than k_{max}, and then execute the *replacement* operations. In the following we present an online algorithm for online top-*k* query processing. We also describe how to maintain the materialized view using the replacement and insertion operations, after obtaining the top-*k* result of a top-*k* query.

4.2 Maintenance algorithm based on a materialized view

Assume that a top-(k_t, [t_s, t_e]) query is issued at time step t. The materialized view at the base station is V_t. At time step t the expired points are removed from V_t. Denote by $V_t(t_s, t_e) = \{p | t_s \le p.time \le t_e, p \in V_t\}$ the set of points in the view generated within the interval [t_s, t_e] and the points in $V_t(t_s, t_e)$ are sorted in decreasing order of their values. If $|V_t(t_s, t_e)| \ge k_t$, the value of the k_tth point in $V_t(t_s, t_e)$ is broadcast along with the query as the filter to the sensor network. The proposed filter algorithm for top-k evaluation is then applied on the set of remaining points with values not small than the filter in $P_t[t_s, t_e]$, to return the result of top-(k_t, [t_s, t_e]) query, referred to as $Top(t)$. The set of points in $Top(t)$ that are not in $V(t)$ is referred to as $New_t = \bigcup_{t_i=t-w+2}^{t} New_t(t_i)$, where $New_t(t_i) = \{p | p.time = t_i, p \in Top(t), p \notin V_t\}$. We exclude the points generated at time step $t - w + 1$ because they will expire at the next time step and will not contribute to the next query. Recall that k_{max} is the given maximum value of k in all issued queries. The base station calculates $E_{min} = \sum_{t_i=t-w+2}^{t} (|V_t(t_i)| \times \frac{t_i}{w} \times \frac{w-t_i+1}{w})$. If $E_{min} \ge k_{max}$, we will execute replacement operations until no point in New_t can increase the value of E_{min}. Otherwise, we will execute insertion operations and update the value of E_{min} of the new view until $E_{min} \ge k_{max}$, followed by executing the replacement operations. The two operations are described as follows.

The *replacement* operation replaces a point $p \in V_t$ with a point $q \in New_t$ if existent. Point p with $p.time = t_1$ is chosen from V_t such that

$$p.val = min\{x.val | x \in V_t(t_1)\}, \tag{21}$$

$$\frac{t_1 - t + w}{w} \times \frac{t - t_1 + 1}{w} = min\left\{\frac{t_i - t + w}{w} \times \frac{t - t_i + 1}{w} |V_t(t_i)| > 0\right\}. \tag{22}$$

Having chosen point p, another point q with $q.time = t_2$ is chosen from New_t such that

$$q.val = max\{x.val | x \in New_t(t_2)\}, \tag{23}$$

$$\frac{t_2 - t + w}{w} \times \frac{t - t_2 + 1}{w} = max\left\{\frac{t_i - t + w}{w} \times \frac{t - t_i + 1}{w} ||New_t(t_i)| > 0\right\}, \tag{24}$$

$$\frac{t_2 - t + w}{w} \times \frac{t - t_2 + 1}{w} \ge \frac{t_1 - t + w}{w} \times \frac{t - t_1 + 1}{w}. \tag{25}$$

Note that the ranges of t_1, t_2, and t_i in Eqs. (21)–(24) and inequality (25) are within [$t - w + 2$,t]. Having replaced point p with point q, the value of E_{min} will increase. The replacement operation will terminate if there are no such pair of points p and q meeting conditions (21)–(25).

The *insertion* operation proceeds by moving a point $q \in New_t$ that meets equations (23) and (24) into V_t when $E_{min} < k_{max}$. The value of E_{min} of the new materialized view will then be recalculated. If the updated E_{min} is not smaller than k_{max}, the replacement operations will be executed; otherwise, another point in New_t satisfying equations (23) and (24) will be inserted into the materialized view until the updated E_{min} is not smaller than k_{max}.

Having updated the materialized view based on the current top-k results, the filter for next query is then drawn

Algorithm 3: $Materalized\_View\_Maintenance(t, top(k_t, [t_s, t_e]))$

```
begin
    receive a top-($k_t$, [$t_s$, $t_e$]) query;
    if $|V_t(t_s, t_e)| \ge k_t$ then
        broadcast the value of the $k_t$th point in $V_t(t_s, t_e)$ along with the query;
    else
        broadcast the query into network;
    receive $New_t$ from the results of query;
    if $E_{min} \ge k_{max}$ then
        while points p, q meeting equations (21)-(24) and inequality (25) can be chosen do
            execute the replacement operation;
    else
        while $E_{min} \le k_{max}$ do
            execute the insertion operation and calculate $E_{min}$ of the new view;
        while points p, q meeting equations (21)-(24) and inequality (25) can be chosen do
            execute the replacement operation;
    wait for next top-$k$ query;
end
```

from the updated view if there are at least $k_{t'}$ points generated within the time interval of the next query top-($k_{t'}$, [$t_{s'}$, $t_{e'}$]). The pseudo-code for maintaining the materialized view is described in Algorithm 3.

The time complexity of Algorithm 3 is $max\{|V_t|*(|V_t| + |New_t|), |New_t|*(|V_t| + |New_t|)\}$, and the view maintenance is performed by the base station. Thus, the time spent on the view maintenance can be ignored in comparison with the duration of the next query issued.

5 Performance evaluation

In this section, we evaluated the performance of the proposed algorithms in terms of the total energy consumption, the maximum energy consumption among the sensors, the network lifetime, and the average energy consumption of sensors at different layers of the routing tree. The network lifetime is defined as the number of top-*k* queries answered before the first sensor exhausts its energy [3]. In other words, the network with a longer lifetime can answer more top-*k* queries. In the following algorithm `QF` with the optimal value of α, a fixed $\alpha = 0.3$ and a fixed $\alpha = 0.5$ are referred to as algorithm `Optimal-QF`, algorithm `0.3-QF`, and algorithm `0.5-QF`, respectively. The online algorithm is referred to as algorithm `View-Filter`. The performance of algorithm `Naive-`*k* in [26] will be used as the benchmark for the comparison purpose.

5.1 Experimental setting

We assume that the sensor network is deployed to monitor a 100 m $\times$ 100 m square region of interest, in which *n* sensors are randomly deployed by the `NS-2` simulator [20] and the base station is located at the square center. Assume that all the sensors have the same transmission range (5 meters in this paper). The initial energy at each sensor is 10^5 mJ. A routing tree will be found in the network, e.g., the *TAG* routing tree [1] (a Breadth-First-Search tree) can be used as the routing tree. As mentioned earlier, the energy consumption of a sensor on wireless communication dominates all its other types of energy consumptions, we therefore only take into account the radio energy consumption. In our experiments we adopt the transmission and reception energy consumption parameters of a real sensor MICA2 mote [18], where the energy consumption on transmitting and receiving a header and handshaking are $\rho_t = 0.4608$ mJ and $\rho_r = 0.1152$ mJ, and the energy consumption of transmitting and receiving one byte are $R = 0.0144$ mJ and $r_e = 0.00576$ mJ, respectively [18, 26]. The sensing readings (points) are drawn from a real dataset of temperature traces, collected by the Intel Berkeley Research Lab from February 28 to April 5, 2004 [19]. The temperature traces are generated by different sensors, where a fraction of sensors generates new points per a half minute.

5.2 Performance evaluation of algorithms

We first evaluate the performance of various algorithms for top-*k* query, where the range of *k* is from 80 to 150. In our experiments, two instances of sensor networks consisting of 1,500 and 2,500 sensors are considered.

Figure 4 shows the curves of the total energy consumption, the maximum energy consumption among the sensors, and the network lifetime derived by various algorithms. It can clearly be seen that the total energy consumption and the maximum energy consumption by algorithms `Optimal-QF`, `0.5-QF`, and `0.3-QF` are substantially less than that by algorithm `Naive-`*k*. This highlights the efficiency of adopting quantile filters. Meanwhile, algorithm `Optimal-QF` is the best among all mentioned algorithms. This implies the optimal quantile filter can filter out much more points from transmission. The network lifetime delivered by algorithm `Optimal-QF` is 1.25, 1.15 or 1.4 times longer than that by algorithm `0.3-QF`, `0.5-QF` or `Naive-`*k* on average for the network instances with 1,500 and 2,500 sensors.

Figure 5 plots the curves of the average energy consumption of sensors at different layers of the routing tree by applying different algorithms on both network instances consisting of 1,500 and 2,500 sensors when $k = 80$, $k = 100$, and $k = 120$, respectively. A sensor at the *i*th layer is *i*-hop away from the base station and the sensors near to the base station play more important roles in prolonging network lifetime. As shown in Fig. 5, the average energy consumption of sensors at the first 10 layers by algorithms `Optimal-QF`, `0.3-QF`, and `0.5-QF` are less than that by algorithm `Naive-`*k*. The average energy consumption of sensors in the first 3 layers by algorithm `Optimal-QF` is less than that by algorithms `0.3-QF` and `0.5-QF`. The curves imply that algorithm `Optimal-QF` efficiently balances the load among the sensors, thereby reducing the energy consumptions of sensors near to the base station. Through the performance analysis of various mentioned algorithms, algorithm `Optimal-QF` clearly prolongs the network lifetime significantly.

5.3 Impact of routing tree updating frequency and unreliable links on the algorithm performance

In this subsection we first evaluate the impact of the overhead of building routing trees and the updating frequency of routing trees on the network lifetime by algorithms `Optimal QF` and `Naive-`*k*. We then evaluate the impact of unreliable link communications on the

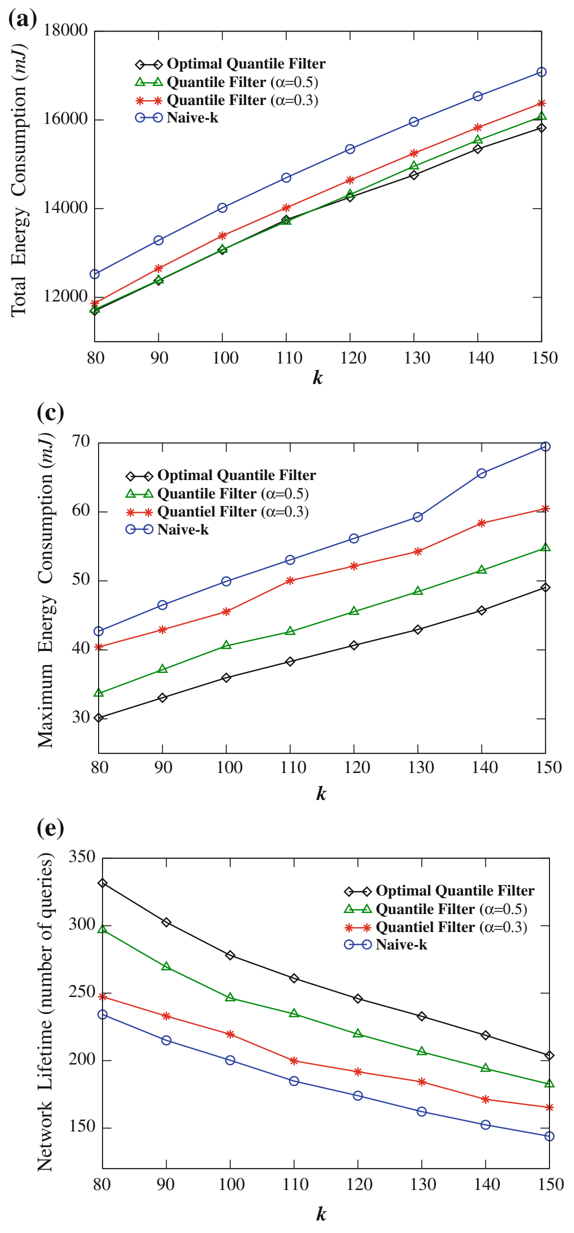


Fig. 4 The performance of various evaluation algorithms for the top-*k* query on network instances with $80 \leq k \leq 150$. **a** Total energy consumption (n = 1,500). **b** Total energy consumption (n = 2,500). **c** Maximum energy consumption (n = 1,500). **d** Maximum energy consumption (n = 2,500). **e** Network lifetime (n = 1,500). **f** Network lifetime (n = 2,500)

performance of algorithms `Optimal QF` and `Naive-`*k* in terms of the total energy consumption and the maximum energy consumption as follows.

We start with building different routing trees and different tree updating frequencies. We assume that the network size is 1,500 or 2,500 respectively. A new routing tree is built after *x* queries where the range of *x* is in [100, 1,000]. We consider three different types of routing trees: the BFS tree, the MML tree which is built based on the residual energy of nodes [29], and the CDS-backbone tree that is built based on the connected dominating set of the network [25, 30]. Figure 6 demonstrates that algorithm `Optimal-QF` outperforms algorithm `Naive-`*k* in network lifetime. Also, from which it can be seen that a higher frequency of tree updating may not necessarily lead to a longer network lifetime, because of the energy overhead on building routing trees, where the energy overhead on building a tree includes the energy consumption on message transfers within the network for the tree building and the energy consumption on resetting top-*k* query filters at sensor nodes.

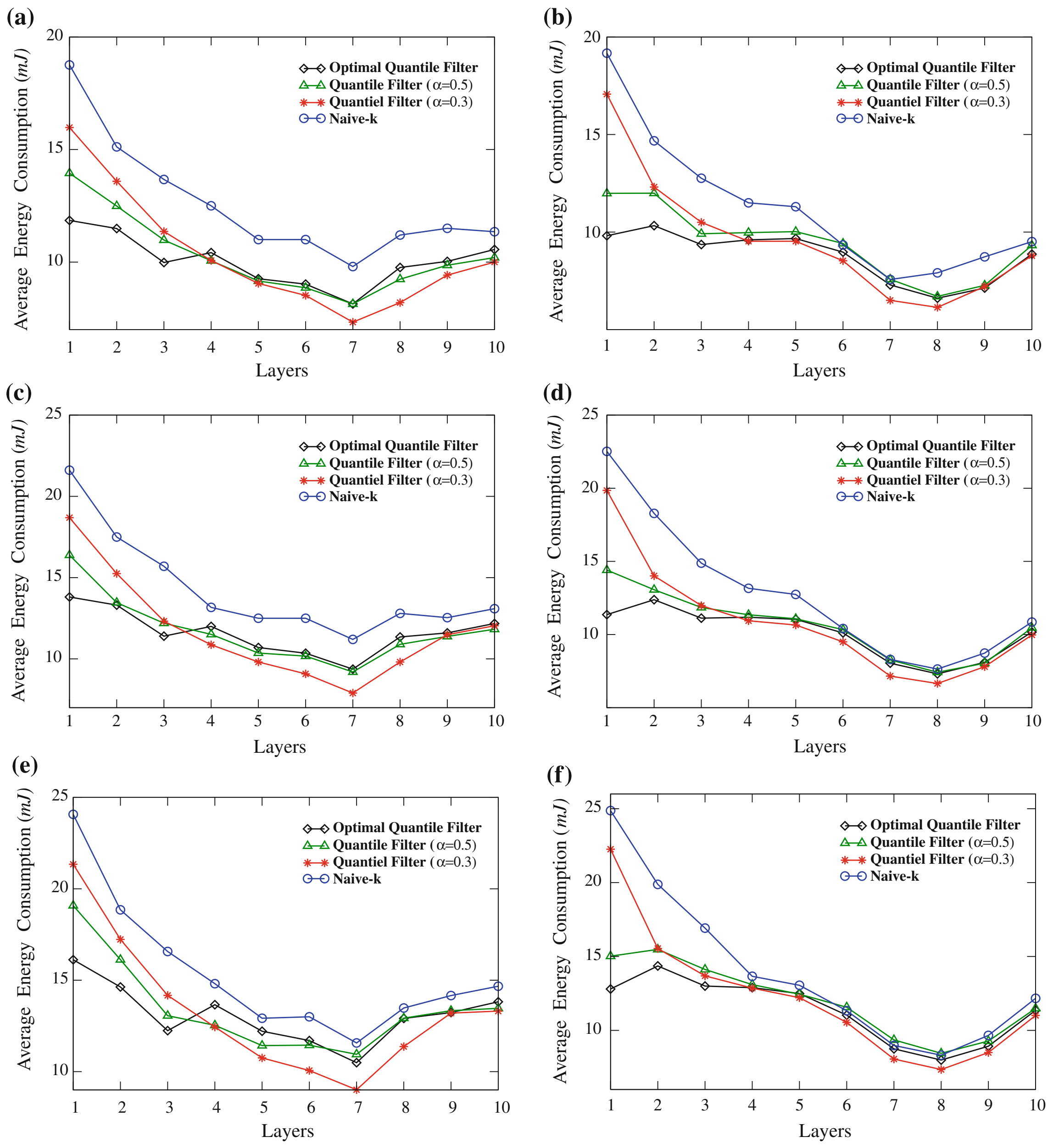


Fig. 5 The average energy consumption of sensors in different layers of a routing tree derived by different top-k evaluation algorithms. **a** Average energy consumption (n = 1,500, k = 80). **b** Average energy consumption (n = 2,500, k = 80). **c** Average energy consumption (n = 1,500, k = 100). **d** Average energy consumption (n = 2,500, k = 100). **e** Average energy consumption (n = 1,500, k = 120). **f** Average energy consumption (n = 2,500, k = 120)

We then evaluate the impact of unreliable link communications on the total energy consumption and the maximum energy consumption by algorithms `Optimal-QF` and `Naive-`*k*. We consider top-k query evaluation with different values of k under link failure rates of 10, 30, and 50 %, respectively. Figure 7 clearly indicates that algorithm `Optimal-QF` outperforms algorithm `Naive-`*k* in both the total energy consumption and the maximum energy consumption for all different link failure rates.

5.4 Performance of the online maintenance algorithm

We then study the performance of different algorithms for online answering top-k queries with various ks and time intervals on real sensing datasets [19]. We assume that each time step lasts τ minutes in the simulation and the lifespan of all points is w time steps, where τ is set as 2.5 and 5 min respectively, while the range of w is from 100 to 500. A fraction of sensors among the n sensors generate

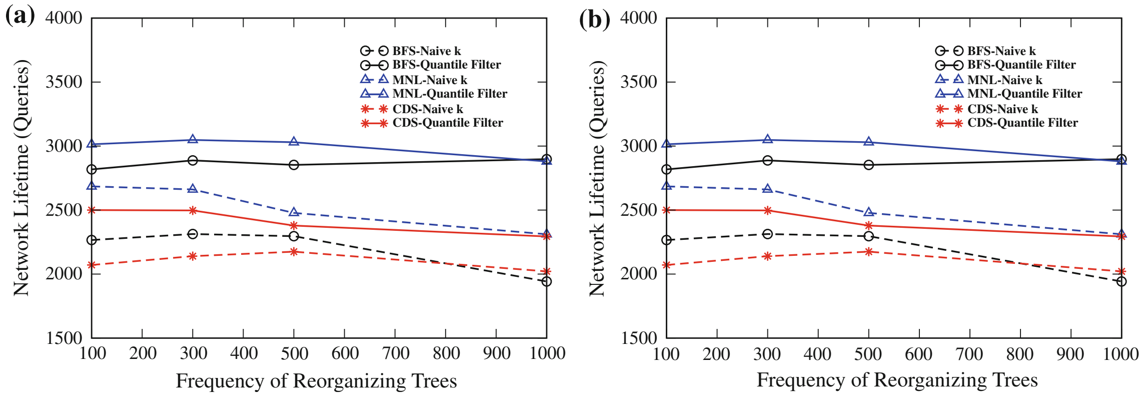


Fig. 6 The network lifetime delivered by algorithms `Optimal-QF` and `Naive-`*k* with different routing trees and updating frequencies. **a** Total energy consumption. **b** Maximum energy consumption

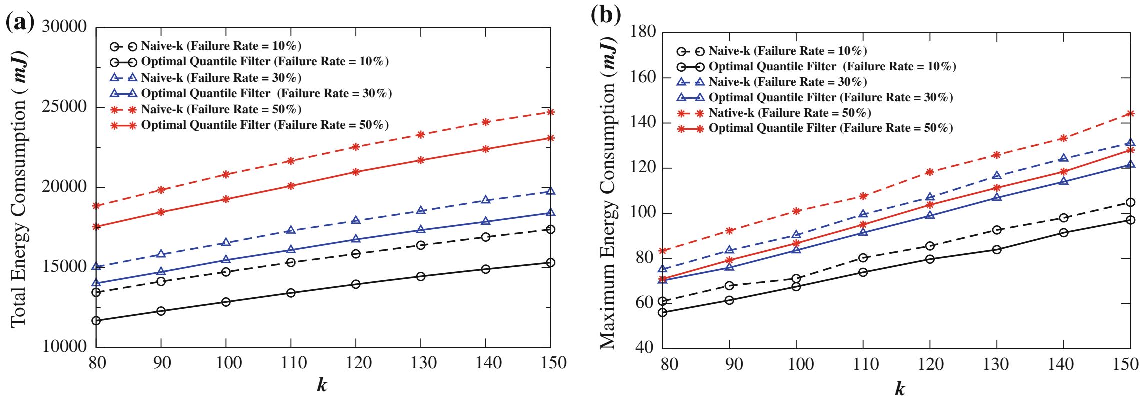


Fig. 7 The performance of algorithms `Optimal-QF` and `Naive-`*k* with different link failure rates. **a** Total energy consumption. **b** Maximum energy consumption

new points per a half minute in the datasets, thus within each time step, there are τ*2 new points updated at each sensor. With the increase of values of τ and *w*, more points will be updated (generated or expired) at each time step and the lifespan of each point will become much longer, and the number of valid points in the sensor network will be larger. Specifically, the real dataset is a collection of sensing data collected by a 54-sensor network [19]. To map the sensing data generated from the 54-sensor network into 1,500-sensor and 2,500-sensor networks in our experiments, we use $\frac{n}{50}$ (1,500 or 2,500) sensors in our networks to simulate the behaviors of one sensor in the 54-sensor network. If sensor *i* in the real sensor network generates a point at time step *t*, the corresponding sensor i' in the *n*-sensor network also generates the point at time step *t*, where $i' = (i*(n/50) + t \; mod \; (n/50))$ and $1 \leq i \leq 50$. We generate a list of $(t, k_t, [t_s, t_e])$ which represents a top-$(k_t, [t_s, t_e])$ query issued at different time step *t*, where k_t is in the range from 30 to 100 while *t* is in the range from 1 to 3,600. The query sequence follows the Poisson distribution, and the value of *k* and the starting and ending time steps of time interval $[t_s, t_e]$ are randomly generated, where $t - w + 1 \leq t_s \leq t_e \leq t$. We evaluated the performance of algorithm `View-Filter` against algorithm `Optimal-QF` and algorithm `Naive-`*k* for each query with various values of τ and *w*.

Figure 8(a) and (b) illustrate the curves of the total energy consumption by algorithms `View-Filter`, `Optimal-QF` and `Naive-`*k* at time step 3,600, while the curves of the maximum energy consumption among the sensors by algorithms `View-Filter`, `Optimal-QF`, and `Naive-`*k* at time step 3,600 are illustrated in Fig. 8(c) and (d). It can be seen that algorithm `View-Filter` is the best in terms of the total energy consumption and the maximum energy consumption among the sensors. For example, the total energy consumption by algorithm `View-Filter` is only 14 or 21 % of that by algorithm `Naive-`*k* or `Optimal-QF` in the network instance with 1,500 sensors when $w = 500$ and τ = 5min,

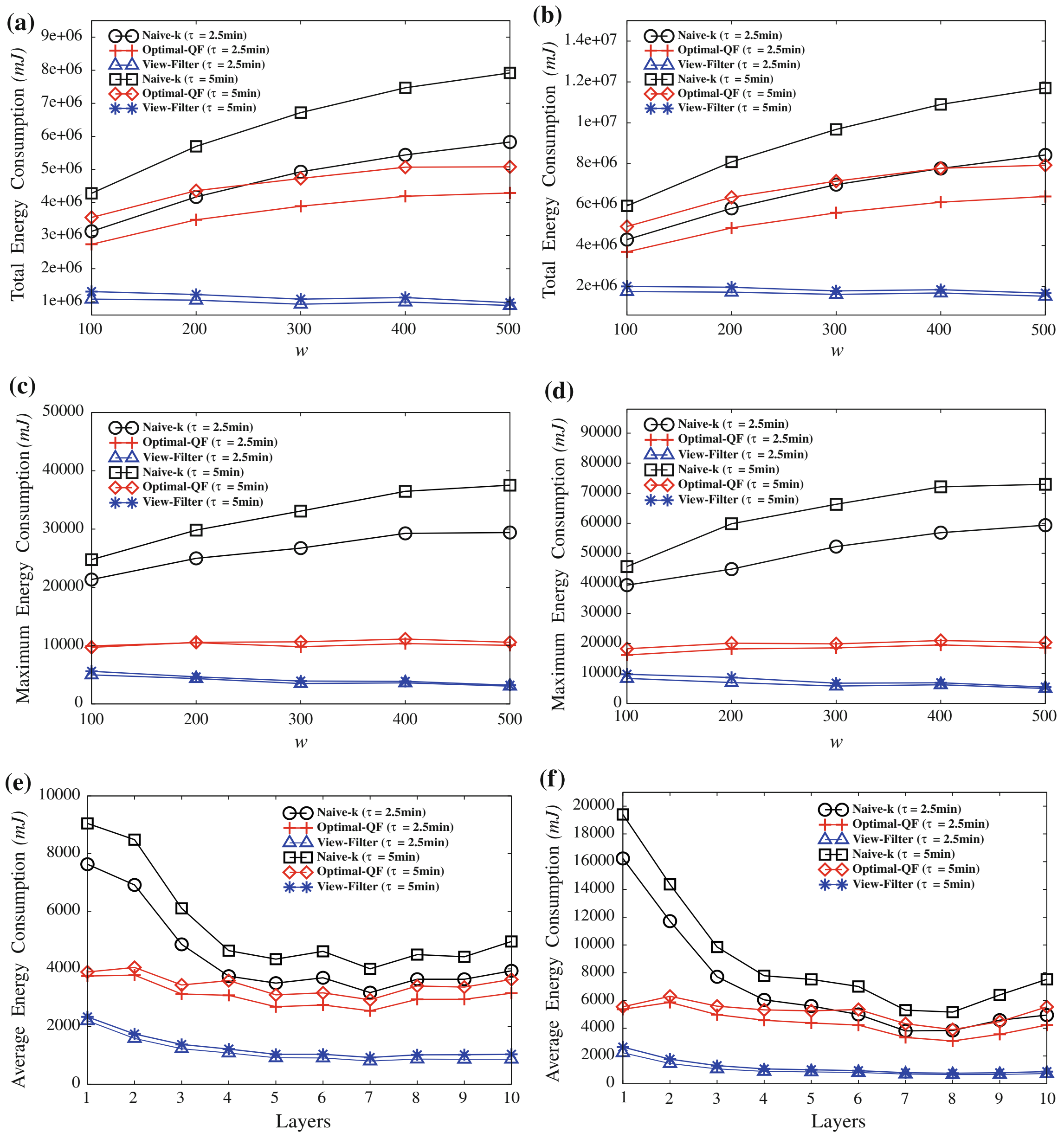


Fig. 8 The performance of different algorithms for answering various top-k queries with different values of k. **a** Total energy consumption (n = 1, 500). **b** Total energy consumption (n = 2,500). **c** Max energy consumption (n = 1,500). **d** Max energy consumption (n = 2,500). **e** Average energy consumption with w = 300, n = 1,500. **f** Average energy consumption with w = 300, n = 2,500

while the maximum energy consumption by algorithm `View-Filter` is only 8 or 27 % of that by algorithm `Naive-`*k* or `Optimal-QF`. Furthermore, the ratio of the total energy consumption as well as the maximum energy consumption among the sensors by algorithm `View-Filter` to that by algorithm `Naive-`*k* becomes smaller and smaller with the growth of w. This is because the filters in the materialized view will have a much longer lifespan and is able to filter out much more unlikely top-k points. Figure 8(e) and (f) plot the curves of the average energy consumption among the sensors at different layers by various algorithms with $w = 300$ at time step 3,600, from which it can be seen that in both network instances, the average energy consumption of sensors at the first 10 layers by algorithm `View-Filter` is smaller than that by all the other algorithms. In terms of the total energy consumption and the maximum energy consumption among the sensors, algorithm `View-Filter` consumes much less than the optimal-quantile algorithm, thereby prolonging the network lifetime significantly.

5.5 Performance evaluation of the proposed algorithms under duty-cycling

In this subsection, we investigate the impact of duty cycles on various energy consumptions by the proposed algorithms. In the following we start with the evaluation algorithm, followed with the maintenance algorithm.

5.5.1 *Algorithm evaluation*

We study the impact of duty cycles on the total and the maximum energy consumptions by algorithms `Optimal-QF` and `Naive-`*k*. We divide each time step within 10 time slots, i.e., a query evaluation will be finished within 10 time slots. The same duty cycle rate *dcr* is set for all the sensors, and each sensor is randomly assigned its first active time slot *ts* that is within 1 to $10 - \lfloor 10*dcr \rfloor$ and the active period of the sensor is from time slot *ts* to $ts + \lfloor 10*dcr \rfloor$. For example, when *dcr* is 30 % and a sensor is assigned its first active time slot 3, then it will keep active at time slots 3, 4 and 5. In our experiments, we set *K* as 80, and run top-80 query 1,000 times with duty cycles of each sensor from 10 to 60 %. Each time when all sensors are assigned with a different duty-cycle rate, the first active time slot of each sensor is randomly assigned as well. The value in Fig. 9 is the mean of the values obtained based on 1,000 experiments.

From Fig. 9 we can see that when adopting the duty cycles in both algorithms `Optimal-QF` and `Naive-`*k* prolongs the network lifetime significantly. With the rate increase of duty cycles, algorithm `Optimal-QF` clearly outperforms algorithm `Naive-`*k*. When the rate of duty cycles is low, each sensor will receive much fewer points from its descendant sensor nodes due to the fact that many of these sensors are in sleep statuses. Thus, in many cases algorithm `Optimal-QF` will not be applied due to the inequality (12) and inequality (13). However, with the growth of the duty cycle rate, algorithm `Optimal-QF` shows its superiority of energy savings on both total and

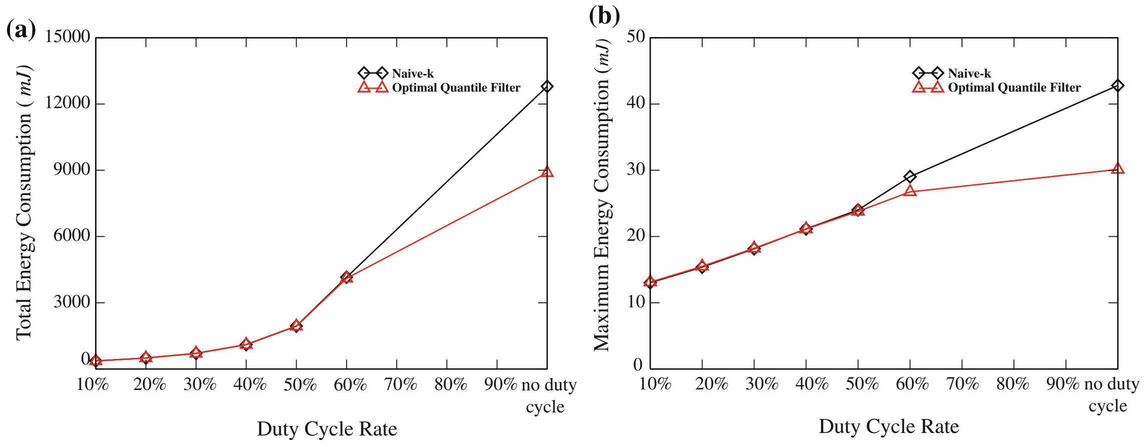


Fig. 9 The performance of algorithms `Optimal-QF` and `Naive-`*k* with different duty cycle rates. **a** Total energy consumption. **b** Maximum energy consumption

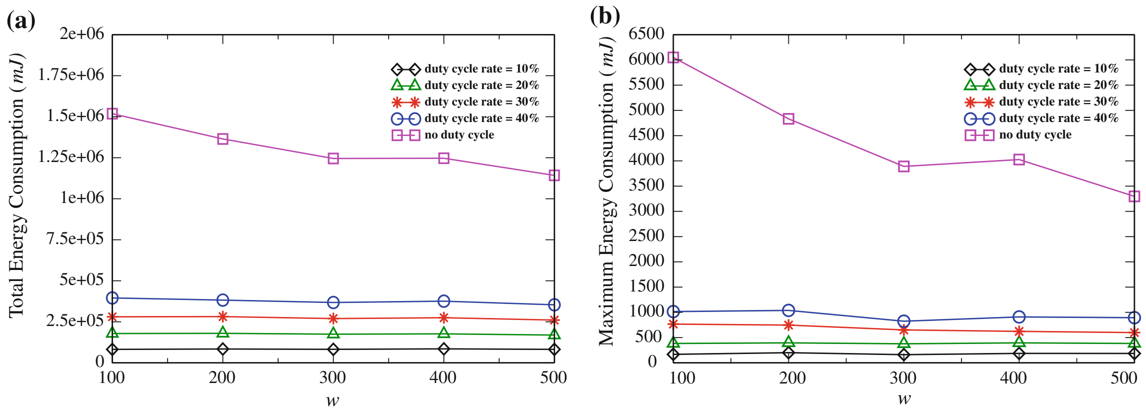


Fig. 10 The performance of algorithm `View-Filter` answering various time-dependent top-*k* queries with different duty cycle rates. **a** Total energy consumption. **b** Maximum energy consumption

maximum energy consumptions. For example, when the duty cycle rate reaches 60 %, the maximum energy consumption by algorithm `Optimal-QF` is no more than 90 % of that by algorithm `Naive`-k. Although duty cycling prolongs the network lifetime, it must be pointed out that the top-k results with duty cycling are not as accurate as the top-k results without duty-cycling.

5.5.2 Maintenance algorithm

We then study the impact of duty cycles on the energy consumption by the proposed maintenance algorithm, by varying the rate of duty cycles from 10 to 40 %. Denote by dcr the rate of duty cycle. Assume that t is the current time step and the window length $\tau = 10$. A sensor is activated at t, it keeps active in the next $\lceil dcr * 10 \rceil$ time steps before it enters the sleep status. And it will be re-activated at time step $t + 10$. We assume that each time step lasts 5 min and the lifespan of all points is w time steps. The network contains 1,500 sensors, and the rest setting is the same as the one in previous subsections.

Figure 10 plots the curves of the total energy consumption and the maximum energy consumption of the solution delivered by algorithm `View-Filter` under different duty cycle rates. Clearly adopting duty cycles can prolong the network lifetime substantially. For different window length w and adopting a duty cycle rate 40 %, the total energy consumption ranges only from 25 to 30 % of it without duty cycle, while the maximum energy consumption varies from 17 to 27 % of it without duty cycle. The algorithm performance is much better when adopting a lower duty cycle rate. However, due to the inactive sensors cannot sense and transfer the data during their sleeping periods, the adoption of duty cycle will result in significant data loss, thereby the base station cannot get the accurate top-k results in this period.

6 Conclusions

In this paper, we have studied energy-efficient evaluating top-k query in wireless sensor networks with an objective to maximize the network lifetime. We first proposed a novel, filter-based distributed algorithm that enables to filter out as many unlikely top-k results as possible from the network-wide transmission. We then devised an online algorithm for answering various time-dependent top-k queries with different values of k in sliding window environments. We finally conducted extensive experiments by simulations to evaluate the performance of proposed algorithms on real sensing datasets. The experimental results showed that the proposed algorithms are efficient and can prolong network lifetime significantly. Particularly, the network lifetime delivered by the proposed optimal quantile algorithm is at least 142 % times longer than that by the existing algorithm.

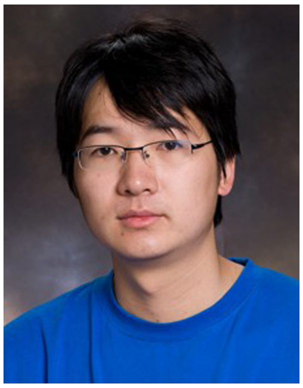
Acknowledgments We would like to thank Xiaojiang Ren and Wenzheng Xu for their help in experimental evaluation.

References

1. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). TAG: A tiny aggregation service for ad hoc sensor networks. *ACM SIGOPS Operating Systems Review, 36*, 131–146.
2. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proceedings of the ACM SIGMOD*, ACM, pp. 491–502.
3. Chang, J.-H., & Tassiulas, L. (2000). Energy conserving routing in wireless ad hoc networks. In *Proceeding of the INFOCOM'00, IEEE*.
4. Greenwald, M. B., & Khanna, S. (2004). Power-conserving computation of order-statistics over sensor networks. In *Proceedings of ACM PODS*, ACM, pp. 275–285.
5. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine*, pp. 102–114.
6. Cheng, J., Jiang, H., Liu, J., Liu, W., & Wang, C. (2011). On efficient processing of continuous historical top-k queries in sensor networks. *IEEE Transactions on Vehicular Technology, 60*, 2363–2367.
7. Babcock, B., & Olston, C. (2003). Distributed top-k monitoring. In *Proceeding of ACM SIGMOD*, ACM, pp. 28–39.
8. Ilyas, I. F., Beskales, G., & Soliman, M. A. (2008). A survey of top-k query processing techniques in relational database systems. *ACM Computing Survey, 40*(4).
9. Wu, M., Xu, J., Tang, X., & Lee, W.-C. (2006). Monitoring top-k query in wireless sensor networks. In *Proceedings of ICDE*, IEEE, 2006.
10. Wu, M., Xu, J., Tang, X., & Lee, W-C. (2007). Top-k monitoring in wireless sensor networks. *IEEE Transaction on Knowledge and Data Engineering, 19*(7), 962–976.
11. Chen, B., Liang, W., & Yu, J. X. (2010). Online time interval top-k queries in wireless sensor networks. In *Proceedings of MDM, IEEE*, pp. 177–182.
12. Chen, B., Liang, W., Yu, J. X. (2012). Energy-efficient skyline query optimization in wireless sensor networks. *Wireless Network, Springer, 18*, 985–1004.
13. Chen, B., Liang, W., Zhou, R., & Yu, J. X. (2010). Energy-efficient top-k query processing in wireless sensor networks. In *Proceedings of CIKM*, ACM, pp. 329–338.
14. Fagin, R., Lotem, A., & Naor, M. (2001). Optimal aggregation algorithms for middleware. *Journal of Computer and System Science, 1*, 614–656.
15. Marian, A., Gravano, L., & Bruno, N. (2004). Evaluating top-k queries over web-accessible databases. *ACM Transaction on Database Systems, 29*(2), 319–362
16. Das, G., Gunopulos, D., Koudas, N., & Sarkas, N. (2007). Ad-hoc top-k query answering for data streams. In *Proceedings of VLDB*, pp. 183–194.
17. Pottie, G. J., & Kaiser, W. J. (2000). Wireless integrated network sensors. *Communication of ACM, 43*, 51–58.
18. Crossbow Inc. *MPR-Mote Processor Radio Board Users Manual*.
19. http://db.csail.mit.edu/labdata/labdata.html.
20. Network simulator-ns2. http://www.isi.edu/nsnam/ns, 2006.
21. Yao, Y., & Gehrke, J. (2002). The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record, 31*, 9–18.
22. Yao, Y., & Gehrke, J. (2003). Query processing for sensor networks. In *Proceedings of the 2003 conference on innovative data systems research*, January, 2003.

23. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., & Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of international workshop on wireless sensor networks and applications*, ACM, 2002.
24. Liang, W., Chen, B., & Yu, J. X. (2008). Energy-efficient skyline query processing and maintenance in sensor networks. In *Proceedings of ACM CIKM*, ACM, pp. 1471–1472.
25. Basagni, S., Mastrogiovanni, M., Panconesi, A., & Petrioli, C. (2006). Localized protocols for ad hoc clustering and backbone formation: A performance comparison. *IEEE Transactions on Parallel and Distributed Systems, 17*, 292–306.
26. Silberstein, A., Braynard, R., Ellis, C., Munagala, K., & Yang, J. (2006). A sampling-based approach to optimizing top-*k* queries in sensor networks. In *Proceedings of ICDE, IEEE*, pp. 68–79.
27. Fagin, R. (1996). Combining fuzzy information from multiple systems. In *Proceedings of PODS*, pp. 216–226.
28. Yi, K., Yu, H., Yang, J., Xia, G., & Chen, Y. (2003). Efficient maintenance of materialized top-*k* views. In *Proceedings of ICDE*, IEEE, pp. 189–200.
29. Liang, W., & Liu, Y. (2007). Online data gathering for maximizing network lifetime in sensor networks. *IEEE Transactions on Mobile Computing, 6*, 2–11.
30. Wu, J., & Lin, H. (1999). On calculating connected dominating set for efficient routing in ad hoc wireless networks. *Prof DIALM'99*, ACM, 1999.
31. Cao, P., & Wang, Z. (2004). Efficient top-*k* query calculation in distributed networks. In *Proceeding of PODC'04*, ACM, pp. 206–215.
32. Malhotra, B., Nascimento, M. A., & Nikolaidis, I. (2010). Exact top-*k* queries in wireless sensor networks. *IEEE TKDE, 23*, 1513–1525
33. Chen, B., Liang, W., & Min, G. (2011). Top-*k* query evaluation in sensor networks with guaranteed accuracy of query results. *Proc of DEXA*, LNCS, pp. 156–171.

Author Biographies



Baichen Chen received the Ph.D. degree from the Australian National University in 2012, the M.E. and the B.Sc. degree from Northeastern University, China in 2007 and 2004 respectively, all in computer science. He is currently a financial software engineer in Research and Development Department of Bloomberg company. His research interests include information processing in wireless sensor networks, design and analysis of distributed algorithms and graph theory.

Weifa Liang received the Ph.D. degree from the Australian National University in 1998, the M.E. degree from the University of Science and Technology of China in 1989, and the B.Sc. degree from Wuhan University, China in 1984, all in computer science. He is currently an Associate Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of energy-efficient routing protocols for wireless ad hoc and sensor networks, information processing in wireless sensor networks, cloud computing, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Jeffrey Xu Yu received the B.E., M.E., and Ph.D. degrees in computer science, from the University of Tsukuba, Japan, in 1985, 1987, and 1990, respectively. Currently he is a Professor in the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong. His major research interests include graph mining, graph database, keyword search, and query processing and optimization. He is a senior member of the IEEE, a member of the IEEE Computer Society, and a member of ACM.