Efficient and Fast Distributed Top-k Query Protocol in Wireless Sensor Networks

Shao-Jie Tang[†], Xufei Mao[‡], Xiang-Yang Li[‡†]
Email: {stang7, xmao3}@iit.edu, xli@cs.iit.edu
[†]Department of Computer Science, Illinois Institute of Technology, USA
[‡] TNLIST, School of Software, Tsinghua University

***Abstract*—In this paper, we focus on designing efficient query of top-k data produced by sensor nodes in a wireless sensor network (WSN). Assume that we are given a connected WSN of diameter D, consisting of n nodes with maximum node degree Δ. Two different models are studied. In the first model, each node holds a numeric element, the goal is to determine the top-k smallest (or biggest) of these elements from all nodes. In the second model, there are m objects in set $\mathcal{L}$, each node $v_i, 1 \leq i \leq n$ holds a numeric value $s_j(v_i)$ for each object $L_j \in \mathcal{L}, 1 \leq j \leq m$, the goal is to find the k objects in $\mathcal{L}$ with the k smallest (or biggest) aggregated values $f(s_j(v_1), s_j(v_2), \cdots, s_j(v_n))$, where f is an aggregation function given in advance. We propose both fast and message efficient methods for conducting top-k queries in the two aforementioned models. Following that we study the minimum delay and messages required by *any* distributed method for top-k queries in both models. Our analysis shows that our methods are almost optimum. We conducted extensive experiments in both testbed and simulations to study the practical performances of our methods.**

***Index Terms*—Wireless networks, top-k query, aggregation, scheduling.**

I. Introduction

In this paper, we design efficient top-k query methods with small delay and small exchanging messages, and study some fundamental complexities for various top-k queries in wireless sensor networks (WSNs). This work is motivated by many top-k related applications in WSN. For instance, one of important applications [17] in agriculture using WSNs is to monitor the light level of different areas inside a forest. In this application, agriculture researchers are usually more interested at some highest light readings. The other application is forest fire monitoring, in order to detect potential forest fire using WSN, we should be aware of those nodes with highest temperature or smoke readings.

A typical or naive implementation of top-k query would be using a centralized approach where all sensor readings are collected by the base station (or the sink node), which then computes the top-k result set. However, this is overkill in most cases since collecting all sensor readings is less efficient and effective [28]. Designing efficient top-k queries has been widely studied in the database community [1], [6], [15], [24]. Surprisingly, only a few results [22], [27]–[30] have been proposed to deal with top-k queries in WSNs.

In this paper, we study two types of top-k queries in WSNs. The first type focuses on the scenario that each wireless sensor node holds some data, *e.g.*, the data could be the temperature readings, moisture readings or light lever readings during the last several minutes or hours. Since all readings of all wireless nodes are kind of independent of each other, we call this "single object model", *i.e.*, each reading can be considered as a single object. The top-k query under *single object model* is then to find the top-k ranked items, *i.e.*, the k highest (or lowest) temperature (or moisture) readings during a period.

The second type focuses on the scenario that the data items collected by different sensor nodes can be treated as different measurements of same objects. In this model, we assume that there are totally m objects. The value of an object is an aggregation of the observed values of all sensors for this object. For example, the aggregation functions could be summation, average and so on. We call this type as "multiple objects model". The top-k query under *multiple objects model* is then to find the top-k objects whose aggregated observed value are top-k ranked.

To the best of our knowledge, we are the first to study message efficient and minimal delay top-k query for WSNs, to present lower bounds on the minimum number of messages, and minimum delay, required for these two types of top-k queries by *any* distributed methods for multihop WSNs. The main contributions of this paper are as follows.

1) For single object model, we present efficient randomized algorithms for top-k queries. One algorithm is able to find top-k objects in expected time $O(\Delta + D\log_D k + k)$ for any wireless sensor network G with diameter D, maximum degree Δ, and each wireless node has at most $O(1)$ different data items (objects). We show that any randomized distributed algorithm needs at least $\Omega(D\log_D k + k)$ time slots to find the top-k data items and any deterministic algorithm needs at least $\Omega(\Delta + D\log_D k + k)$ time slots to find the top-k data items. Thus our method is almost optimum. We also propose distributed top-k query method that incurs small message overhead. We prove that our method costs at most $O(\min(n + kn_{\mathbf{c}}, n + n_{\mathbf{c}}(k + \log n_{\mathbf{c}}) + kD))$ messages to find the top-k items. Here $n_{\mathbf{c}}$ is the size of a connected dominating set. We further show that there exists a network G of n nodes such that $\Omega(kD + n)$ messages are required by any method (deterministic or randomized) to compute the k-th smallest (or biggest) elements (objects) in G under the single object model.

2) Under multiple objects model, we assume that there are m different objects totally in wireless network G with n wireless nodes. We design efficient deterministic distributed methods that can find the k objects whose aggregated values are top-k ranked. Our methods answer the top-k queries in

$O(D+m\Delta)$ time slots, with $O(mn)$ messages. We show that any deterministic distributed top-k query method in this model needs $\Omega(D+m\Delta)$ time slots, and $\Omega(mn)$ messages.

3) We studied the practical performances of our methods by extensive testbed (consisting of 36 Telosb nodes) experiments. The experimental results verify the feasibility of all proposed algorithms on small sensors with limited energy and memory. We further evaluate our algorithms in large scale network through simulation, simulation results show that our algorithms perform well as network scales.

II. Preliminaries and Problem Formulation

A. Wireless Sensor Network Model

We mainly focus on designing efficient methods for, and study the complexities of, top-k query in WSNs. We assume a simple and general model that is widely used in the community. Assuming that there are $n+1$ wireless sensor nodes $V=\{v_0,v_1,\cdots,v_n\}$ deployed in a certain geographic region, where v_0 is the sink node. Each wireless sensor node corresponds to a vertex in a graph G and two vertices are connected iff their corresponding sensor nodes can communicate directly. The graph G is called the *communication graph* of this sensor network. We say that wireless communication links are "reliable": when a node v_i sends some data to a neighboring node v_j within $v_i's$ communication range, the total message cost is only $O(1)$, although v_i may need to re-transmit several times in practice. In addition all wireless sensor nodes transmit at a fixed power, and there is a fixed integer ϱ such that a transmitting node z will cause interference to a node v (receiving data from another sender u) if the hop distance between v and z is at most ϱ. For example, $\varrho=2$ or 3 typically in the literature.

Let $h_G(v_i,v_j)$ be the hop number of the minimum hop path connecting v_i and v_j in graph G, and $D(G)$ be the diameter of the graph. Here, we assume that $D(G)\geq 2$. Otherwise, the graph G is simply a completed graph and all questions studied in this paper can either be trivial or have been solved [10]–[12]. For a graph G, we denote its maximum degree as $\Delta(G)$. When each node v_i has n_i data items, we define the weighted degree, denoted as $\tilde{d}_{v_i}(G)$, of a node v_i in graph G as $n_i+\sum_{v_j:v_iv_j\in G}n_j$. The maximum weighted degree of a graph G, denoted as $\tilde{\Delta}(G)$, is defined as $\max_i \tilde{d}_{v_i}(G)$.

For some of the results, we further assume that the wireless sensor network G is growth-bounded by a polynomial function g. A network G is said to be growth-bounded by function g if for every node v and any integer ℓ, any set S of independent nodes within ℓ hops of node v has size at most $g(\ell)$.

We use A_i to denote all data items (*e.g.*, temperature, moisture and so on) collected by wireless node v_i. We assume that one packet (*i.e.*, message) can only contain one data item $a\in A_i$, plus the node ID and additional constant number of bits, *i.e.*, the packet size is at the order of $\Theta(\log n+\log U)$, where U is the upper-bound on values of a. Such a restriction on the message size is realistic and needed, otherwise a single convergecast would suffice to accumulate all data items to the sink which will subsequently solve the problems easily. We consider a TDMA MAC schedule and assume that one time-slot duration allows transmission of exactly one packet.

For data queries in WSNs, we often need to build a spanning (routing) tree T of the communication graph G first for pushing down queries and propagating back the intermediate results. Given a tree T, we use $H(T)$ denote the height of the tree.

B. Problems and Complexities

We mainly study the time complexity and message complexity of distributed top-k query in multihop wireless networks. The complexity measures used to evaluate the performance of a given protocol are worst-case measures. The *message complexity* of a protocol is defined as the maximum number of total messages transmitted by all nodes over all inputs, *i.e.*, over all possible wireless networks $\mathcal{G}$ of n nodes (with additional requirement of having diameter D and/or maximum nodal degree Δ) and all possible data distributions of A over V. The *time complexity* is defined as the elapsed time from the time when the first message was sent by some node to the time when the last message was received by the sink. The *lower bound* on a complexity measure is the minimum complexity required by *all* protocols that answer the queries correctly. We now formally define top-k query problem under two different models.

Single Object Model: In this model, we assume that $A=\{a_1,a_2,\cdots,a_N\}$ is a totally ordered multi-set of N elements collected by all n nodes. Each node v_i has n_i number of raw data, denoted as $A_i\subset A$. Since A is a multi-set, we assume $A_i\cap A_j=\emptyset$ and $A=\bigcup_{i=1}^{n}A_i$. Then $\langle A_1,A_2,\cdots,A_n\rangle$ is called a distribution of A at sites of V. The top-k query is then to find the top-k ranked items under a certain ranking function $r()$. Here we assume that, given any two data items a_i and a_j, we can compare the ranks $r(a_i)$ and $r(a_j)$ in constant time.

Multiple Objects Model: The second type focuses on the scenario that the data items collected by different sensor nodes can be treated as different measurements of same objects. Under this model, we assume that there are totally m data objects $\mathcal{L}=\{L_1,L_2,\cdots,L_m\}$. Each wireless sensor node $v_i\in V$ will have a measurement $s_j(v_i)$ for the object L_j. In other words, each wireless node $v_i\in V$ will have a vector $s(v_i)=[s_1(v_i),s_2(v_i),\cdots,s_m(v_i)]$. Assume that there is an aggregation function f such that for object L_j, its *aggregated observed value*, denoted as b_j, is defined as $f(s_j(v_1),s_j(v_2),\cdots,s_j(v_n))$. Here we assume that the aggregation function f is distributive and symmetric. Assume that we are given an aggregation function f that can be expressed as the combination of q distributive functions $g_1()$, $g_2()$, $\cdots$, $g_q()$, for some integer constant q, *i.e.*, $f(X)=h(g_1(X),g_2(X),\cdots,g_q(X))$. For example, when f is *average*, then $q=2$, g_1 can be set as *sum*, g_2 can be set as *count* (obviously both g_1 and g_2 are distributive) and h can be set as $h(y_1,y_2)=y_1/y_2$. Hereafter, we assume that an algebraic function f is given in formula $h(g_1(X),g_2(X),\cdots,g_q(X))$ by providing functions $h()$, $g_i()$, for $1\leq i\leq q$. Thus, instead

of computing f, we can just compute $y_i = g_i(X)$ distributively for $i \in [1, q]$ and $h(y_1, y_2, \cdots, y_q)$ at the sink node.

When v_i does not have a measurement for the object L_j, we use $s_j(v_i) = \emptyset$ to denote this and v_i will not participate in the final aggregation of the value $f(s_j(v_1), s_j(v_2), \cdots, s_j(v_n))$. The top-$k$ query is then to find the top-k ranked items whose aggregated value are the k smallest (or biggest) under a ranking function $r()$. From now on, we assume that we want k smallest ranked items.

C. Connected Dominating Set

Our results are based on some connected dominating set (CDS) of the original communication graph G. We can construct a dominating set using a distributed method in [26]. For a graph G, let $\mathbf{C}(G) = (V_{\mathbf{C}}, E_{\mathbf{C}})$ be the CDS constructed using method in [26]. In the rest of the paper, we assume that such CDS has been constructed in advance. Here, our assumption is reasonable since constructing CDS is one-time work after all wireless nodes are deployed. Let $d_G(u)$ denote the degree of node u in a graph G.

Let $T_{\mathbf{C}}$ be a BFS tree of $\mathbf{C}$. For a node $v \in V \setminus V_{\mathbf{C}}$, we define an unique dominator, denoted as $d(v)$, which is the one having the shortest hop distance to the sink v_0. Then our top-k query is based on the following data communication tree (DCT).

Definition 1: For a WSN G and its CDS, the *data communication tree (DCT)* $T(G) = (V, E_T)$ is defined as $E_T = E_{\mathbf{C}} \cup \{\overline{vd(v)} \mid v \in \overline{V \setminus V_{\mathbf{C}}}\})$. Here $E_{\mathbf{C}}$ is the set of edges used in the CDS $\mathbf{C}(G)$, $\overline{vd(v)}$ denotes the edge between v and $d(v)$.

Given a data communication tree $T = (V, E_T)$, an aggregate operation consists of (possibly repeated) two phases: a *propagation* phase where the query demands are pushed down into the sensor network along the tree; and an *aggregation* phase where the aggregated values are propagated up from the children to their parents. Next, we present one theorem and one lemma proposed in [20], which will be used in our proof later.

Theorem 1: [20] Let G be a growth-bounded communication graph by a function $g()$, and $\mathbf{C}$ be the constructed CDS graph on G. The data communication tree $T(G)$ has the following properties:

1) $\Delta(T_{\mathbf{C}})$ is at most a constant $\mathbf{d}$
2) For any edge $e \in E_T$, let $I(e)$ be the set of edges in E_T that have interferences with e, then $|I(e)| \leq c \cdot \Delta(G)$ for some constant c depending on ϱ.

All our methods are based on **data clustering**: given a good CDS, for a node $v \in V \setminus V_{\mathbf{C}}$, it sends the data items to its dominator $d(v)$ in a TDMA manner.

Lemma 2: [20] Given a *good* CDS of the graph G, data clustering can be done in time $O(\tilde{\Delta}(G))$.

After data clustering, all data elements are clustered in $T_{\mathbf{C}}$. In other words, each node v_i in the connected dominating set now will have data from all nodes dominated by v_i.

III. Single Object Model

In this section, we study the top-k query for the single-object model (SO). For simplicity, we assume $|A_i| = p$ for every node v_i and further define $w = \min\{p, k\}$. We not only present efficient algorithms to minimize the latency, or minimize the number of messages communicated, but also give lower bounds on the latency and communication cost of any top-k query method under this model for multihop WSNs.

A. Time Complexity: Efficient Methods and Lower Bounds

1) Delay Efficient Methods: We first present our method (Algorithm 1) for distributed top-k query in growth-bounded multihop WSNs. The main idea is as follows. At the beginning, each dominator node collects top-k data items from all its dominatee nodes. Next, the distributed selection method (Algorithm 2) is run over the CDS to find the k-th ranked data item. This selection method has time complexity $O(D \log_D N)$ in wired communication model [14]. When taking interference into account, the time complexity is still $O(D \log_D k)$ since the wireless networks have the growth-bounded property and the interference only happens within ϱ hops. Finally, the sink node distributes the k-th smallest data item (assume a_k) to all the wireless nodes and the top-k data items can be collected by returning all the data items which are larger than a_k from each wireless node.

Algorithm 1 Distributed Top-k Query With Low Latency

Input: Growth-bounded (conflict) graph $G = (V, E)$ and ρ.
Output: top-k data items.

1: Construct a CDS with bounded degree $\mathbf{d}$ as in [2], [3]. Then construct the BFS tree $T_{\mathbf{C}}$ of the CDS $\mathbf{C}$.
2: **Data Clustering**: Each dominator collects the local top-k data items from all its dominatee nodes within time $O(w\Delta)$ when we consider the interferences. Here each node v_i has to report $\leq w$ data items, where $w = \min(p, k)$.
3: **Rank Selection**: Using BFS tree $T_{\mathbf{C}}$, we run randomized Algorithm 2, with $t = 8\lambda D$ for a constant $\lambda > 1$ to find the k-th smallest data item a_k using only nodes and links of the CDS, *i.e.*, only nodes in CDS will participate.
4: Sink node broadcasts a_k to all the wireless nodes in CDS.
5: Collect all the data items which are larger than a_k from all wireless nodes, which can be done in time $O(k + D)$.

In Algorithm 2, **getRndElementsInRange** will find t random elements within range (L, U) from data items of all wireless sensor nodes (Main idea and related proof shown in Lemma 3). Procedure **countElementsInRange**$((a, b])$ is an aggregation function that counts the total number of elements a_i in the range of $(a, b]$ (*i.e.*, $a < a_i \leq b$) among all wireless nodes. This counting function can clearly be done using $n_{\mathbf{c}}$ messages and in time $\Theta(D)$ since *counting* is a distributive function. We then show that our method will finish in certain time-slots with high probability (*w.h.p*).

Lemma 3: Procedure **getRndElementsInRange** can find t random items within range (L, U) with equal probability. The total messages used to get t random elements is at most $2tn_{\mathbf{c}}$. The time delay is at most $O(t + D)$ when we let nodes report the selected data items in sequel.

Proof: Assume that the BFS tree $T_{\mathbf{C}}$ of the CDS has been constructed in advance. The procedure **getRndElementsInRange**(L, U) will be implemented as follows. For each node

Algorithm 2 Random Data Selection $RDS(t,k)$

1: $L \leftarrow -\infty$; $U \leftarrow \infty$; phase$\leftarrow$ 0;
2: **repeat**
3: $x_0 \leftarrow L$; $x_{t+1} \leftarrow U$; phase$\leftarrow$phase+1;
4: $\{x_1, \ldots, x_t\} \leftarrow$ **getRndElementsInRange**$(t, (L,U))$
5: **for** $i = 1, \ldots, t$ in sequel **do**
6: $r_i =$**countElementsInRange**$((x_{i-1}, x_i])$
7: **if** $x_0 \neq -\infty$ **then**
8: $r_1 \leftarrow r_1 + 1$
9: $j \leftarrow \min_{l \in \{1,\ldots,t+1\}} \sum_{i=1}^{l} r_i > k$
10: $k \leftarrow k - \sum_{i=1}^{j-1} r_i$
11: **if** $k \neq 0$ and $j \neq 1$ **then**
12: $k \leftarrow k + 1$
13: **until** $r_j \leq t$ or $k = 0$
14: **if** $k = 0$ **then**
15: return x_j
16: **else**
17: $\{x_1, \ldots, x_s\} =$**getElementsInRange**$([x_{j-1}, x_j])$;
18: return x_k

v in the CDS, let $c(v)$ be the total number of data items stored in v (including its own data items and data collected from its dominatee nodes) within the range (L,U). First for each node v, we get a counting on the total number (denoted as $n(v)$) of data items in the range (L,U) from the subtree rooted at v. This clearly can be done by simple aggregation function *sum* with $n_{\mathbf{c}}$ number of messages and $\Theta(D)$ delay, considering the wireless interferences. Then the root node v_0 knows the total number of data items $n(v_0)$ in the range (L,U). The root randomly chooses t random integers, say $p_1 < p_2 < \cdots < p_t$, in range $[1, n(v_0)]$. Let $p_0 = 0$. These integers p_i, $1 \leq i \leq t$ denote the sorted order of the random data items in (L,U) to be picked from all nodes. We then implement another procedure **fetchRndElementsInRange**$(v_0, [p_1, p_2, \cdots, p_t])$ as follows:

1) Let $s = 1$. While $p_s \leq c(v_0)$, let $s = s+1$. We randomly pick $s-1$ data items in range (L,U) from node v_0.
2) Let u_1, u_2, $\cdots$, u_d be the d children nodes of v_0 in the BFS tree $T_{\mathbf{C}}$. For each child u_i, find $l(i)$ and $r(i)$ such that $c(v_0) + \sum_{j=1}^{i-1} n(u_j) < p_{l(i)} < p_{l(i)+1} < \cdots < p_{r(i)} \leq c(v_0) + \sum_{j=1}^{i} n(u_j)$. Let $Q = c(v_0) + \sum_{j=1}^{i-1} n(u_j)$, $q_1 = p_{l(i)} - Q$, $q_2 = p_{l(i)+1} - Q$, $\cdots$, $q_{r(i)-l(i)+1} = p_{r(i)} - Q$. We then call **fetchRndElementsInRange**$(u_i, [q_1, q_2, \cdots, q_{r(i)-l(i)+1}])$ to get $r(i) - l(i) + 1$ random data items from the subtree rooted at u_i.

■

Lemma 4: For any growth-bounded wireless networks, the k-th smallest data item can be found with delay at most $O(D + D\log_D k)$ in Line 3 of Algorithm 1 *w.h.p.*

Proof: Let $n_{\mathbf{c}}$ denote the size of CDS constructed in Algorithm 1. Then the total number of data items contained in all dominators is $n_{\mathbf{c}}k$ before Line 3 runs. This is because when data are collected to the dominators (after Line 2 is done), every dominator maintains the set of local top-k data items including its dominatees and its own. As proved in [14], finding the k-th smallest data item will cost $O(D\log_D(n_{\mathbf{c}}k))$ rounds[1] of communications *w.h.p*, *i.e.*, $O(D\log_D(n_{\mathbf{c}}k))$ time-slots *w.h.p*, since (1) each dominator node contains at most k data items, and (2) each round is composed of $\beta \leq g(2)$ timeslots because the maximum degree in CDS is $g(2)$ for growth-bounded G.

For multihop WSNs modeled by a growth-bounded network G using a function $g()$, the size of the minimum dominating set (MDS) is $\leq g(D)$. It is easy to show that the size of the connected dominating set found by our method is at most $2g(1)$ times of the MDS. Thus, $n_{\mathbf{c}} \leq 2g(1) \cdot g(D)$. Therefore, $O(D\log_D(n_{\mathbf{c}}k)) \leq O(D\log_D(k \times 2g(1)g(D))) = O(D + D\log_D k)$.

because $g()$ is a polynomial. This finishes the proof. ■

Observe that $O(D + D\log_D k) = O(\lceil D\log_D k \rceil)$ when $k = \Omega(D)$. For simplicity, we will use $D\log_D k$ to denote $\lceil D\log_D k \rceil$. Then $O(D + D\log_D k) = O(D\log_D k)$.

Theorem 5: Algorithm 1 can find the top-k data items in expected time $O(w\Delta + D\log_D k + k)$.

Proof: This lemma follows from the time complexity analysis for each phase as follows: (1) the data clustering step has time complexity $\Theta(w\Delta)$; (2) the rank selection step will cost $O(D + D\log_D k)$ time-slots *w.h.p* based on Lemma 4; (3) the broadcast step costs $\Theta(D)$ time slots to broadcast a_k; (4) the last step costs $O(D + k)$ time slots to collect the real top-k data items to sink node since we can collect these k items in parallel without interference, *e.g.* the k items can be transmitted one by one. This finishes the proof. ■

Note that, we can derandomize our algorithm as in [14] to find a deterministic top-k query algorithm. The proof of the following theorem is a simple combination of a proof similar to Theorem 5 and the proof of Theorem 4.4 of [14].

Theorem 6: There is a deterministic algorithm that can find the top-k data items in time $O(w\Delta + D(\log_D k)^2 + k)$.

2) Lower Bound on Query Latency: In this subsection, we give a lower bound on the time complexity for any top-k query method in multihop WSNs.

Lemma 7: For any Δ, there exists a wireless network with maximum degree Δ such that *any* deterministic algorithm needs at least Δ time slots to compute the top-k set.

Proof: Given any $\Delta > 1$, we construct a simple network with $\Delta + 1$ nodes such that all other Δ nodes are located in the transmission range of the sink. We then prove that for any deterministic algorithm, there always exists a data placement such that every node needs to be active (*i.e.*, send a message to sink) at least once during the computation for the top-k set. Since we study deterministic distributed algorithms here, all sensor nodes will have a uniform method in determining their actions (whether to send a message, if send then send what message) based on the messages they received historically

[1]In wired networks, a round is the time-duration such that, given any node u, each of the neighboring nodes of the node u can send one message to u. Clearly, in wireless networks, such communications cannot be finished in one time-slot. For arbitrary wireless sensor networks, such communications need at least $\Delta(G)$ time-slots. Since we run the **Rank Selection** on top of the CDS structure, such communications can be finished in at most $g(2)$ time-slots when networks G is growth-bounded by a function $g()$. Recall that $\Delta(\mathbf{C}(G)) \leq g(2)$ in this case.

and its own data. Without loss of generality, assume we have a binary function $f(x_i, m_t, M_{t-1})$ for any deterministic algorithm. Here, x_i denotes the data located at the ith node, m_t denotes the message received from the sink node at time t, M_{t-1} denotes all historical messages received from the sink node upto time $t-1$. Each node decides whether to be active or not according to the value of $f(x_i, m_t, M_{t-1})$, *e.g.*, the node will send message to sink if $f(x_i, m_t, M_{t-1}) = 1$, or keep silent otherwise.

In the initial phase, without receiving any information from the other nodes, the sink will send some fixed message m_1 to all the other nodes. There are two possible cases we need to consider:

1) there are infinite number of values a such that $f(a, m_1, \emptyset) = 1$, we may set x_i such that $f(x_i, m_1, \emptyset) = 1$ for every node v_i. In this case, all n nodes will send message to the sink node.
2) there are infinite number of values a such that $f(a, m_1, \emptyset) = 0$. In this case, we set x_i such that $f(x_i, m_1, \emptyset) = 0$ for every node v_i. The algorithm gets no information and must send another message $m_2 \neq m_1$ under some fixed rules (otherwise, the sink will never get any information).

Thus, as long as for some message m_t, there exists enough a such that $f(a, m_t, M_{t-1}) = 1$, we set $x_i = a$ for every node. Therefore, every node needs to be active at least once under our data placement. Obviously, we need at least Δ time slots to finish all the data collection due to the interference. This finishes the proof. ■

Theorem 8: There are multihop WSNs and data distributions such that any distributed method (deterministic or randomized) needs at least $\Omega(D + D\log_D k + k)$ time slots to find the top-k data items.

Proof: The time complexity of finding the top-k data items for a wireless network is at least as expensive as that of finding the k-th smallest data item in a corresponding wired network (by assuming that no interferences exist among all transmission links). For wired networks, it was proved in [14] that any k-th selection method needs $\Omega(D + D\log_D k)$ time-slots. Thus, any top-k query algorithm for WSNs needs at least $\Omega(D + D\log_D k)$ time-slots. In addition, no matter what method is implemented, we need to spend at least $\Omega(k)$ time slots in collecting the real top-k data items with size k to the sink. This finishes the proof. ■

Based on Lemma 7, we have

Theorem 9: There are multihop WSNs and data distributions such that any deterministic distributed method needs at least $\Omega(\Delta + D\log_D k + k)$ timeslots to find the top-k data items.

Thus, when each node has only $p = O(1)$ data items, the time-slots needed by our method matches the lower bound of any deterministic algorithm. It remains an interesting question to close the gap between the lower bounds and upper bounds for deterministic algorithms when $p = O(k)$, and to further find a possibly better lower bound for randomized algorithms by proving a better lower bound needed by any randomized algorithm in data collection phase.

B. Message Complexity: Efficient Methods and Lower Bounds

1) Message Efficient Methods: In this section, we propose two algorithms in order to minimize the number of messages incurred for top-k query. Here we include both the data messages and the control messages. The first algorithm is called Breadth First Searching Tree-Based Algorithm (BFS-BA) which is shown in Algorithm 4. The second one (Algorithm 3) is Data Selection Based Algorithm (DS-BA). Based on further analysis on their message costs, we choose the better one depending on the value of k.

Algorithm 3 DS-BA

Input: A CDS with bounded node degree **d**. A control parameter $d \geq 2$.
Output: top-k data items.

1: **Data Cluster**: As in Algorithm 4, each dominator collects the local top-k data items among its dominatees.
2: **Rank Selection**: Then the k-th smallest data item a_k is found using only the CDS nodes by running randomized Algorithm 2, with $t = 8d\lambda$ for a constant $\lambda > 1$.
3: Sink node broadcasts a_k to all nodes in CDS.
4: Collect the top-k data items by letting each node in CDS report its data items that are larger than a_k.

The basic idea of BFS-BA is as follows. We first construct a BFS tree of the CDS **C** rooted at sink node v_0. Each dominator node $v_i \in \mathbf{C}$ will collect the local top-k data items from its all dominatees using one of the two following methods: (1) dominatee nodes directly report their data to v_i; or (2) v_i uses the binary search to find the k-th ranked data item and all dominatee nodes report the data items smaller than this one. Then for each node v_i, if it has received local top-k items from all its children nodes in T, it sends the aggregated top-k items to the sink (choosing the top-k items among its own data and received data). Finally, returning the top-k items computed by sink node as the desired top-k set.

Algorithm 4 BFS-BA

Input:BFS tree $T_{\mathbf{C}}$ of CDS **C** rooted at sink node v_0. Each node with $\leq p$ items. Network G with maximum degree Δ.
Output: top-k data items.

1: **Data Cluster**: Each dominator node v_i in CDS **C** finds the top-k data items among data items stored in its $d(v_i)$ dominatee nodes. If $pd(v_i) \leq k$, each dominatee node simply reports its local top-k items to v_i. Otherwise, v_i applies Algorithm 2 to find the ranked k items among data items stored in its $d(v_i)$ dominatee nodes. Then local top-k items are reported to v_i.
2: **for** each node v_i in **C do**
3: If node v_i has received local top-k items from each of its children nodes in T, it sends the aggregated top-k items, by choosing the top-k items among its own data items and received local top-k items from each of its children nodes, to its parent node.
4: Return the top-k items computed by sink node v_0.

Theorem 10: Given a wireless network G with n nodes (each with $\leq p$ data items) and diameter D, Algorithm 4 costs at most $(n + 3n_{\mathbf{c}}k)$ messages to find the top-k items.

Proof: For a dominator node v_i, to find the local top-k data items of its dominatee nodes, it will cost $wd(v_i)$ messages using direct reporting for $w = \min(p, k)$, or will cost $\log(wd(v_i)) + k < 2k + \log\Delta$ messages using binary search since $w \leq k$ and $d(v_i) \leq \Delta$. Thus, in total, step 1 costs at most $\sum_{i=1}^{n_{\mathbf{c}}} \min(2k+\log d(v_i), pd(v_i)) \leq 2kn_{\mathbf{c}}+n$ messages. Obviously, each node on $\mathbf{C}$ then needs to transmit at most k data items. Then the total number of messages transmitted at step 2 is at most $n_{\mathbf{c}}k$ by all nodes. Then the theorem follows. ■

We then present DS-BA, a selection based top-k query algorithm. Our method will first find the k^{th} smallest data item, saying a_k, using selection and then let each node in CDS report its data items smaller than a_k.

Lemma 11: Given a wireless network G with n nodes (each with p data items) and diameter D, Algorithm 3 (by setting $t = 8d\lambda$) can find the top-k data items with at most $\min(n + 2kn_{\mathbf{c}}, wn) + 3n_{\mathbf{c}}d\log_d(n_{\mathbf{c}}k) + kD$ messages *w.h.p*, where $n_{\mathbf{c}}$ is the number of nodes in CDS.

Proof: First, in data cluster phase of Algorithm 3 (Line 1), it costs $\min(n+2kn_{\mathbf{c}}, wn)$ messages to find the local top-k data items for all the dominators. Then we will prove in Lemma 12 that the rank selection (Line 2 in Algorithm 3) takes at most $3\log_d(n_{\mathbf{c}}k)$ phases (variable *phase* is defined in Algorithm 2) *w.h.p*, when the k-th smallest number is found. Since the number of links in the tree spanning the nodes of CDS is $n_{\mathbf{c}} - 1$, and in each round every node needs to send at most $t = 8d\lambda$ messages, we get an upper bound on the number of messages sent in rank selection phase: $3n_{\mathbf{c}}d\log_d(n_{\mathbf{c}}k)$, *w.h.p*. Furthermore, we need at most $O(kD)$ messages to collect the real top-k data items from all nodes. Thus, the number of messages (both data messages and control messages) used by Algorithm 3 is, by setting $t = 8\lambda d$, at most $\min(n + 2kn_{\mathbf{c}}, wn) + 3n_{\mathbf{c}}d\log_d(kn_{\mathbf{c}}) + kD$, *w.h.p*. ■

Lemma 12: Variable *phase* takes at most $3\log_d N$ rounds *w.h.p*, when the k-th smallest data item is found by using Algorithm 2 with $t = 8d\lambda$.

Proof: First, we compute an upper bound on the probability that after any phase i, the wanted element is in a fraction of size at least $c\frac{\log d}{d}$ times the size of the fraction after phase $i-1$ for a suitable constant c, *i.e.*, $n^{(i)} \geq c\frac{\log d}{d} \cdot n^{(i-1)}$. Here $n^{(i)}$ is the size of the all data items we have to check to find the k^{th} smallest data before the phase i starts. Notice $n^{(0)} = N$. Let $\{a_1, a_2, \cdots, a_{n^{(i)}}\}$ be the sorted list of the $n^{(i)}$ data items that we will check for the k^{th} smallest element in phase $i+1$. The probability that none of the $t = d\lambda$ randomly selected elements is in $\{a_k, a_{k+1}, \cdots, a_{k+c\frac{\log d}{d}n^{(i)}/2}\}$ is at most $(1-\frac{c\log d}{2d})^{8\lambda d}$. Same argument holds for data items $\{a_{k-c\frac{\log d}{d}n^{(i)}/2}, \cdots, a_{k-1}, a_k\}$. Thus,

$$\mathbf{Pr}\left(n^{(i)} \geq c\cdot\frac{\log d}{d}\cdot n^{(i-1)}\right) \leq 2(1-\frac{4\lambda c\log d}{8\lambda d})^{8\lambda d} \leq 2e^{-4c\lambda\log d}.$$

When $n^{(i)} \leq c\frac{\log d}{d}\cdot n^{(i-1)}$ the phase i is considered to be *successful*; otherwise it is considered to be *failed*. Clearly, we need at most $\log_{c\frac{d}{\log d}} N$ successful phases to find the k-th smallest element. By setting $c = \frac{16}{17}$, less than $S = 2\log_d N$ successful phases are required to find the k^{th} smallest number based on the following inequality:

$$\log_{c\frac{d}{\log d}} N = 2\log_{(c\frac{d}{\log d})^2} N < 2\log_d N$$

A phase i will fail with probability at most $p = 2e^{-4c\lambda\log d}$. Then among $\frac{3}{2}S$ phases, the probability that we have less than S successful phases (*i.e.*, at least $\frac{1}{2}S$ failed phases) is at most

$$\sum_{i=\frac{1}{2}S}^{\frac{3}{2}S}\binom{\frac{3}{2}S}{i}p^i(1-p)^{\frac{3}{2}S-i} \leq \binom{\frac{3}{2}S}{\frac{1}{2}S}p^{\frac{1}{2}S} \leq (\frac{1}{d^\lambda})^{\frac{1}{2}S} = \frac{1}{n^\lambda}$$

This holds because $4\lambda c\frac{\ln d}{\ln 2} - \ln 6e > \lambda\ln d$ for $c = \frac{16}{17}$ and $d \geq 2$, Hence, *w.h.p*, Algorithm 2 terminates after less than $3\log_d N$ phases when $t = d\lambda$. Each phase will cost at most $2\times 8\lambda d n_{\mathbf{c}}$ messages. This finishes the proof. ■

To minimize the number of messages, we may set t as some constant, *e.g.* $t = 16\lambda$, in our DS-BA. Thus, we have

Theorem 13: When $n_{\mathbf{c}}k \geq (w-1)n/2$, the message cost of Algorithm 3 is at most $wn + 3n_{\mathbf{c}}\log(kn_{\mathbf{c}}) + kD$.

Assume that each node has p data items initially. Recall that $w = \min(p, k)$. By combining the results in Theorem 10 and Theorem 13, our optimal decision between the above two algorithms can be made as follows:

- When $wn > n + 2n_{\mathbf{c}}k$, we use BFS-BA.
- When $wn \leq n + 2n_{\mathbf{c}}k$,
  - if $\log n_{\mathbf{c}} > k$, choose BFS-BA; Otherwise, DS-BA

2) Lower Bound on Message Complexity: Instead of studying the lower bound of message complexity on top-k query problem directly, we first focus on the lower bound on data selection problem. Our lower bound of message complexity is based on the result on two-party model. For two nodes connected by a link, each with $N/2$ data, finding the median need $\Theta(\log N)$ messages; or generally, the k^{th} smallest element ($k < N/2$) can be found using $\Theta(\log k)$ messages.

Results by [14] imply that there is a wireless network with n nodes and data distribution in these nodes such that, for *any* top-k query method, $\Omega(n\log k)$ messages are needed to compute the top-k items. Here we give a better lower bound on message complexity for top-k query in any general wireless network.

Theorem 14: For any wireless network G with n nodes and diameter D, for any top-k query method, when $k = O(D)$, there exists a data placement such that at least $\Omega(Dk)$ messages are needed to compute the top-k data items.

Proof: Based on the definition of diameter D, we can always find a pair of nodes, u and v, such that the hop distance between them is at least D. If we randomly place half of the top-k data items on u and the other half of the top-k items on v, at least $\Omega(Dk)$ messages are needed to just collect the top-k items, regardless of where the sink node is. When each node can store upto p data items and $k < D$, we can place

p data items on every node on the shortest path from u to v. We randomly place top-k data items on these nodes. Simple calculation shows that the theorem still holds. ■

Obviously, for any deterministic method, each wireless node needs to send at least one message. In summary, we get the following lower bound on the total number of messages needed by *any* deterministic top-k query method.

Theorem 15: For any wireless network G with n nodes and diameter D, for any top-k query method, there exists a data placement such that any *deterministic* top-k query method needs at least $\Omega(n + Dk)$ messages.

In the previous analysis of lower-bound, we essentially present lower-bounds using the diameter D of the network. We make the following conjecture on the message complexity of any top-k query algorithm, for any general network.

Conjecture 16: For any wireless network G with a minimum dominating set of size $n_{\mathbf{c}}$, for any top-k query method (deterministic or randomized), there exists a data placement such that at least $\Omega(n_{\mathbf{c}} \log k)$ messages are needed to compute the top-k data items.

IV. Multiple Objects Model

In this section, we propose two methods to compute the top-k objects under multiple objects model. We further give upper bounds of the time complexity and message complexity for both methods respectively. Remember that under multiple objects model, there are totally m objects and each wireless sensor node hold some value of each object, and the rank of each object is determined by aggregating all values from all nodes.

A. Time and Message Efficient Methods

1) Data Aggregation Based Algorithm: The main idea of our first method, named as Data Aggregation Based Algorithm(DA-BA), is as follows. We first construct a CDS, and every dominator v_i aggregates the local score of each object L_i from all its dominatees including $v_i's$ as well. In the second phase, the sink node aggregates all the data contained in dominators along BFS tree of CDS. Finally, the sink node locally computes the top-k set based on these aggregated data. Refer to Algorithm 5 for details. We will show that the time complexity of DA-BA is of order $\Theta(D + m\Delta)$.

Theorem 17: DA-BA (Algorithm 5) performs top-k query in $O(D + m\Delta)$ time slots with $O(mn)$ messages.

Proof: We first show that Algorithm 5 costs $O(D+m\Delta)$ time slots. For simplicity of analysis, we first study a basic case when $m = 1$, *e.g.* there is only one object. The first step in which each dominatee node sends its data to its dominator node will take at most $O(\Delta)$ time slots. Then we perform aggregation in rounds, where each round is composed of β time slots (where constant β is the number of colors needed to color the interference graph induced by all CDS nodes). In round 1, all nodes in level D (all leaves) send a message to their parents. In round t, all nodes in level $D - t + 1$ should have received all the messages from their children, they then compute the aggregation of all data received so far, and then send the aggregated values to their parents. In all, the total number of rounds to finish data aggregation is D. Recall that each round is composed of β time-slots. We get the total time latency $\leq \beta D = O(D)$.

Algorithm 5 Efficient Data Aggregation Based Algorithm

Input: A CDS **C**, a distributive function f.
Output: top-k objects.

1: **for** each dominator node v_i **do**
2: v_i collects all data items of each object from all its dominatees.
3: Constructing a BFS tree rooted at the sink node v_0 among all nodes in CDS.
4: **for** $t = 1$ to D **do**
5: **for** each node $v_i \in V_{\mathbf{C}}$ **do**
6: For each object L_j, if node v_i has received aggregated data from all its children nodes in $T_{\mathbf{C}}$, it sends the aggregated value to its parent node.
7: After received all information from all dominator nodes, the sink node computes the top-k objects by choosing the k objects with the smallest aggregated observed value.

The more general case is $m > 1$, under which we can deliver the messages one by one. We call this as sequential aggregation. Considering the first phase, the dominators need at most $\beta m\Delta$ time slots to aggregate all m object's local score due to the interference; In the second phase, we do the scheduling using sequential aggregation, then for each object, we can do it in parallel way without interference, the time latency is increased by at most βm. So, the total time latency is $\beta(D + m\Delta + m) = O(D + m\Delta)$.

The message complexity then follows from the fact that the length of the tree connecting n nodes is $n-1$ and we have m objects. Therefore, we need to send at most $m \times n$ messages. This finishes the proof. ■

2) Threshold Algorithm Based Algorithm: Top-k query (with multiple objects) in distributed databases is well studied before, a number of efficient algorithms have been proposed. Among these, the Threshold Algorithm (TA) [7] is the most well-known due to its simplicity and memory requirements. TA is based on an early-termination condition and can evaluate top-k queries without examining all the tuples. In particular, TA terminates when a certain threshold is achieved.

Here we propose a TA based algorithm (TA-BA) to compute the top-k set efficiently. The basic idea of our method is as follows. First, imagining that the root node v_0 has asked each of its children nodes to find the top-r elements and its aggregated values for these top-r elements using data items contained by nodes in the subtree rooted at that child. The root node v_0 can find the *partial* aggregated value using these partial results for each object $L_i \in \mathcal{L}$. Since these top-k elements based on partial knowledge may be not the correct final top-k elements, the root node computes a threshold τ that is the aggregated value of all the r^{th} objects from each of the children. If τ is already larger than the kth smallest value in those partial aggregated values, we find the corrected top-k elements. Otherwise, we increase the search range by increasing r to $r + 1$ and continue the same procedure until

Algorithm 6 TA-BA(T, v_0, k)

Input: BFS tree T rooted at a root v_0.

Output: Return top-k ranked objects.

Let $v_{0,j}$, $1 \leq j \leq q$ be the q children nodes of node v_0 in the tree T. For the original data items $[s_1(v_0), s_2(v_0), \cdots, s_m(v_0)]$ of node v_0, let $b_1(v_0) \leq b_2(v_0) \leq \cdots \leq b_m(v_0)$ be the sorted list in increasing order.
$\mathcal{L}_R = \emptyset$. DONE = FALSE; $r \leftarrow 1$.
repeat
  for each children node $v_{0,j}$ of the root v_0 in T **do**
    Let $v_{0,j}$ compute the top-r element by calling TA-BA$(T_{v_{0,j}}, v_{0,j}, r)$. Here $T_{v_{0,j}}$ is the subtree of T rooted at $v_{0,j}$. Let $L(r, j)$ be the set of r objects returned and let $a_1(v_{0,j})$, $a_2(v_{0,j})$, $\cdots$, $a_r(v_{0,j})$ be the sorted list of r aggregated values of all nodes in the subtree $T_{v_{0,j}}$ for these r objects $L(r, j)$.
    $\mathcal{L}_R \leftarrow \mathcal{L}_R \cup L(r, j)$.
  for each object $L_t \in \mathcal{L}_R$ **do**
    Let $f_t(v_{0,j})$ be the aggregated value computed by node $v_{0,j}$ for this object L_t using data from all nodes in the subtree $T_{v_{0,j}}$.
    Compute the aggregated value of this object L_t, $f_t(v_0) \leftarrow f(s_t(v_0), f_t(v_{0,1}), f_t(v_{0,2}), \cdots, f_t(v_{0,q}))$;
  Sort the aggregated values of all objects in $\mathcal{L}_R$. Let $a_1(v_0)$, $a_2(v_0)$, $\cdots$, $a_m(v_0)$ be the sorted list. Find the top-k objects that has the k smallest values.
  if $\mathcal{L}_R = \mathcal{L}$ **then**
    Then node v_0 can simply return the top-k objects correctly. DONE=TRUE;
  else
    $\tau_r \leftarrow f(b_r(v_0), a_r(v_{0,1}), a_r(v_{0,2}), \cdots, a_r(v_{0,q}))$. Here τ_r is the threshold value for round r. If $(\tau_r \geq a_k(v_0))$ then DONE=TRUE, otherwise $r \leftarrow r + 1$.
until DONE
Return top-k objects.

finding the correct top-k objects. Refer to Algorithm 6 for details.

Theorem 18: TA-BA (Algorithm 6) will answer the top-k query in $O(D + m\Delta)$ time slots with $O(mn)$ messages.

B. Lower Bounds on Message Complexity and Latency

Next, we show that our methods for top-k query in multiple-objects model is asymptotically optimum.

Theorem 19: For any network G of diameter D, and maximum degree Δ, for any deterministic distributed top-k query method under multiple-objects model, there is a data placement among nodes such that the method needs $O(D + m\Delta)$ time slots, and $O(mn)$ messages.

Proof: First of all, the lower bound D is obvious since the information about the data on the furthest node u from root v_0 should be known by the root node v_0. To propagate any information from node u to v_0, it takes time at least D. Notice that, here the method may not need to know the actual data items hold by the node u. The lower bound $O(m\Delta)$ on time-slots, and the lower bound $O(mn)$ on the number of messages come from the following statement: for any node u, node u must send its data about at least $m - k$ objects. This will be proved using a special star network, with root v_0 and Δ neighbors v_1, v_2, $\cdots$, v_Δ. If there are two nodes v_i and v_j, both send only at most $m - k$ objects based on a deterministic rule of selecting which objects to report. Then we can carefully define values of the rest of k objects so that the computed top-k depending on the values of these k objects. This finishes the proof. ■

V. Performance Evaluation

To illustrate the feasibility and performance of our distributed top-k query algorithms, we implemented and tested our algorithms on testbed consisting of 36 Telosb nodes. We performed experiments of our algorithms under both single object model and multiple objects model to measure message complexity and time complexity.

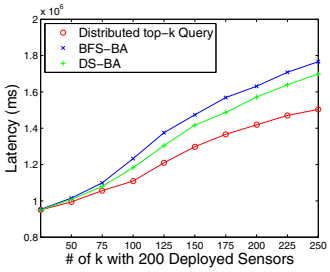
A. Experiment Design

1) System Design: Our testbed consists of 36 Telosb sensor nodes, each node has a unique ID between 0 and 35. In addition, the node with ID 0 is the sink node and connected to the laptop by USB port. We use Java 6 to implement the related GUI on the laptop, which can exchange information with the sink node, monitor the network and perform statistics of the experimental results. We deployed 36 Telosb sensor nodes with 6×6 mesh topology. The distance between two vertical (horizontal) wireless nodes is 60cm. Here, we set the transmission power of each Telosb node to level 2 such that the valid transmission range of each wireless node is around 70cm.

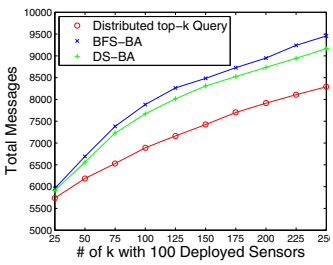
For all top-k algorithms we have implemented, there are two main layers: routing layer and the application layer. In routing layer, all nodes continuously send beacon messages to neighbors to maintain necessary routing information. In the application layer, all nodes will cooperate to collect (disseminate) necessary data messages (used to find final top-k results) to/from the sink node. To test our algorithms efficiently, we let each algorithm run 20 rounds, and study the average performance. In our experiment, we choose $\max$ as the function of interest f for both single object and multiple object model. The main metrics used in our experiment to evaluate different algorithms are the overall delay (latency) and total messages (including all the retransmissions) exchanged among all nodes. Notice that the real wireless links are not stable due to background noise and wireless communication interference, to compare all implemented top-k algorithms fairly, we use explicit ACK messages to guarantee the "stable" property of a wireless link (with cost of retransmissions).

B. Experimental Results

Under single-project model, we compare the performance of our Distributed top-k Query algorithm (Algorithm 1), BFS-BA algorithm (Algorithm 4), and DS-BA algorithm (Algorithm 3). For each sensor node, after booted, it continues to sense the

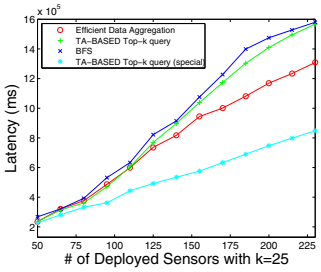


(a) Latency

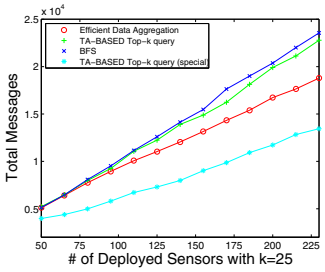


(b) number of messages

Fig. 1. Simulation results for single object model. k continues to increase from 25 to 250 with step 25.



(a) Latency



(b) number of messages

Fig. 2. Simulation results for multiple objects model where $k = 25$ and $p = 50$, n continues to increase.

environmental illuminance (photo value) per second. Every node maintains a window with size p ($p = 20$ in our experiment) which records the photo value of last p seconds. Therefore, for each round, the system totally has 36×20 readings, and the objective of the sink node is to find the top-k (we let k vary from 25 to 250 with step 25 in our experiment) largest readings.

As we can see from Fig. 3(a) and (b), the performances of all three algorithms are similar. This is mainly because of the limited network size and regular network topology. Notice that both the message complexity and latency increase almost linearly with k for all three algorithms, this validates our theoretical analysis. More importantly, the above experimental results validate the feasibility of all three algorithms in sensors with limited energy and memory.

For multiple objects model, we implement both Efficient Data Aggregation Based Algorithm (Algorithm 5) and Threshold Based Top-k Query Algorithm (Algorithm 6). We also implement a simple BFS-based algorithm by which all wireless nodes construct a BFS tree rooted at the sink node. In our experiments, given a set of p time-slots, we want to find the top-k time-slots with the highest average observed photo values by all nodes. Here, the objects the set of time-slots and the value of an object is the average of photo values collected by all nodes. For each node, when it starts the top-k algorithm, it will consider its last p ($p = 10$ in our test cases.) monitored photo values as the values for 10 objects respectively. In our experiment, we let k vary from 1 to 6 with step 1 and test the latency and total messages used for each algorithm. The latency and total used messages are shown in Fig. 4(a) and Fig. 4(b) respectively.

For each algorithm, the total latency and used messages increase slowly with the increment of k. This is because for multiple object model, regardless of the value of k, the sink node has to collect all values of all objects from all nodes in order to find the final top-k objects, except for the TA-based methods.

C. Simulation Results

In order to verify the performance of our algorithms when applied to large-scale WSN, we also implement and test our top-k algorithms on TOSSIM (TinyOS 2.0.2) [31] on Mac OS X 10.5.6. We randomly deployed n wireless nodes with transmission range 100 meters on a 500×500 $meter^2$ square region.

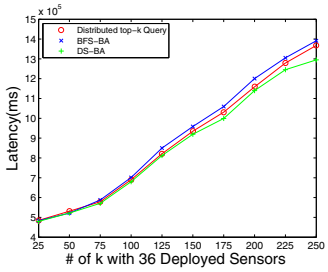
Fig. 1 and 2 demonstrate the simulation results for both single object model and multiple objects model, under which we either fix the number of sensors or k in order to verify the performance. Clearly, our distributed top-k method has smaller latency and message complexity. Again, it is worth noting that both the latency and amount of required messages appear to increase linearly with k, which verified our theoretical analysis in large scale wireless sensor network.

VI. Related Work

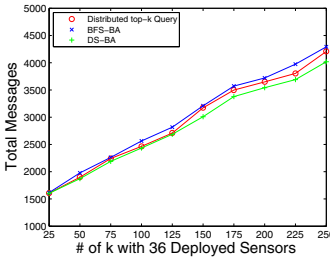
To the best of our knowledge, there is no previous work addressing the complexity problem on distributed top-k query in multihop WSNs. For top-k query in traditional distributed databases, one of the milestone result is [7]. They proposed a simple, yet efficient algorithm, called Fagin's algorithm (FA), that works on sorted lists stored in different databases. However, they did not strike to optimize the communication costs and the latency. The most efficient algorithm over sorted lists is the TA algorithm, proposed by several groups [8]. Very recently, Akbarinia *et al.* [1] proposed another threshold based method, called Best Position Algorithm (BPA) for top-k query.

Several TA-style algorithms, *i.e.*, extensions of TA, have been proposed for processing top-k queries in distributed environments, *e.g.*, [5], [16]. In [14], Kuhn *et al.* studied the problem of distributed selection of median for general *wired* networks with N data items distributed in a network of n nodes and diameter D. Babcock *et al.* [4] study a useful class of queries that continuously report the k largest values obtained from distributed data streams ("top-k monitoring queries"). Mouratidis *et al.* [18] studies continuous monitoring of top-k queries over a fixed-size window W of the most recent data.

To the best of our knowledge, the only results in literature that deal with related topics in wireless sensor networks are [9], [13], [19], [21]–[23], [25], [27]–[30]. Most of the results did not directly address the top-k query problems studied in this paper, except [22], [28], [30]. In [22], Silberstein *et al.* proposed a method to formulate the problem of optimizing approximate top-k queries under an energy constraint as a linear program by using the historical data to predict the future data. In [28], [30], the authors address the top-k queries in

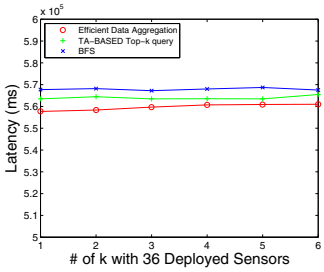


(a) Latency

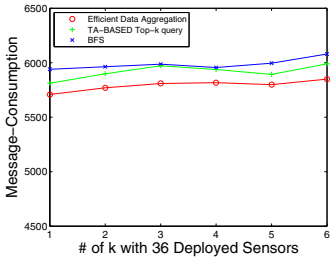


(b) number of messages

Fig. 3. Testbed results using 36 nodes for single object model. k continues to increase from 25 to 250 with step 25.



(a) Latency



(b) number of messages

Fig. 4. Testbed results using 36 nodes for multiple object model.

streaming data and they exploit the semantics of top-k query and propose a novel energy-efficient monitoring approach, called FILA. No complexity bounds on the required time-slots and messages by efficient top-k methods are known in the literature.

VII. Conclusion

A number of interesting and challenging questions are left for future study. The first question is to close the gap between the lower bounds and upper bounds achieved by our methods on various models. The second question is to relax the requirement of exact top-k queries to approximate top-k queries. The third question is to design efficient methods for top-k query in data streams where each sensor node v_i produces a data item every d_i time-slots. We also would like to know the time and message complexity of performing top-k query in this model.

VIII. Acknowledgement

The authors would like to thank Laura Marie Feeney for her constructive feedback. The research of authors is partially supported by NSF CNS-0832120, NSF CNS-1035894, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program).

References

[1] Akbarinia, R., Pacitti, E., and Valduriez, P. Best position algorithms for top-k queries. In *VLDB* (2007)

[2] Alzoubi, K., Li, X.-Y., Wang, Y., Wan, P.-J., and Frieder, O. Geometric spanners for wireless ad hoc networks. *IEEE TPDS 2003*

[3] Alzoubi, K. M., Wan, P.-J., and Frieder, O. Message-optimal connected dominating sets in mobile ad hoc networks. In *ACM Mobihoc* (2002), pp. 157–164.

[4] Babcock, B., and Olston, C. Distributed top-k monitoring. In *ACM SIGMOD* (2003).

[5] Chang, K. C.-C., and won Hwang, S. Minimal probing: supporting expensive predicates for top-k queries. In *ACM SIGMOD* (2002),

[6] Das, G., Gunopulos, D., Koudas, N., and Sarkas, N. Ad-hoc top-k query answering for data streams. In *VLDB* (2007).

[7] Fagin, R. Combining fuzzy information from multiple systems (extended abstract). In *ACM PODS* (1996)

[8] Fagin, R., Lotem, A., and Naor, M. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci. 66*, 4 (2003)

[9] Gandham, S., Zhang, Y., and Huang, Q. Distributed minimal time convergecast scheduling in wireless sensor networks. In *IEEE ICDCS* (2006), pp. 50.

[10] Kashyap, S., Deb, S., Naidu, K. V. M., Rastogi, R., and Srinivasan, A. Efficient gossip-based aggregate computation. In *ACM PODS* (2006), pp. 308–317.

[11] Kempe, D., Dobra, A., and Gehrke, J. Gossip-based computation of aggregate information. In *IEEE FOCS* (2003), p. 482.

[12] Kesselman, A., and Kowalski, D. R. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. *JPDC 66*, 4 (2006)

[13] Kollios, G., Byers, J., Considine, J., Hadjieleftheriou, M., and Li, F. Robust Aggregation in Sensor Networks. *IEEE Data Engineering Bulletin* (2005).

[14] Kuhn, F., Locher, T., and Wattenhofer, R. Tight bounds for distributed selection. In *ACM SPAA* (2007)

[15] Mamoulis, N., Yiu, M., Cheng, K., and Cheung, D. Efficient top-k aggregation of ranked inputs. In *ACM TODS 2007*.

[16] Marian, A. Evaluating top-k queries over web-accessible databases. In *IEEE ICDE* (2002), p. 369.

[17] Mo, Lufeng, He, Y., Liu, Y.-H., Zhao, J.-Z., Tang, S.-J., Li, X.-Y., and Dai, G.-J. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest, ACM SenSys, 2009.

[18] Mouratidis, K., Bakiras, S., and Papadias, D. Continuous monitoring of top-k queries over sliding windows. In *ACM SIGMOD* (2006)

[19] Patt-Shamir, B. A note on efficient aggregate queries in sensor networks. In *ACM PODC* (2004), pp. 283–289

[20] Ren, C., Mao, X.-F., Xu, P., Dai, G.-J., and Li, Z.-H., Delay and Energy Efficiency Tradeoffs for Data Collections and Aggregation in Large Scale Wireless Sensor Networks. In *IEEE WiNA-2009, co-located with 6th IEEE MASS* (2009)

[21] Santini, S., and Römer, K. An adaptive strategy for quality-based data reduction in wireless sensor networks. *Proc. of (INSS)* (2006).

[22] Silberstein, A., Braynard, R., Ellis, C., Munagala, K., and Yang, J. A sampling-based approach to optimizing top-k queries in sensor networks. *ICDE* (2006).

[23] Silberstein, A., Braynard, R., and Yang, J. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. *ACM SIGMOD* (2006), 157–168.

[24] Soliman, M. A., Chang, K. C.-C., and Ilyas, I. F. Top-k query processing in uncertain databases. In *IEE ICDE)* (2007), pp. 896–905.

[25] Tulone, D., and Madden, S. PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks. *3rd EWSN* (2006).

[26] Wan, P.-J., Alzoubi, K. and Frieder, O. Distributed construction of connected dominating set in wireless ad hoc networks. In *IEEE INFOCOM* (2002)

[27] Jiang, H. and Cheng, J. and Wang, D. and Wang, C. and Tan, G. Continuous multi-dimensional top-k query processing in sensor networks INFOCOM, 2011

[28] Wu, M., Xu, J., Tang, X., and Lee, W. Monitoring Top-k Query in Wireless Sensor Networks. *Proc. of ICDE* (2006).

[29] Chen, B. and Liang, W. and Zhou, R. and Yu, J.X. Energy-efficient top-k query processing in wireless sensor networks. CIKM 2010

[30] Wu, M., Xu, J., Tang, X., and Lee, W. Top-k monitoring in wireless sensor networks. *IEEE TKDE., 17*, 7 (2007)

[31] http://en.wikipedia.org/wiki/TinyOS.