Optimization of Accurate Top-k Query in Sensor Networks with Cached Data

Qunhua Pan, Minglu Li, Min-You Wu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
{oct-pan,li-ml,wu-my}@cs.sjtu.edu.cn

Wei Shu
Department of Electrical and Computer Engineering
The University of New Mexico
Albuquerque, NM 87131, USA
shu@ece.unm.edu

Abstract — **It is crucial to design algorithms for energy-efficient query processing for a sensor network since sensor nodes are battery-powered, and thus their lifetimes are limited. We propose a history-based approach to optimizing query processing. We apply the approach to the top-k query problem and design new algorithms. Energy consumption can be reduced by pruning unnecessary sub-queries or guiding the query to right directions. Simulation results show that energy cost can be significantly reduced. This approach can be generalized for other query problems.**

Keywords-Sensor networks; Query processing; Top k; Cached data

I. Introduction

How to effectively and efficiently query sensor networks is an important and challenging task. Since sensors are often battery-powered, the lifetime of the network is tied to the rate at which it consumes energy. Additionally, the radio communication is orders of magnitude more energy demanding than the local processing. Hence, minimizing communication in query execution can save a significant amount of energy and help to prolong the lifetime of the network.

In this paper we focus on optimizing top-k query in sensor networks. We present an efficient top-k query processing approach for distributed sensor networks. A top-k query is to find the k highest ranked answers for a user. The top-k query is practically useful and can serve as a case study to demonstrate the advantages of our proposed optimization framework.

For example, consider the temperature monitoring in a large exhibition with many rooms. To automatically monitor and adjust the temperature of the rooms, it needs to collect the real-time temperature of each room. Temperature sensors are deployed in all rooms. These sensors are self-organized into a wireless sensor network. The sensory data are sent back to the control room, where an operator can monitor the temperature of each room and be able to know which rooms having the k highest or lowest temperatures. The controller may run a top-k query over the network to find out the target rooms in order to adjust their air conditioners or heaters. This example of top-k query might be "*Find four highest data values of temperature in the exhibition hall.*"

Optimization of top-k query is significantly more complex than one for ordinary queries (e.g. return all readings greater than x). Top-k query must know all the sensory data in the sensor network. Flooding the query to the entire sensor network can obtain all information but will consume too much energy. Every sensor is visited no matter whether its data will be useful or not. The goal of our work is to develop an optimization framework that is efficient in exploiting past behavior and flexible enough to incorporate energy and topology considerations. Our contributions are as follows:

- We propose to utilize past sensor readings to optimize the query processing. By using the aggregated historical data in intermediate nodes, we develop a query optimization framework so those nodes with less useful data will be bypassed.
- We apply the above concepts and techniques to the top-k query distributed algorithms. We evaluate these algorithms with simulated and real-world data. The top-k data can be accurately obtained if the data do not change from the last base query reading. The communication cost has been reduced.

The rest of this paper is organized as follows: Section II provides an overview of related works in sensor network query processing. Section III presents preliminaries of our top-k query processing. The algorithms are discussed in Section IV, followed by performance evaluation in Section V. We conclude in Section VI.

II. Related Works

A substantial amount of work has been done on querying sensor networks. Many papers focus on saving energy and extending the life of the network. One main strategy is to do at least some query work in-network, as suggested by [1] and [2]. In-network aggregation reduces overall energy consumption by performing computation within the network and reducing the size of transmissions propagated upward.

A strategy for conserving energy by using approximation has been proposed. Work in [3] introduced the idea of using models to encode the relationships between sensor values at different motes, as well as between different types of sensors. Silberstein et al. [4] presented the sampling-based top-k queries in sensor networks. In this work, sampling and linear programming complement each other to form an optimization framework. This framework is applied to the top-k query problem and the PROSPECTOR series of algorithms are designed. Zeinalipour-Yazti et al. [5] present the Threshold Join Algorithm (TJA), which is an efficient top-k query processing algorithm for distributed sensor networks. The algorithm seeks to optimize the use of the network resources by pushing computation into the network. Wu et al. [6] present a local threshold filter algorithm to monitor the top-k query in sensor network. The problem of continually providing top k answers in a distributed environment is discussed in [7].

There have been a lot of works in the area of top-k query processing in the web and database access. Bruno et al. [8] discuss the problem of answering top-k queries over web accessible databases and present several algorithms for processing such queries. Reference [9] studied the advantages and limitations of processing a top-k query by translating it into a single range query that can be efficiently processed by a traditional relational database management system (RDBMS).

Most of the above approaches focus on in-network processing and increasing the accuracy of top-k query in a distributed environment. Threshold is set in sensor node to filter the redundant data value received from its children nodes. Though the threshold can filter data, all the sensors are responded to the query. The total energy consumed is high. Our approach can prune the query nodes with less useful data, therefore, saving energy as shown in our simulation results.

III. Preliminaries

A. Network model

Query processing in sensor networks concerns with routing tree, aggregation [1][10] and semantics [11]. We present the properties of the sensor network in this section.

A sensor network consists of N nodes $n_0,\ldots,n_{N-1}$. Here n_0 is a *root* where a query is started and the result is subsequently collected. Initially, every sensor node has some immediate neighbor nodes and the sensor network is connected. A basic topology desired in data-gathering wireless sensor networks is a spanning tree, as the traffic is mainly in the form of many-to-one flows. We use a nearest-first tree [12] as our network topology. The node will choose the nearest candidate node as its parent. Intuitively, this should result in a better channel quality on all links, assuming all nodes have the same transmitting power. During the initiation phase, the root starts a flooding process to construct a nearest-first tree. As a result, every node n_i except node n_0 will have its parent node as $p(n_i)$. Also, every node n_i will have m_i (≥ 0) children nodes: $c_{i,1}$, $c_{i,2}$, …, $c_{i,mi}$. We assume that the query is pushed down into the network from the root node in a *distribution phase*, and the results are routed up from children to parents and eventually to the root in a *collection phase*.

The primary source of energy consumption in a sensor network is radio communication. Therefore, we use the total amount of energy spent on communication as the primary yardstick for measuring the cost of a query. We use a simplified model shown in [13] for the radio hardware energy dissipation as follows. The energy to transmit a l-bit data to a distance d is

$$E_{Tx}(l,d) = \begin{cases} lE_{elec} + l\varepsilon_{fs}d^2 \\ lE_{elec} + l\varepsilon_{mp}d^4 \end{cases}$$

If the distance is less than a threshold d_0 , the free space (*fs*) model is used; otherwise, the multipath (*mp*) model is used. The energy for receiving data is $E_{Rx}(l) = l \times E_{elec}$. The total amount of energy spent in sending and receiving a unicast message of l bytes is ($E_{Tx} + E_{Rx}$).

A naive top-k query computes the answer bottom-up in one pass over the network. Each node simply collects the top k values from each of its children, computes the top-k among all such values and its own, and passes them on to its parent. This algorithm guarantees an exact answer. But, the total size of the messages used by a naive top-k query is quite large. A node with fan-out f will receive $f \times k$ values from its children when there are at least k values in children, but at least $(f\text{-}1) \times k$ of them will not be in the final result.

For simplicity, it is assumed a single set of value X is queried, which has a key represented as an integer number x. Thus, each node n_i will keep a measured value x_i and is able to sense a new value x_i' at any time as needed. We assume there is a synchronized clock accessible by every sensor node as T. Based on this clock, every x_i is time-stamped when the value is sensed and updated.

B. Base query

Due to changes in the monitoring environment, a base query is designed to request all sensors in the sensor network to periodically sense the field. Normally, a base query is an aggregation operation performed every time period of P.

An example can be a sensor network that monitors the forest fire. A sensor in the field should wake up every $P = 10$ minutes, sense and transmit the temperature to its parent. The aggregation operation is to extract the maximum of values from its children as well as itself. The base station will receive information of the highest temperature in the field. Once a potential fire is detected, an alarm notifies the operator to generate normal queries to understand the degree and ranges of the fire. As an example, a top-k query may be sent to find out the most severe part of the fire.

In summary, a base query is described as follows. First, it is a periodical query. Second, it is a query that is sent to every node through a broadcast tree. Third, it is for surveillance purpose so it must be energy-efficient. To this extend, the length of the period must be carefully selected and should be application-aware. It must be frequent enough so the field is sufficiently monitored and must be long enough to ensure a long lifetime of the sensor network. Furthermore, the message sent to the base station must be minimized. So the base query might be "*find the maximum value of the sensor network*" and there is only one maximum data value should be sent in each node. In the above example, aggregation of maximum of temperature values from every node is performed to monitor the fire in the field so that only N-1 messages with a single value is transmitted every time period. Finally, all messages are cached in their destination. Thus, these cached data will provide a guide to subsequent queries.

There might be some changes in the field during a time period. Subsequent queries may return inaccurate top k values due to this dynamically change data. However, when a time period is over, a new round of base query is executed so the error will not accumulate.

IV. Algorithms for History-Based Approach

We now introduce the history-base approach for top-k query processing in sensor networks. With an additional probing and filtering phase, this approach reduces the number of nodes required to respond the query. In addition to the baseline algorithm: *Query-Ordinary*, two optimized query algorithms, *Query-Prune* and *Query-Estimate*, are presented to utilize the cached historical data in sensors to gain better performance or additional features. *Query-Prune* exploits the cached data for optimization, but does not consider the topology of the sensor network. *Query-Estimate* is topology-aware and considers the cached data in nodes near the root. *Query-Ordinary* performs the naive query algorithm described in Section III.

A. Technical Terms

Before the presentation of query algorithms, the technical terms used are described here.

1) Active Query

Different from traditional queries, an active query will determine its route to sensor nodes and its operation at a node. A query can be defined in many aspects. For simplicity, here we use a simple query to ask for top-k values with the following attributes:

k	requests top-k values ($k \geq 1$).
T	specifies the validation deadline of the queried data. If $T = \infty$, it queries the most currently available data.
P	specifies a time interval to re-invoke process of data sensing and aggregation. If $P = \infty$, the corresponding process will be executed only once.

2) Data collection

Every node n_i, either periodically or as requested, will sense from its own sensor a new value X_i. It may also receive m_i sets of data $V_{i,ci}$ from its children. These m_i+1 set of data to form a vector U_i, where, $U_i = [X_i, V_{i,c1}, \ldots, V_{i,mi}]$. U_i needs to be aggregated to produce a vector V_i to be sent to its parent $p(n_i)$.

3) Data aggregation

Data aggregation is a process to produce V_i.. A top-k query will return a set of data whose keys are of top-k ranks. In general, it will take the newly collected data U_i, or the current existing data vector V_i, the previous top-k value k and the new top-k value k_{new} as inputs and output a new data vector V_i'. When k_{new} is larger than k, procedure AggregationU2V is called, otherwise procedure AggregationV2V is called. Both procedures are illustrated in Figure1.

```
Vi' = AggregationU2V (Ui, k, knew)
    flat vector Ui into a one-dimension array of element X with
        size = f where f ≤ mi × k + 1
    apply application specified criteria to reduce f elements into
        knew elements  and assign them into vector Vi'
  return Vi'

Vi' = AggregationV2V (Vi, k, knew)
    assert knew ≤ k
    apply application specified criteria to reduce k elements into
        knew elements and assign them into vector Vi'
    return Vi'
```

Figure 1. Procedures of Data Aggregation

4) Status and data cache

At each node n_i, $W_i = (U_i, V_i)$ serves as a current status at node n_i and will be cached with timestamps. Availability of W_i and its cache can significantly help execution of active queries as illustrated later. Every time a new W_i is computed, its old value will be added into the cache.

B. Query-Ordinary

We begin with the ordinary query algorithm that guarantees exact answers to top-k queries without optimization. In the query distribution phase, once a query *Q* arrived at node n_i, it will disseminate *Q* onto its children via the query tree. After that, the node will enter its data collection phase, during which the environment is sensed to produce a new data value and data are also collected from the node's children. Based on these newly collected data, the node will perform the aggregation operation and send them upwards to its parent node. Meanwhile, data are cached in all intermediate nodes.

```
QueryOrdinary(ni, Q, knew, k, P)
    for each child ci,j, 1≤ j≤ mi // mi is # of children of ni
        send QueryOrdinary(ni, Q, knew, k, P) to ci,j
    Vi' = DataUpward(ni, Q, knew, k, P)
    return Vi'
DataUpward(ni, Q, knew, k, P)
    if P ≠ ∞  // need periodically re-invocation
      set timer=P, when timer expires
          call DataUpward(ni, Q, knew, k, P)
    perform sensing to get a new value Xi
    reset update[ci,j] = false, 1 ≤ j ≤ mi // mi is # of children
    while not all update[ci,j] = true and not timeout
        upon arrival of Vci,j ; update[ci,j] = true;
    form new vector Ui'= [Xi, Vci,1', Vci,2', …, Vi,mi']
    Vi' = AggregationU2V (Ui', knew, knew);
    send Vi' to its parent p(ni)
    if knew > k
        Ui = Ui'; Vi = Vi'; k = knew
```

Figure 2. Procedure of Query-Ordinary

This process will complete at the root n_0, where the exact top-k data values will be obtained by the return value from *DataUpward*. Detailed procedures are shown in Figure 2. Based on this *Query-Ordinary* algorithm, a *base query* is an invocation of *QueryDownward* with k = 1 at n_0. A node with m_i children receives no more than m_i values. Since each value is transmitted in a separate message, base query may require a large number of messages. Overall, the base query collects the initial data to form a basis to assist the subsequent queries, such as from *Query-Prune* or *Query-Estimate*.

C. *Query-Prune*

Based on the results generated with the previous query for top k data, if the subsequent query requests for a richer data set, that is, with a larger k_{new} ($k_{new} > k$), retrievals of every node in network can be pruned with utilization of W_i at intermediate nodes. On the other hand, if a new query is made with a smaller k_{new} ($k_{new} \leq k$), W_0 at root node n_0 can provide data immediately. Figure 3 is a modified procedure with consideration of prunes.

```
QueryPrune($n_i$, Q, $k_{new}$, k)
    if $k \geq k_{new}$ // $W_0$ data can directly satisfy the query
        $V_i' = AggregationV2V(V_i, k, k_{new})$
    else
        $V_i' = QueryDownPrune(n_i, Q, k_{new}, k, 0, True)$
    return $V_i'$
QueryDownPrune($n_i$, Q, $k_{new}$, k, Y, changeY)
    if (changeY)
        $Vi' = AggregationU2V(U_i, k, k_{new})$
        // re-discover more data from $U_i$
        if $\|V_i'\| \geq k_{new}$
        Y= max(Y,$V_i'[k_{new}-1]$) // a threshold for top-k
    reset update[$c_{i,j}$] =true, $1 \leq j \leq m_i$ // $m_i$ is # of children
    for each child $c_{i,j}$, if $V_{ci,j}[0] > Y$
        // node $c_{i,j}$ needs to be queried
        send QueryDownPrune($n_i$, Q, $k_{new}$, k, Y) to $c_{i,j}$
        update[$c_{i,j}$] = false; // $c_{i,j}$ has to send data back to $n_i$.
    while not all update[$c_{i,j}$] = true and not timeout
        upon arrival of $V_{ci,j}'$ ; update[$c_{i,j}$] = true;
    form new vector $U_i= [X_i, V_{ci,1}', V_{ci,2}', \ldots, V_{i,mi}']$
    $V_i = AggregationU2V(U_i, k_{new}, k_{new})$;
    return $V_i$ to its parent p($n_i$)
    $k = k_{new}$
```

Figure 3. Procedure of Query-Prune

Most previous works on top-k selection provide an approximate answer of the original top-k query. In fact, in order to obtain an accurate answer of the top-k query, the entire sensor network needs to be searched. However, when some previous data, either from the base query or from previous normal queries, are cached in nodes, it becomes possible that not every nodes to be searched to find an accurate answer to the top-k query. In the query dissemination procedure with prune shown in Figure 3, an approximation of the threshold is obtained by searching a small number of nodes. Then many unnecessary sub-queries can be pruned according to the threshold, *Y*. The set of *l* values obtained by *Query-Prune* are indeed the top *l* values in the sensor network.

D. *Query-Estimate*

Imprecise estimation based on the previous cached data can help further with prune. Assume there is a lookup table *L* built once the tree is established at the root. There will be d elements in the lookup table *L*, $l_1, l_2,..., l_d$, where d is the depth of the tree. Here, l_i represents the total number of children at depth i. In order to retrieve the cached data, H(k_{new},k) value is calculated as the smallest number that satisfies $k_{new} \geq \Sigma_{1 \leq i \leq H} (k \times l_i)$. When $k_{new} > k$, *QueryExtended* will be invoked so vector E_0 will be finally collected back to be used to estimate a prune threshold.

The *QueryEstimate* algorithm is shown in Figure 4. Here, the threshold is not necessarily determined at the root node. More data from the children nodes may be collected back in the *QueryExtended* procedure. At least k data will be collected from which the threshold can be determined. This estimated threshold could be equal or less than the accurate threshold.

```
QueryEstimate($n_i$, Q, $k_{new}$, k)
    if $k \geq k_{new}$ // $W_0$ data can directly satisfy the query
        $V_i' = AggregationV2V(V_i, k, k_{new})$
    else
        compute h = H($k_{new}$, k) from lookup table L
        $E_0$ = QueryExtended($n_0$, Q, $k_{new}$, k, h); Y = 0;
        Y = $E_0[k_{new}-1]$ // Y is a threshold for top-$k_{new}$
        $V_i' = QueryDownPrune(n_i, Q, k_{new}, k, Y, False)$
    return $V_i'$
QueryExtended($n_i$, Q, $k_{new}$, k, h)
    if (h≡1) // the last level
        $EV_i' = AggregationU2V(U_i, k, k_{new})$;
    else // not the last level, waiting for children
        for each child $c_{i,j}$, $1 \leq j \leq m_i$
            send QueryExtendeded($n_i$, Q, $k_{new}$, k, h-1) to $c_{i,j}$
            reset update[$c_{i,j}$] = false // $c_{i,j}$ has to send data back
        while not all update[$c_{i,j}$] = true and not timeout
          upon arrival of $EV_{ci,j}'$ ; update[$c_{i,j}$] = true;
        new vector $E_i' = [X_i, EV_{ci,1}', EV_{ci,2}', \ldots, EV_{i,mi}']$
        $EV_i' = AggregationU2V(E_i', k_{new}, k_{new})$;
    return $EV_i'$ to its parent p($n_i$)
```

Figure 4. Procedure of Query-Estimate

The procedure to obtain the threshold H, though may consume some energy, provides an estimation which may reduce the number of hops compared to the query dissemination with prune algorithm. On the other hand, the threshold in *Query-Estimate* is fixed. If the initial threshold is set too low, more nodes will be visited.

V. Simulation And Performance Evaluateion

In this section, we describe our experimental framework which consists of a distributed simulator written in C# that implements *Query-Ordinary*, *Query-Prune* and *Query-Estimate*. We model communication costs, as discussed in Section III to includes both the cost of query execution and the cost of data collection.

We tested two cases of top k query execution. In the first case (Case−Ⅰ), every query is applied individually after the base query; and in the second case (Case-Ⅱ), query k is always based on query k-1.

1) *Intel Lab Data*

We evaluated our algorithms on a dataset from Intel Berkeley Research Lab [14] where the sensory data was generated by light sensors. In addition, since the lab area is small, we set radio range to 6 meters that keeps the sensors connected. The generated nearest-first tree is shown in Figure 5.

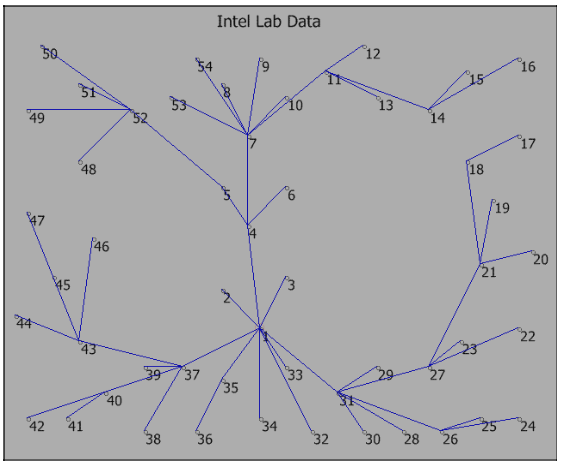

Figure 5. The nearest-first tree generated from Intel Lab Data

We test our optimization algorithms using the data collected at 1:10 am, Feb. 28, 2004 and run base query (k = 1) on the data. The normal queries (Case I or Case II) are then applied. In the collection phase, when the node transmits data value, the communication cost is $4\times\alpha$, where α is proportional to ($E_{Tx} + E_{Rx}$) and data value is expressed by 4 bytes.

The results of Case-I are shown in Figure 6. It shows that the optimization by cached data can reduce the total communication cost in top-k query. The initial threshold has a major impact on the performance. The *Query-Estimate* fetched more data from children nodes, the initial threshold in *Query-Estimate* is higher than that in *Query-Prune* especially when k is higher. Fewer nodes are visited and communication cost is reduced.

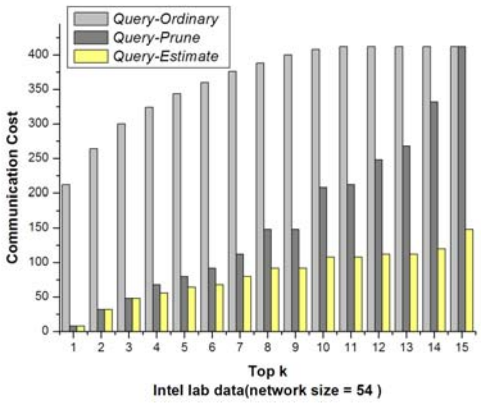

Figure 6. Communication costs of Case-Ⅰ

Nest, we evaluate the performance for Case-II queries, where the top-k (k = 1, …, 15) queries are injected into the sensor network in order after the base query has been executed. The results of communication costs in CaseⅡusing the Intel Lab Data along with CaseⅠare shown in Figure 7. As the later query can use the aggregated data produced by the previous top-k query in Case-II, more information is collected by top-k (k > 1) query than the base query. Thus, the threshold generated in root node is closer to real top k-*th* data. Therefore, more sensor nodes are pruned. Moreover, *Query-Prune* and *Query-Estimate* have similar performance in Case-II.

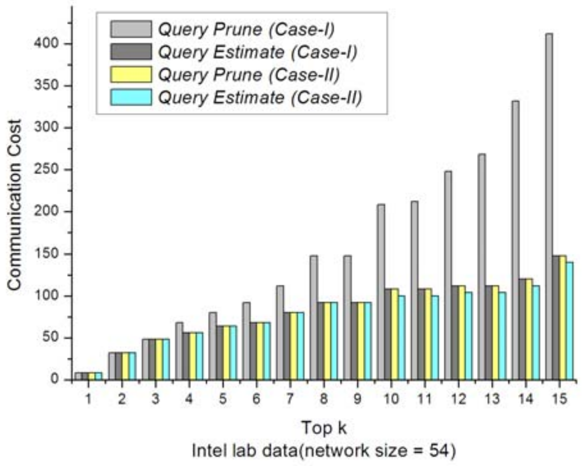

Figure 7. Communication costs of Case-I and Case-II

2) Synthetic Data

In order to compare the performance of different query approaches in a large-scale sensor network, we place 400 nodes uniformly in a square area with 20 × 20 grids. Each grid is of 1 square meter. The location of a node in a grid is generated randomly. All nodes in the network have the same communication range. The sink is located at the center grid of the area. The data value in each node is random selected from 150 to 1000. The communication range of node is 2.5 meter.

The results of communication cost of Case-I for the synthetic data are shown in Figure 8. The *Query-Prune* reduces the total energy cost by pruning the queried nodes. The *Query-Estimate* spends the least energy cost.

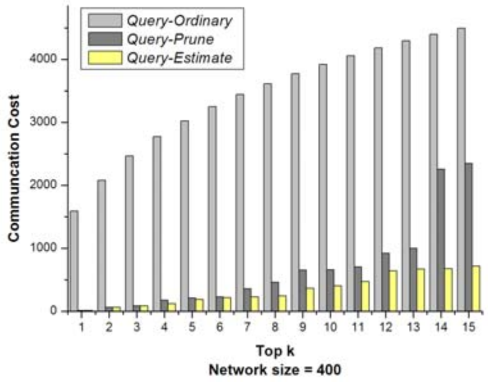

Figure 8. Communication cost of Case-Ⅰ

Figure 9 shows the communication costs of both Case-I and Case-II queries for synthetic data. When k is increasing, the

Case-II query caches more and more data in intermediate nodes. By pruning more sensor nodes, it reduces energy cost. The similar performance trend of the synthetic data and the Intel Lab Data shows that the performance results apply to both small-scale and large-scale networks.

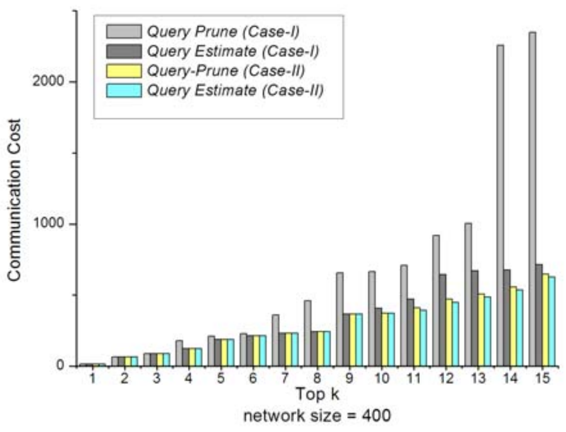

Figure 9. Communication cost of Case-I and Case-II

VI. CONCLUSIONS

In this paper, we studied the problem of optimal top-k query in sensor networks. We proposed a new querying approach based on cached data. By utilizing the cached data collected by the base query as well as previous top-k queries, we designed *Query-Prune* and *Query-Estimate* algorithms. Performance results showed that this approach is both efficient and practical. The goal of obtaining accurate top-k values with significant energy saving has been achieved.

ACKNOWLEDGMENT

This paper was supported partially by Natural Science Foundation of China grant No.60573138, the National Grand Fundamental Research 973 Program of China under Grant No.2006CB303000 and the National 863 Program grant No.2005aa12232.

REFERENCES

[1] M. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis, "Balancing energy efficiency and quality of aggregate data in sensor networks". The VLDB Journal, 13(4):384–403, 2004.

[2] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks". ACM SIGMOD, 2003. pp. 491~502.

[3] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. "Model-driven data acquisition in sensor networks". Intl. Conf. of VLDB, 2004.

[4] A. Silberstein, R. Braynard, C. Ellis, K. Munagala and J. Yang, "A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks". Intl. Conf. of ICDE, 2006.

[5] D. ZeinalipourYazti, Z. Vagena, D. Gunopulos, V. Kalogeraki, V. Tsotras, "The Threshold Join Algorithm for Top-k Queries in Distributed Sensor Networks", DMSN'05, August 29, 2005.

[6] M. Wu, J. Xu, X. Tang, W. Lee, "Monitoring Top-k Query inWireless Sensor Networks". Intl. Conf. of ICDE, 2006.

[7] B. Babcock and C. Olston. "Distributed top-k monitoring." ACM SIGMOD Intl. Conf. on Management of Data, June 2003.

[8] A. Marian, N. Bruno. "Evaluating top-k queries over web-accessible databases". ACM Trans. Database Syst, 29(2):319-362, 2004.

[9] N. Bruno, S. Chaudhurl, L. Gravano, "Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation", ACM Trans on Database Systems, Vol. 27(2), pp.153-187, June 2002.

[10] S. Madden, R. Szewczyk, Michael J. Franklin. "Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks", Workshop on Mobile Computing and Systems Applications, 2002.

[11] Q. Pan, M. Li, M.Y. Wu, "A semantic-based architecture for sensor networks", Annals of telecommunications, Vol.60 No.7-8, pp.928-943, July-August 2005.

[12] C. Zhou and B. Krishnamachari, "Localized Topology Generation Mechanisms for Wireless Sensor Networks", GlobeCom, pp. 1269–1273, 2003.

[13] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks", IEEE Transactions on Wireless Communications, 2002,1(4):660-669, 2002.

[14] Intel Berkeley Research Lab. http://berkeley.intelresearch. net/labdata/