Processing Precision-Constrained Approximate Queries in Wireless Sensor Networks*

Minji Wu Jianliang Xu
Hong Kong Baptist University
Kowloon Tong, Hong Kong
{alexwu,xujl}@comp.hkbu.edu.hk

Xueyan Tang
Nanyang Technological University
Singapore
asxytang@ntu.edu.sg

Abstract

A lot of research efforts have been devoted to improving energy efficiency for wireless sensor networks by exploring distributed data storage and in-network query processing techniques. In this paper, we present a generic two-tier data storage strategy for answering precision-constrained approximate queries in a sensor network. The basic idea is to keep two versions of data in the network. A high-precision version is kept at the sensor node that captures the data while a low-precision version is maintained at the base station. We develop query processing and node refreshment strategies for various types of approximate queries under the two-tier storage. Our extensive experiments show that the two-tier storage strategy outperforms the basic centralized storage scheme by an order of magnitude in terms of network lifetime under various system configurations.

1 Introduction

The rapid development in sensing and wireless communication technologies has made the availability of wireless sensor networks. Wireless sensor networks can be used in a wide range of practical applications such as habitat monitoring and environment monitoring. For example, the conservation of endangered species in Hong Kong (such as Chinese White Dolphins and Romer's Tree Frog) is hampered by insufficient knowledge of their status [19]. The current practice is to send human beings to the habitat sites of these species to collect their status information. However, this approach introduces potential disturbance. The use of networked sensors not only eliminates the potential impacts of human presence, but also enables data collection at scales and resolutions that are difficult to achieve through traditional instrumentation [17].

*This work was supported by Research Grants Council of Hong Kong SAR, China under Project No. HKBU 2115/05E.

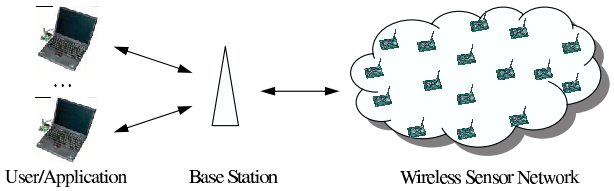


Figure 1. Sensor Network Architecture

A wireless sensor network is typically constructed of a base station and a large number of sensor nodes scattered in an area of interest (see Figure 1). These sensor nodes are equipped with sensing, data processing, and communication components to collect local measurements, process and exchange information about the environment. They are usually battery powered. Replacing the batteries is not only costly but also impossible in many situations. As such, energy efficiency is a critical consideration in the design of sensor networks. There have been significant research efforts towards energy-conserving sensor networks. However, most of the existing studies have focused on the design of sensing architectures and protocols in support of *exact* answers to user queries.

Here we take a different approach to improve energy efficiency. We exploit the trade-off between data quality and energy consumption to extend network lifetime by investigating approximate queries with precision guarantees. Many sensor applications are willing to tolerate a certain degree of error in data due to either the application nature or the high resource constraints in sensor networks. For example, to save energy, a user retrieving the temperature reading of a sensor may allow an error of one degree. In this case, the retrieved value is acceptable as long as it is within ± 1 degree of the actual reading.

In this paper, we present a generic two-tier data storage strategy for answering precision-constrained approximate queries. The basic idea is to keep two versions of data in the network. A high-precision version is kept at the sensor node that captures the data, while the same data with a lower precision is replicated at the base station. The imprecision of

low-precision data at the base station is bounded by an *approximation range*. An update in reading will not be sent to the base station if the new value remains within the approximation range. Thus, a query can be answered by the base station if the user's required precision is weaker than that of the result computed based on low-precision data. Otherwise, some of the sensor nodes need to be refreshed to improve the data precision, in which two fundamental issues arise: 1) How to determine the to-refresh node set? As the costs for sensor nodes to report their readings to the base station differ from one another, it is important to pick the set of sensor nodes that incurs the minimal energy consumption. 2) Upon selecting the set of to-refresh nodes, how to refresh them in an energy-efficient way? In some types of queries, we do not need to refresh all the nodes in the to-refresh set to resolve the answer.

We develop detailed query processing and node refreshment strategies for various types of approximate queries (including ID-based, range, top-k, and aggregate queries) under the proposed two-tier data storage. Extensive experiments are conducted to evaluate the performance of the two-tier storage strategy using real trace data. The results show that the two-tier storage strategy substantially outperforms the basic centralized storage and local storage schemes under various system configurations.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 gives some preliminaries of the work. Section 4 presents the two-tier data storage strategy and develops the query processing and node refreshment techniques for various types of queries. We evaluate the performance of our proposed techniques in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

Distributed data storage for wireless sensor networks has been investigated in the literature [3, 10, 15, 21]. In the TinyDB project, Madden *et al.* [10] proposed a pull-based acquisitional query processing (ACQP) model, where the sensors control where, when, and how often the data is acquired and delivered to query processing operators. The Cougar project [3] employed a hybrid pull-push model: sensed data is pushed to some selected view nodes, from which the data is pulled by queries. Ratnasamy *et al.* [15] proposed a data-centric storage model: each sensor reading is pushed to the sensor node nearest to some geographical location hashed from a predefined key. Only equality queries are supported by data-centric storage.

In-network query processing techniques have been studied for various data storage models [5, 7, 22]. These studies examined exact queries only. Query evaluation techniques over imprecise data have been investigated by Lazaridis and Mehrotra [8], which quantified data quality with *set-based uncertainty* and *value-based uncertainty*. They then proposed a cost efficient processing technique for quality-aware relational queries. However, how to collect imprecise data was not discussed in [8]. Deshpande *et al.* [2] have most recently developed a model-driven data acquisition architecture that employs statistical modeling techniques to efficiently answer one-shot queries with high confidence.

In-network data aggregation, where data values are aggregated as forwarded by the network, has been receiving increasing attention recently [1, 9]. Continuous precision-constrained aggregate queries were studied in [4, 16]. The key issue is how to allocate the error budget to the sensor nodes involved in the aggregation tree. Sharaf *et al.* [16] implemented a uniform error allocation scheme. Deligiannakis *et al.* [4] improved it by developing an adaptive algorithm to allocate more error tolerances to the nodes that can reduce more network traffic. Neither of these studies considered balancing the energy consumption of sensor nodes to extend network lifetime. In a recent work [18], we proposed an error allocation algorithm to optimize network lifetime. However, these techniques developed for continuous queries are not applicable to one-shot queries. To the best of our knowledge, this is the first effort to investigate one-shot approximate queries for wireless sensor networks.

Other inspiring work includes querying approximate data over distributed caches and streams. Olston *et al.* [13] studied error-bounded aggregate queries over distributed data streams. An adaptive scheme for precision adjustment at each individual source was proposed to reduce the communication cost. Inspired from [14], Han *et al.* [6] developed an adaptive precision setting algorithm for precision-constrained data collection in a single-hop sensor network. However, their work was confined to collection of individual sensor readings. In contrast, this paper presents a generic two-tier data storage strategy in support of various types of queries in multi-hop sensor networks.

3 Preliminaries

We assume the wireless sensor network is composed of a base station and many sensor nodes. Each sensor node measures the local physical phenomena (e.g., temperature, humidity, and light) at a *fixed* sampling rate and reports to the base station if necessary. The base station and sensor nodes are equipped with wireless interfaces to communicate with each other. Since the wireless transmission range is limited, a routing infrastructure (such as TAG tree [9]) is established to transmit data between the base station and the sensor nodes in the network.

The base station serves as an interface for external users to pose queries to the sensor network. Users are interested in various types of precision-constrained approximate queries, e.g. (their definitions will be detailed in Section 4):

Q1 Find the temperature reading (within $\pm 1^{\circ}$C) of Sensor Node 1. (ID-based Query)

Q2 Find the sensors whose temperature readings are above 100°C (within an error of 5°C). (Range Query)

Q3 Find the k sensors (within an error of 1°C) with the highest temperature readings. (Top-k Query)

Q4 Find the average temperature reading (within an error of 1°C) of the sensors. (Aggregate Query)

To answer these queries, two basic data storage strategies exist:

- **Centralized Storage (CS):** Each sensor node reports to the base station whenever a new reading is sampled. Note that the report of sensor readings cannot make use of the in-network aggregation technique. When the base station receives a query, the result can be immediately computed based on the stored up-to-date readings.
- **Local Storage (LS):** The sensed data is stored on the local node only. When the base station receives a query, the query is sent to the node involved (for ID-based queries) or flooded throughout the whole network (for range, top-k, and aggregate queries). The query result is then collected by the base station via the routing infrastructure. For top-k and aggregate queries, the result collection can take advantage of in-network aggregation to save energy costs.

Both of these two strategies have some performance disadvantages. The CS strategy suffers from a high updating cost while the LS strategy incurs a high querying traffic. Furthermore, they do not take advantage of the error allowed in the query answer to improve system performance. In the next section, we propose a more efficient data storage strategy.

4 Two-Tier Data Storage

4.1 Overview

Taking advantage of users' error tolerances, we propose a generic *two-tier* data storage strategy to support processing of various types of approximate queries (including ID-based, range, top-k, and aggregate queries). The base station serves as the first tier (referred to as *centric storage*) that stores imprecise sensor data, while each sensor node serves as the second tier (referred to as *local storage*) that stores exact up-to-date data. Consider a sensor node i. The imprecision of the data stored at the base station is bounded by a certain error represented by an *approximation range*, i.e., a stored value v_i with an approximation range of e_i means that the actual value must lie in the *approximate interval* $[l_i, h_i]$, where $l_i = v_i - \frac{e_i}{2}$ and $h_i = v_i + \frac{e_i}{2}$. At each sampling instance, if the newly sensed value v_i' is within a difference of $\frac{e_i}{2}$ from the previously reported value v_i, the new value v_i is kept at the sensor node locally, otherwise an update message is sent to the base station to replace v_i by v_i' in the centric storage (this is called *source-initiated update*). In this way, a lot of updating traffic can be saved. However, if the precision of stored data is insufficient to answer a query issued to the base station, we will have to refresh the data from the local storage (this is called *query-initiated refreshment*), which incurs communication overhead.

The general query processing under the two-tier storage takes three steps. First, the base station computes a tentative result based on stored imprecise data. Second, if the tentative result is not sufficiently precise, the base station refreshes the readings from a (selected) subset of the sensor nodes. After refreshment, the approximation ranges of those refreshed nodes are shrunk to zero. Note that the refreshed values remain up-to-date till the next sampling instance, after which the approximation range of node i returns to e_i. Finally, the base station re-evaluates the query based on refreshed data. In the following, we detail the query processing techniques for different types of queries.

4.2 ID-based Query

An approximate ID-based query is interested in the reading of a particular sensor node (e.g., Node 1), with a precision constraint of R. It is acceptable as long as the returned value is within a deviation of R of the true reading.

Recall that the sensor reading kept at the base station is in the form of $[l_i,\ h_i]$. If $R \geq h_i - l_i$, meaning the stored data has a higher precision than the expected, it is immediately returned to the user. Otherwise, the stored data does not meet the precision requirement, we have to send a refresh message to the desired sensor node to probe its latest reading. By doing so, we shrink the approximation range to zero (till the next sampling instance), thereby satisfying the precision requirement of the query.

4.3 Range Query

In this type of queries, we are interested in the sensor nodes whose readings are within a specified range $[L,\ H]$. With a precision constraint of R, we are required to find out all sensor nodes whose readings are in $[L + R,\ H - R]$ and to exclude those whose readings are not in $[L - R,\ H + R]$. The nodes whose readings are within $[L - R,\ L + R]$ or $[H - R,\ H + R]$ may or may not be returned.

By examining the approximate value $[l_i,\ h_i]$ of each node i, we divide the sensor nodes into three groups T^+,

$T^-, T^?$, which respectively represent the nodes who can be returned, the nodes who are not returned, and the rest. A node i is categorized in T^+ if $l_i > L - R$ and $h_i < H + R$. It is categorized in T^- if $h_i < L + R$ or $l_i > H - R$. If none of these conditions is satisfied, the node is categorized in $T^?$. The nodes in $T^?$ must be refreshed because we are not sure whether they should be included in the query result.

Take Q2 as an example, in which the query asks for the nodes whose readings are greater than 100°C with a precision constraint of 5°C. Thus, as illustrated in Figure 2, for the nodes that hold $l_i > 95$, we throw them into T^+, and for the nodes who hold $h_i < 105$, we throw them into T^-. We will refresh all the other nodes and combine the qualified nodes with T^+ as the final result.

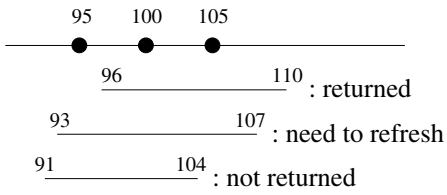


Figure 2. Processing Range Query

4.4 Top-k Query

In a top-k query, the user wants to get the k nodes with the highest (or lowest) readings. Recall that the reading of node i is approximated with an interval of $[l_i, h_i]$. Given a precision constraint of R, an approximate top-k query retrieves the (ordered) set of sensor nodes $\mathcal{T}$ with the highest readings:

$$\mathcal{T} =< n_1, n_2, \cdots, n_k >,$$

where $\forall i > j, h_{n_i} \leq l_{n_j} + R$ and $\forall l \neq n_i (i = 1, 2, \cdots, k), h_l \leq min\{l_{n_1}, l_{n_2}, \cdots, l_{n_k}\} + R$. Intuitively, if two sensor readings are within a difference of R, their order can be arbitrary.

The evaluation of an approximate top-k query is much different from that of the previous two types of queries. Given an ID-based query or range query, the set of to-refresh nodes is uniquely determined. However, for a top-k query, whether a node needs to refresh depends on the relative order of its reading against the other sensor nodes. We divide the refreshing process into two steps: selecting to-refresh nodes and processing refreshment.

When the base station receives a top-k query with a precision constraint of R, it sorts the sensor nodes based on their current approximate readings. Without loss of generality, the nodes are sorted by the upper bounds of their approximate intervals. Suppose $n_1, n_2, \cdots, n_k$ is the tentative top-k list. We will return this list immediately if no node in the list has an overlap with any other node by greater than R in the approximate interval. Otherwise, we need to refresh some nodes to resolve the top-k order. To do so, we define *refreshing candidates* (RC_i) with respect to each node i in the tentative top-k list as follows:

$$RC_i = \begin{cases} \emptyset & \text{if } \forall j, h_j - l_i \leq R, \\ \{i\} \cup \{j \mid h_j - l_i > R\} & \text{otherwise.} \end{cases}$$

Note that the refreshing candidate sets with respect to different nodes may overlap. Figure 3 shows an example top-2 query among 4 nodes (with approximate intervals of $[5, 8]$, $[3, 7]$, $[2, 6]$, and $[1, 5]$, respectively). Assume the precision constraint $R = 1$. The RC_i sets for nodes 1 and 2 are $\{1, 2\}$ and $\{2, 3, 4\}$, respectively.

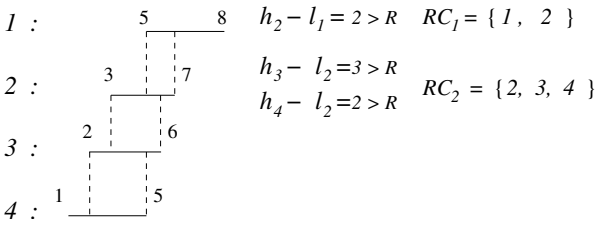


Figure 3. Finding out RC_i in Top-k Query

A straightforward refreshment strategy is to refresh all nodes in $RC = \bigcup_{i=1}^{k} RC_i$. We call it *full refreshment*. However, this might not be necessary because the refreshments of some nodes may eliminate the need to refresh other nodes. Consider the early example. $RC = \{1, 2, 3, 4\}$. Suppose we choose to refresh node 2 first, and assume that the current reading of node 2 is 5.5. After refreshment, the approximation interval $[3, 7]$ of node 2 is replaced by its exact reading of 5.5. Hence, RC_1 and RC_2 are updated with empty sets. Thus, we can assert that the top-2 list is $\langle 1, 2 \rangle$ without refreshing nodes 1, 3, and 4 anymore.

This fact suggests that we can refresh in rounds. In each round, we choose a subset of RC to refresh. When we get the refreshed reading(s), we update the RC_i set for each node i in the tentative top-k list. This process is repeated until all the RC_i sets become empty. We propose two round-based refreshment strategies:

- **Batch:** Starting from the top-1 node, the RC_i set of one top-k node is refreshed in each round.
- **Sequential:** One node is refreshed per round. In each round, the node that appears in most RC_i sets is selected to refresh. Refreshing such a node is expected to quickly resolve the order confusion.

4.5 Aggregate Query

There are five types of standard aggregate queries: SUM, MAX, MIN, COUNT, AVG. MAX and MIN queries can be viewed as top-1 queries. While COUNT can be processed in

a way similar to SUM, AVG and SUM queries differ by only a constant which is the number of sensor nodes. Therefore, we shall focus our discussion on SUM queries here.

If the reading of each sensor node i maintained at the base station has an approximation range e_i, the SUM aggregation can be computed with an approximation range $E = \sum_{i=1}^{n} e_i$, where n is the number of sensor nodes in the network.

If the query precision constraint R is greater than E, the result is returned by the base station without refreshing the reading of any sensor node. Otherwise, if R is smaller than E, some sensor readings have to be refreshed to refine the query result. Let T be the to-refresh node set. Since refreshing the reading of a sensor node reduces its approximation range to zero, to meet the precision requirement of the query, T must satisfy

$$\sum_{i \in T} e_i \geq E - R. \quad (1)$$

The refreshment can make use of in-network aggregation to improve energy efficiency. Specifically, on receiving up-to-date readings from more than one children, an intermediate node aggregates the readings before forwarding them upstream. For SUM aggregation, the partial aggregate result is simply the sum of the readings received. In-network aggregation cuts down the volume of data sent over the upper-level links in the routing tree.

We define the subtree rooted at each child of the base station as a *region*. Since these children relay packets between the base station and the other nodes in their respective regions, they consume much more energy than the others. We therefore call these nodes the *hot-spot nodes*. In order to prolong the network lifetime, we should conserve the energy at these hot-spot nodes. This implies the following design philosophy of refreshing:

- We should distribute the to-refresh nodes in as few regions as possible. This is because due to in-network aggregation, the volume of data sent by a hot-spot node to the base station is independent of the number of sensor nodes refreshed in the corresponding region. To save the energy consumption at hot-spot nodes, it is desirable to reduce the number of regions involved in the refreshment.
- When selecting regions for refreshment, we favor those with more residual energy.
- When the number of to-refresh sensor nodes is smaller than that in one region, we should choose the nodes that lie closer to the base station. This helps reduce the number of sensor nodes involved in relaying the up-to-date readings and thus the network-wide total energy consumption.

We propose to construct the to-refresh node set as follows. Starting from an empty to-refresh node set, we continue to insert nodes into the set until the total approximation range of the nodes in the set adds up to $E - R$. In this process, the regions are sequentially examined. For each region, all nodes in the region are inserted to the to-refresh node set if the insertion does not make the total approximation range greater than $E - R$. Otherwise, only a subset of the nodes in the region are inserted to increase the total approximation range of the to-refresh node set to $E - R$. The subset of nodes are selected in increasing order of their distances to the base station. We propose two examination orders of the regions:

- **Max-Size:** Our first strategy favors large regions to minimize the number of regions involved in the refreshment, i.e., the regions are examined in decreasing order of their sizes.
- **Max-Energy:** The second strategy favors the regions with more residual energy to balance the energy consumption among regions. That is, we examine the regions in descending order of the residual energy levels of their hot-spot nodes. Note that since the hot-spot nodes are located near the base station, it is practically easy to maintain the residual energy levels of these nodes (e.g., by piggybacking the energy information on the refresh messages).

5 Performance Evaluation

5.1 Simulation Setup

We have developed a simulator based on ns-2 (version 2.26) [11] and NRL's sensor network extension [12] to evaluate the proposed two-tier storage strategy. The simulator includes the detailed models of the MAC and physical layers for wireless sensor networks. The sensor nodes can operate in one of three modes: sending message, receiving message, and sleeping. These modes differ in energy consumption. The energy consumption for sending a message is determined by a cost function: $s \cdot (\alpha + \beta \cdot d^q)$, where s is the message size, α is a distance-independent term, β is the coefficient for a distance-dependent term, q is the component for the distance-dependent term, and d is the distance of message transmission. We set α=50 nJ/b, β=100 pJ/b/m^2, and q=2 in the simulation [18]. The energy consumption for receiving a message is given by $s \cdot \gamma$, where γ is set at 50 nJ/b. The power consumption in sleeping mode is set at 0.016 mW. For simplicity, the energy overhead of mode switching is ignored. We set the size of a data update message at 8 bytes, and the size of a refresh message at 4 bytes. The initial energy budget at each sensor node was set at 0.1 Joule.

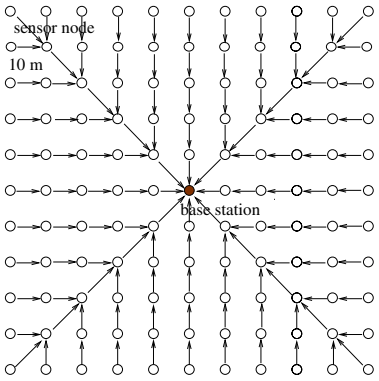


Figure 4. Network Layout

We simulated a multi-hop network of 120 sensor nodes (see Figure 4 for the layout). The sensor readings were simulated using real traces provided by the Live from Earth and Mars (LEM) project [24] of University of Washington. We used the temperature and humidity traces logged by the station at the University of Washington from Aug. 2004 to Aug. 2005 in our experiments. Each trance consists of more than 500,000 readings captured at a sampling interval of one minute. We extracted many different subtraces starting at randomly selected timepoints. Each subtrace contained 20,000 readings. The subtraces were used to simulate the physical phenomena in the immediate surroundings of different sensor nodes. In the simulation, the interval between two successive readings was assumed to be one time unit. The following two metrics are used in the performance comparison:

- **Network Lifetime**: As in the previous work [20, 23], the network lifetime is defined as the time duration before the first sensor node runs out of power. It serves as the primary metric in the performance evaluation.
- **Average Energy Consumption**: It is defined as the average amount of energy consumed by a sensor node.

In what follows, we first evaluate the query processing heuristics developed for top-k queries and aggregate queries. We then compare the performance of the proposed two-tier storage strategy against the basic centralized storage and local storage schemes with mixed types of queries.

5.2 Refreshment Strategies for Top-k Queries

In this section, we evaluate the performance of the three node refreshment strategies (proposed in Section 4.4) for top-k queries. We set k at 5 and the query precision constraint of each query at a value randomly selected from an interval of [0,1]. The temperature trace was used in this set of experiments. Figure 5 plots the network lifetime under different approximation range settings. When the approximation range is smaller than 0.5 (i.e., the stored data at the base station is relatively precise), the refreshment strategies have a similar performance since refreshments are rarely needed. With increasing the approximation range, the three strategies achieve different network lifetimes. The batch and sequential strategies are much better than (sometimes double the lifetime of) the full refreshment. By maximizing the utility of each refreshment, the sequential refreshment shows the best performance in all cases tested. It is also interesting to observe that the performance curve of each strategy forms a '$\cap$' shape. The network lifetime shortens when the approximation range is set too small (due to a large amount of source-initiated updates) or too large (due to a large amount of query-initiated refreshments). This suggests there exists an optimal setting of approximation range. We leave the study of optimization of the approximation range as an important future work.

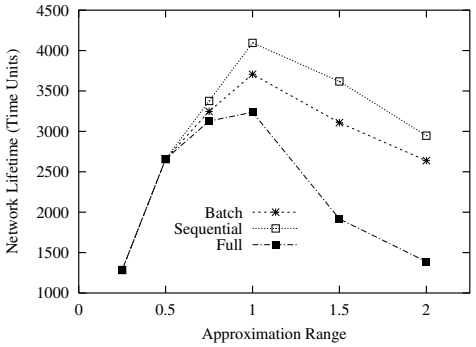


Figure 5. Performance for Top-k Queries

5.3 Node Selection for Aggregate Queries

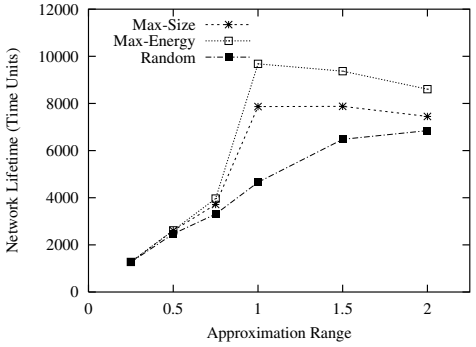
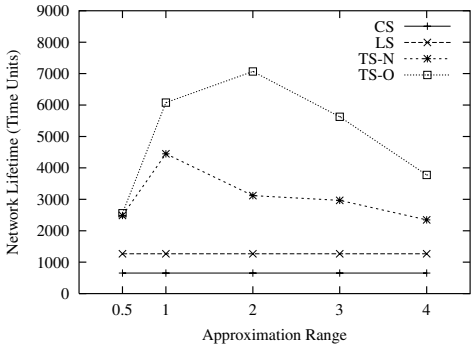


Figure 6. Performance for Aggregate Queries

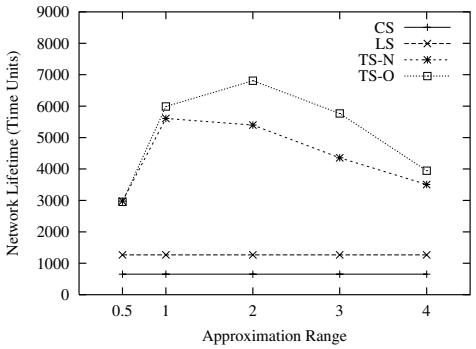
In this section, we evaluate the schemes for selection of to-refresh nodes in processing aggregate queries. In addition to the Max-Size and Max-Energy schemes proposed in Section 4.5, we also include a *Random* selection scheme for comparison. The Random scheme randomly selects the to-refresh nodes, which serves as a baseline scheme. We tested AVG aggregate queries using the temperature trace with precision constraints uniformly distributed in the range of [0, 1]. Figure 6 shows the result under different approximation range settings. As observed in the last subsec-

tion, when the approximation range is small, all the three schemes show a similar performance. Their performance differences are obvious with an approximation range larger than 0.75. Clearly, the Max-Energy scheme beats the other two schemes by more than 25%. Max-Energy performs better than Max-Size, which implies that it is more important to balance the energy consumption of each region rather than the overall network traffic.

5.4 Performance Evaluation of Two-Tier Storage



(a) Temperature



(b) Humidity

Figure 7. Lifetime vs Approximation Range

In this section, we evaluate our proposed two-tier storage (TS) against the basic centralized storage (CS) and local storage (LS) strategies under a mixed-type query environment. We simulated four types of queries, i.e., ID-based, range, top-10, and AVG queries. We set the query rate at one per time unit by default. At each querying instance, a query type is randomly selected among the four types and a precision constraint is set to a random value in the range of [0, 1]. We evaluate two versions of TS strategy: *TS-O* in which the best node selection and refreshment schemes for top-k and AVG queries (i.e., Sequential and Max-Energy) are used, and *TS-N* in which the basic node selection and refreshment schemes (i.e., Full and Random) are used. As shown in Figures 7a and 7b, both TS-O and TS-N substantially outperform CS and LS. In particular, TS-O improves the lifetime against CS by an order of magnitude and against LS by several times. For a similar reason explained in Section 5.2, the network life increases first and drops next as the approximation range increases.

Figure 8 shows the average energy consumption for each of the storage strategies under comparison. It is interesting to observe that the improvement of TS-O over CS, LS, TS-N in terms of energy consumption is not as high as that in terms of network lifetime. This suggests that the node selection and refreshment schemes in TS-O are particularly optimized for the metric of network lifetime.

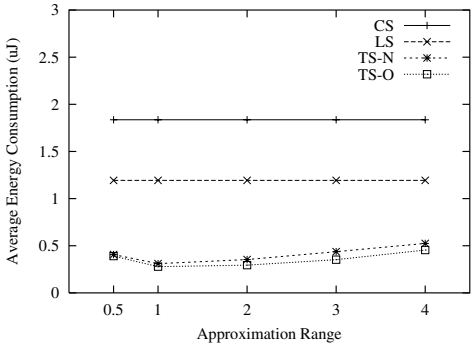


Figure 8. Energy vs Approximation Range (Temperature)

We also evaluate the proposed two-tier storage with different query patterns. Figure 9 shows the result by varying the query rate. The approximation range was set at 1. Again, TS-O and TS-N perform much better than CS and LS. As expected, all the storage strategies except CS deteriorate with increasing query rate. When the query rate is increased from 0.25 to 2, compared to LS, the performance downgrade of TS-O and TS-N is a bit smaller (i.e., 56% and 58% vs 66%). Figure 10 plots the result with different settings of query precision constraints. While CS and LS remain constant in performance as the precision constraint is relaxed, TS-O and TS-N can take advantage of the relaxed precision requirement to further improve network lifetime significantly.

6 Conclusion

This paper has presented a two-tier data storage strategy in support of precision-constrained approximate queries in wireless sensor networks. By storing high-precision data at the sensor nodes while maintaining low-precision duplicates at the base station, the proposed strategy attempts to balance the energy consumption between data updating and querying. We have developed the query processing and node refreshment strategies for various types of approximate queries under the two-tier storage. Extensive experiments have been conducted to evaluate the performance of

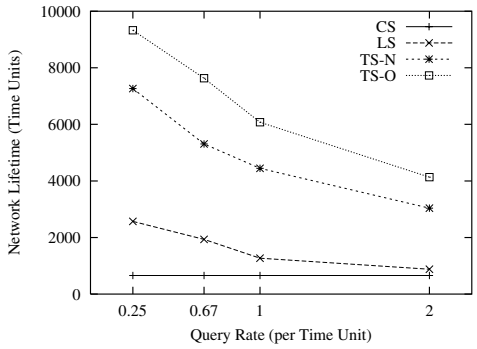


Figure 9. Lifetime vs Query Rate (Temperature)

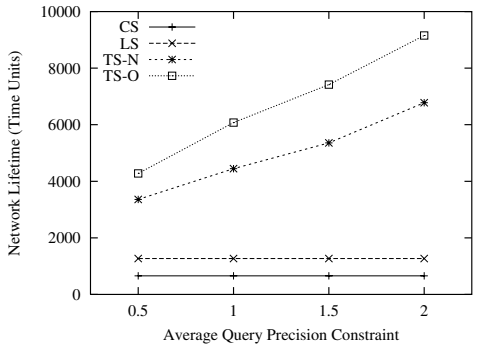


Figure 10. Lifetime vs Precision Constraint (Temperature)

the proposed two-tier storage strategy using real trace data. The results show that the two-tier storage strategy soundly outperforms the basic centralized storage and local storage schemes.

As for future work, we are going to investigate the optimal setting of approximation range for the two-tier storage. This paper did not consider the query predicates; we plan to extend the work to the queries with predicates. We also plan to build a small-scale testbed using Motes to measure the performance of different storage strategies.

References

[1] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *IEEE ICDE*, March 2004.

[2] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, August 2004.

[3] A. Demers, J. Gehrke, R. Rajaraman, J. Trigoni, and Y. Yao. The Cougar project: A work-in-progress report. In *SIGMOD Record*, 32(4): 53-59, Dec. 2003.

[4] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical in-network data aggregation with quality guarantees. In *EDBT*, March 2004.

[5] J. Gehrke and S. R. Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 2004.

[6] Q. Han, S. Mehrotra, and N. Venkatasubramanian. Energy efficient data collection in distributed sensor environments. In *IEEE ICDCS*, March 2004.

[7] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multidimensional range queries in sensor networks. In *ACM SenSys*, Nov. 2003.

[8] I. Lazaridis and S. Mehrotra. Approximate selection queries over imprecise data. In *IEEE ICDE*, March 2004.

[9] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *USENIX OSDI*, Dec. 2002.

[10] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *ACM SIGMOD*, June 2003.

[11] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[12] NRL's sensor network extension to ns-2. http://nrlsensorsim.pf.itd.nrl.navy.mil/.

[13] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *ACM SIGMOD*, June 2003.

[14] C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In *ACM SIGMOD*, May 2001.

[15] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with GHT, a geographic hash table. *ACM/Kluwer MONET*, 8(4), 2003.

[16] M.A. Sharaf, J. Beaver, A. Labrinidis, and P.K. Chrysanthis. TiNA: A scheme for temporal coherency-aware in-network aggregation. In *ACM MobiDE*, Sept. 2003.

[17] R. Szewczyk, *et al.* Habitat monitoring with sensor networks. *Communications of ACM*, June 2004.

[18] X. Tang and J. Xu. Extending network lifetime for precision-constrained data aggregation in wireless sensor networks. In *IEEE INFOCOM*, April 2006.

[19] World Wildlife Fund - Hong Kong. http://www.wwf.org.hk/eng/conservation/spe_cons/.

[20] M. Wu, J. Xu, X. Tang, and W.-C. Lee. Monitoring top-k query in wireless sensor networks. In *IEEE ICDE*, April 2006. (Poster)

[21] J. Xu, and X. Tang, and W.-C. Lee. EASE: Energy-conserving Approximate StoragE for querying object tracking sensor networks. In *IEEE SECON*, Sept. 2005.

[22] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchel. Processing window queries in wireless sensor networks. Proc. In *IEEE ICDE*, April 2006.

[23] O. Younis and S. Fahmy. Distributed clustering for ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proc. IEEE INFOCOM*, March 2004.

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