Quality Aware Query Scheduling in Wireless Sensor Networks

Hejun Wu
Department of Computer Science
Sun Yat-sen University
whjnn@cse.ust.hk

Qiong Luo
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
luo@cse.ust.hk

Jianjun Li
School of Computer Science and Technology
Huazhong University of Science and Technology
jianjunli@smail.hust.edu.cn

Alexandros Labrinidis
Department of Computer Science
University of Pittsburgh
labrinid@cs.pitt.edu

ABSTRACT

We study query scheduling in Wireless Sensor Networks (WSNs) with a focus on two important metrics: Quality of Service (QoS) and Quality of Data (QoD). The motivation comes from our observation that most WSN scheduling techniques ignore the quality requirements of queries. As a result, they are inefficient or inapplicable to quite a few applications that have different quality requirements. In this paper, we propose a distributed *Quality Aware Scheduling (QAS)* framework to address this problem. QAS works on top of existing *quality-unaware* query scheduling protocols and allows individual users to specify their QoS and QoD requirements on their queries. Given these quality requirements, QAS determines the target qualities to be provided in scheduling and the execution order of these queries so as to maximize the total system profit. Our preliminary results show that QAS significantly outperforms the baseline scheduling algorithms in terms of system profit.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Experimentation

1. INTRODUCTION

In-network query processing techniques for wireless sensor networks have been widely accepted for efficient on-line sensory data management in WSNs [11], [21]. To guarantee the efficiency and reliability of such query processing, scheduling schemes have been proposed [4], [7], [10], [17], [20]. These scheduling schemes are able to coordinate the communication timings of nodes for communication reliability, and to schedule nodes to sleep for energy efficiency.

However, existing query scheduling protocols ignore the potentially different quality requirements from different applications and users. As a result, those queries with high quality requirements may be unsatisfied, whereas the queries with lower quality requirements may be overly satisfied.

Therefore, in this paper we propose a distributed Quality Aware Scheduling framework (QAS) for these quality-unaware scheduling protocols to address the above problem. In QAS, users can specify their quality requirements on queries by giving revenues to different qualities in quality functions [6], [14]. Given the quality function, a WSN attains a profit from each processed query in accordance with the quality it served. The profit is the ratio between the attained revenue and the query processing cost. With the quality functions of the queries, QAS tries to find the best qualities and processing order of the queries to get the maximum profit for the underlying scheduling protocol. This profit is only the highest one for the current underlying protocol, but may not be the highest for others due to the efficiency differences among various scheduling protocols.

2. BACKGROUND AND RELATED WORK

In this section, we define QoS, QoD, and system profit, and discuss related work. The symbols (excluding those commonly used, e.g., U - voltage, the temporary variables, and those in the algorithm) used throughout this paper is summarized in Table 1.

2.1 QoS

In this paper, QoS refers to response time and/or query lifetime in query processing. The response time is the period from the start time of query processing to the time when all of the nodes have reported their query results in one epoch (sample interval). The query lifetime is described using the number of epochs from the time of query injection to the time when the query stops running. The lifetime requirement of a snapshot query is 1, since a snapshot query needs only one epoch of processing. The reason of including these two performance metrics in QoS is as follows: some queries desire short response time, e.g., event monitoring queries, some queries prefer long query lifetime, e.g., data collection queries, and some queries may require both short response time and long life time, e.g., queries in factory or health monitoring applications.

2.2 QoD

In QAS, QoD is defined by Equation (1). In this equation, D is the number of query results received by the sink of a

Table 1: Summary of symbols used

Symbol	Definition
D	Number of query results received by the sink
A	Number of all query results generated in a WSN
$D(i)$	Number of transmitted query results in hop i
$A(i)$	Number of all results should be generated in hop i
D_Q	Data quality on a node
H	Maximum number of hops
I_h	Electric current in hibernating (sleeping)
I_p	Electric current in computation
I_r	Electric current in receiving
I_t	Electric current in transmitting
L	Query lifetime
N_c	Average number of children
N_d	Average number of descendants
$N(i)$	Number of nodes in hop i
r	Ratio of parallel transmission among all nodes
T_a	Aggregating time on a node
T_c	Communication time on a node
T_h	Hibernating time
$T(j)$	Execution time of the j-th operator in a query
T_{Np}	Network processing time of a query
T_{Nc}	Network communication time of a query
T_{Nq}	Network query evaluation time of a query
T_o	Overlap between the operations on a node
T_q	Query evaluation time on a node
T_r	Time for receiving data from the children of a node
t_s	Length of a time slot

WSN, while A is the number of all query results generated in the WSN. If A is 0, i.e., there is no satisfying query result in the network, then $\frac{D}{A} = 1$.

Note that $\frac{D}{A}$ alone does not describe the data quality of a network well: A WSN may want to return query results from nodes that are closer to the sink node as much as possible to save energy. In this scenario, the value of $\frac{D}{A}$ will still be high but the data quality under such scenario may be low in effect, since there are few results from the nodes that are far from the sink. To avoid such a problem, Equation (1) uses the average weighted difference between $\frac{D}{A}$ and $\frac{D(i)}{A(i)}$ of each hop i", $\rho(\frac{D}{A} - \frac{D(i)}{A(i)})$, as a punishment. Here $D(i)$ is the number of transmitted query results in hop i, H is the maximum number of hops, and $A(i)$ is the number of all results that should be generated in hop i. Similarly, if $A(i)$ is 0, which indicates that there is no satisfying query result generated in hop i, then $\frac{D(i)}{A(i)} = 1$.

$$QoD:\ D_Q = \frac{D}{A} - \frac{1}{H}\zeta\sum_{i=1}^{H}(\frac{D}{A} - \frac{D(i)}{A(i)}) \qquad (1)$$

In Equation (1), ζ is a user specified weight ($0 < \zeta \leq 1$ and $\zeta = 0$ when $\frac{D}{A} \leq \frac{D(i)}{A(i)}$). A larger ζ indicates that a user expects the received source data to be more evenly distributed among hops.

2.3 System Profit

As mentioned, the system profit is the ratio of award and the cost. Due to the severe limitations of hardware and energy resources in WSNs, the energy cost is critical for a sensor network to attain the highest award during its lifetime. Hence, we define the system profit gained from a query as Equation 2, where φ is the price of battery energy, with a unit of dollar per joule ($\$/joule$). The problem for QAS is to maximize the profit of the queries processed.

$$Profit = \frac{AwardtoQoS + AwardtoQoD}{\varphi \cdot cost} \qquad (2)$$

2.4 Related Work

Previous work on scheduling in WSNs [4], [7], [17], [20] for both computation and communication are tightly related to ours, as QAS is designed to work on top of them. FPS [7], SS [17] and DCS [20] are the representative scheduling protocols. These protocols can either be directly used or be adopted for query processing. There are also some other schemes for event detection [4] or data collection that needs only one time wake-up per epoch [10]. These protocols allocate slots so as to reduce wireless competition and the idle listening periods. The problem with them is that they are not quality-aware and thus are not applicable for queries with quality requirement.

Although QoS and QoD scheduling are well studied in traditional database [1], [2], [5], [6], [12], [14], there are few studies on QoS and QoD in query scheduling for wireless sensor networks yet. However, these studies of QoS and QoD in traditional databases share a common goal of profit maximization with our work.

Qu *et al.* proposed the concept of quality contract to integrate QoS and QoD metrics [14]. With quality contract, the scheduling scheme is able to perform the tradeoff between QoD and QoS to maximize the total system profit. We adopted the concept of quality contract in designing our QAS. In Borealis [1], Abadi *et al.* proposed a QoS model that aggregates multiple metrics with different weights to be a single metric for evaluating the QoS. We adopt this concept of total system profits in QAS.

There are extensive studies on priority (or value) based and deadline oriented scheduling in the database community. For instance, Haritsa *et al.* proposed Value over Relative Deadline (VRD) to enable the queries whose deadlines are closer to the current time to be executed earlier [6]. The purpose is to complete as many queries as possible. These deadline and value based scheduling methods are mainly used in real-time databases [8], [12], but they are also helpful in query scheduling of WSNs where queries post fixed deadlines for result reporting.

Recently, there are initial investigations in query processing quality of WSNs. For instance, Amirijoo et al. defined the sensory data quality as the length of the sample interval in continuous data collection [3]. The authors proposed mechanisms to lengthen the WSN lifetime by dynamically adjusting the sampling period. However, their definition is not applicable to continuous queries with sample interval specified already [11]. Yates et al. defined the data quality as the normalized delay and proposed an approximation approach to reduce the delay [22]. These essentially focused on the query processing delay (QoS) as defined in this paper.

There are also two representative studies [15], [13], on the quality of data (QoD) similar to the QoD defined in this paper. Among these, Ren et al. proposed an algorithm to select the most related nodes as the active nodes to answer queries in a WSN to reduce the energy consumption of nodes without undermining the data quality much. Peng et al. elaborated the assessment models of data quality in in-network data processing. Their methods are quite helpful to improve the admission control and the active node selection in QAS, although they did not consider query and communication scheduling. We are planning to adopt and extend these methods as our future work.

To the best of our knowledge, QAS is the first work that considers both QoS and QoD in scheduling of WSNs. Next, we will show how QAS improves the QoS and QoD while observing energy efficiency.

3. OVERVIEW

Before coming to the details of the cost model in QAS, we present the overview of QAS to show its general idea. Fig. 1 illustrates the architecture of QAS.

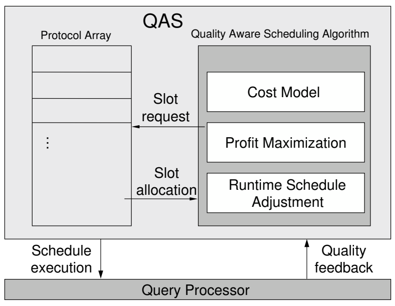

Figure 1: QAS architecture

On each node, the protocol array adopts a set of existing scheduling protocols to schedule the query operators and communications of each query [7], [17], [4], [20]. For a scheduling protocol to be loaded to the array, we design a uniform interface that allows QAS to specify the number of transmission, receiving and query execution slots for each query. This way, QAS is able to control the target quality of each query to be scheduled by the protocols. The underlying protocol to run is system specified before query processing.

The quality-aware scheduler determines the quality and execution order of queries for the underlying protocol using a cost model so that the WSN attains the total maximum profit. To avoid the scenario in which the target qualities of nodes differ much due to this distributed quality determination, it uses the same set of network parameters (see Section 4) in the cost model on nodes. Extending our model to allow nodes to have different parameter settings is a challenging direction of future work. The details of the cost model will be presented in the next section.

4. COST MODEL

The cost model in QAS estimates the network processing time and the energy consumption of a single query. The motivation of this model is to express the relation between the QoS, QoD, time and energy costs, and the system profit of a query, so that QAS is able to determine the target QoS and QoD to get the maximum system profit.

4.1 Network Processing Time

The network processing time, T_{Np}, is divided into query evaluation time, T_{Nq}, and communication time, T_{Nc}. The network query evaluation time of a WSN, T_{Nq}, is called network query evaluation time, which is the accumulated time of non-overlapping evaluation time on the nodes of the WSN. Similarly, T_{Nc} of a WSN is the accumulated time of non-overlapped communications on all nodes of the WSN. By definition, the time when a node transmits or receives messages in the midst of query evaluation is considered in T_{Nc}, not T_{Nq}.

T_{Nq}**:** We first investigate the query evaluation time on a single node, T_q so that we can calculate T_{Nq} with T_q. T_q is calculated as $T_q = D_Q \sum_{j=1}^{n} T(j) - T_o$, where D_Q is the QoD to be provided by this node, $T(j)$ is the execution time of the j-th operator and T_o is the overlapping time between the query execution and the communication on the node.

In a WSN, the nodes in upper hops are usually started at the same time or later than lower hop nodes [11], [4]. Therefore, the query evaluation time of the upper hop nodes and the query evaluation of the lower hop nodes will overlap, or the query evaluation time and the communication of other nodes will overlap. The non-overlapping network query evaluation time thus is roughly the same as that of a single node, i.e., $T_{Nq} \approx T_q$.

T_{Nc}**:** We estimate T_{Nc} by Equation (3). In (3), D (Directly forward) refers to a node that directly forwards the results from its children towards the sink; A (Aggregate) refers to a node that aggregates the results of its children first and then reports the aggregated result. The equations in the remainder of this paper use the same denotations. In the two formulas for $forward$ and $aggregate$, r is the ratio of parallel transmission among nodes in the network. r is used to remove time of simultaneous transmissions on the nodes. $N(i)$ is the number of nodes in hop i and H is the maximum number of hops in the network. t_s is the length of a slot. D_Q is the QoD. The reason of using D_Q in the $forward$ formula is that D_Q determines the number of packets to be forwarded on each node. N_c is the average number of children and T_a is the average time for aggregating a result. r, $N(i)$, H, N_c, and T_a are called the *network parameters*, which are managed by the sink and disseminated to the other nodes before query processing.

$$T_{Nc} = \begin{cases} (1-r)D_q t_s \sum_{i=1}^{H} iN(i), & D, \\ N_c T_a + \sum_{i=1}^{H} N(i), & A. \end{cases} \tag{3}$$

LEMMA 1. *Given a WSN whose routing paths are fixed, the network processing time of a query, T_{Np}, is a linear function of data quality D_Q: $T_{Np} = \alpha D_Q + \beta$.*

PROOF. As described above, $T_{Np} = T_q + T_{Nc}$. From the equations of T_q and T_{Nc}, T_{Np} can be estimated as follows.

$$T_{Np} = D_Q \sum_{j=1}^{n} T(j) - T_o + \begin{cases} (1-r)D_q t_s \sum_{i=1}^{H} iN(i), & D, \\ N_c T_a + \sum_{i=1}^{H} N(i), & A. \end{cases} \tag{4}$$

In (4), once the network and the scheduler is fixed, the network parameters such as H, $N(i)$, t_s, and T_o are all constants. Denoting these constants using α and β as shown in Equations (5) and (6), we have $T_{Np} = \alpha D_Q + \beta$, hence the lemma follows.

$$\alpha = \sum_{j=1}^{n} T(j) + \begin{cases} (1-r) \sum_{i=1}^{H} iN(i)t_s, & D, \\ 0, & A. \end{cases} \tag{5}$$

$$\beta = -T_o + \begin{cases} 0, & D, \\ N_c T_a + \sum_{i=1}^{H} N(i), & A. \end{cases} \quad (6)$$

□

4.2 Energy Consumption

The average energy consumption for processing a query on a node in a WSN can be estimated from the node running time and electric current: $E = U \cdot I \cdot T$, where E is the energy consumption of a node within T length of time, during which the voltage and the electric current of the node are U and I, respectively. Considering the different operations in query processing, the energy consumption should be $E = U \sum I_j \cdot T(j) \cdot L$, where I_j and $T(j)$ are the electric current and the execution time of each type of the j-th operation per epoch, L is the total number of epochs in processing the query.

With the above analysis, the node energy consumption is modeled in Equation (7). In (7), T_q is the query evaluation time and T_o is the same as that in Section 4.1. I_p is the electric current of the computation for query evaluation. Usually, a node can turn off the radio chips to lower the electric current. T_r is the time for receiving the results from the children. I_r is the electric current of receiving on the node. T_c is the communication time as described in Equation (4). $T_c - T_r$ is the total transmitting time on the node. I_t is the electric current of transmitting. Finally, T_h and I_h are time and electric current in hibernating (sleeping), respectively.

$$E = (T_q I_p + T_r I_r + (T_c - T_r) I_t + T_h I_h) \cdot U \cdot L \quad (7)$$

In Equation (4), the electric current of transmission in each epoch is assumed to be constant. This assumption is realistic because in the scheduling protocols and current query processing systems, the transmission power and the corresponding transmission range are all fixed in each epoch. In addition, the average retransmission time due to transmission failures is included in the transmitting time per epoch.

Given the processing time on each node, the energy consumption can be expressed as $E = (\lambda D_Q + \delta)L$. Here we omit the procedure of computing λ and δ, and list them in Equation (8) and (9). N_d in Equation (8) is the average number of descendants of each node in the WSN. It is used to calculate the receiving and forwarding slots for the descendants. The detailed procedure of computing λ and δ can be referred to our QAS technical report [19].

$$\lambda = U I_p \sum_{j=1}^{n} T(j) + U \cdot \begin{cases} N_d t_s I_r + (N_d + 1) t_s I_t, & D, \\ 0, & A. \end{cases} \quad (8)$$

$$\delta = -T_o U \cdot I_q + U \cdot \begin{cases} 0, & D, \\ N_c T_a I_p + N_c t_s I_r + t_s I_t, & A. \end{cases} \quad (9)$$

5. QUALITY AWARE SCHEDULING

Given the cost model, QAS uses Equation (10) to calculate the profit. In this equation, $k_1, k_2, b_1, b_2, k_3, b_3$ are the coefficients in the quality functions. With this equation, QAS uses the partial derivatives of P, such as $\frac{\partial^2 P}{\partial^2 D_Q}$, $\frac{\partial^2 P}{\partial D_Q \partial L}$, and $\frac{\partial^2 P}{\partial^2 L}$ to calculate the target qualities of queries that enable the nodes to get the maximum system profit.

$$P = \frac{k_1 L + b_1 + k_2(\alpha D_Q + \beta) + b_2 + k_3 D_Q + b_3}{(\lambda D_Q + \delta) L} \quad (10)$$

Algorithm 1 shows the steps of the determination of the target qualities and the execution order of queries and then the process of calling scheduling protocol in QAS. Note that Algorithm 1 is run on each node instead of on the sink to reduce the communication overhead, since the communication is more costly than computation.

In this algorithm, the first segment (Lines 1 - 6) uses the partial derivatives of the cost model to calculate the target QoS and QoD, $tQoS[i]$ and $tQoD[i]$ for the i-th query. The system profit would be $tP[i]$, if the target qualities $tQoS[i]$ and $tQoD[i]$ were realized, as shown in Line 3. Then Lines 4 -5 calculate the number of receiving slots needed, $sr[i]$, and transmission slots, $st[i]$, using the target quality $tQoS[i]$ and $tQoD[i]$. In these two lines, $Nr[i]$ and $Nt[i]$ are the number of needed receiving and transmission slots if $tQoD[i] = 1$, i.e., the number of receiving and transmission slots that the node would allocate previously without QAS.

Algorithm 1 Scheduling for Profit Maximization

```
Input  n queries with quality functions;
Output  Query execution order of these queries and the
    target QoS and QoD for each query; schedule of each
    query;
 1: for $i = 1$ to $n$ do
 2:     find $tQoS[i], tQoD[i]$ for query $i$;
 3:     compute the profit, $tP[i]$ of the i-th query under
        $tQoS[i], tQoD[i]$;
 4:     $sr[i] = Nr[i] \cdot tQoD[i]$;
 5:     $st[i] = Nt[i] \cdot tQoD[i]$;
 6: end for
 7: for $j = 1$ to NUM_STRATEGIES do
 8:     sort the queries in the descending order of their
        weights, the order is denoted as $O[j]$;
 9:     move the query in $O[j]$ whose quality requirement can-
        not be satisfied into the waiting list $w[j]$;
10:     $P[j]$ = the sum of system profit of queries in $O[j]$;
11: end for
12: find the execution order, $O[k]$, that has the highest profit
    ($1 \leq k \leq NUM\_STRATEGIES$);
13: send $O[k]$ and the $sr, st$ of the queries in $O[k]$ to the
    underlying scheduling protocol to build up the schedule
    for each query in $O[k]$;
```

The second segment (Lines 7 - 13) optimizes the query execution order of multiple queries to be scheduled and processed to get the maximum total system profit. This algorithm uses different weights to sort the queries and can get a number of orders of query execution of the set of queries as shown in Lines 5 - 9. QAS currently uses two strategies (NUM_STRATEGIES = 2) to define the weights: (1) maxP[i] and (2) maxP[i] / tQoS[i]. We found in our experiments that in most occasions, these two types of weights allowed QAS to get higher profits than a single type.

6. EVALUATION

6.1 Experiment Setup

We ran the prototype of QAS on four scheduling protocols of WSNs, DCS [20], AHS [18], FPS [7], and SS [17]. We compared QAS with the following quality-aware scheduling schemes applied on the protocols: (1) Low-quality(low), which always serves the lowest acceptable QoD and life time of a query; (2) Medium-quality(Mid), which serves the medium level QoD and life time within the range of QoD and lifetime requests; (3) Random-quality(Rand), which randomly chooses the target quality from the range of QoD and lifetime; and (4) High-quality (High), which always serves the highest QoD and life time of a query.

We evaluated the performance of the scheduling schemes through simulation at this stage and take the experiments on real sensor motes as future work. In the simulation experiments, the sensory data for the queries were synthetic data, since there were no available sensory dataset for up to 100 nodes. Hence, we used the source dataset from Intel lab [9] and expanded it to up to 100 nodes using a data generation tool [16]. We fixed the WSN to be one with randomly deployed 100 nodes (including the sink) in an 100 meter * 100 meter area, in which there was at least one route from each node to the sink node. The transmission range is 25 meters (a MICA series sensor mote can reach this distance when the transmission current is about 22 mA). WSNs with such a configuration are widely used in the sensor networking studies [4], [7], [11], [20], [21]. We used the following query Q1 as the workload for scheduling:

Q1: SELECT nodeid, light
FROM sensors
WHERE light $< \lambda$
SAMPLE INTERVAL 60s

6.2 Experiment Results

In the experiments, we found the following major factors affecting the system profits from processing queries in a WSN: (1) scheduling protocol, (2) quality functions, and (3) query execution order.

Scheduling Protocol. We compared QAS on the four scheduling protocols for WSNs using Q1 (see Section 2). The query predicate is designed to make the selectivity to be about 70%. A query with a selectivity lower than 100% is the common case in both data collection and monitoring applications. Here we choose a little higher selectivity (70%) to make the protocols work in a relatively high communication traffic network to investigate their performance under a heavy workload. The quality function is as follows: $k1 = b1 = 0; k2 = -10, b2 = 100; k3 = 333.33, b3 = -133.33$. Such a quality function specifies that, for a network to attain profits, the minimum quality it should serve is as follows: the shortest query lifetime is 1 epoch, longest response time is 10s and the lowest data quality is 0.4. Such requirements are common in real world applications.

The results of the four schemes are shown in Fig. 2. As shown in this figure, different underlying scheduling protocols achieve different system profits with a given quality-aware scheme. Some scheduling protocols combined with a quality-aware scheme may get a negative profit, e.g., FPS with Random. No matter what underlying scheduling protocols used, QAS outperformed the other three quality-aware schemes. The High scheme was close to QAS on AHS, due to the highest profit was achieved near the high end of the range of the qualities. Overall, AHS achieves the highest system profits across all schemes. In the following experiments, we use AHS as the underlying scheduling protocol.

Quality Function. A network running the same scheduling scheme may get different system profits from a query given different quality functions. Since QAS allows a user to specify the quality function of each query, the effect of quality function on system profits should be studied.

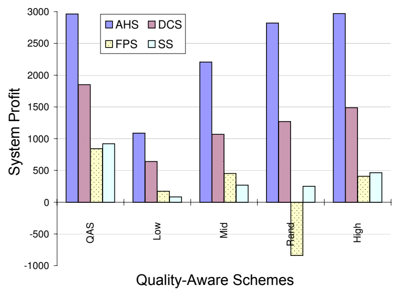

Figure 2: Different scheduling protocols

Fig. 2 only shows the system profit from executing queries with *standard* quality functions. A standard quality function for a query refers to a function in which the revenue is proportional to the cost of the query. We also tested Q1 with two non-standard quality functions to evaluate QAS as shown in Table 2. These two non-standard quality functions are used to favor certain performance metrics. For instance, Function A in this table desires a high QoD and Function B prefers longer lifetime (at least 1000 epochs).

Table 2: Non-standard Quality Functions for Q1

Function	k1	b1	k2	b2	k3	b3
A	0	0	-1.667	100	1000	-800
B	1	-1000	-60	600	166.667	-66.667

The attained system profits are shown in Figure 3. The results show that, QAS achieved the highest profit for both non-standard quality functions. Especially for Function B, QAS was able to avoid negative profit.

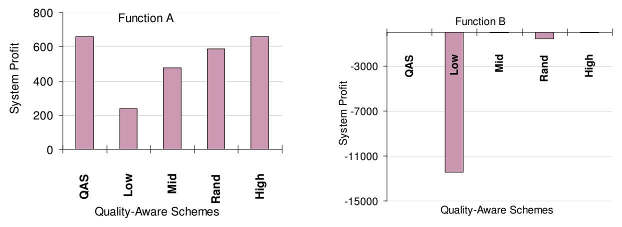

Figure 3: Q1 with different quality functions

Query Execution Order. We now show the comparison of three strategies, the algorithm in QAS, a greedy algorithm, and an exhaustive searching algorithm on scheduling of multiple queries. The greedy algorithm always arranges the queries in descending order of system profits. The exhaustive algorithm traverses all of the permutations of queries and arranges the queries in the order that will get the highest total system profits.

We tested a group of five queries, where the required query life time and QoD of each were 1 and 0.4, respectively. The acceptable response time of the queries were 30s ($Q1_3$), 40s ($Q1_4$), 10s ($Q1_1$), 50s ($Q1_5$), 20s ($Q1_2$) (The query injection order was: $Q1_3$, $Q1_4$, $Q1_1$, $Q1_5$, $Q1_2$). We used different quality functions of the queries, so that different execution orders of queries would not get the same system profits.

As shown in Figure 4(a), when the quality functions spec-

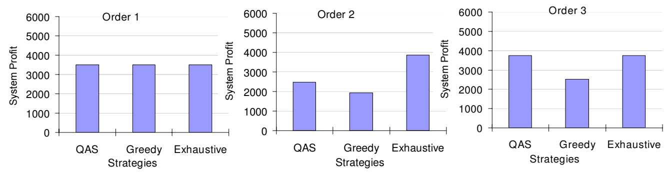

Figure 4: System Profits of the three strategies

ified the revenues in a way that made the order of possible highest system profit (PHSP) as $Q1_1 > Q1_2 > Q1_3 > Q1_4 > Q1_5$, then the three strategies got the same total system profits. The PHSP is the highest system profit that can be attained from processing a query given a WSN. We found that the strategies determined the same query execution order and it is the optimal query execution order.

However, when the quality functions did not enable the PHSP of $Q1_1$ to be the largest one, the execution orders given by the strategies were much different. As demonstrated in Figure 4, in general the performance of QAS was better than the greedy algorithm and close to the exhaustive algorithm. In the worst case, QAS attained the same system profit as that of the greedy algorithm.

Finally, we evaluated the network lifetime. In this paper, we define the network lifetime as the time from a network starts to the time that the first node runs out of energy. Due to space constraints, we omit the performance figures on network time. In general, QAS enabled a much longer network lifetime than the other schemes. Interested readers are referred to our technical report.

7. CONCLUSION

In this paper, we presented a quality aware scheduling framework, QAS, which efficiently satisfies the user quality requirements on queries. QAS runs on existing quality-unaware scheduling protocols of WSNs and enables users to specify their quality requirements using quality functions. Given these quality functions, QAS determines the target quality of each query, at which the ratio between the revenue and the energy cost is maximal.

QAS effectively solves the energy waste problem in sensor query processing that causes high quality requirement queries to be unsatisfied but low quality requirement queries to be overly satisfied. As shown in the experimental results, QAS outperforms the baseline quality scheduling strategies.

Acknowledgments

This work was supported by grant 617307 from the Hong Kong Research Grants Council and by NSF Career award #IIS 0746696.

8. REFERENCES

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. B. Zdonik, "The design of the borealis stream processing engine," in *CIDR*, 2005.

[2] R. K. Abbott and H. Garcia-Molina, "Scheduling real-time transactions: A performance evaluation," *ACM Trans. Database Syst.*, vol. 17, no. 3, pp. 513–560, 1992.

[3] M. Amirijoo, S. Son, and J. Hansson, "Qod adaptation for achieving lifetime predictability of wsn nodes communicating over satellite links," in *INSS*, June 2007.

[4] Q. Cao, T. F. Abdelzaher, T. He, and J. A. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *IPSN*, 2005.

[5] H.-R. Chen and Y.-H. Chin, "An adaptive scheduler for distributed real-time database systems," *Inf. Sci.*, vol. 153, pp. 55–83, 2003.

[6] J. R. Haritsa, M. J. Carey, and M. Livny, "Value-based scheduling in real-time database systems," *VLDB J.*, vol. 2, no. 2, pp. 117–152, 1993.

[7] B. Hohlt, L. Doherty, and E. A. Brewer, "Flexible power scheduling for sensor networks," in *IPSN*, 2004.

[8] D. Hong, T. Johnson, and S. Chakravarthy, "Real-time transaction scheduling: A cost conscious approach," in *SIGMOD*, 1993.

[9] IntelLabData, "http://berkeley.intel-research.net/labdata."

[10] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *INFOCOM*. IEEE, 2005.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[12] H. Pang, M. J. Carey, and M. Livny, "Multiclass query scheduling in real-time database systems," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 4, pp. 533–551, 1995.

[13] L. Peng and K. S. Candan, "Data-quality guided load shedding for expensive in-network data processing," in *ICDE*, 2007.

[14] H. Qu and A. Labrinidis, "Preference-aware query and update scheduling in web-databases," in *ICDE*, 2007, pp. 356–365.

[15] Q. Ren and Q. Liang, "Energy and quality aware query processing in wireless sensor database systems," *Inf. Sci.*, vol. 177, no. 10, pp. 2188–2205, 2007.

[16] SDGEN, "http://www.cse.ust.hk/catalac/."

[17] M. L. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," in *INFOCOM*, 2004.

[18] H. Wu and Q. Luo, "Adaptive holistic scheduling for query processing in sensor networks," *HKUST-CSE Technical Report, http://www.cse.ust.hk/tahoe*, April, 2007.

[19] H. Wu, Q. Luo, J. Li, and A. Labrinidis, "Quality aware scheduling for query processing in wireless sensor networks," *HKUST-CSE Technical Report, http://www.cse.ust.hk/tahoe*, January, 2009.

[20] H. Wu, Q. Luo, and W. Xue, "Distributed cross-layer scheduling for in-network sensor query processing," in *PerCom*. IEEE Computer Society, 2006, pp. 180–189.

[21] Y. Yao and J. Gehrke, "Query processing in sensor networks," in *CIDR*, 2003.

[22] D. Yates, E. Nahum, J. Kurose, and P. Shenoy, "Data quality and query cost in wireless sensor networks," in *Pervasive Computing and Communications Workshops*, March 2007.