2009 Third International Conference on Sensor Technologies and Applications

Energy-Efficient Multiple Query Optimization for Wireless Sensor Networks

Yu Won Lee, Ki Yong Lee, Myoung Ho Kim
Department of Computer Science, KAIST,
373-1 Guseong-Dong, Yuseong-Gu, Daejeon 305-701, Republic of Korea
{*ywlee, kylee, mhkim*}*@dbserver.kaist.ac.kr*

Abstract

A wireless sensor network is a collection of sensor nodes distributed over a geographic area to monitor physical conditions, such as temperature, light, humidity, or pressure. Since a sensor node has limited energy supply, running multiple continuous monitoring queries may impose significant overhead on the sensor network. In this paper, we propose an energy-efficient multiple query optimization method to reduce the number of monitoring queries that are running in the sensor network. When a new monitoring query is submitted to the base station, the proposed method checks whether the new query can be evaluated using results of currently running queries. If so, the proposed method rewrites the new query using currently running queries and then evaluates the rewritten query at the base station without injecting it into the sensor network. Consequently, the amount of data transmitted from sensor nodes to the base station is reduced, resulting in lower energy consumption. Through experiments, we show the effectiveness of the proposed method.

Index Terms

Wireless Sensor Networks, Query Optimization, Energy-Efficient

1. Introduction

A wireless sensor network is a collection of sensor nodes distributed over a geographical area to monitor physical conditions, such as temperature, light, humidity, or pressure. In many applications, a sensor network is viewed as a relational database which provides query services on sensor data [2] [3]. For example, in TinyDB [2], sensor readings produced by sensor nodes logically belong to a table called *sensors*. Each row of the *sensors* table contains the node id and readings of a sensor node, as illustrated in Fig. 1(a). When a user submits a continuous monitoring query like the one in Fig. 1(b) to the base station, the query is injected into the sensor network and then each sensor node periodically sends its readings towards the base station if its readings satisfies the condition of the query. The *sample period* of a query specifies the time interval between the re-evaluation of the query.

nodeid	temperature	light	humidity
1	28	221	129
2	36	237	216
…	…	…	…

(a) the sensor table

```
SELECT nodeid, temperature
FROM sensors
WHERE temperature > 35
SAMPLE PERIOD 4s
```

(b) a monitoring query

Figure 1. The *sensors* table and a monitoring query

For each sample period, the base station collects data relevant to the query from the sensor network and returns the final results to the user. Fig. 1(b) shows an example of a continuous monitoring query that returns the node ids and temperature readings of sensor nodes every 4 seconds whose temperature readings are greater than 35°C.

As the number of monitoring queries that are running in the sensor network increases, the energy consumption of the sensor network increases. This is because, for each query, each sensor node has to send its reading towards the base station whenever its reading satisfies the condition of the query. Thus, in order to reduce the energy consumption of the sensor network, it is crucial to reduce the number of monitoring queries injected into the sensor network.

In practice, monitoring queries often have similar expressions among them. Instead of running each query independantly, if we reuse the results of queries to answer other queries, the overall energy consumption of the sensor network can be reduced because duplicate data requests can be eliminated. For example, suppose that two monitoring queries q_1 and q_2 in Fig. 2(a) are currently running in a sensor network. If a new query q_{new} in Fig. 2(a) is submitted to the base station, we can answer q_{new} using the results of q_1 and q_2 without injecting q_{new} into the sensor network. In other words, we can rewrite q_{new} using q_1 and q_2 to reuse the results of those two queries. Fig. 2(b) shows an example where q_{new} is rewritten using q_1 and q_2 into q'_{new}. Here, q'_{new} does not need to be injected into the sensor network and can be evaluated at the base station using the results of q_1 and q_2. Since the base station has abundant computing resources and has no energy constraint, this reduces the overall energy consumption of the sensor network.

978-0-7695-3669-9/09 $25.00 © 2009 IEEE
DOI 10.1109/SENSORCOMM.2009.89

IEEE computer society

```
q1: SELECT nodeid, temp
    FROM sensors
    WHERE temp > 35
    SAMPLE PERIOD 2s

q2: SELECT nodeid, light
    FROM sensors
    WHERE light > 200
    SAMPLE PERIOD 4s

qnew: SELECT nodeid, temp, light
      FROM sensors
      WHERE temp > 50 AND light > 300
      SAMPLE PERIOD 8s
```

(a) example queries

```
q'new: SELECT nodeid, temp, light
       FROM q1, q2
       WHERE q1.nodeid = q2.nodeid AND temp > 50 AND light > 300
       SAMPLE PERIOD 8s
```

(b) a rewritten query of q_{new}

Figure 2. query rewriting example of q_{new}

Until now, most existing work on query processing in sensor networks has focused on the optimization and execution of a single long-running query [4]. Although the query rewriting problem and multiple query optimization have been extensively studied in the database literature [5] [6] [7], these studies cannot be directly applied to the sensor network as will be described in Section 2. In this paper, we propose an energy-efficient multiple query optimization method for wireless sensor networks to reduce the number of monitoring queries injected into the sensor network. When a new query is submitted to the base station, the proposed method checks whether the new query can be rewritten using currently running queries. If so, the proposed method uses a sophisticated query rewriting algorithm to rewrite the new query using currently running queries. The rewritten query is then evaluated at the base station using the results of currently running queries without being injected into the sensor network. Consequently, the number of queries injected into the sensor network is reduced, resulting in lower energy consumption. Since query rewriting is done at the base station, the proposed method is not affected by network conditions such as network topology, network size, or routing protocols. Our experimental results based on the TOSSIM simulator [9] show that the proposed method can significantly reduce the amount of data transmitted from sensor nodes to the base station, which leads to lower energy consumption.

The rest of the paper is organized as follows. The related work is described in Section 2. We define some notations and present the problem description in Section 3. In Section 4, we describe the proposed method in detail. Section 5 presents the results of our performance evaluation. Finally, we conclude our work in Section 6.

2. Related Work

For energy-efficient query processing in sensor networks, a lot of work has been done on various aspects of sensor networks. [10] has proposed energy-efficient routing protocols for query processing in sensor networks. [11] has investigated in-network query processing that pushes the processing task deeper into the network to reduce communication cost. [12] has studied approximated query processing to use as little information from sensor nodes as possible to answer user queries. However, these researches have mainly focused on the optimization and execution of a single long-running query.

Traditionally, the multiple query optimization and query rewriting problem have been extensively studied in the database literature [5] [6] [7]. However, these studies cannot be directly applied to the sensor network due to the following reasons. First, the main goal of the traditional multiple query optimization is to speed up query processing. Thus, query results may not be reused if reusing them slows down query processing. On the other hand, the main goal of multiple query optimization in sensor networks is to reduce energy consumption. Thus, query results should be reused as much as possible to reduce the amount of data transmissions, even if they may slow down query processing. In this regard, the proposed method uses a more complex and sophisticated query rewriting algorithm than the traditional ones. Second, a monitoring query in a sensor network is specified with a *sample period*, after which the query is re-evaluated. However, the traditional multiple query optimization and query rewriting problem do not consider the sample periods of monitoring queries.

There has also been some work for multiple query optimization for streaming sensor data. The Fjords architecture proposed in [13] manages multiple queries on streaming data received from sensor proxies, which serve as the interface between sensor nodes and the query processor. However, it focuses on efficient processing of data streams and does not consider the problem of query rewriting.

Recently, there have been a few studies on multiple query optimization in wireless sensor networks. [14] has proposed an algorithm for optimizing multiple aggregation queries in sensor networks. However, their work focuses on *region-based* aggregation queries and cannot be applied to other types of queries. [4] [15] have proposed multiple query optimization for sensor networks that considers other types of queries, e.g., value-based queries. Their techniques are based on *query merging*, which merges similar queries into a single, more general query to reduce the number of queries injected into the sensor network. However, merged queries may require more sensor data, or may have to be evaluated

more frequently, than the original queries in order to be used to answer all the original queries.

3. Problem Description and Notations

In this paper, we assume that a table *sensors* contains only the most recent reading of each sensor node. Each row of the *sensors* table contains the node id and the most recent reading of a sensor node. The *sensors* table is updated whenever new readings become available. The *sensors* table has the following schema:

$$sensors\_schema \ = \ (nodeid, A_1, A_2, ..., A_s) \qquad (3.1)$$

where $nodeid$ is the node id of a sensor node and A_1, A_2, ..., A_s are sensor attributes, e.g., temperature, light, humidity, etc. Note that $nodeid$ is the primary key of the *sensor* table. Let Q = $\{q_1, q_2, ..., q_N\}$ be the set of monitoring queries that are currently running in the sensor network. In this paper, we consider the simple select-project query, which is one of the most frequently used types of monitoring queries in sensor networks [2]. A select-project query q is expressed in the relational algebra as follows:

$$q = \Pi_l(\sigma_c(sensors)) \qquad (3.2)$$

where l is a projection list and c is a selection condition. We assume that the sample period of q, denoted by $SP(q)$, is specified separately. Here, we assume that l contains $nodeid$ and c has the following form:

$$c \ = \ p_1 \wedge p_2 \wedge ... \wedge p_m \qquad (3.3)$$

where p_i ($1 \leq i \leq m$) is a simple predicate, which has the form:

$$p_i \ : \ A_j \ \theta \ Value \qquad (3.4)$$

where $A_j \in \{nodeid, A_1, A_2, ..., A_s\}$, $\theta \in \{=, <, \leq, >, \geq\}$, and $Value$ is chosen from the domain of A_j. For a query q = $\Pi_l(\sigma_c(sensors))$, $P(q)$ is the set of attributes that appear in l except $nodeid$ and $S(q)$ is the set of attributes that appear in c. Let $C(q)$ denote the selection condition of q and $C(q, A_i)$ denote the selection condition of q consisting of only those predicates that involve A_i. For example, when $C(q)$ = '$(light > 200) \wedge (temp < 25)$', $C(q, light)$ = '$light > 200$' and $C(q, temp)$ = '$temp < 25$'.

Let q_{new} be a new monitoring query submitted to the base station. Our goal is to rewrite q_{new} using q_1, q_2, ..., q_N. If such rewriting is possible, we do not inject q_{new} into the sensor network and evaluate it at the base station using the results of q_1, q_2, ..., q_N. Consequently, the energy consumption of the sensor network is reduced.

4. The Proposed Method

In this section, we describe the proposed query rewriting method in detail. Our proposed query rewriting method consists of the following four steps: (1) query decomposition, (2) candidate query set construction, (3) query rewritability test, and (4) query transformation. In the following subsections, we describe each step in detail.

4.1. Query decomposition

Suppose that a new query q_{new} is submitted to the base station, where $P(q_{new}) \cup S(q_{new})$ = $\{a_1, a_2, ..., a_n\}$. Then, we decompose q_{new} into d_{a_1}, d_{a_2}, ..., d_{a_n} as follows:

$$q_{new} = d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n} \qquad (4.1)$$

where

$$d_{a_i} = \begin{cases} \Pi_{nodeid,\ a_i}(\sigma_{C(q_{new},a_i)}(sensors)), \\ \qquad\qquad \text{if } a_i \in P(q_{new}) \\ \Pi_{nodeid}(\sigma_{C(q_{new},a_i)}(sensors)), \\ \qquad\qquad \text{if } a_i \in S(q_{new}) - P(q_{new}) \end{cases} \qquad (4.2)$$

for $1 \leq i \leq n$. Since each d_{a_i} ($1 \leq i \leq n$) contains $nodeid$, which is the primary key of the *sensor* table, in its projection list, it is easy to verify that q_{new} can be reconstructed by taking the natural join of d_{a_1}, d_{a_2}, ..., d_{a_n} on $nodeid$. For example, consider q_{new} = $\Pi_{nodeid,temp}(\sigma_{(temp<25)\wedge(light<200)}(sensors))$, where $P(q_{new}) \cup S(q_{new})$ = $\{temp, light\}$. q_{new} can be decomposed into d_{temp} and d_{light} as follows:

$$\begin{aligned} q_{new} &= d_{temp} \bowtie d_{light} \\ d_{temp} &= \Pi_{nodeid,temp}(\sigma_{temp<25}(sensors)) \\ d_{light} &= \Pi_{nodeid}(\sigma_{light<200}(sensors)) \end{aligned}$$

In what follows, we will rewrite each d_{a_i} ($1 \leq i \leq n$) using queries in Q = $\{q_1, q_2, ..., q_N\}$.

4.2. Candidate query set construction

Obviously, not all queries in Q = $\{q_1, q_2, ..., q_N\}$ can be used to rewrite d_{a_1}, d_{a_2}, ..., d_{a_n}. In this subsection, we construct a candidate set of queries $Q' \subseteq Q$ that can be used to rewrite d_{a_1}, d_{a_2}, ..., d_{a_n} by filtering out queries that cannot be used to answer q_{new}. Initially, we set Q' to Q.

First, we remove from Q' those queries whose sample period is not a divisor of the sample period of q_{new}. Since q_{new} has to be re-evaluated for each sample period, those queries whose sample period is not a divisor of the sample period of q_{new} cannot be used to answer q_{new}. For example, if the sample period of q_{new} is 8 seconds, only those queries whose sample period is 1, 2, 4, or 8 seconds can be used to answer q_{new}. Second, we remove from Q' those

queries whose selection condition conflicts with the selection condition of q_{new}. That is, we remove q from Q' if $C(q) \wedge C(q_{new}) = false$. For example, if $C(q_{new})$ = '$light > 300$' and $C(q)$ = '$light < 200$', we remove q from Q' because $(light > 300) \wedge (light < 200) = false$.

After constructing a candidate query set Q', we construct Q'_{a_1}, Q'_{a_2}, ..., Q'_{a_n} from Q', where Q'_{a_i} $(1 \leq \text{i} \leq n)$ is defined as follows:

$$Q'_{a_i} = \{q \mid (q \in Q') \wedge (a_i \in P(q))\} \quad (4.3)$$

That is, Q'_{a_i} is the set of queries in Q' that contain a_i in the projection list. Then, we will rewrite each d_{a_i} using queries in Q'_{a_i} $(1 \leq i \leq n)$.

4.3. Query rewritability test

Before we rewrite d_{a_i} using queries in Q'_{a_i}, we test whether d_{a_i} can be rewritten using queries in Q'_{a_i} $(1 \leq i \leq n)$. For d_{a_i} to be rewritten using queries in Q'_{a_i} = {q_1, q_2, ..., q_t}, the following condition must hold:

$$C(q_{new}, a_i) \rightarrow C(q_1) \vee C(q_2) \vee ... \vee C(q_t) \quad (4.4)$$

That is, the query region of d_{a_i}, which is represented by $C(q_{new}, a_i)$, must be contained in the union of the query regions of queries in Q'_{a_i}, which is represented by $C(q_1) \vee C(q_2) \vee ... \vee C(q_t)$. If the above condition does not hold for any d_{a_i} $(1 \leq i \leq n)$, q_{new} cannot be rewritten using the currently running queries. In this case, we stop the rewriting process and inject q_{new} into the sensor network.

The problem of testing whether the above condition holds is equivalent to the *CNF-satisfiability problem* [16]. Since the CNF satisfiability problem is NP-complete, we use an heuristic algorithm, called *cube elimination*, proposed in [16]. Refer to [16] for more details.

4.4. Query transformation

If the condition in Section 4.3 holds for all d_{a_i} and Q'_{a_i} $(1 \leq i \leq n)$, we can rewrite each d_{a_i} using queries in Q'_{a_i}. We rewrite each d_{a_i} using queries in Q'_{a_i} as follows:

$$d_{a_i} = \begin{cases} \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid,\ a_i}(\sigma_{C(q_{new},a_i)}(q)), \\ \qquad \text{if } a_i \in P(q_{new}) \\ \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid}(\sigma_{C(q_{new},a_i)}(q)), \\ \qquad \text{if } a_i \in S(q_{new}) - P(q_{new}) \end{cases} \quad (4.5)$$

Recall that Q'_{a_i} is the set of queries that contain a_i in the projection list. Thus, by applying $C(q_{new}, a_i)$ to each $q \in Q'_{a_i}$ and taking the union of them, we can obtain d_{a_i}. After we rewrite each d_{a_i} using queries in Q'_{a_i}, we finally rewrite q_{new} as $q'_{new} = d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n}$. Then, q'_{new} is evaluated at the base station without being injected into the sensor network. Fig. 3 presents the proposed query rewriting method described so far.

Procedure QueryRewriting
Input
$Q = \{q_1, q_2, ..., q_N\}$: the set of currently running queries;
q_{new}: a new query;
Output
q'_{new}: a rewritten query of q_{new} defined over $q_1, q_2, ..., q_N$;
Begin
Let $P(q_{new}) \cup S(q_{new}) = \{a_1, a_2, ..., a_n\}$;

/* Candidate query set construction */
$Q' = Q$;
for each $q \in Q'$ do
  if $SP(q)$ is not a divisor of $SP(q_{new})$ then
    remove q from Q';
  else if $C(q) \wedge C(q_{new}) = false$ then
    remove q from Q';
for each a_i in $P(q_{new}) \cup S(q_{new})$ do
  $Q'_{a_i} = \{q \mid (q \in Q') \wedge (a_i \in P(q))\}$;

/* Query rewritability test */
for each Q'_{a_i} $(1 \leq i \leq n)$ do
  test whether $C(q_{new}, a_i) \rightarrow \vee_{q \in Q'_{a_i}} C(q)$ holds;
  if test fails then return;

/* Query transformation */
for each a_i in $P(q_{new})$ do
  $d_{a_i} = \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid,\ a_i}(\sigma_{C(q_{new},a_i)}(q))$;
for each a_i in $S(q_{new})$ - $P(q_{new})$ do
  $d_{a_i} = \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid}(\sigma_{C(q_{new},a_i)}(q))$;

/* Return result */
return $q'_{new} = d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n}$;
End

Figure 3. The proposed query rewriting method

4.5. Correctness of the proposed method

Theorem 1: Let q_{new} be a new query submitted to the base station. Let q'_{new} be a rewritten query of q_{new} by the proposed query rewriting method. Then, q_{new} and q'_{new} are equivalent, i.e., they produce the same result on all instances of the $sensors$ table.

Proof: See Appendix. □

4.6. Example

Assume that the *sensor* table has a schema ($nodeid$, $light$, $temp$) and the following four queries q_1, q_2, q_3, q_4 are

currently running in the sensor network:

$$q_1 = \Pi_{nodeid,light}(\sigma_{light \leq 200}(sensors)), SP(q_1) = 4s$$
$$q_2 = \Pi_{nodeid,light}(\sigma_{200 < light}(sensors)), SP(q_2) = 8s$$
$$q_3 = \Pi_{nodeid,temp}(\sigma_{25 < temp}(sensors)), SP(q_3) = 2s$$
$$q_4 = \Pi_{nodeid,temp}(\sigma_{35 < temp < 40}(sensors)), SP(q_4) = 5s$$

Suppose that the following new query q_{new} is submitted to the base station.

$$q_{new} = \Pi_{nodeid,light}(\sigma_{(150 < light < 250) \wedge (30 < temp)}(sensors)), SP(q_{new}) = 8s$$

Note that $P(q_{new}) \cup S(q_{new})$ = $\{light, temp\}$. First, we decompose q_{new} into d_{light} and d_{temp} as follows:

$$\begin{aligned} q_{new} &= d_{light} \bowtie d_{temp} \\ d_{light} &= \Pi_{nodeid,light}(\sigma_{150 < light < 250}(sensors)) \\ d_{temp} &= \Pi_{nodeid}(\sigma_{30 < temp}(sensors)) \end{aligned}$$

Next, we construct a candidate query set Q' from Q = {q_1, q_2, q_3, q_4}. Since $SP(q_4)$ = 5s is not a divisor of $SP(q_{new})$ = 8s, we filter out q_4 so that Q' becomes {q_1, q_2, q_3}. Then, we construct Q'_{light} and Q'_{temp} from Q' as Q'_{light} = {q_1, q_2} and Q'_{temp} = {q_3}, respectively. After that, we perform the query rewritability test for d_{light} and d_{temp} as follows:

$$\begin{aligned} d_{light} &: C(q_{new}, light) \rightarrow C(q_1) \vee C(q_2) \\ d_{temp} &: C(q_{new}, temp) \rightarrow C(q_3) \end{aligned}$$

Note that $C(q_{new}, light)$ = '$150 < light < 250$', $C(q_1)$ = '$light \leq 200$', $C(q_2)$ = '$200 < light$', $C(q_{new}, temp)$ = '$30 < temp$', and $C(q_3)$ = '$25 < temp$'. Since the above conditions hold for both d_{light} and d_{temp}, we can see that d_{light} and d_{temp} can be rewritten using queries in Q'_{light} and Q'_{temp}, respectively. We rewrite d_{light} and d_{temp} using queries in Q'_{lgiht} and Q'_{temp}, respectively, as follows:

$$\begin{aligned} d_{light} &= \Pi_{nodeid,light}(\sigma_{150 < light < 250}(q_1)) \\ &\quad \cup \Pi_{nodeid,light}(\sigma_{150 < light < 250}(q_2)) \\ d_{temp} &= \Pi_{nodeid}(\sigma_{30 < temp}(q_3)) \end{aligned}$$

Finally, we rewrite q_{new} using d_{light} and d_{temp} as follows:

$$\begin{aligned} q_{new} &= d_{light} \bowtie d_{temp} \\ &= (\Pi_{nodeid,light}(\sigma_{150 < light < 250}(q_1)) \\ &\quad \cup \Pi_{nodeid,light}(\sigma_{150 < light < 250}(q_2))) \\ &\quad \bowtie \Pi_{nodeid,temp}(\sigma_{30 < temp}(q_3)) \end{aligned}$$

5. Performance Evaluation

In this section, we evaluate the effectiveness of the proposed method. We have implemented the proposed method in TinyDB, which is one of the most famous query processing systems for sensor networks. We have simulated a sensor network using TOSSIM [9], an event-driven simulator for TinyOS-based sensor networks. In the experiment, we used real sensor data obtained from [17]. The data obtained from [17] contain about 2.3 million sensor readings collected from 54 Mica2Dot sensor nodes deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004. Each sensor reading has the information about date, time, temperature, humidity, light, voltage, etc.

With this setting, we have compared the performance of the following three methods for running multiple monitoring queries in the sensor network: (1) *NAIVE*: each query is independently executed in the sensor network. (2) *MERGE*: queries are optimized using the merge-based method [4] described in Section 2. (3) *QR+MERGE*: our proposed query rewriting method (QR) is used together with MERGE. In fact, MERGE can be used as a complementary method to QR. In this method, if a new query can be rewritten using currently running queries, the new query is rewritten by QR and evaluated at the base station. If not, the new query is optimized using MERGE. For the performance measure, we have counted the number of sensor readings transmitted from sensor nodes to the base station. For each query injected into the sensor network, each sensor node sends its sensor reading towards the base station whenever its sensor reading satisfies the condition of the query. Note that the energy consumption of the sensor network increases as the number of transmissions increases. Since the performance of our method does not depend on network conditions such as network topology, network size, or routing protocols, we have not considered these factors in the experiments.

In the experiments, we used two workloads, $QuerySet1$ and $QuerySet2$, as shown in Fig. 4. We submitted queries in the workloads, q_1, q_2, ..., q_8, one by one to the base station. $QuerySet1$ is a query set designed to evaluate the effectiveness of the proposed method when many queries can be rewritten using other queries. In $QuerySet1$, q_6, q_7, q_8 are rewritten using q_1, q_2, ..., q_5 by QR and q_5 is merged with q_1 by MERGE. q_2, q_3, and q_4 run independently in the sensor network. Note that all queries run independently in NAIVE. On the other hand, $QuerySet2$ is a query set where many queries can be merged but only a few queries can be rewritten using other queries. In $QuerySet2$, only q_5 is rewritten using q_1 and q_3 by QR and q_3, q_4, and q_7 are merged with q_1, q_2, and q_6, respectively, by MERGE. q_8 runs independently in the sensor network.

Fig. 5 and Fig. 6 show the number of sensor readings transmitted from sensor nodes to the base station when $QuerySet1$ and $QuerySet2$ are executed using the three methods, respectively. In each experiment, we have counted the number of sensor readings transmitted as the number of sensor readings produced increases. In the case of $QuerySet1$ (Fig. 5), QR+MERGE significantly reduced the number of transmissions compared to the other methods. When a total of 2,100,000 sensor readings were produced, 1,441,204 and 1,166,271 sensor readings were transmitted

$[QuerySet1]$
q_1 = $\Pi_{nodeid,temp}(sensors)$, $SP(q_1) = 32s$
q_2 = $\Pi_{nodeid,light,temp}(\sigma_{(5\leq nodeid\leq 20)\wedge(200\leq light\leq 800)}(sensors))$, $SP(q_2) = 8s$
q_3 = $\Pi_{nodeid,light}(\sigma_{(5\leq nodeid\leq 15)\wedge(250\leq light\leq 700)\wedge(20\leq temp\leq 30)}(sensors))$, $SP(q_3) = 8s$
q_4 = $\Pi_{nodeid,light,temp}(\sigma_{(0\leq nodeid< 15)\wedge(150\leq light\leq 950)}(sensors))$, $SP(q_4) = 16s$
q_5 = $\Pi_{nodeid,light,temp}(sensors)$, $SP(q_5) = 64s$
q_6 = $\Pi_{nodeid,temp}(\sigma_{20<temp<35}(sensors))$, $SP(q_6) = 32s$
q_7 = $\Pi_{nodeid,temp}(\sigma_{(0\leq nodeid<10)\wedge(temp<27)}(sensors))$, $SP(q_7) = 64s$
q_8 = $\Pi_{nodeid,light,temp}(\sigma_{(7<nodeid<14)\wedge(400\leq light\leq 850)\wedge(20<temp<30)}(sensors))$, $SP(q_8) = 16s$

$[QuerySet2]$
q_1 = $\Pi_{nodeid,light,temp}(\sigma_{(0\leq nodeid\leq 30)\wedge(50\leq light\leq 800)\wedge(22\leq temp\leq 29)}(sensors))$, $SP(q_1) = 4s$
q_2 = $\Pi_{nodeid,light,temp}(\sigma_{(150\leq light\leq 500)\wedge(23\leq temp\leq 32)}(sensors))$, $SP(q_2) = 32s$
q_3 = $\Pi_{nodeid,light,temp}(\sigma_{(10\leq nodeid\leq 20)\wedge(100\leq light\leq 750)\wedge(20\leq temp\leq 28)}(sensors))$, $SP(q_3) = 4s$
q_4 = $\Pi_{nodeid,light,temp}(\sigma_{(90\leq light\leq 600)\wedge(22\leq temp\leq 30)}(sensors))$, $SP(q_4) = 32s$
q_5 = $\Pi_{nodeid,light,temp}(\sigma_{(0<nodeid<10)\wedge(150\leq light\leq 700)\wedge(23\leq temp\leq 30)}(sensors))$, $SP(q_5) = 8s$
q_6 = $\Pi_{nodeid,light}(\sigma_{0\leq nodeid<7}(sensors))$, $SP(q_6) = 16s$
q_7 = $\Pi_{nodeid,temp}(\sigma_{(2\leq nodeid<10)\wedge(temp<29)}(sensors))$, $SP(q_7) = 32s$
q_8 = $\Pi_{nodeid,light}(\sigma_{(0\leq light\leq 900)\wedge(27\leq temp\leq 32)}(sensors))$, $SP(q_8) = 16s$

Figure 4. Two query sets used in the experiments

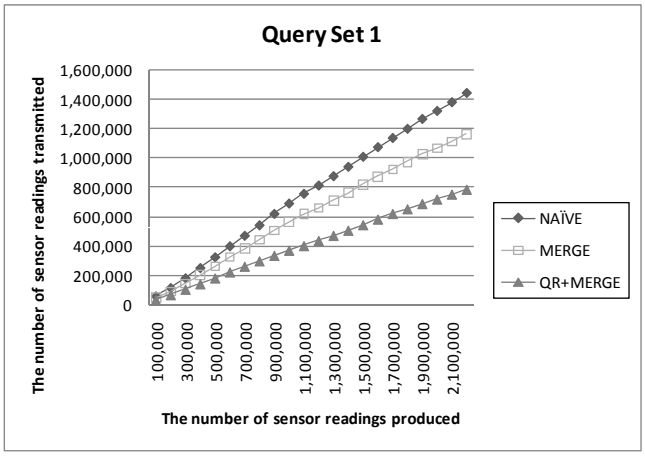

Figure 5. The number of readings transmitted when $QuerySet1$ is executed

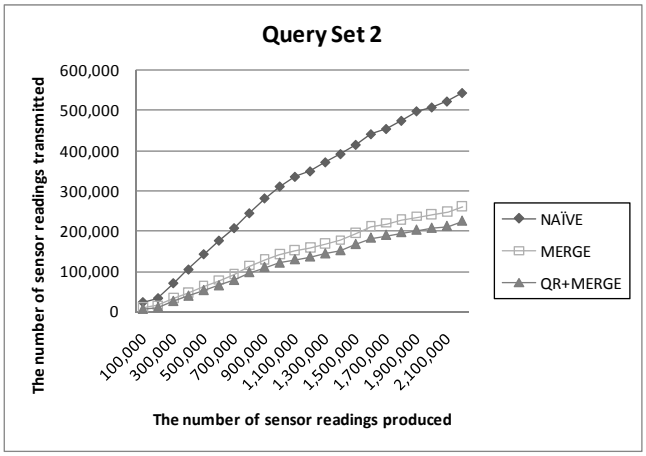

Figure 6. The number of readings transmitted when $QuerySet2$ is executed

in NAIVE and MERGE, respectively, while only 786,387 sensor readings were transmitted in QR+MERGE. That is, QR+MERGE reduced the number of transmissions by 45.4% and 32.6% compared to NAIVE and MERGE, respectively. Also in the case of $QuerySet2$ (Fig. 6), the number of transmissions was reduced in QR+MERGE compared to the other methods. When a total of 2,100,000 sensor readings were produced, 543,377 and 261,137 sensor readings were transmitted in NAIVE and MERGE, respectively, while only 224,041 sensor readings were transmitted in QR+MERGE. In this case, QR+MERGE reduced the number of transmissions by 58.8% and 14.2% compared to NAIVE and MERGE, respectively. This means that even when there are only a small number of queries rewritten, QR+MERGE can benefit from MERGE.

From the experimental results, we can confirm that our proposed method can help reduce the number of transmissions for running multiple monitoring queries. Obviously, the performance gain of the proposed method will increase as the number of queries rewritten increases. In addition, as Fig. 5 and Fig. 6 show, the benefit of QR+MERGE increases as the running time of monitoring queries increases.

6. Conclusions

Since sensor nodes have limited energy capacity, it is important to minimize the energy consumption of sensor nodes to prolong the lifetime of a sensor network. In this paper, we propose an energy-efficient multiple query optimization for sensor networks based on a sophisticated query rewriting algorithm. When a new monitoring query is submitted to the base station, the proposed method rewrites the new query

using currently running queries if possible. The rewritten query is then evaluated at the base station using the results of other queries without being injected into the sensor network. As a result, the energy consumption of the sensor network is reduced. Considering the characteristics of the wireless sensor network environment, the proposed method uses a more complex and sophisticated query rewriting algorithm than the traditional ones. Through the experiments, we show that our method reduces the number of sensor readings transmitted from sensor nodes to the base station, resulting in lower energy consumption.

Acknowledgment

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R0A-2007-000-10046-0), and Brain Korea 21 Project, the School of Information Technology, KAIST in 2009.

References

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCOM), 2000.

[2] S. R. Madden, M. J. Franklin, and J. M. Hellerstein, "TinyDB: an acquisitional query processing system for sensor networks", ACM Trans. on Database Systems, Vol. 30, No. 1, 122-173, 2005.

[3] P. Bonnet, J. E. Gehrke, and P. Seshadri, "Towards Sensor Database Systems", In Proc. of the Second International Conference on Mobile Data Management, Hong Kong, January 2001.

[4] S. Xiang et al., "Two-Tier Multiple Query Optimization for Sensor Networks", In Proc. of the 27th ICDCS, 2007.

[5] P. -A. Larson, H. Z. Yang, "Computing Queries from Derived Relations: Theoretical Foundation", University of Waterloo, Technical Report, CS-87-35, 1987.

[6] P. Roy et al., "Efficient and extensible algorithms for multi query optimization", In SIGMOD, 2000.

[7] A. Y. Halevy, "Answering queries using views: A survey", In VLDB Journal, Vol. 10, 270-294, 2001.

[8] S. R. Madden, J. Hellerstein, and W. Hong, "TinyDB: In-Network Query Processing in TinyOS", http://telegraph.cs.berkeley.edu/tinydb/.

[9] P. Levis et al., "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", In Proc. Of SenSys 2003

[10] S. R. Madden, M. Frankln, J. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for adhoc sensor networks", In Proc. of OSDI, 2002.

[11] D. J. Abadi, S. Madden, and W. Lindner, "REED: Robust, efficient filtering and event detection in sensor networks", In VLDB, 2005.

[12] A. Deshpande et al., "Model-driven data acquisition in sensor networks", In VLDB, 2004.

[13] S. R. Madden, M. J. Franklin, "Fjording the Stream : An Architecture for Queries over Streaming Sensor Data", In ICDE Conference, 2002.

[14] N. Trigoni et al., "Multi-query optimization for sensor networks", In DCOSS, 2005.

[15] R. Muller, G. Alonso, "Efficient Sharing of Sensor Networks", In Proc. of MASS, 2006

[16] J. Hoffmann, S. Kupferschmid, "A Covering Problem for Hypercubes". In Proceedings of IJCAI. pp.1523-1524, 2005.

[17] Intel Lab Data, "http://www.select.cs.cmu.edu/data/labapp3/index.html".

Appendix

We prove the correctness of the proposed query rewriting method.

Theorem 1: Let q_{new} be a new query submitted to the base station. Let q'_{new} be a rewritten query of q_{new} by the proposed query rewriting method. Then, q_{new} and q'_{new} are equivalent, i.e., they produce the same result on all instances of the $sensors$ table.

Proof: Let $q_{new} = \Pi_{nodeid,a_1,a_2,...,a_j}(\sigma_{c_{new}}(sensors))$, where $P(q_{new}) \cup S(q_{new}) = \{a_1, a_2, ..., a_n\}$. Let $q'_{new} = d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n}$. We show that $d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n} = \Pi_{nodeid,a_1,a_2,...,a_j}(\sigma_{c_{new}}(sensors))$. By definition, d_{a_i} $(1 \leq i \leq n)$ is written as follows:

$$d_{a_i} = \begin{cases} \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid,\ a_i}(\sigma_{C(q_{new},a_i)}(q)), \\ \qquad\qquad \text{if } a_i \in P(q_{new}) \\ \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid}(\sigma_{C(q_{new},a_i)}(q)), \\ \qquad\qquad \text{if } a_i \in S(q_{new}) - P(q_{new}) \end{cases} \quad \text{(A.1)}$$

Since $\sigma_{c_1}(\sigma_{c_2}(sensors)) = \sigma_{c_1 \wedge c_2}(sensors)$, d_{a_i} can be written as follows:

$$d_{a_i} = \begin{cases} \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid,\ a_i}(\sigma_{C(q_{new},a_i) \wedge C(q)}(sensors)), \\ \qquad\qquad \text{if } a_i \in P(q_{new}) \\ \bigcup_{q \in Q'_{a_i}} \Pi_{nodeid}(\sigma_{C(q_{new},a_i) \wedge C(q)}(sensors)), \\ \qquad\qquad \text{if } a_i \in S(q_{new}) - P(q_{new}) \end{cases} \quad \text{(A.2)}$$

Let $Q'_{a_i} = \{q_1, q_2, ..., q_t\}$. Since $\Pi_l(\sigma_{c_1}(sensors)) \cup \Pi_l(\sigma_{c_2}(sensors)) = \Pi_l(\sigma_{c_1 \vee c_2}\ (sensors))$, d_{a_i} can be written as follows:

$$d_{a_i} = \begin{cases} \Pi_{nodeid,\ a_i}(\sigma_{C(q_{new},a_i) \wedge (C(q_1) \vee ... \vee C(q_t))}(sensors)), \\ \qquad\qquad \text{if } a_i \in P(q_{new}) \\ \Pi_{nodeid}(\sigma_{C(q_{new},a_i) \wedge (C(q_1) \vee ... \vee C(q_t))}(sensors)), \\ \qquad\qquad \text{if } a_i \in S(q_{new}) - P(q_{new}) \end{cases} \quad \text{(A.3)}$$

Note that $P(q_{new}) = \{a_1, a_2, ..., a_j\}$. Thus, $d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n}$ can be written as follows:

$$d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n} = \Pi_{nodeid,a_1,a_2,...,a_j}(\sigma_c(sensors))$$

where

$$\begin{aligned} c \quad = \quad & (C(q_{new}, a_1) \wedge (\vee_{q \in Q'_{a_1}} C(q))) \\ & \wedge\ (C(q_{new}, a_2) \wedge (\vee_{q \in Q'_{a_2}} C(q))) \\ & \wedge\ ... \\ & \wedge\ (C(q_{new}, a_n) \wedge (\vee_{q \in Q'_{a_n}} C(q))). \end{aligned} \quad \text{(A.4)}$$

Since $C(q_{new}, a_i) \rightarrow \vee_{q \in Q'_{a_i}} C(q)$ by the condition in Section 4.3, c can be reduced as follows:

$$\begin{aligned} c \quad &= \quad C(q_{new}, a_1) \wedge C(q_{new}, a_2) \wedge ... \wedge C(q_{new}, a_n) \\ &= \quad C(q_{new}) \ = \ c_{new} \end{aligned} \quad \text{(A.5)}$$

Therefore, we have shown that $d_{a_1} \bowtie d_{a_2} \bowtie ... \bowtie d_{a_n} = \Pi_{nodeid,a_1,a_2,...,a_j}(\sigma_{c_{new}}(sensors))$. □