Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 978694, 18 pages
doi:10.1155/2010/978694

Research Article

Energy-Efficient Query Management Scheme for a Wireless Sensor Database System

Guofang Nan[1,2] and Minqiang Li[3]

[1] *Institute of Systems Engineering, Tianjin University, Tianjin 300072, China*
[2] *Department of Electronics, Polytechnic University of Turin, Turin 10129, Italy*
[3] *Department of Information Management and Management Science, Tianjin University, Tianjin 300072, China*

Correspondence should be addressed to Guofang Nan, guofangnan@gmail.com

Received 5 November 2009; Revised 6 April 2010; Accepted 3 June 2010

Academic Editor: Xinbing Wang

Minimizing the communication overhead to reduce the energy consumption is an essential consideration in sensor network applications, and existing research has mostly concentrated on data aggregation and in-network processing. However, effective query management to optimize the query aggregation plan at the gateway side is also a significant approach to energy saving in practice. In this paper, we present a multiquery management framework to support historical and continuous queries, where the key idea is to reduce common tasks in a collection of queries through merging and aggregation, according to query region, attribute, time duration, and frequency, by executing the common subqueries only once. In this framework, we propose a query management scheme to support query partitioning, region aggregation and approximate processing, time partitioning and aggregation rules, multirate queries, and historical database. In order to validate the performance of our algorithm, a heuristic routing protocol is also described. The performance simulation results show that the overall energy consumption for forwarding and answering a collection of queries can be significantly reduced by applying our query management scheme. The advantages and disadvantages of the proposed scheme are discussed, together with open research issues.

1. Introduction

With the development of low-power hardware manufacturing and integration, it is possible to design tiny sensor devices combining the abilities of sensing, computation, storage, and communication [1]. These nodes collect sensor data and communicate with each other, forming a network to monitor objects, animals, people, temperature, humidity, and so on in a given area [2]. The appearance of wireless sensor networks has significantly changed various kinds of remote sensing applications such as environmental and ecological monitoring of natural habitats, smart homes, and military areas in recent years [3].

Due to the facts that sensor nodes are physically small and must use extremely limited power or energy, the network lifetime is still a vital problem. Many of the WSN (wireless sensor network) techniques designed to extend the network lifetime are concentrated on modified routing protocols [4–6], in-network processing [7, 8], node sleep scheduling [9, 10], and aggregation strategies [11–13]. Recent studies have shown that radio communication is significantly more expensive than computation or sensing in most existing sensor node platforms, hence the main consideration is to minimize the communication overhead of forwarding queries and transmitting queried data between gateway and source nodes [14]. Thus, information aggregation is effective to save energy and extend the network lifetime. According to the location where aggregation occurs, aggregation strategies can be divided into two main types: data aggregation and query aggregation. Data aggregation belongs to the category of in-network filtering and processing techniques, which combine the data coming from different sources and eliminate data redundancy to minimize the number of transmissions, thus saving energy [15]. Most of the existing work in this area focuses on data aggregation. Query aggregation occurs at a query manager node, which is usually located at the gateway, and has been the subject of much less research [16]. On the other hand, for many

applications, especially those with high query rates, for example, a large number of very similar queries are issued to the network within a short period, due to a large number of users, while the response data to each individual query is comparatively simple [16]. If we assume that all the queries are processed individually, this will lead to large amount of energy consumption due to query dissemination and data transmission within the network. Therefore, a proper query aggregation and management scheme should be used to reduce redundant queries in order to minimize wasted bandwidth, power, and energy.

The problem of efficient query aggregation on the gateway side for a sensor database system has been treated previously in the literature [14, 16–19]. In order to provide efficient data services for sensor network applications, an overlay-based query aggregation approach including a query manager and an effective query aggregation algorithm were presented in [16]. The former defines the query aggregation plan that is executed by the latter. In their query model, only spatial information is aggregated to minimize the number of queries that are actually sent out. A spatial- and attribute-based query aggregation method (SAQA) was introduced in [19], which extends the aggregation to include also attributes. However, their models assume that most queries are snap-shot (queries that ask for current value of the sensors), which may not be the case in practice, because users sometimes ask for sensor values during a period of time (continuous query). Similar to their work, several other models supporting continuous queries were discussed in [14, 17, 18], which propose two significant ideas: exploiting common subqueries in a group and using a proper routing algorithms to minimize the total system cost.

However, there are still two main limitations of the works mentioned above. One is that there is still some redundancy in the aggregated queries, due to time duration: for instance, two queries that ask for the same type of sensing data in the same region at an overlapping duration can be aggregated into one. The other is that the historical database is not fully utilized. We assume that all the (recent) query records are stored in a historical database located at the gateway side, where the query manager can check for already present information before injecting new queries into the network. Thus, in this paper we propose a complex query optimization framework to support historical and continuous queries, and we describe the corresponding query processing scheme.

In contrast to previous work, our contributions can be summarized as follows. We propose a multiquery optimization framework suitable for historical and continuous queries. In contrast to previously proposed query optimization schemes, the query partition module, the aggregation module, and the result merging module are also included in the framework to optimize the objective function. A storage mechanism of historical query records is kept at the gateway side, and we propose an algorithm for querying it. Query partitioning, region aggregation and approximate processing, time partitioning and aggregation rules, as well multirate query processing algorithms are presented.

The rest of this paper is organized as follows. Section 2 presents some research works related to ours. Section 3 introduces the multiquery problem. In Section 4, we propose our energy efficient framework including the query partition module, query aggregation module, query result merging module, and historical database module. Section 5 presents the corresponding algorithms to support our query framework. In Section 6, we present simulation and experiment results to demonstrate the efficiency of the work and compare it with other query processing techniques. Finally, the advantages and disadvantages of the proposed scheme are discussed, together with open research issues in Section 7.

2. Related Work

Generally, when the gateway node receives queries from applications by end-users to ask for sensor data they are interested in, it will directly forward them to the sensor network according to given query dissemination schemes. Within the network, the nodes must respond to these queries in an energy-efficient manner using a variety of in-network processing techniques and cross-layer optimizations to report answers to the end-users at the appropriate rates [20]. A wireless sensor network can be regarded as a distributed database, due to the fact that sensing information is often reported from a number of different sources and is often held in a number of databases that may be distributed among computing and communication facilities at different locations [21]. Thus, the previous work related to this paper consists of two main aspects, namely, complex query optimization in traditional distributed databases and multiquery aggregation in wireless sensor databases.

2.1. Complex Query Optimization in Traditional Distributed Databases. The problems of complex query optimization [22, 23] and multiquery aggregation have been studied in the traditional database literature for more than 40 years. The core idea is to exploit the common tasks among groups of queries and perform them only once to reduce the execution cost [24, 25]. Most studies emphasized efficiently generating alternative plans that maximize shared operations and minimize system cost [26]. Several heuristic algorithms [24, 27] have been studied to identify common tasks and to select a plan for each query. In order to solve the multiquery optimization problem, the partitioning of complex queries was discussed in [28], which leads to a better interpretation of complex aggregate queries and a better execution plan. The author also presented two algorithms to decompose a complex aggregate query into its group query components, and the experiments show the validity of complex query partitioning. The concept of sketch sharing for approximate multiquery stream processing was presented in [29] to optimize multiqueries. Given a collection of queries to be processed over incoming streams, the same sketches over their input streams are optimized by performing space allocation and coalescing rules. The final results clearly demonstrate that sketch sharing is efficient to solve the multiquery problem, especially with respect to the quality of query answering. Even though these studies cannot be directly applied to sensor networks due to the very different

storage, communication, and energy constraints, the core idea about query partitioning and query aggregation is significant to our work.

2.2. Multiquery Aggregation in Sensor Databases. In order to provide efficient data services for sensor network applications, an overlay-based query aggregation approach including a query manager and an effective query aggregation algorithm was presented in [16]. The query manager, located at the base station, is mainly devoted to defining the global query aggregation plan, and the query aggregation algorithm is designed to aggregate and optimize queries issued by end-users. The corresponding protocols for query dissemination and data transmission are also included. Contrary to traditional query processing, in their framework, queries from applications cannot be directly forwarded to the network, but are collected and evaluated at the gateway, to be aggregated if possible according to zone merging rules. Only the merged queries are delivered to the access node in the network by an appropriate selection scheme using a query delivery overlay construction protocol. Finally the queried data would then be passed back from the access node to the gateway. However, their aggregation protocol is mainly based on region operation. As mentioned above, spatial- and attribute-based query aggregation (SAQA) was introduced in [19], assuming that most queries are snap-shot queries. Similar to above work, a kind of multiquery optimization technique was also presented in [17], which supports multiple users submitting both continuous and snapshot queries. The query optimizer groups queries from applications with the same aggregate operator and optimizes each group separately. All queries gathered during the previous epoch are sent to the network together for evaluation in the query preparation phase, and query answers are forwarded back to the gateway in the result propagation phase. The query preparation protocol and result propagation protocol are also described. The effect of multiquery optimization in sensor networks was discussed in [30] to study the benefit of exploiting common subexpressions among these queries, with significant performance improvements. The author also presented a two-tier query framework for optimizing multiple queries to improve the service quality of the sensor networks [18].

A related aspect in sensor networks is query rate. Support for multiple rates plays a significant role for the performance of sensor networks. If source nodes disseminate the data streams to users at the frequency that they request, the result is very costly in terms of energy. A more efficient framework to process multirate queries was proposed in [31], where, the construction of a path-sharing routing tree was also discussed. Another related aspect is the query/storage techniques for sensor networks in real applications the author in [32] describes an effective middleware that was specifically designed for proactive urban monitoring and exploits node mobility to opportunistically diffuse sensed data summaries among neighboring vehicles and to create a low-cost index to query monitoring data. To make their works more persuasive, the related protocols were validated and their effectiveness in terms of indexing completeness, harvesting time, and overhead are demonstrated. To avoid the content redundancy and storage imbalance in distributed storage system, a cooperative storage solution for mobile surveillance in vehicular sensor networks [33] called VStore is presented in [34] to maximize the average lifetime of sensory data in sensor networks.

We conclude this section by observing that there are two key ideas among these works: exploiting the common tasks among groups of queries and using proper routing algorithms to minimize the total system cost. As noted above, most of these solutions are unaware of query time duration and historical database. The former is useful to reduce query redundancy, while the latter contributes to energy saving. Therefore, this paper presents a generic query management scheme including query partitioning, common task optimization exploiting a historical database, and query result merging to further improve the performance.

3. Problem Definition

We consider a multihop sensor network with one gateway node at the centre and L sensors distributed randomly in a rectangular field. The gateway node receives queries from users and processes them by using its query manager component, then sends queries into the appropriate regions of the sensor network. Eventually sensors in the region may respond to one of the queries. In this paper, we assume that all the sensors have the same fixed transmission range and the same minimum connectivity transmission range [35]. We consider a sensor network similar to the network model used in [1, 35], with the following assumptions

(i) There are L energy constrained sensor nodes that are distributed randomly in a rectangle-shaped region. The batteries cannot be changed after the sensors are deployed.

(ii) Each node, including the gateway node, can obtain its own location information.

(iii) There is no coverage hole in the sensor network and all the sensors can communicate with the gateway node through a routing protocol.

The goal of this study is to reduce the total number of queries and the energy consumption. Similar to prior work in [16, 17, 19], we use the following definitions.

Definition 1 (Query Region). Query region R indicates the geographical area that the application is interested in. Without loss of generality, we assume query regions to be a union of rectangular shapes. A two-dimensional query region element is thus represented by a bounding box, for example, the minimum and maximum values of the coordinates. For any query point, its coordinates (x, y) should satisfy $x \in [x_1, x_2]$ and $y \in [y_1, y_2]$, where x_1 and x_2 are the minimum and maximum values on the x-axis of the bounding box, and y_1 and y_2 are the minimum and maximum values on the y-axis of the bounding box.

Definition 2 (Query Time Duration). Query time duration T is the duration of the query. For example, it is often needed to continuously report the temperature and humidity value of the monitoring area from t_1 to t_2, so $T = \{t_1 \text{ to } t_2\}$. It is noted that if T is a time point, rather than a time duration, it represents a snap-shot query.

For the query model in [16, 17, 19], the attribute information A and frequency information F are two other important elements to be considered. The attribute information indicates the list of attributes that the application is interested in, and the period information is the inverse of the frequency at which the data should be reported. In this work we also consider the query identifier, defined as follows.

Definition 3 (Query ID). Each query must have a query ID denoted by ID, which is the information used by the gateway node to analyze and identify which user the query is from. After the query is injected into the sensor network from a gateway node, the corresponding information the user is interested in will be sent back by the routing protocol via the gateway node to the user, and throughout the process the query ID is its unique identifier. Furthermore, the query ID is also used by query partitioning, aggregation, and result merging.

Definition 4 (Query: Q). A query consists of five types of information: query ID, query region, query time, attribute information, and query period, so it can be denoted by a 5-tuple.

$$Q = \langle ID, R, T, A, F \rangle, \tag{1}$$

where ID is query ID, see Definition 3, R is Query Region, see Definition 1, T is Query Time, see Definition 2, A is Attribute information, the list of attributes which the application is interested in, and F is Query period, at which the attribute information should be reported.

Example 1. User 1 wants to know the temperature and humidity of regions R_1, R_2, and R_4 from t_1 to t_2 every five seconds.

So the query q_1 can be represented as follows:

$$Q_1 = \langle ID_1, \{R_1, R_2, R_4\}, \{t_1 \text{ to } t_2\}, \{\text{Temp\&Humidity}\}, 5s \rangle. \tag{2}$$

User 2 is interested in the wind speed of regions R_2 and R_3 from t_3 to t_4 every two seconds. So

$$Q_2 = \langle ID_2, \{R_2, R_3\}, \{t_3 \text{ to } t_4\}, \text{WindSpeed}, 2s \rangle. \tag{3}$$

For simplicity, the query ID is omitted in the rest of the paper, because it is usually related to the header information of a packet.

In order to produce better execution plans and improve query performance, performing complex analysis at the gateway side is essential and requires nontrivial partitioning and aggregation operations over different attribute sets, query time durations, and query regions.

Definition 5 (Attribute Partitioning). For each query $Q = \langle R, T, A, F \rangle$, the attribute set A can be disjointly partitioned into several subattribute sets, $A = \{A_1, A_2, \ldots, A_k\}$ and $A_1 \cap A_2 \cap \cdots \cap A_k = 0$. Hence also the query Q can be partitioned into several subqueries:

$$\begin{aligned} Q &= \langle R, T, A, F \rangle \\ &= \langle R, T, A_1, F \rangle + \langle R, T, A_2, F \rangle \\ &\quad + \cdots + \langle R, T, A_k, F \rangle. \end{aligned} \tag{4}$$

Definition 6 (Query Region Partitioning). For each query $Q = \langle R, T, A, F \rangle$, the query region set R can be disjointly partitioned into several subregions, $R = \{R_1, R_2, \ldots, R_j\}$ and $R_1 \cap R_2 \cap \cdots \cap R_j = 0$. Hence also the query Q can be partitioned into several subqueries:

$$\begin{aligned} Q &= \langle R, T, A, F \rangle \\ &= \langle R_1, T, A, F \rangle + \langle R_2, T, A, F \rangle \\ &\quad + \cdots + \left\langle R_j, T, A, F \right\rangle. \end{aligned} \tag{5}$$

Definition 7 (Query Time Partitioning). For each query $Q = \langle R, T, A, F \rangle$, the query time set T can be disjointly partitioned into several subdurations, $T = \{T_1, T_2, \ldots, T_n\}$ and $T_1 \cap T_2 \cap \cdots \cap T_j = 0$. Hence also the query Q can be partitioned into several subqueries:

$$\begin{aligned} Q = \langle R, T, A, F \rangle &= \langle R_1, T_1, A, F \rangle + \langle R_2, T_2, A, F \rangle \\ &\quad + \cdots + \left\langle R_j, T_n, A, F \right\rangle. \end{aligned} \tag{6}$$

Definition 8 (Query Partitioning). Given N queries: $Q_1, Q_2, \ldots, Q_N$, the overall objective of query partitioning is to divide them into disjoint subqueries, in order to find the common parts between them. Each Q_j can be partitioned into a group of subqueries denoted by set $\{Q'_{j1}, Q'_{j2}, \ldots, Q'_{jK_j}\}$. Hence given a query set $\{Q_1, Q_2, \ldots, Q_N\}$, the output partition is also a set.

$$\begin{aligned} &\left\{Q'_{11}, Q'_{12}, \ldots, Q'_{1K_1}, Q'_{21}, Q'_{22}, \ldots, Q'_{2K_2}, \ldots, Q'_{N1}, Q'_{N2}, \ldots, Q'_{NK_N}\right\} \\ &\quad = \{Q'_1, Q'_2, \ldots, Q'_M\} \quad M = \sum_{j=1}^{N} K_j. \end{aligned} \tag{7}$$

The overall objective of query aggregation is to reduce the overall energy consumption in both query transmission and data delivery by eliminating and merging queries whenever possible. So query aggregation can be divided into two steps: (1) finding the common subqueries; (2) Merging and recombination of these subqueries according to query ID, query region, query time, and query attributes.

Definition 9 (Query Aggregation). Given W partitioned queries $Q'_1, Q'_2, \ldots, Q'_W$, through the aggregation operation set $\{Q'_1, Q'_2, \ldots, Q'_W\}$ can be merged into another set $\{Q''_1, Q''_2, \ldots, Q''_P\} (P \leq W)$, where each $Q''_i (1 \leq i \leq P)$

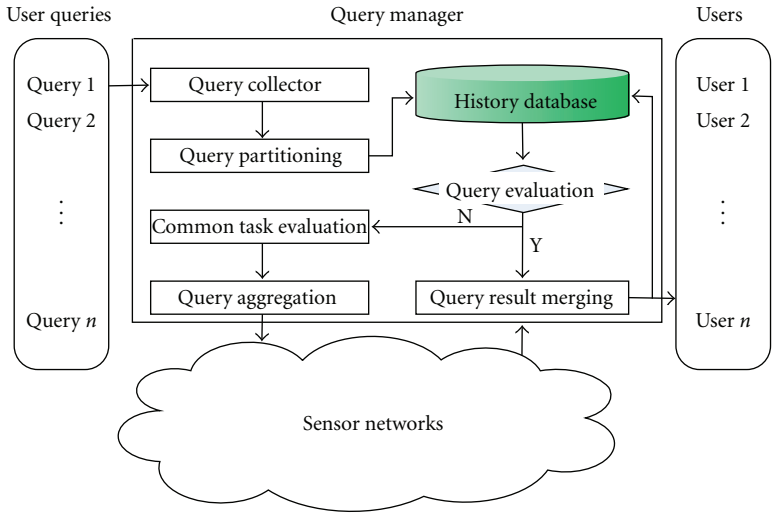

Figure 1: Query management framework.

represents the output of query aggregation. Motivated by [16], we also define the + as the aggregation operator, that is, $Q_i'' = Q_1' + Q_2' + \cdots + Q_W'$ means that $Q_i''(1 \le i \le P)$ is aggregated from the partitioned queries $Q_1', Q_2', \ldots, Q_W'$.

4. Query Management Framework

In this section, we present our framework and a number of components for multiquery optimization at the gateway side. Our framework quantifies the performance impact of processing multiple queries with our query management scheme. To perform an effective decision making to produce a better query plan is vital in an energy-efficient query management system, and the core of such system is a query management scheme and a historical database which stores historical query records and consolidated data from the sensor network, supporting complicated queries that return interesting information [36]. The basic idea of the proposed query optimization framework is to minimize the number of queries injected into the sensor network by querying the historical database and aggregating queries to improve the query processing performance.

Our energy-efficient framework for multiquery optimization in sensor network is built upon a number of components, including query collector, query partitioning, common task evaluation, query aggregation, query result merging, and historical database, and the first five components are integrated into a query manager. Query collector collects queries from different users within a given time duration. The complex queries are partitioned into several subqueries by the query partitioning module. We evaluate these subqueries and recognize the common tasks between them in a centralized way through the process of common task evaluation. The process of query aggregation eliminates the duplicate common tasks of a collection of queries through merging and aggregating them according to query region, query attribute, query time duration, and query frequency. Query result merging collects the answers from both the sensor network and the historical database, merges them according to the query ID, and sends them back to the users.

The historical database contains records of user queries and query results. These are often captured automatically by the system and may be manually complemented or annotated by the database manager after query answers are sent back to the query manager. A query record, as described in this paper, is one type of interaction history which specifically records user queries. It typically contains search queries, result sets, and relevant contextual information, such as user profiles, system settings, and statistics. By searching the historical database, full or partial query results may be obtained without sending these queries into the sensor network, thus saving energy and reducing network traffic.

In order to eliminate the duplicated common tasks, when the gateway node receives queries from end-users to ask for sensor data they are interested in, the query manager collects and processes them first to produce an optimal query plan for a collection of queries. The processing of the initial queries is as follows (See Figure 1).

Step 1. Query collector receives queries from the applications within a given time duration.

Step 2. Check all the collected queries. If a complex query is included, partition it into subqueries according to the query partitioning scheme, as described more in detail below.

Step 3. For each partitioned query, search the historical database for the queried data; (1) if the historical query records fully meet one query requirements, the corresponding query records are directly sent to the query result merging module; (2) if the historical query records have no relation with the query requirements, that is to say, these queries cannot be answered by historical database, thus, the queries have to be evaluated with other queries to find common

subqueries between them; (3) if the historical query records partially meet the query requirements, the query is reduced to the part that cannot be answered by historical database according to Definition 7.

Step 4. The queries not answered by historical database are evaluated to exploit their common tasks.

Step 5. Aggregate these queries and eliminate the duplicated parts among them, and then produce an optimal query plan.

Step 6. Route the queries to the appropriate regions to achieve the query results.

Step 7. Query results from both the historical database and the sensor network are merged according to query ID, and forwarded to the end-users.

Step 8. In order to ensure the accuracy and reliability of the data stored in the historical database, only the elementary queries without aggregation, together with their data, are stored into the historical database.

The query manager translates the application queries to the format that sensor network understands. Generally, the outputs of the query manager are complex queries which may include several attributes, regions, frequencies, or different time durations [16]. For each complex query, the query manager calculates an access point of the queried region which is defined as the geometrical centre of the region in this paper. A heuristic routing protocol is also adopted by multihop transmission from the gateway node to the access point, by which the sensor node closest to the access point called access node receives the complex query, reroutes it to the other sensor nodes in the region. When data is sent back from the sensor nodes, the reverse process is used.

We can summarize the main features of the framework as follows:

(1) The system provides an energy efficient query management framework for sensor networks, which allows their components to effectively cooperate with others to manage and optimize the complex queries received from users.

(2) It uses a historical database, which makes full use of query records and their results. It is especially suitable for the applications with many queries and comparatively simple query answers.

5. Query Management Scheme

The query management scheme is designed to support the framework described in Section 4. It includes schemes for partitioning the initial queries according to time and region partitioning and aggregation rules, approximate processing, multirate processing, historical database query, and routing protocol.

5.1. Initial Query Partitioning. Query region, query time, and query attributes are three important components to be considered when analyzing how different queries overlap in terms of these three components. In the query framework that we discussed in this paper, the historical database is a vital part for the users to directly access the sensing data that they are interested in at the gateway side, without the need for the queries to be transmitted to the sensor networks. But the historical database only includes the data reported by a subset of past queries, since not all the sensing data are transmitted from the sensor network to the historical database in time or kept forever. Sometimes the historical database cannot fully answer one query, but some subqueries may be answered in terms of query regions, query time, and query attributes. In order to fully utilize the data in the historical database, we partition a collection of queries according to specific granularity. Another important aspect of query partitioning is that it is useful for the query manager to evaluate the common subqueries.

For example, one query asks for the temperature and humidity value of region R_1, R_2 and R_4 from t_1 to t_2 and from t_3 to t_4 every five seconds, so it can be represented as follows:

$$Q = \langle \{R_1, R_2, R_4\}, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T\&H\}, 5s \rangle. \quad (8)$$

Here, a step-by-step partitioning strategy is adapted (See Table 1).

5.2. Region Aggregation and Approximate Processing. Query region is the most significant component in our framework, and it is the main criterion to select sensor nodes for query execution and data forwarding. Therefore, all the queries sent to the sensor networks should follow the regional priority rule, meaning that region overlap is the first and foremost criterion to decide about query merging. Other types of operations, including query time merging, attribute aggregation, and multirate query processing, are all based on the fact that these operations occur approximately in the same region. Otherwise, there is no advantage in aggregating the queries. The regional priority rule is also adopted even if we search the historical database to check the previous query record for one query. For instance, consider two queries $Q_1 = \langle R_1, \{0 \text{ to } 10\}, H, 2s \rangle$ and $Q_2 = \langle R_2, \{0 \text{ to } 10\}, H, 2s \rangle$. If their query regions are completely different, even though the query time, attribute, and query frequency of Q_1 are in full accord with that of Q_2, it is useless to merge the two queries.

Adopting the main idea in [16] about query region processing, we divide the rules of processing query region into two basic types: approximate region aggregation and overlapped region aggregation.

Consider two queries $Q_1 = \langle R_1, \{4 \text{ to } 20\}, H, 2s \rangle$ and $Q_2 = \langle R_2, \{0 \text{ to } 10\}, H, 2s \rangle$, if R_1 and R_2 satisfy $R_1 \subset R_2$, sending out both queries to the sensor network would be redundant as the result of Q_1 can be inferred from that of Q_2. So the rule of approximate region processing can be defined as follows (this case is also suitable for searching historical database for a single query). If R_1 in one query Q_1 and R_2

Table 1: Process of query partitioning.

Initial query	Region partitioning	Time partitioning	Attribute partitioning
$\langle\{R_1, R_2, R_4\}, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T\&H\}, 5s\rangle$	$\langle R_1, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T\&H\}, 5s\rangle$	$\langle R_1, \{t_1 \text{ to } t_2\}, \{T\&H\}, 5s\rangle$	$\langle R_1, \{t_1 \text{ to } t_2\}, T, 5s\rangle$
			$\langle R_1, \{t_1 \text{ to } t_2\}, H, 5s\rangle$
		$\langle R_1, \{t_3 \text{ to } t_4\}, \{T\&H\}, 5s\rangle$	$\langle R_1, \{t_3 \text{ to } t_4\}, T, 5s\rangle$
			$\langle R_1, \{t_3 \text{ to } t_4\}, H, 5s\rangle$
	$\langle R_2, \{t_1 \text{ to } t_2, t_3 \text{to } t_4\}, \{T\&H5s\}, \rangle$	$\langle R_2, \{t_1 \text{ to } t_2\}, \{T\&H\}, 5s\rangle$	$\langle R_2, \{t_1 \text{ to } t_{24}\}, T, 5s\rangle$
			$\langle R_2, \{t_1 \text{ to } t_2\}, H, 5s\rangle$
		$\langle R_2, \{t_3 \text{ to } t_4\}, \{T\&H\}, 5s\rangle$	$\langle R_2, \{t_3 \text{ to } t_4\}, T, 5s\rangle$
			$\langle R_2, \{t_3 \text{ to } t_4\}, H, 5s\rangle$
	$\langle R_4, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T\&H\}, 5s\rangle$	$\langle R_4, \{t_1 \text{ to } t_2\}, \{T\&H\}, 5s\rangle$	$\langle R_4, \{t_1 \text{ to } t_2\}, T, 5s\rangle$
			$\langle R_4, \{t_1 \text{ to } t_2\}, H, 5s\rangle$
		$\langle R_4, \{t_3 \text{ to } t_4\}, \{T\&H\}, 5s\rangle$	$\langle R_4, \{t_3 \text{ to } t_4\}, T, 5s\rangle$
			$\langle R_4, \{t_3 \text{ to } t_4\}, H, 5s\rangle$

in the historical query record satisfy $R_1 \subset R_2$, the result of Q_1 can be inferred from the historical query record.

If any two queries $Q_i = \langle R_i, T_i, A_i, F_i\rangle$ and $Q_j = \langle R_j, T_j, A_i, F_i\rangle$, are such that R_i and R_j satisfy $R_i \subset R_j$, then they can be merged to

$$Q_{i+j} = \left\langle R_j, \left\{T_i, T_j\right\}, A_i, F_i\right\rangle. \tag{9}$$

Like the rule of approximate region processing, the rule of region overlapped aggregation can be defined as follows.

If any two queries $Q_i = \langle R_i, T_i, A_i, F_i\rangle$ and $Q_j = \langle R_j, T_j, A_i, F_i\rangle$, are such that R_i and R_j satisfy $R_i \cap R_j = R_{ij}$ and $S_{R_i \cap R_j}/S_{R_i \cup R_j} \geq \beta$, then the two queries can be merged to

$$Q_{i+j} = \left\langle R_{ij}, \left\{T_i, T_j\right\}, A_i, F_i\right\rangle. \tag{10}$$

Here, β is a constant, $0 \leq \beta \leq 1$, $S_{R_i \cap R_j}/S_{R_i \cup R_j}$ is the region overlapping degree, namely, the ratio between the intersection and the union area of R_i and R_j.

5.3. Multirate Query Processing. In a multirate query system, the sensor network serves multiple queries to send data at different frequencies to users. The data sources disseminate the data streams to the users at the frequencies they request. In order to reduce the amount of transmitted data, we can modify the data streams according to a multirate processing rule which is illustrated as follows.

For example, user 1 requests the data from all the nodes in region R with period F_1. At the same time, user 2 requests the data from all the nodes in the same region with period F_2. Without loss of generality, we find an appropriate time unit such that all frequencies can be represented as integers. Without aggregation, the gateway node initiates independently each data query by sending out a query request to the data sources; the query request is routed to the appropriate source nodes within the queried region, and the source nodes will start sending data back to the gateway node. However, this method is prone to produce redundant data transmission.

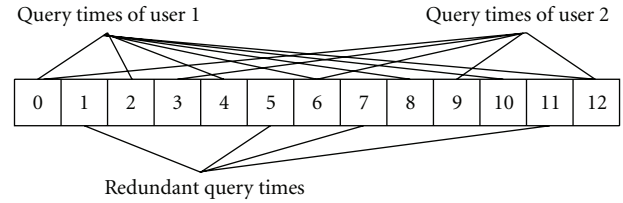

Figure 2: Example of multirate query.

Let $F_i (1 \leq i \leq n)$ be the requested periods of all the user queries to the same region R. Then the aggregate period F of these n queries can be described by the next equation.

$$F = \text{GCD}(F_1, F_2, \ldots, F_n), \tag{11}$$

where the GCD (Greatest Common Divisor) function returns the greatest common divisor of one or more integers; for instance, if a sensor network is used for collecting the temperature of the environment, user 1 might need the temperature every 2 minutes, and user 2 might need the temperature every 3 minutes. Assuming that these two queries are issued at time 0, this will result in a multirate query in the sensor network. Given that $F_1 = 2$, and $F_2 = 3$, $F = \text{GCD}(F_1, F_2) = 1$, a simplistic method would request data at times 0,1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and so on, while it can be seen clearly that the data sent at time 1, 5, 7,11 are not needed (See Figure 2).

To avoid the redundant queries resulting by multirate processing, the query time interval is set by each user at one of several discrete values, according to the next equation.

$$F = 2^k \ (k = 1, 2, \ldots, i, \ldots). \tag{12}$$

For instance, if a sensor network is used for collecting the temperature of the environment, user 1 might need the newest temperature every 2 minutes, and user 2 might need the newest temperature every 4 minutes. Assuming that these two queries are issued at time 0, this will result in multirate queries in the network, for with periods, $F_1 = 2$, and $F_2 = 4$,

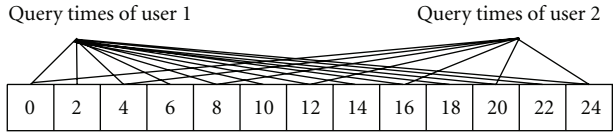

Figure 3: Example for processing redundant data.

$F = \mathrm{GCD}(F_1, F_2) = 2$, demanding data at times 0, 2, 4, 6, 7, 8, 10,12,... (See Figure 3). Thus no redundant query time is produced.

The choice of query time interval is based on two aspects: (1) each user selects a discrete time interval according to his requirement, based on the rate of change of the sensed quantity and the user's required precision; (2) for any two time intervals, one is exactly divided by another, so no redundant query time is yielded, which therefore saves energy consumption in data transmission.

5.4. Time Partitioning and Aggregation Rules. Consider the case where the gateway receives the query $Q = \langle R, \{0 \text{ to } 10\}, H, 2s\rangle$, that is, the user wants to know the humidity value of region R from $t_1 = 0$ to $t_2 = 10$ every two seconds, while there is information about the humidity value of region R from $t_1' = 0$ to $t_2' = 4$ only every two seconds in the historical database. Hence we partition the query in time, as $Q_1 = \langle R, \{0 \text{ to } 4\}, H, 2s\rangle$ and $Q_2 = \langle R, \{4 \text{ to } 10\}, H, 2s\rangle$, so that without sending Q_1 to the sensor networks, the queried information is available in the local historical database.

As we discussed in the query framework, the partitioned queries are not directly evaluated and aggregated, but used for direct search in the historical database to check whether there are historical query records that partially or fully satisfy each partitioned query.

Consider another case where the gateway nodes receives two queries $Q_1 = \langle R, \{0 \text{ to } 10\}, H, 2s\rangle$ and $Q_2 = \langle R, \{6 \text{ to } 20\}, H, 2s\rangle$ from two different users. There exists a common time interval $\{6 \text{ to } 10\}$ between the two queries. Therefore, the common query time interval can be found by partitioning these two queries as follows:

$$\begin{aligned} Q_1 &= \langle R, \{0 \text{ to } 10\}, H, 2s\rangle \\ &= \langle R, \{0 \text{ to } 6\}, H, 2s\rangle + \langle R, \{6 \text{ to } 10\}, H, 2s\rangle, \\ Q_2 &= \langle R, \{6 \text{ to } 20\}, H, 2s\rangle \\ &= \langle R, \{6 \text{ to } 10\}, H, 2s\rangle + \langle R, \{10 \text{ to } 20\}, H, 2s\rangle, \\ Q_1 + Q_2 &= \langle R, \{0 \text{ to } 20\}, H, 2s\rangle. \end{aligned} \tag{13}$$

Query time aggregation only occurs in the sub-query evaluation process, if any two different queries have a common time interval and have the same query region and query attributes. In our query management scheme, the rules of time partitioning are designed to follow exactly these two cases.

5.5. Search of Historical Database. One of the most important components of the framework illustrated in Figure 1 is the historical database. Storage techniques for query processing in sensor networks can be divided into two categories, one storage method is to send raw data of sensors to the gateway through a network routing tree rooted at the gateway node. Another possible storage approach is local storage, for example, sensors collect and store data local. When queries are injected from gateway node, sensors send back their reply [37]. However, two traditional storage techniques mentioned above cannot be directly applied to the scenario of energy effective multiple similar queries. The first method is more costly because energy is needed for sending data to the gateway where it may never be used, while the second method does not insert query records into historical database. Therefore, in this paper, in order to ensure the accuracy and reliability of the data stored in the historical database, only the elementary queries without aggregation, together with their answering data, are stored into the historical database, when a partitioned query arrives, the historical database is checked to see if the query had been executed previously, by searching the historical database, full or partial query results may be obtained without sending these queries into the sensor network, which is helpful to energy saving and data accuracy.

The motivation of storing historical data is to support historical data queries for various applications and most of existing approaches to historical data storage of sensor networks are distributed [38]. The proposed storage mechanisms are mainly focused on local storage of historical data. Therefore, in this paper, we propose a centralized storage method that stores historical query and their answers at the gateway side, which makes queries processing more efficient in terms of energy consumption by using historical database. Since a query consists of five types of information as defined in Section 3, we also use it as the basic data format for the historical database. Thus, the query manager translates the user requirements and their answers into the format according to query definition. For instance, one query asks for the temperature of region R_1 from t_1 to t_2 every 2 seconds, $Q_1 = \langle R_1, \{t_1 \text{ to } t_2\}, T, 2s\rangle$, and R_1 can be represented by two zones [0,20] and [0,10], which means that for any point $(x, y) \in R_1$satisfies $0 \le x \le 20$ and $0 \le y \le 10$, $t_1 = 2$, $t_2 = 10$. Assumed that three sensors (node 1, node 4, and node 10) in this region, for each sensor, the location information and the temperature value at time 2, 4, 6, 8, 10 should be sent back to the gateway node. After the gateway node receive the answers for Q_1, it creates a storage index for Q_1 based on the received data as Figure 4 shows. Clearly, query Q_1 and their answers are stored in the historical database.

We compare each partitioned query with each recorded query, only if the query region, query attribute, query time duration, and query frequency satisfy $R_i \subset R_j$, $V_i = V_j$, $F_i \ge F_j$ and $T_i \cap T_j \ne \phi$, that is, if full or partial results can be obtained from historical database. The algorithm is as follows in Algorithm 1.

As for the partial query result, consider the case where the gateway receives the query $Q = \langle R_i, \{0 \text{ to } 10\}, H, 2s\rangle$, that is, the user wants to know the humidity value of region

Input: a simple query and a historical query record set with n queries.
Output: initial simple query or partial query results and sub-query or query results Begin
// compare the initial query with historical query record
// (the initial query $Q_i = \langle R_i, V_i, T_i, F_i\rangle$ and any historical query
// record $Q_j = \langle R_j, V_j, T_j, F_j\rangle$ use the same query model.)
if $R_i \subset R_j$ and $V_i = V_j$ and $F_i \geq F_j$
if $T_i \cap T_j \neq \phi$
if $T_i \subset T_j$, Q_i output full results, else output partial results
$Q_i = \langle R_i, V_i, T_i - T_i \cap T_j, F_i\rangle$
else
$Q_i = \langle R_i, V_i, T_i, F_i\rangle$
end if
else
$Q_i = \langle R_i, V_i, T_i, F_i\rangle$
end if

Algorithm 1: Query historical database.

Temperature value region $R_1[0, 20][0, 10]$ $T = \{2 \text{ to } 10\}$ $F = 2s$			
	Node 1	Node 4	Node 10
$t = 2$	25.1	25.2	25.3
$t = 4$	25.1	25.1	25.1
$t = 6$	25.2	25.2	25.3
$t = 8$	25.1	25.1	25.2
$t = 10$	25.2	25.1	25.3

Figure 4: An example storage structure for historical queries.

R_i from $t_1 = 0$ to $t_2 = 10$ every two seconds, while there is information about the humidity value of region R_j from $t_1' = 0$ to $t_2' = 4$ only every two seconds in the historical database. Hence we partition the query in time, as $Q_1 = \langle R, \{0 \text{ to } 4\}, H, 2s\rangle$ and $Q_2 = \langle R, \{4 \text{ to } 10\}, H, 2s\rangle$, if $R_i \subset R_j$or $R_i = R_j$, so that without sending Q_1 to the sensor networks, the queried information is available in the local historical database, thus, partial query result of Q can be obtained.

5.6. Query Aggregation Scheme. The main idea of the proposed query aggregation scheme is to exploit and eliminate the duplicated common tasks of a collection of queries for minimizing the cost of processing multiple queries. However, a number of system considerations have to be taken into account to apply it to a real sensor network. In this section, we develop a multiquery optimization scheme for these queries according to query region, query attribute, query time interval, and query period.

The input of the query aggregation scheme is a set of partitioned queries Q from users, each partitioned query $Q_i \in Q$ is denoted by $\langle R_i, V_i, T_i, F_i\rangle$, where R_i represents the query region, V_i represents the query attribute, T_i is the query time interval, and F_i is the query period. It is noted that these partitioned queries has been compared with each recorded query stored in the historical database, only those queries cannot be answered by the historical query records have to be evaluated with other queries to find common subqueries between them by our proposed query aggregation scheme. The output of the query aggregation scheme is another set of queries Q' where each query $Q_i' \in Q'(1 \leq i \leq M)(M \leq N)$ is denoted by $\langle R_i', V'i, T_i', F_i'\rangle$, where R_i' represents the combined region, V_i' represents the query attribute set, T_i' is the new query time interval, and F_i' is the query period determined from several initial queries. According to Definition 9 and the region aggregation and approximate processing rules, Q is approximately equal to Q' in that the rules of approximate region aggregation and overlapped region aggregation are applied to our query aggregation scheme.

The operator *QuerAggregation*() is used to perform the aggregation function, which is the main idea of the proposed query aggregation scheme to exploit and reduce the common tasks among groups of queries Q, then recombine them into another set of queries Q'. For any two partitioned queries Q_i and Q_j in Q, *QuerAggregation*() offers the algorithm to merge Q_i and Q_j into one to perform optimization if these two queries satisfy all of the next 4 conditions. First, if R_i of Q_i and R_j of Q_j can be overlapped, that is, $R_i \cap R_j \neq \phi$ and $R_i \cap R_j = R_{ij}$, meanwhile, the region overlapping degree of R_i and R_j satisfy $S_{R_i \cap R_j}/S_{R_i \cup R_j} \geq \beta$, which means that only those queries asking for information of the approximate same region may possibly be merged into one and the aggregated region is R_{ij}. Second, if two partitioned queries ask for the same or different attribute information of the approximate same region, they can be possibly merged into one. If V_i of Q_i and V_j of Q_j are the same attribute, that

Input: A set of partitioned queries Q from users, each partitioned query $Q_i \in Q$ is denoted by $\langle R_i, V_i, T_i, F_i\rangle$, where R_i represents the query region, V_i represents the query attribute, T_i is the query time interval and F_i is the query period.

Output: A set of queries Q' where each query $Q_i{}' \in Q'(1 \le i \le M)(M \le N)$ is denoted by $\langle R_i{}', V'_i, T_i{}', F_i{}'\rangle$, where $R_i{}'$ represents the combined region, $V_i{}'$ represents the query attribute set, $T_i{}'$ is the new query time interval and $F_i{}'$ is the query period determined from several initial queries.

Begin:

for $i = 1$ to $N - 1$

for $j = i + 1$ to N

$(Q_i{}', Q'_j) = QueryAggregation(Q_i, Q_j)$ // for any two queries, aggregate them according to R_i, V_i, T_i, F_i

end for

end for

// description of query aggregation scheme $(Q_i{}', Q'_j) = \text{QueryAggregation}(Q_i, Q_j)$ for any two initial queries

Algorithm $(Q_i{}', Q'_j) = QueryAggregation(Q_i, Q_j)$

Input: $Q_i = \langle R_i, V_i, T_i, F_i\rangle$ and $Q_j = \langle R_j, V_j, T_j, F_j\rangle$

Output: $Q_i{}' = \langle R_i{}', V_i{}', T_i{}', F_i{}'\rangle$ and $Q_j{}' = \langle R_j{}', V_j{}', T_j{}', F_j{}'\rangle$

Begin:

if $R_i \cap R_j \neq \phi$ and $S_{R_i \cap R_j}/S_{R_i \cup R_j} \ge \beta$

if $V_i = V_j$

$T'_j = T_i \cup T_j$

$V'_j = V_i = V_j$

$F'_j = GCD(F_i, F_j)$ // Multirate processing

$R'_j = R_i \cap R_j$

$Q_i{}' = \phi$; $Q_j{}' = \langle R_j{}', V_j{}', T_j{}', F_j{}'\rangle$ // Merging two queries into one

else // $V_i \neq V_j$

if $T_i = T_j$

$T'_j = T_i = T_j$

$V'_j =$

$F'_j = GCD(F_i, F_j)$ // Multirate processing

$R'_j = R_i \cap R_j$

$Q_i{}' = \phi$; $Q_j{}' = \langle R_j{}', V_j{}', T_j{}', F_j{}'\rangle$ // Merging two queries into one

else // $T_i \neq T_j$

$Q_i{}' = \langle R_i{}', V_i{}', T_i{}', F_i{}'\rangle$

$Q_j{}' = \langle R_j{}', V_j{}', T_j{}', F_j{}'\rangle$

end if

end if

else //$R_i \cap R_j = \phi$

$Q_i{}' = \langle R_i{}', V_i{}', T_i{}', F_i{}'\rangle$

$Q_j{}' = \langle R_j{}', V_j{}', T_j{}', F_j{}'\rangle$

end if

Algorithm 2: Query region, attribute, and time- and frequency-based query aggregation scheme.

is, $V_i = V_j$, the attribute in merged query is V_i, otherwise, the attribute in merged query is $V_i \cup V_j$, therefore, two or more answers to different attributes information can be obtained through forwarding only one combined query into the sensor network. Third, if two partitioned queries ask for the same attribute information at the same or different time interval of the approximate same region, and if T_i of Q_i and T_j of Q_j are the same time interval, that is, $T_i = T_j$, the time interval in merged query is T_i, otherwise, the time interval in merged query is $T_i \cup T_j$. Forth, if two partitioned queries ask for the same attribute information at the same or different time interval of the approximate same region with different frequencies, and if F_i of Q_i and F_j of Q_j satisfy the condition we set in Section 5.3, the frequency in merged query is $\text{GCD}(F_i, F_j)$.

The proposed query aggregation scheme is given in Algorithm 2.

5.7. Routing Protocol. The process of query dissemination and data transmission is greatly influenced by energy considerations. Multihop routing will often consume less energy than direct communication [39]. Here, we adopted the main idea of GPSR (greedy perimeter stateless routing) in this paper. GPSR [40] is a geographic routing protocol that eliminates the need to maintain state information while performing routing, and assumes that every sensor node in the field knows the geographical location of its neighbors. In this paper, we also assume that sensors including the gateway node know the geographical location of all sensor nodes within their power range [41]. Therefore, path selection

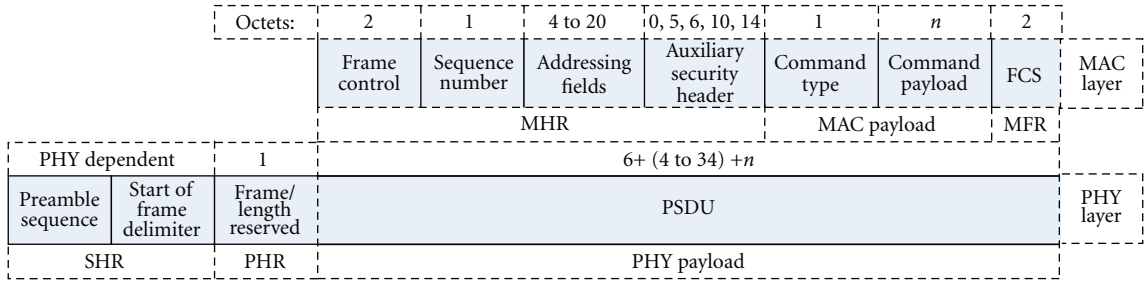

Figure 5: MAC command frame and the PHY packet.

Input: n_c is the current routing node;
p is the destination;
$n_1, n_2, \ldots, n_m$ are m neighboring nodes of n_c.
Output: next hop node n_j
Begin
for =1 to m
Computes distance from nodes in its RT to access point, $d_i = d(P, n_i)$
end for
n_j is the next hop node which satisfies minimum $\{d_i\}$

Algorithm 3: Next hop selection scheme within neighbours.

in this work is the reverse process of GPSR. Each node in sensor network maintains a RoutingTable (RT) which contains information about the nodes from which the node has heard directly, that is to say, the RT of one node maintains the information about all its neighboring nodes. When a packet arrives, for each hop between the gateway node and the access point, a node chooses its neighbor within radio range that is closest (in terms of geometric distance) to the access point as the next hop destination and forwards the message to it. When data is sent back from the sensor nodes, the reverse process applies.

After all the paths from the gateway node to each sensor node in the field are confirmed, the whole routing of the sensor network is build. In this approach, if a sensor node is selected to be the next potential routing node, but it has been in the routing path, it should go back to the last step and select another neighboring node inferior to routing node selected in last step. The purposes that we use GPSR-based routing scheme is to compare the total energy consumption with other query processing techniques, there are many routing protocols to handle both query packets and data packets, and they all performs better in energy consumption under our query management scheme in our previous study, we adopt GPSR-based routing scheme in this paper only because it is easy implemented.

6. Performance Evaluation

We first describe the assumed system settings and the energy model used to evaluate the proposed query management scheme, and then discuss the simulation results.

6.1. System Setting and Energy Model. In this section, we evaluate the performance of the proposed scheme via simulations. Unless otherwise specified, we assume that 200 sensor nodes are randomly scattered in a field with dimensions 200×200 where the gateway node is located at position $x = 100$, and $y = 100$. We assume that the ratio radius for the sensor nodes is 20 m. Every result shown is an average of 25 experiments, each using a different randomly generated position for each node. In addition, all the queries are randomly generated by a query generator.

To validate the performance of the proposed scheme, we consider a static and homogeneous sensor network. In our simulations, we calculate the query packet and data packet size according to the IEEE 802.15.4 standard, as shown in Figure 5. The size of the preamble sequence is 4 bytes, the size of the start of frame delimiter is 1 byte, the size of the addressing field is 4 bytes and the size of the auxiliary security header is 0 bytes. As a data delivery model, we simulate a query-driven sensor network in which sensor nodes report information only if a query occurs.

In our simulation, we utilize a simple model [1, 16, 42, 43] for radio hardware energy dissipation, where the energy dissipation is mainly from transmitting and receiving data. The energy consumption of transmitting each k-bit packet is calculated as

$$E_{\text{tx}}(k, d) = E_{\text{elec}} \cdot k + E_{\text{amp}} \cdot k \cdot d^{\alpha} \tag{14}$$

The energy consumption of receiving a k-bit packet is calculated as

$$E_{\text{rx}}(k) = E_{\text{elec}} \cdot k, \tag{15}$$

Table 2: Simulation parameter settings.

Parameter	Meaning
N_q	Number of queries collected from users
N_h	Number of query records in the historical database
N_{hm}	Number of queries which can be fully answered by the historical database
N_{phm}	Number of queries which can be partially answered by the historical database
N_a	Number of aggregated queries
E_{df}	Energy consumed by the DF method for N_q queries collected from users
E_{hf}	Energy consumed by the HDQF method for N_q queries collected from users
E_{hfa}	Energy consumed by the HDQA method for N_q queries collected from users
$S_{R_1 \cap R_2}/S_{R_1 \cup R_2}$	Region overlapping degree: ratio of the overlapping area to the union area
$a \times b$	Region size with length a and width b
β	A constant used to select two queries to be aggregated when $S_{R_1 \cap R_2}/S_{R_1 \cup R_2} > \beta$

where d is the message transmission distance between the sender and receiver, k is the packet size, and α is a value between 2 and 4; and in this paper, $\alpha = 2$, $E_{\text{elec}} = 50\,\text{nJ/bit}$, $E_{\text{amp}} = 100\,\text{pJ/(bit} * \text{m}^2)$.

The energy consumed for processing queries and sensing data is usually a very small portion of the total (in a common scenario, the energy consumed to process 100 million instructions is comparable to the energy required to transfer 10 bits of data). Therefore, we do not take it into consideration.

It is also important to note that the radio channel is symmetric, which means that the cost of transmitting a message from A to B is the same as the cost of transmitting a message from B to A [44].

In order to analyze the performance of our query management scheme, we compare the following query processing approaches [19] with our historical database supported query aggregation scheme (HDQA)

Direct forwarding (DF): in this approach, the gateway node just simply forwards queries to the sensor network without any aggregation.

Historical database supported query forwarding (HDQF): In this approach, the gateway node acts as a centralized database system that for each query, searches the historical database first, then forwards it to the network.

In this paper, all the initial queries are produced by a Random Query Generator (RQG) that conforms to Definition 4.

6.2. Simulation Results. We conduct four sets of experiments with the objective of evaluating the impact of the number of queries collected from users, the number of query records in historical database, and the ratio of the overlapping area to the union area and the region size. Table 2 summarizes the simulation parameter settings for all the experiments.

Before evaluating the impact of these parameters on the proposed query management scheme, we obtain N_{hm}, N_{phm}, N_a, E_{df}, E_{hf}, and E_{hfa} through 25 experiments using our algorithm, with the following parameters setting:

$$N_q = 50;\ N_h = 500;\ \beta = 0.3;\ a \times b = 20 \times 10. \tag{16}$$

The values of E_{hf} and E_{hf} obtained by the other two methods and the improvement of E_{hfa}VS E_{hf} and E_{hfa}VS E_{df} are also listed in Table 3. We ran each algorithm 25 times.

Table 3 summarizes the overall performance of the proposed algorithm. Notice that the energy consumption of HDQA is smaller than that of both DF and HDQF. HDQA reduces the energy consumption between 15% and 37% (26% on average) with respect to HDQF, and between 31% and 52% (42% on average) with respect to DF. The energy reduction comes from both historical query and query aggregation; Table 3 also shows an average of 6.64 queries which can be fully answered by historical database, an average of 7.84 queries which can be partially answered by historical database and an average of 7.44 aggregated queries. However, the variance of the values of N_{hm}, N_{phm}, and N_a obtained from our algorithm is high, because the initial queries and the historical query records are all randomly generated.

6.2.1. Impact of Query Number. By varying the number of queries collected from users, we can validate the impact of the query number on the performance of the proposed query system, and further control query redundancy and data reduction after aggregation. For example, a small number of queries have the less probability to be aggregated, while an extremely large number of queries has more redundancy among them, therefore, more aggregation will occur. In this simulation, we fix $N_h = 500$, $\beta = 0.3$, and $a \times b = 20 \times 10$. The query number N_q varies from 40 to 100.

Figure 6 compares the impact of N_q to N_{hm}, N_{phm}, and N_a using the proposed method. Observe that N_{hm}, N_{phm}, and N_a increase with N_q since more queries are processed by thus query manager and thus N_{hm}, N_{phm}, and N_a also increase. Moreover, the increase of N_{hm}, N_{phm} and N_a with N_q is nearly linear. Figure 7 compares the energy consumption using the three approaches presented above. Of course, energy consumption increases with N_q since more queries are processed. However, the HDQA approach achieves a better energy usage when compared to the other approaches.

Table 3: Comparison of the Energy consumption obtained by HDQA, DF, and HDQF.

No.	N_{hm}	N_{phm}	N_a	E_{df} (MnJ)	E_{hf} (MnJ)	E_{hfa} (MnJ)	Improvement E_{hfa}VS E_{hf} (%)	Improvement E_{hfa}VS E_{df} (%)
1	5	6	5	6676	5607	4489	20	33
2	12	6	9	6574	4602	3619	21	45
3	7	8	6	5769	4500	3404	24	41
4	9	9	10	6317	4612	3137	32	50
5	6	10	3	6074	4738	3836	19	37
6	10	12	8	6233	4114	3464	16	44
7	10	10	7	6756	4729	3855	18	43
8	4	7	10	7312	6215	4030	35	45
9	6	12	6	6877	5227	4022	23	42
10	3	6	13	6121	5387	3524	35	42
11	8	12	7	6724	4841	4043	16	40
12	8	3	7	6607	5352	3892	27	41
13	7	7	8	6251	4938	3295	33	47
14	4	6	7	7512	6310	4365	31	42
15	5	11	8	6231	4923	3461	30	44
16	6	3	6	5539	4708	3683	22	34
17	7	8	5	6245	4871	3293	32	47
18	6	8	8	6472	5177	3532	32	45
19	6	5	6	7259	6025	4540	25	37
20	4	7	5	6115	5197	4074	22	33
21	10	9	5	6392	4539	3661	19	43
22	7	11	11	7279	5168	3501	32	52
23	7	5	8	6548	5304	3718	30	43
24	2	6	7	6518	5866	4472	24	31
25	7	9	11	6809	5243	3257	38	52

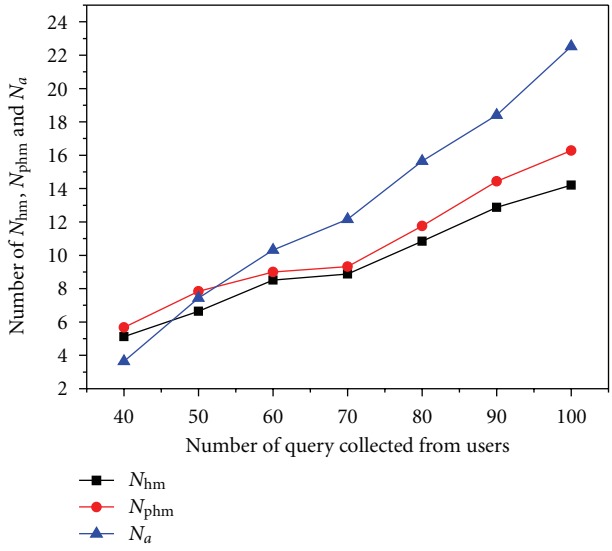

Figure 6: Impact of N_h on N_{hm}, N_{phm} and N_{phm}, N_a.

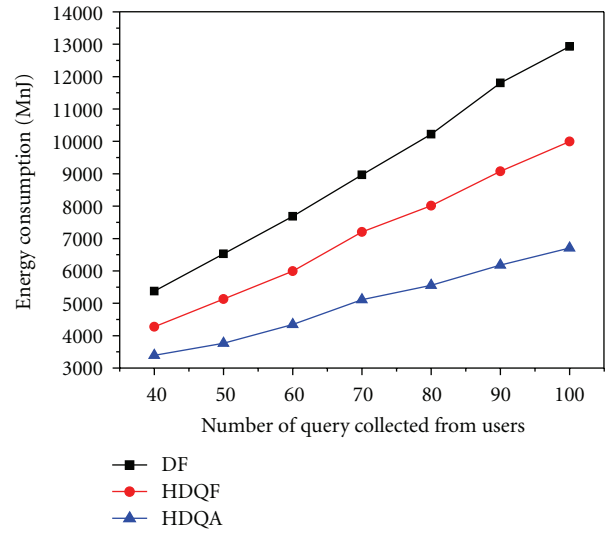

Figure 7: Energy sensitivity of total query.

6.2.2. Impact of Historical Query Record. In this experiment, we first evaluate the impact of the number of historical query records on N_{hm}, N_{phm} and N_a obtained by our algorithm, and then evaluate the total energy consumption by HDQA, DF, and HDQF. Here, we fix $N_q = 50$, $\beta = 0.3$, and $a \times b = 20 \times 10$. The historical query record N_h varies from 400 to 1000.

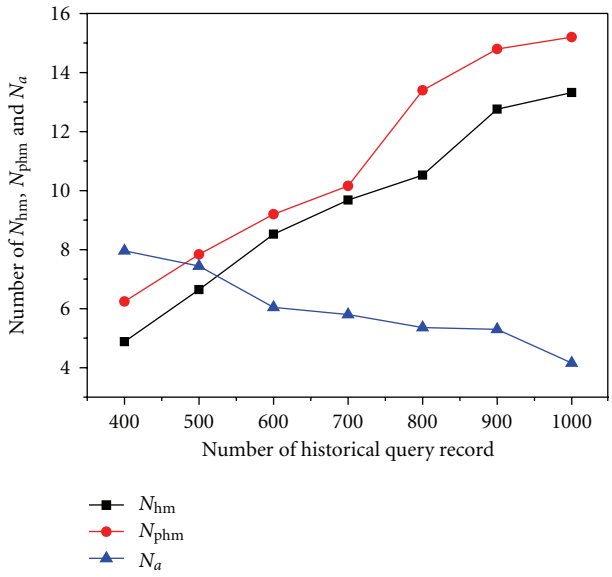

Figure 8: Impact of N_h on N_{hm}, N_{phm} and N_a.

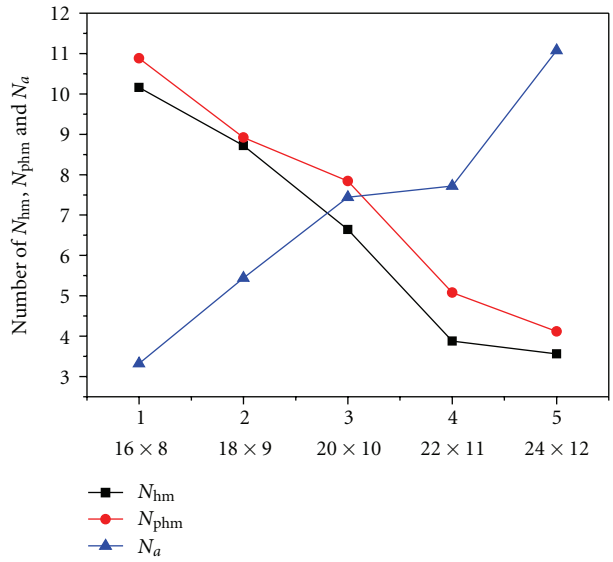

Figure 10: Impact of region size on N_{hm}, N_{phm} and N_a.

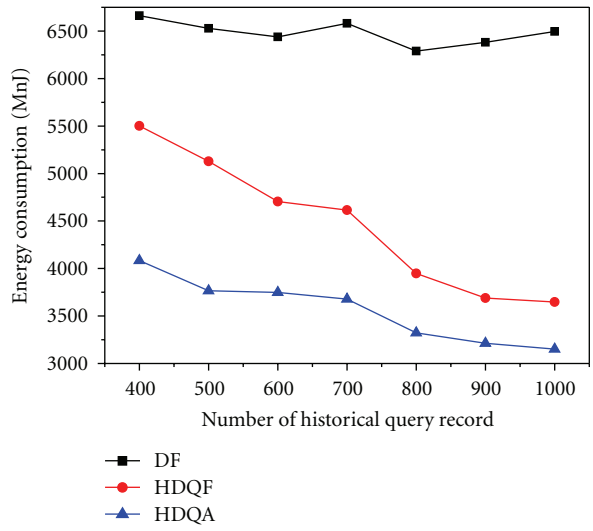

Figure 9: Energy sensitivity of historical query record.

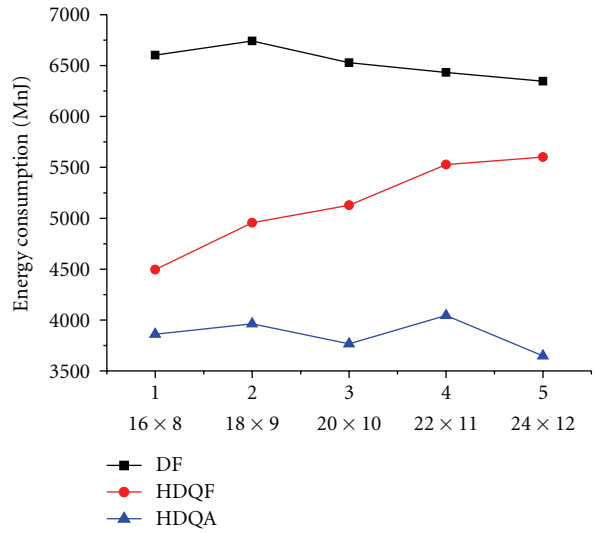

Figure 11: Energy sensitivity of query region size.

Table 4: Setting query region size.

No.	Region size $a \times b$
1	16×8
2	18×9
3	20×10
4	22×11
5	24×12

Figure 8 shows the impact of N_h on N_{hm}, N_{phm}, and N_a using the proposed algorithm. We can observe that N_{hm} and N_{phm} increases with N_q, while N_a decreases. This can be explained by the following two reasons. First, with the increase of N_h, each query has more probability to find full or partial answers from the historical database, therefore N_{hm} and N_{phm} increase with N_q. Second, queries with full answer from historical database will not participate in the process of query aggregation, which reduces the number of queries that participate in query aggregation, even if those queries with partial answer from historical database will be partitioned into subqueries, which have less probability to be aggregated with others. Thus, N_a decreases with the increase of N_h.

Figure 9 compares energy consumption using HDQA, DF, and HDQF. Observe that energy consumption using HDQA and HDQF decreases with the increase of N_h, since more queries get answers from the historical database. The energy consumption using DF does not change with the increase of N_h, because this method does not use the historical database.

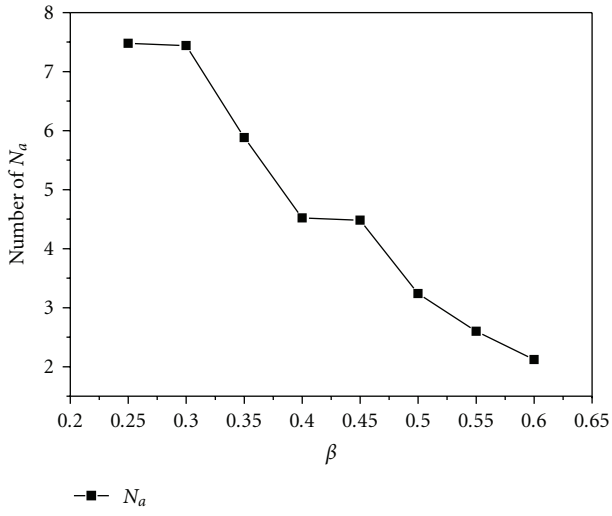

Figure 12: Impact of β on N_a.

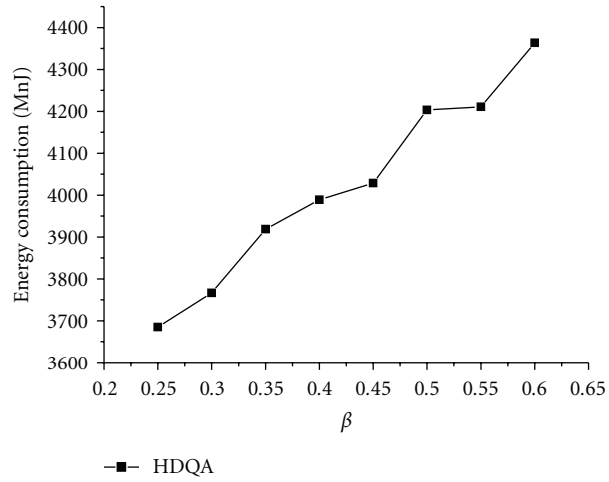

Figure 13: Energy sensitivity of β.

6.2.3. Impact of Region Size. In this analysis, we first evaluate impact of region size on N_{hm}, N_{phm}, and N_a by our algorithm, and then evaluate the total energy consumption by HDQA, DF, and HDQF. In this experiment, we fix $N_q = 50$, $N_h = 500$, and $\beta = 0.3$, The query region size is set as Table 4.

Figure 10 illustrates the impact of region size on N_{hm}, N_{phm}, and N_a on our approach. By changing query region size from 16×8 to 24×12, N_{hm}, N_{phm}, and N_a show different trends. N_{hm} and N_{phm} decrease, while N_a increases. Since a strict limit is used for the query region in our historical database, the region of one query must be inside in the region of one record query in the historical database. Moreover, the region size of the record query is randomly generated within a certain range. It is more difficult for a query with larger region size to find full or partial answer than a query with small region size. Thus, N_{hm}, N_{phm} decreases with the increase of query region size. However, two queries with larger query region size are prone to be aggregated, so N_a increases with the increase of query region size. Figure 11 shows the energy consumption by HDQA, DF, and HDQF with the change of query region size from 16×8 to 24×12. The energy consumed using DF and HDQA does not change with the control of query region size. Since N_{hm} and N_{phm} decreases and N_a increases with the change of query region size, more energy is required by HDQF, and less energy is required to process queries by aggregation.

6.2.4. Impact of Region Overlapping Degree. The region overlapping degree is only used in the query aggregation phase, and it is possible for two queries to be aggregated when $S_{R_1 \cap R_2}/S_{R_1 \cup R_2} > \beta$, therefore, we only evaluate the impact of β on N_a. In this experiment, we fix $N_q = 50$, $N_h = 500$ and $a \times b = 20 \times 10$, β is from 0.25 to 0.6.

It can be seen from Figure 12 that N_a decreases with the increase of β, because it becomes more difficult for two queries to satisfy the aggregation condition with the increase of β. Therefore, N_a decreases, while the energy consumption increases (See Figure 13). Only an average of 2 queries are aggregated when β is 0.6 while an average of 7.5 queries are aggregated when β is 0.25.

6.2.5. Comparison with IQAF and SAQA. In order to further demonstrate our proposed scheme (HDQA), we compared the performance of our scheme with that of IQAF (IQAF: Integrated Query Aggregation-based Framework) and SAQA (SAQA: Spatial and Attribute Based Query Aggregation) by applying them to our query record and query set. IQAF is an overlay-based query aggregation approach including a query manager and an effective query aggregation algorithm that was presented in [16], and the main idea of the proposed aggregation scheme is mainly based on region operation of user queries. SAQA is a region- and attribute-based query aggregation for sensor networks which was introduced in [19]. Through the process of experiments, our routing protocol was applied to IQAF and SAQA, meanwhile, both of these two approaches support our historical database query.

In this experiment, we first evaluate the total energy consumption and query latency by HDQA, IQAF, and SAQA under the impact of the number of queries from users, here, we fix $N_h = 500$, $\beta = 0.3$ and $a \times b = 20 \times 10$. The query number N_q varies from 40 to 100. Second, we evaluate the total energy consumption and query latency by HDQA, IQAF, and SAQA under the impact of the number of query record; here, we fix $N_q = 50$, $\beta = 0.3$ and $a \times b = 20 \times 10$. The historical query record N_h varies from 400 to 1000.

Figure 14 shows the data on the sensitivity of energy performance for different query number. In this figure, the x-axis represents the different query number, and y-axis represents the total energy consumption by HDQA, IQAF, and SAQA. As the query number is enlarged, the overall energy consumption by three approaches also increases. This is because a larger query number means that more queries are involved and more query/data transmissions are performed. Given a fixed number of user query, our HDQA performs better than the other two schemes in that HDQA performs more aggregation operations, thus saving energy consumption.

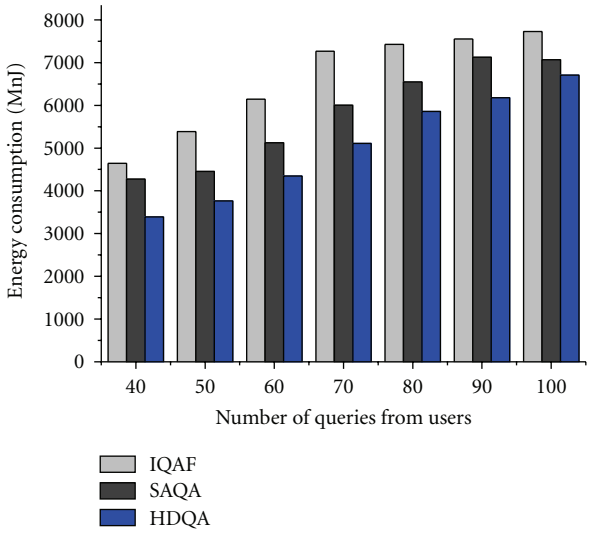

Figure 14: Energy sensitivity of total query.

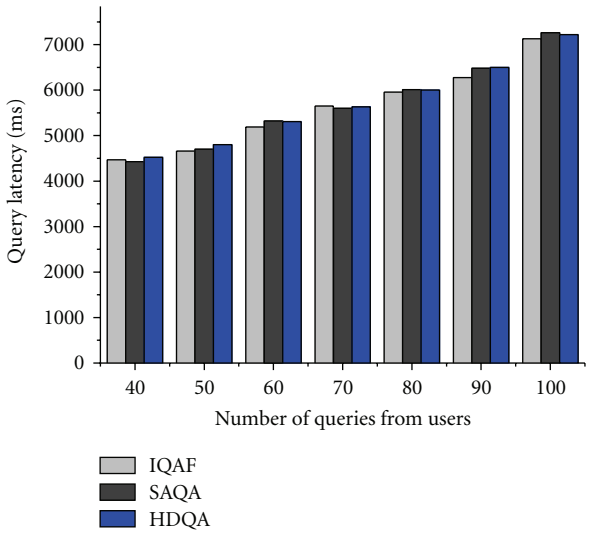

Figure 15: Query latency sensitivity of total query.

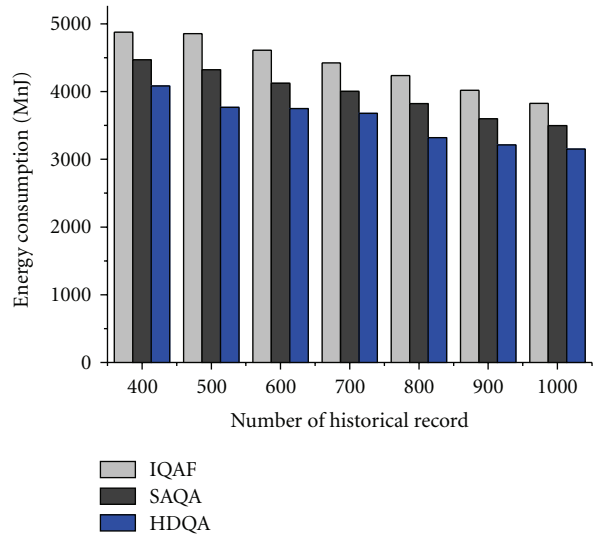

Figure 16: Energy sensitivity of historical query record.

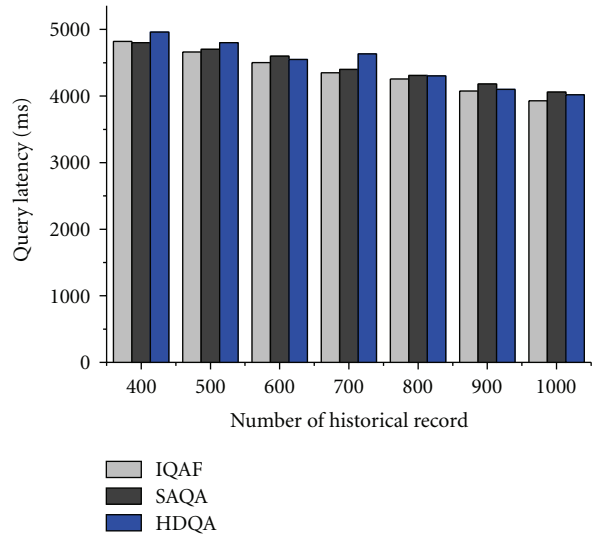

Figure 17: Query latency sensitivity of historical query record.

Figure 15 compares the query latency under different query number by HDQA, IQAF, and SAQA, it is noted that query latency here means the average elapsed time between a query being issued and results being received. In this figure, the x-axis represents the different query number and y-axis represents the query latency. As the query number increases, the query latency by three approaches also increases in that more aggregations are performed and less data and queries are issued in sensor networks. However, for any fixed query number, the query latencies caused by three approaches are almost equal because more time is used for aggregations by HDQA which simultaneously reduced the latency for data and query transmissions, while it reduces the aggregations time by IQAF which simultaneously increases the data and query delivery.

Figure 16 evaluates the sensitivity of energy performance for different historical query record by HDQA, IQAF, and SAQA. In this figure, the x-axis represents the different historical query record and y-axis represents the total energy consumption. As the historical query record increases, the overall energy consumption by three approaches decreases. This is because more query answer can be obtained from historical database and less query/data transmissions are performed. Given a fixed number of user query, our HDQA performs better than the other two schemes in that HDQA performs more aggregation operations which contributes to energy saving.

Figure 17 demonstrates the query latency for different algorithms, and varies historical query record from 400 to 1000; the query latency dramatically decreases as historical query record further increases. This is because more query answer can be obtained from historical database and less query/data transmissions are performed. However, for any fixed historical query record, the query latencies caused by

three approaches are almost equal because more time is needed for aggregation operation by HDQA which simultaneously reduces the time for data/query transmissions while less time is needed for aggregation by IQAF which simultaneously increases the time for data/query delivery.

7. Conclusions

Energy consumption is a crucial factor affecting the application and effectiveness of a wireless sensor network. In this paper, we proposed an energy-efficient query management framework that copes with multiple queries in a sensor network. In summary, our contributions include the following: (1) an energy-efficient query management framework to process multiple queries; (2) an effective historical database query method to make full use of past queries; (3) a novel query aggregation mechanism to process duplicated tasks among queries in order to save to energy. Both analytical and simulation results reveal that our strategy can lead to a significant saving of communication cost, thereby extending the effective lifetime of the sensors.

The main idea of this paper is to use an efficient query management framework to optimize complex queries, search for similar subqueries, optimize each representative sub-query, and share the optimization result with other similar subqueries through aggregation techniques and historical database queries. However, the query execution plan generated by the query management scheme is based on collecting a group of queries, which will lead to latency in query answering, especially for the queries collected at the early stage. Moreover, the query management scheme performs a significant amount of preprocessing and postprocessing work, which also directly affects the query response time.

There are several directions to extend our study. First, in the routing protocol, we use a greedy routing mechanism based on GPSR to handle both query packets and data packets, which is not the best solution for energy-balanced routing. Therefore, it would be interesting to analyze the impact of an energy-balanced routing algorithm to further optimize energy. Second, we used randomly generated initial queries and historical query records to evaluate our query management scheme. Thus, analyzing the performance with a real dataset is another possible direction.

Acknowledgments

This paper is partially supported by a research grant from the National Science Foundation of China under Grant no. 70701025, the Doctoral Foundation for young scholars of Education Ministry of China under Grant no. 20070056002, the Program for New Century Excellent Talents in Universities of China under Grant no. NCET-08-0396, and a National Science Fund for Distinguished Young Scholars of China under Grant no. 70925005. The authors would like to express great appreciation to Professor Luciano Lavagno for paper editing and his valuable comments on improving the quality of this paper.

References

[1] S. Ci, M. Guizani, and H. Sharif, "Adaptive clustering in wireless sensor networks by mining sensor energy data," *Computer Communications*, vol. 30, no. 14-15, pp. 2968–2975, 2007.

[2] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 239–249, November 2004.

[3] K. Kalpakis, K. Dasgupta, and P. Namjosh, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, 2003.

[4] X. X. Huang, H. Q. Zhai, and Y. G. Fang, "Robust cooperative routing protocol in mobile wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5278–5285, 2008.

[5] K. Zeng, K. Ren, W. Lou, and P. J. Moran, "Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply," *Wireless Networks*, vol. 15, no. 1, pp. 39–51, 2009.

[6] C. Pandana and K. J. R. Liu, "Robust connectivity-aware energy-efficient routing for wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 10, pp. 3904–3916, 2008.

[7] P. Edara, A. Limaye, and K. Ramamritham, "Asynchronous in-network prediction: efficient aggregation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 4, article 25, 2008.

[8] A. Brayner, A. Lopes, D. Meira, R. Vasconcelos, and R. Menezes, "An adaptive in-network aggregation operator for query processing in wireless sensor networks," *Journal of Systems and Software*, vol. 81, no. 3, pp. 328–342, 2008.

[9] L.-H. Yen and Y.-M. Cheng, "Range-based sleep scheduling (RBSS) for wireless sensor networks," *Wireless Personal Communications*, vol. 48, no. 3, pp. 411–423, 2009.

[10] V. P. Sadaphal and B. N. Jain, "Random and periodic sleep schedules for target detection in sensor networks," *Journal of Computer Science and Technology*, vol. 23, no. 3, pp. 343–354, 2008.

[11] J. M. Zhu and X. D. Hu, "Improved algorithm for minimum data aggregation time problem in wireless sensor networks," *Journal of Systems Science and Complexity*, vol. 21, no. 4, pp. 618–628, 2008.

[12] H. F. Chen, H. Mineno, and T. Mizuno, "Adaptive data aggregation scheme in clustered wireless sensor networks," *Computer Communications*, vol. 31, no. 15, pp. 3579–3585, 2008.

[13] Y. J. Zhu, R. Vedantham, S.-J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 354–369, 2008.

[14] N. Trigoni, A. Guitton, and A. Skordylis, "Interplay of processing and routing in aggregate query optimization for sensor networks," in *Proceedings of the International Conference of Distributed Computing and Networking*, vol. 4904 of *Lecture Notes in Computer Science*, pp. 401–415, 2008.

[15] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pp. 575–578, 2002.

[16] W. Yu, T. N. Le, J. Lee, and D. Xuan, "Effective query aggregation for data services in sensor networks," *Computer Communications*, vol. 29, no. 18, pp. 3733–3744, 2006.

[17] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman, "Multi-query optimization for sensor networks," in *Proceedings of the IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS '05)*, vol. 3560 of *Lecture Notes in Computer Science*, pp. 307–321, 2005.

[18] S. L. Xiang, H. B. Lim, K.-L. Tan, and Y. L. Zhou, "Two-tier multiple query optimization for sensor networks," in *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*, p. 39, Toronto, Canada, June 2007.

[19] Y. Jie, Y. Bo, S. Lee, and J. Cho, "SAQA: spatial and attribute based query aggregation in wireless sensor networks," *Lecture Notes in Computer Science*, vol. 4096, pp. 15–24, 2006.

[20] A. Woo, S. Madden, and R. Govindan, "Networking support for query processing in sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 47–52, 2004.

[21] B. Scotney and S. McClean, "Database aggregation of imprecise and uncertain evidence," *Information Sciences*, vol. 155, no. 3-4, pp. 245–263, 2003.

[22] O. Vechtomova and H. Zhang, "Articulating complex information needs using query templates," *Journal of Information Science*, vol. 35, no. 4, pp. 439–452, 2009.

[23] C. Goss, S. Lowenstein, I. Roberts, and C. DiGuiseppi, "Identifying controlled studies of alcohol-impaired driving prevention: designing an effective search strategy," *Journal of Information Science*, vol. 33, no. 2, pp. 151–162, 2007.

[24] I. H. Toroslu and A. Cosar, "Dynamic programming solution for multiple query optimization problem," *Information Processing Letters*, vol. 92, no. 3, pp. 149–155, 2004.

[25] N. N. Dalvi, S. K. Sanghai, P. Roy, and S. Sudarshan, "Pipelining in multi-query optimization," *Journal of Computer and System Sciences*, vol. 66, no. 4, pp. 728–762, 2003.

[26] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe, "Efficient and extensible algorithms for multi query optimization," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD '00)*, pp. 249–260, Dallas, Tex, USA, 2000.

[27] T. K. Sellis, "Multiple query optimization," *ACM Transactions on Database Systems*, vol. 13, no. 1, pp. 23–52, 1988.

[28] D. Chatziantoniou and K. A. Ross, "Partitioned optimization of complex queries," *Information Systems*, vol. 32, no. 2, pp. 248–282, 2007.

[29] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Multi-query optimization for sketch-based estimation," *Information Systems*, vol. 34, no. 2, pp. 209–230, 2009.

[30] S. L. Xiang, H. B. Lim, and K. L. Tan, "Impact of multi-query optimization in sensor networks," in *Proceedings of the 3rd Workshop on Data Management for Sensor Networks*, pp. 7–12, 2006.

[31] Y. W. Chen, M. Xu, H.-M. Wang, et al., "An energy-efficient framework for multirate query in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, Article ID 48984, 10 pages, 2007.

[32] L. Uichin, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 882–901, 2009.

[33] X. Li, W. Shu, M. L. Li, H.-Y. Huang, P.-E. Luo, and M.-Y. Wu, "Performance evaluation of vehicle-based mobile sensor networks for traffic monitoring," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 1647–1653, 2009.

[34] X. Li, H. Y. Huang, W. Shu, M. L. Li, and M.-Y. Wu, "Vstore: towards cooperative storage in vehicular sensor networks for mobile surveillance," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '0909)*, pp. 1–6, April 2009.

[35] Y. J. Zhu, R. Vedantham, S.-J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 354–369, 2008.

[36] P. Kalnis and D. Papadias, "Multi-query optimization for online analytical processing," *Information Systems*, vol. 28, no. 5, pp. 457–473, 2003.

[37] T. M. Gil and S. Madden, "Scoop: an adaptive indexing scheme for stored data in sensor networks," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 1345–1349, April 2007.

[38] C. Y. Ai, R. Y. Du, M. H. Zhang, and Y. S. Li, "In-network historical data storage and query processing based on distributed indexing techniques in wireless sensor networks," *Lecture Notes in Computer Science*, vol. 5682, pp. 264–273, 2009.

[39] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[40] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 243–254, August 2000.

[41] S. B. Wu and K. S. Candan, "Power-aware single- and multipath geographic routing in sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 974–997, 2007.

[42] L. F. Yuan, W. Q. Cheng, and X. Du, "An energy-efficient real-time routing protocol for sensor networks," *Computer Communications*, vol. 30, no. 10, pp. 2274–2283, 2007.

[43] Y. Jin, L. Wang, Y. Kim, and X. Z. Yang, "EEMC: an energy-efficient multi-level clustering algorithm for large-scale wireless sensor networks," *Computer Networks*, vol. 52, no. 3, pp. 542–562, 2008.

[44] H. O. Tan and I. Korpeoglu, "Power efficient data gathering and aggregation in wireless senor network," in *Proceedings of the ACM Special Interest Group on Management of Data*, pp. 66–71, 2003.