Multiple Query Optimization based on Data Prediction in Wireless Sensor Networks

Hanieh Bagheri
Computer Engineering and Information Technology dept.
Amirkabir University of Technology
Tehran, Iran
hbagheri@aut.ac.ir

Majid NourHosseini
Computer Engineering and Information Technology dept.
Amirkabir University of Technology
Tehran, Iran
majidnh@aut.ac.ir

Abstract— **multiple query optimization is one of the most recent fields in query processing of wireless sensor networks. It is very likely for a network to be queried by multiple users simultaneously while these queries may have many commonalities and overlapping data. The overlapping data includes geographical locations, data acquisition conditions or specified actions in the query plans. Using the traditional techniques, the queries will be executed independently and execution of redundant and repetitious operations by sensor nodes leads to energy wastage in the sensor network. Therefore, it is possible to prevent the network from consuming too much energy, by applying multiple query optimization methods. In this work, a prediction based component is used for making decisions about the way of merging queries, in environment monitoring applications. Simulation results demonstrate a considerable reduction of transmitted messages and energy consumption in wireless sensor networks.**

Keywords: multiple query, energy optimization, query rewriting, data prediction, network map

I. INTRODUCTION

Wireless sensor networks have found applications in a wide area of fields such as environment monitoring, military applications, traffic control etc., in order to enable users to monitor and interact with the physical world. Since a sensor network generates a large amount of data, sensor nodes are provided with the capability of running queries about the environment. Therefore, each node is a data source and the sensor network can be treated as a database.

In a wireless sensor network, users pose their requests as queries and the base station injects them into the network. Each sensor node senses the environment in given intervals, evaluates incoming queries and answers the ones their condition is satisfied. A routing tree is used to send queries and their likely results between nodes.

In traditional databases, the main purpose of query processing is answering the queries at the shortest possible time. In a wireless sensor network, because of the battery powered nature of sensing devices, energy limitation becomes a significant consideration. As a result, the main challenge is to answer queries effectively, by reducing data transmissions over the network. Another difference between sensor networks and traditional databases is the format of queries; if the queries obey some syntax similar to SQL, in addition to SELECT-FROM-WHERE clauses, there are two extra EPOCH-DURATION ones, which specify that a query must be executed at a given frequency for a specific time [1]. The mentioned differences cause the techniques of query processing in sensor networks to be very different from the ones applicable to traditional databases. In-network processing and data aggregation are two examples of these techniques [2, 3]

The cost of processing a query includes the cost of needed transmissions to propagate the query in the network, the cost of processing in nodes and communication cost of result messages from the responding nodes to the base station. Since the communication cost between nodes is noticeably greater than the cost of processing data in a single node, it is possible to ignore the processing cost. Thus, the total cost of processing multiple queries is equal to the sum of the costs corresponding to the process of each single query; but it is highly expected that similar data is requested by multiple queries.

One of the most recent fields in query processing of wireless sensor networks is multiple query optimization. Earlier methods of query processing were completely concentrated on processing a single independent query. Growth of sensor networks over recent years leads to increasing the number of users, so it is not far that the base station will be expected to process multiple queries concurrently while these queries may have many commonalities and overlapping data. The overlapping data includes geographical conditions, data acquisition conditions or specified actions in query plans. If traditional techniques are used in this case, the queries will be executed independently and executing redundant and repetitious operations by sensor nodes leads to energy wastage in the sensor network. Moreover, if multiple queries are executed independently in the network, a separate routing tree is formed for each of them; so, it is possible for a sensor node to be a member of multiple routing trees, therefore, there will be multiple parents and multiple paths for transmitting the data. In this structure, a node may do similar operations many times and send the same result more than once in various paths. Thus, the network will confront a large energy gradient. Using multiple query optimization, we can prevent the network from such an unpleasant situation.

The work described in this paper was supported by the Iran Telecommunication Research Center

978-1-4244-8185-9/10/$26.00 ©2010 IEEE

Different multiple query optimization techniques are proposed to merge queries to eliminate redundancy of results and operations.

In this paper, we have proposed a multi query optimization method for environment monitoring applications which constructs a map in the base station, which presents a general view of the current values of sensed data of nodes in the sensor network. This map will be used as a decision aid component in the multiple query optimization system. This approximated map provides the query management unit with better information about overlapping queries and redundant results. Using this auxiliary information, selection of appropriate queries to be rewritten together will be a more intelligent process.

Previous works in the field of multiple query optimization can be classified into two categories: First the ones that do the merging according to the overlapped range of the queries. The methods of this category, generally, don't consider the observations related to the sensed values of nodes in various times. In some of the studies, a constant statistical distribution in various levels of routing tree is supposed for each environmental attribute; whereas, the sensed values of different levels don't often obey a specific statistical distribution or by passing time, the model may change significantly. In the second category, the nodes decide locally about identification of the result overlapping queries and merging or eliminating redundant queries. There is only one research [13] which studies the optimization in the range based manner also in a network status one, but these two approaches are applied independently in two distinct layers. We have tried in this paper to take into account semantic factors in merging, in addition to superficial resemblances. For better understanding of the necessity of this work, consider the following examples:

Example 1. Suppose three queries have arrived to the base station simultaneously. For simplicity, we assume they have the same frequency and the same duration. The term *temp* represents the temperature attribute and x, y and z are the desired values from the sensor nodes when the conditions of each query is satisfied.

$$Q_1 = SELECT\ x\ FROM\ network\ WHERE\ 10 < temp < 20$$

$$Q_2 = SELECT\ y\ FROM\ network\ WHERE\ 15 < temp < 25$$

$$Q_3 = SELECT\ z\ FROM\ network\ WHERE\ 20 < temp < 30$$

If we intend to select the more similar ones for merging without paying attention to the sensor nodes and the values they have sensed, there will be no priority or preference for selecting the two more similar ones. But, if somehow we know that the temperature is between 15 and 20 for 10 nodes and it is between 20 and 25 for 50 nodes, now it can be said that merging Q_2 and Q_3 is better than merging Q_1 and Q_2.

Example 2. Suppose three queries have arrived to the base station simultaneously and the common range of Q_1 with Q_2 and Q_3 is as follows, respectively:

$$Q_{12} = \text{SELECT x FROM network WHERE } 10 < temp < 20$$

$$Q_{13} = \text{SELECT y FROM network WHERE } 20 < temp < 100$$

If we decide about the way of merging queries without noticing the sensed values for temperature in the network, Q_{13} sounds better than Q_{12}, because it covers a wider range; but if we know that the number of nodes which answer to Q_{12} is about 30 and the number of nodes which answer to Q_{13} is 3, it might be wiser to revise our decision.

As we know, there are three general models for data collection in a wireless sensor network: Pure Push, Pure Pull, and Hybrid Push-Pull. In the Pure Push model, nodes are programmed to send their readings proactively to the base station. In the Pure Pull model, users inject determined long time queries to the network via the base station. In this model, queries are propagated to the sensors to pull the data toward the base station on demand. In the Hybrid Push-Pull model, the nodes transmit data toward the base station, proactively, but in the case that the data is not precise enough to answer all the supposed queries, additional information is obtained from the network using an on demand approach [4].

We use a lightweight Hybrid Push-Pull method to extract the needed information from the network to build an approximated map of the network situation in the base station [16]. The constructed map can act as a guide that tells the base station which nodes of the network are most likely to answer to a certain query. The obtained information should be about the sensed values from environment in various nodes of the network or their neighbors at a recent time. In this way, the base station can detect which queries get results from similar regions of the network and therefore, the ones which have appropriate overlapping in terms of range and semantic similarity can be selected for merging and rewriting together.

If all the nodes of the network send their data to the base station independent of each other, a huge amount of data will be gathered that is not requested by any user; so energy is consumed unnecessarily. Thus, we have tried to build a sufficiently accurate map of the network situation, so the minimum extra data is pushed to the base station. To achieve this goal, a multiple linear regression model is used which exploits the spatial and temporal correlations of sensor nodes to predict the sensed data of the other nodes. A node transmits its data, only if it exceeds the acceptable bound for the verified attribute. Finally, each query is merged with the appropriate queries, according to the map, using a greedy heuristic algorithm.

The remainder of this paper is organized as follows: Section II introduces the main existing approaches for multiple query optimization. Section III discusses the architecture of multiple query optimization based on a prediction model. Section IV presents a detailed experimental evaluation for our approach. Finally, section V summarizes our findings.

II. RELATED WORKS

Many sensor network query systems, such as TinyDB [5] and Cougar [6], are developed by database research society. Beside the mentioned systems, many research studies have investigated techniques for query processing in sensor networks. Energy efficient routing protocols, in-network query processing techniques, approximate data query processing, strategy adaptive techniques and plan optimization over time

are some of these techniques. Most of these studies are concentrated on optimizing and executing a single long term query. Just a few studies have been carried out in the field of multiple query optimization.

Demers et al studied the effect of different routing trees in data aggregation [7]. In this work, multiple query optimization is done in the nodes of the network. This method should detect when to share partial data between different queries and how the redundant information should be eliminated across the path. A proper encoding method is also used to send the minimum volume of data to the base station.

Trigoni et al presented a formal model for multiple query optimization in the sensor networks for the first time [8]. The concentration of this work is on region based aggregation queries. Arrived queries are not sent to the nodes immediately; instead, the query optimizer in the base station batches the ones with the same aggregation operator into a single group and optimizes each group independently. The main idea of this approach is using linear reduction and a combinational method for reducing the number of regions that are necessary to execute the queries.

Muller at el, considered the multiple query optimization as a rewriting and merging queries problem [9, 10]. The idea of this approach is to share the sensor network among multiple queries. This model contains a processing unit in the base station which merges all the queries together to construct a network query. The user query must be a subset of the network query. In other words, the network query must cover all of the user queries. Also, the sampling frequency of the network query has to be the greatest common devisor of all the sampling frequencies of the user queries. The network query is injected into the network and the nodes return the network result to the base station. Then, the corresponding result of each user is extracted to be delivered. The main advantage of this method is that each node of the network just belongs to a single routing tree and there is no possibility of having multiple parents or paths for propagating results.

Another approach is dividing the queries into two classes: backbone and non-backbone [11]. The backbone queries are propagated in the normal manner and should share their partial results with the queries of non-backbone set. The main goal of this algorithm is to determine the backbone tree and the number of its members in a way that the number of total transmitted messages in the network is minimized. In order to solve this, the problem is mapped to a Max-Cut problem. Having a set of queries, a graph is formed that each of the vertices represents a query and the weight of each edge shows the number of reduced messages in effect of sharing partial results of the two corresponding queries. According to the obtained graph, a heuristic algorithm is used for backbone selection in order to choose the best cut of backbone queries.

TAMPA is a taboo search based algorithm for multiple query optimization which looks for the optimal order of merging queries [12]. Taboo search is a local search algorithm which in addition to cost reduction situation, it also considers cost increasing, in order to escape from local optimum points. The proposed plan uses a repetitive algorithm to search through the appropriate combinations until it converges to an optimal value.

TTMQO is a two layer multiple query optimization method [13]. This system supports both aggregation and data acquisition queries. The first layer optimization takes place in the base station and uses a cost based heuristic algorithm to rewrite user queries in the form of synthetic queries before injection into the network, such that the data requests which have multiple copies in the user queries can be reduced as much as possible. The second layer optimization occurs in the network. The idea of this type of optimization is to focus on the data required by all (synthetic) queries during specific time intervals, and to design a good DAG over the sensor nodes with the base station as the sink point to gather the queried data.

EEMQO is a framework for energy sensitive optimization and processing of queries in sensor networks [14]. The suggested method for multiple query optimization in this work can be considered as a refinement for some other relevant existing works. In this research, it has been tried to increase the optimizer precision of query processing and its sensitivity to energy consumption by considering all query execution phases. Also, by offering some solutions for query dissemination and result propagation, the number of the queries which overlap with each other and consequently the amount of redundant actions is reduced.

Similarity based optimization is another scheme [15] that given a set of queries, it constructs a set of shared intermediate views (SIV). Each SIV identifies a set of shared data among queries. The SIVs are processed only once, but reused by at least two queries. This scheme just verifies range similarities to merge queries.

III. PREDICTION BASED MULTIPLE QUERY OPTIMIZATION SYSTEM

For deciding about the best order to merge and rewrite queries, we have constructed a map, which illustrates a view of the general situation of the nodes in the network at a recent time. The data of this table can be supplied in one of the following ways:

- Results of the previous queries
- Pushed data from the sensor nodes
- Prediction of values from available information of the temporal or spatial neighbors

The base station and the sensor nodes, both participate in the proposed algorithm:

- The nodes collect data in order to construct and update the network map.
- The base station creates the map, updates it by receiving new packets from the network, and makes the appropriate decision about the order of merging.

A. Push based Data Collection in Sensor Nodes

Push based data collection is put into practice because of two properties of the sensor network.

1) *Temporal correlation of the sensed values at a node:* Each node senses similar data values at two close time instances.
2) *Spatial correlation of sensed value at two different nodes:* Two close nodes sense similar data values at the same time

Two sensed data instances of a node are temporally similar at times t_1 and t_2, if their difference is less than than α, where α is a positive real value, called temporal similarity constant.

The sensed data instances of two neighbor nodes are spatially correlated, if their difference is less than β, where β is a positive real value, called spatial similarity constant.

In query processing systems, a sensor node sends data, only if it receives a query and can satisfy its conditions. Therefore, it takes some time for the queries to be propagated in the network and for the results to be routed back to the base station. Even after this time passes, many nodes may exist that do not satisfy the conditions of the received queries. Thus, the base station has not obtained information from many regions of the network after a pretty long time and the map has no view of that region. According to these mentioned facts and in order to speed up the process of building the map in the base station, at the beginning, when the nodes are not involved in processing of queries, each of them, proactively, sends its sensed data to the base station, according the following three steps:

1) Each node sends the sensed data with probability $\frac{1}{m+1}$, where m is the number of its neighbors.
2) If neither a node nor its neighbors send their data, the first step is repeated again; otherwise if the node did not send data and none of the neighbors that have sent data in the previous step satisfy the spatial similarity condition, the node sends data once more, with probability $\frac{1}{m-k+1}$, where k is the number of neighbors that have sent data in the previous step.
3) If neither a node nor its neighbors send data in these two steps, or the spatial similarity condition is not met for at least one of the neighbors that has pushed data, the node proceeds to send its data in the network (with probability 1).

It is visible that, at the beginning, a node has little chance to send its data. If at least one of its spatial neighbors proceeds to send data, its chance to transmit, decreases to zero in the next round. Otherwise, it transmits its sensed data with a probability less than or equal to the probability of the first round. In the first step, we deal with a large number of nodes which has no proxy in the base station. Thus, even if each of the nodes has a little chance for data transmission, the pushed data resulted from this step can cover vast regions of the network. In the next step, there are fewer uncovered regions, so a greater value of probability can be assigned to the members of these regions.

By increasing the number of steps of this algorithm, the network can be covered using as few transmissions as possible, but for our desired goal, three steps sounds to be adequate, and in the third step, if there is a node covered by neither itself nor its spatial neighbors, it sends the sensed data directly, without calculation of any probability.

After the above three steps, the nodes take part in data gathering in a different manner, which follows:

Suppose Threshold (Tr) is the average time that after its passage, the data keeps its temporal similarity with a sensed value, and passed Time (PT) is the time passed since the last moment that a node or one of its spatial neighbors have answered a query or pushed some data. There is a Tr and a Pt variable for each measured attribute in each node of the network. Pt is increased one unit after each sense operation. Whereas a node or one of its spatial neighbors sends any data, pt is reset to 0. The nodes push their data toward the base station, if pt exceeds Tr.

B. Base Station Algorithm

Network map is a table in the base station, including a row for each node. Each row consists of ID, location, value and the time that each attribute of a node has been measured. By arrival of each result or push packet, the base station updates the corresponding rows in the map. For nodes that no relevant packet exists, the value should be predicted from temporal and spatial similarities using an approximation method. Pushing data in the sensor nodes, using the mentioned algorithm guarantees that every region of the network has always at least one proxy in the map to assist the prediction process.

In order to predict the value of a node, we have used a combination of a linear regression and an inverse distance weighted average method. Therefore, by receiving a relevant packet, the corresponding information of the linear regression should be updated; Assume $y = a + bt$ as the predictor function of attribute y at time t, where b and a are resulted from (1) and (2), respectively.

$$b = \frac{N(\sum ty) - \sum t \sum y}{N \sum t^2 - (\sum t)^2} \quad (1)$$

$$a = \bar{Y} - b\bar{X} \quad (2)$$

By arrival of each packet, $\sum ty$, $\sum t \sum y$, $\sum t^2$and $(\sum t)^2$ are updated and when prediction is needed, a and b are calculated, then y is approximated at the desired point. This operation approximates the value of a node using its previous values. For forecasting the value of a node that there is not sufficient data about it, a weighted average method is applied to the approximated values of its spatial neighbors. The weighted average is obtained from (3)

$$y = \sum_{i=1}^{n} w_i y_i \quad (3)$$

where w_i is the assigned weight to the ith neighbor, and n is the number of the temporal-spatial neighbors. Value of w_i is computed from (4)

$$w_i = \frac{\left[\frac{R-h_i}{Rh_i}\right]^2}{\sum_{i=1}^{n}\left[\frac{R-h_i}{Rh_i}\right]^2} \quad (4)$$

where h_i, is the distance between the interpolated point and the i th scattered point, and R is the distance between the interpolated node and the furthest scattered neighbor.

As mentioned in section I, the proposed method is aimed at being used in environment monitoring applications; therefore, in order to predict attributes that exploit both temporal and spatial correlations, i.e. temperature and humidity, the above method can be applied; but for attributes such as light that have no spatial correlation with their neighbors, only temporal correlation is used; thus, in this case, the value of the attribute in one node is obtained from the linear regression of its value at previous time instances.

The base station can use the constructed map to:

- Send queries just to the nodes that may answer them with higher probability, instead of flooding them in the network. This facility is not used, because generating approximated results is not the objective in this work.
- Detect that each query may be answered by which nodes and select queries with more common answering nodes for merging and rewriting.
- Keep constant attributes, such as ID, x and y of nodes in the map. If a query has one of these attributes in its SELECT clause, and if the WHERE clause is satisfied, some transmission can be saved by eliminating these parts from the result message. Just knowing that the query is answered by this node, these fields can be extracted from the map when the result is ready to be delivered to user.

C. Cost Model

The main energy consuming operation in sensor networks is data transmission, and the energy consumption of other operations is ignorable. Thus, it is aimed to minimize energy consumption, by reducing the transmitted messages in the network [13].

Radio messages consist of query result transmission, query propagation and abortion, and periodical network maintenance messages. For continuous queries, result transmission messages dominate, so as it is done in [13], we only count the result message transmissions in our cost model, and to be realistic, the effect of other radio message transmissions are included in the cost of radio transmission in the experimental study.

For calculating the obtained benefit from merging two queries, it is necessary to model the cost of data transmission in the network. Our cost model is a revised version of the model used in [13].

If $F_j = \{f_1, .., f_s\}$ is the set of the members of SELECT clause of query q_j and $\forall k, f_k \epsilon F_j$, $len(f_k)$ demonstrates the needed number of bits to send this field in the network, and the cost of transmission of the generated results for q_j from one node to another can be estimated as (5)

$$Cost_{cell}(q_j) = C_{start} + C_{trans} \sum_{i=1}^{f_s} len(f_j) \qquad (5)$$

where C_{start} is the transmission startup cost and C_{trans} is the transmission cost of each data unit.

Therefore, the overall cost of routing results of q_j from the source node to the base station is estimated from (6).

$$Cost_{Map}(q_j) = \sum_{i=1}^{N} \frac{Cost_{cell}(q_j).hop_i.duration_i}{epoch_i} \qquad (6)$$

If the resulted query from rewriting q_i and q_j, is called q_{ij}, then q_{ij} ought to be a superset of both q_i and q_j to ensure that no data is lost. To achieve this constraint, WHERE clause in q_{ij} must be the union of the corresponding clauses in q_i and q_j, and the EPOCH clause must be the greatest common devisor (gcd) of the corresponding clause in the two queries [13].

If q_i and q_j are aggregation queries (and hence q_{ij}), in order to derive results for both q_i and q_j from the result of q_{ij}, the two queries must have the same predicates. Thus, the integration of two aggregation queries in this way is guaranteed to be beneficial and we do not need to estimate their benefit [13].

The benefit of the integration is estimated by (7).

$$benefit(q_1, q_2) = cost_{\mathrm{Map}}(q_1) + cost_{\mathrm{Map}}(q_2) - cost_{\mathrm{Map}}(q_{12}) \qquad (7)$$

D. Multiple query Optimization using the Network Map

A greedy heuristic algorithm is used to rewrite the queries together to obtain an optimized combination of them. The applied algorithm is much similar to the one that is used in the first tier of TTMQO [13] system because of their close similarity it is not described completely here. This algorithm works as follows:

Having a list of merged queries in the base station that are running in the network, as a new query arrives, if the list is empty, the query is inserted directly to the list; otherwise, it looks in the list for the most beneficial query to be rewritten with. The obtained benefit of merging is calculated using (7), and if the resulted value is greater than zero, then the merging of these two queries is taken into account as beneficial. If none of the list members can produce the necessary benefit, the arrived query is added directly to the list.

IV. TEST AND EVALUATION

For evaluating the effect of our method of multiple query optimization, we have developed the evaluated systems using C#.NET and used the Intel Berkeley Research Lab dataset (http://db.csail.mit.edu/labdata/labdata.html). This dataset contains collected data from readings of 54 nodes along with humidity, temperature and light every 31 seconds.

For evaluating the efficiency of the proposed method, the workload of the network must have the capability of covering all different types of queries. For this aim, we have introduced three different types of queries: light-weight, middle-weight and heavy-weight queries. The criterion for this query categorization is the coverage amount of the network. For separating the queries based on this criterion, there are two ways:

Separation based on the range values of query: In this method, queries which have a small range of values are considered light-weight, queries which cover a large range of values, are considered heavy-weight and the rest, based on the definition of smallness and largeness in the range of the values

are considered middle-weight. This method of categorization is simple in implementation but according to examples 1 and 2 in section I, a categorization based on range, is not accurate and does not meet our goal, which is the measurement of the workload of the network.

Separation based on selectivity of query: In this method, queries to which only a few numbers of nodes respond are considered light-weight, likewise, the queries to which large numbers of nodes respond are heavy-weight, and the rest based on our definition of low and high are considered middle-weight. It is obvious that until the query has not entered the network and its response from nodes has not reached the base station, the identification of query type is not possible in this method. Therefore, for the identification of the query type, we have used the statistical information of the network. Before examining the multiple query optimization methods, we have tested the set of normal queries of the network with Simple algorithm (a simple query processing method with no multi query optimization facility), and so, based on its result, the type of each query is identified. We will assign a weight to each query using (8).

$$W = \frac{n.dur}{N.epoch} \tag{8}$$

In this equation, n demonstrates the number of responsive nodes, N, the total number of nodes, dur, the length of the query and epoch, the period of the query. In this experiment, after calculating this value for each normal query, we will put the query in one of the above mentioned categories. For studying the effect of this suggested method, all of the three following algorithms have been applied on normal, light-weight, and heavy-weight queries.

- Simple: Simple cases in which the processing of each query is done independent of the other queries. In fact, in this case, multiple query optimization is not performed.
- RMQO: Range based Multiple Query Optimization. The case in which the optimization of the multiple queries is done based on the visual criteria and range overlapping (Like the optimization in the first tier of TTMQO)
- PMQO: Prediction based Multiple Query Optimization. The case in which the optimization of multiple queries is done based on the semantic criteria and with the use of approximate data from the network map.

Generally, the main criteria for comparing the different ways of optimizing multiple queries is the number of sent messages through the network. The fewer the number of sent messages in a method, the better the method is and also the better its performance is in contrast to its similar methods. In the following sections, based on the mentioned evaluation method, we will carry out different experiments and show their results. For these evaluations, each experiment has been repeated 20 times and the final result has been obtained by averaging the results from each repetition.

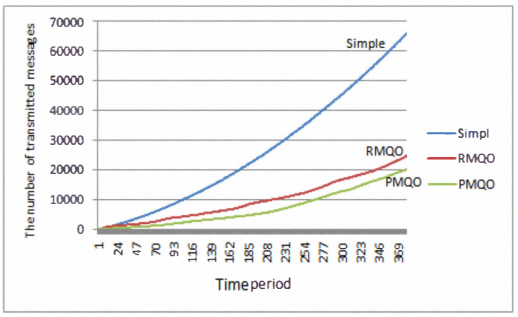

Figure 1. Result Reduction under normal workload

A. *Comparing the Number of Sent Messages under Normal Workload*

In this experiment, the number of sent messages through the network in each period, for each of the three mentioned algorithm, is examined. The queries which are used in this experiment are normal queries. The number of entered queries in each epoch to the base station follows an even distribution between 1 and 20. The number of the sent messages for each method in the different periods is shown in Fig. 1. As you can see, the simple algorithm has the most sent messages in comparison with the other two algorithms. RMQO algorithm, in contrast to Simple, has sent fewer messages and finally PMQO algorithm has sent the fewest messages of all.

B. *Comparing the Number of Sent Messages under Light-Weight Workload*

The conditions for this experiment are similar to the one in part *A*, with the difference that the entering network queries are light-weight ones. As you can see in Fig. 2, in this experiment, Simple and PMQO algorithms respectively have the worst and the best performance. But PMQO Algorithm, comparing to the previous experiment, has outperformed RMQO slightly. The reason is that the injected queries are light-weight and therefore bring rather less information from the nodes to the base station. If the prediction of the network status is made with these small amounts of data, we cannot have a good general insight into this network. Therefore, it is necessary that the nodes push more data to the base station. Ultimately, although by using PMQO we cause less result messages to be sent into the sensor network, but the extra load of the push messages causes the difference between the all sent messages of RMQO and PMQO to be less than the normal experiment.

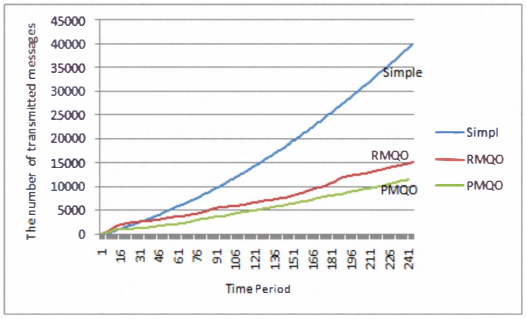

Figure 2. Result Reduction under light-weight workload

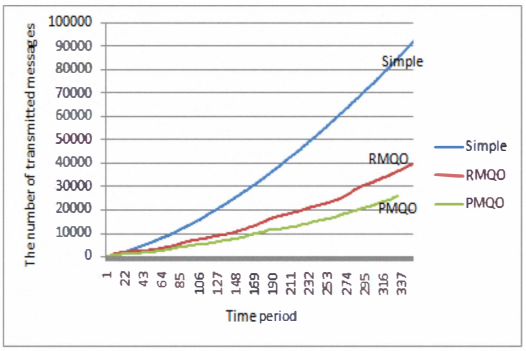

Figure 3. Result Reduction under heavy-weight workload

C. *Comparing the Number of Sent Messages under Heavy-Weight Workload*

The conditions for this experiment are similar to the previous ones, in part *A* and *B*, with the difference that the entering network queries are of the heavy-weight type. As you can see in Fig. 3, in this experiment, Simple and PMQO Algorithms have once more the worst and the best performance, respectively. But PMQO Algorithm, comparing to the previous experiments, has a larger difference with RMQO. The reason is that the injected queries are of the heavy-weight type and bring rather more information from the nodes to the base station. If the prediction of the network status is made with these large amounts of data, we can have a good general insight of this network. Therefore, it is not necessary for the nodes to push a lot of data to the base station. Finally, even though by using PMQO we cause less result messages to be sent into the sensor network, but reduced load of the push messages causes the difference between all sent messages between RMQO and PMQO to be more than the normal and the light-weight experiments.

D. *Comparing the Number of Reduced Queries under Normal Workload*

In this experiment, the number of injected queries into the network is demonstrated with changing the number of arrived user queries to the base station under normal workload. Obviously, the slope of the diagram for Simple Algorithm is equal to 1, because in this method, no query is eliminated. As shown in Fig. 4, in RMQO and PMQO, the number of injected queries into the network is decreased significantly because of the effect of query merging and rewriting in the form of synthetic queries. The amount of reduction of synthetic queries in PMQO is more than RMQO which shows that by having information about nodes of the network, the queries which are not similar apparently, but acquire data from identical nodes, can be identified and merged together. As a result, PMQO sends fewer queries to the network.

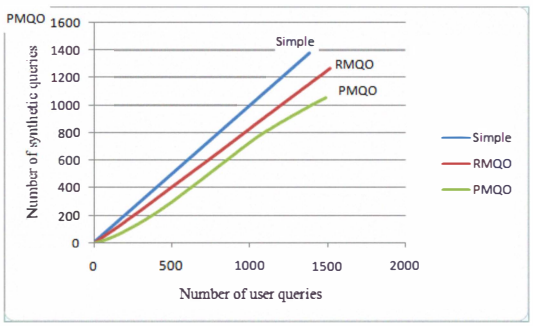

Figure 4. Query Reduction under normal workload

E. *Comparing the Number of Reduced Queries under Light-Weight Workload*

In this experiment, the test of part *D* is repeated for the light-weight workload.

As it can be seen in Fig. 5, the number of user queries and the synthetic queries for Simple algorithm are equal, similar to the previous test. Also, the reduction in the number of queries for a specific number of user queries is larger in PMQO than RMQO. The noticeable point in this figure is the slope of the diagrams, which remains almost constant in RMQO and increases in PMQO moving close to RMQO. The slope value for RMQO and PMQO, under normal workload is equal to 0.84 and 0.67, and under light-weight workload is equal to 0.83 and 0.75, respectively. It means that RMQO which has no information about the network acts similar to the previous experiment and the light-weight workload has not influenced the performance of the method. About PMQO, it can be said that the lighter the queries are, the less overlapping queries may exist; thus, more queries will be injected into the network.

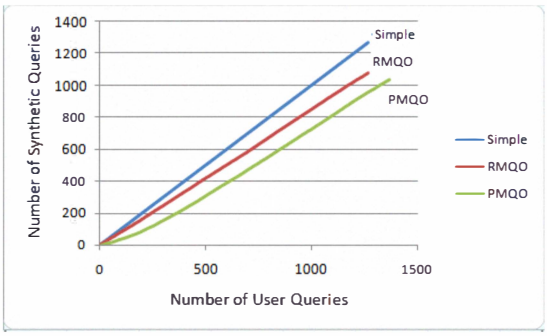

Figure 5. Query Reduction under light-weight workload

F. *Comparing the number reduced queries under heavy-weight workload*

In this experiment, a test similar to the ones in *D* and *E* part is done for the heavy-weight workload. As it can be seen in Fig. 6, the number of the user queries and the synthetic queries for Simple algorithm are equal, like the previous tests. Also, the number of reduced queries for a specific number of user queries is larger in PMQO than RMQO. The slope of RMQO has remained approximately constant in RMQO, but has decreased in PMQO to become far from RMOQ. The slope value for RMQO and PMQO, under

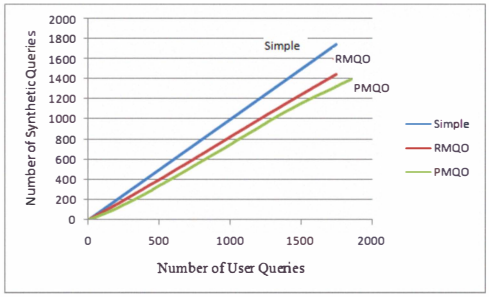

Figure 6. Query Reduction under heavy-weight workload

heavy-weight workload is equal to 0.82 and 0.61, respectively. It means that RMQO that has no information about the network acts the same as the previous experiments and the heavy-weight workload has not influenced the performance of the method. About PMQO, it can be said that the heavier the queries are, the more overlapping queries may exist; therefore, less queries will be injected into the network.

V. CONCLUSION

Multiple query Optimization is one of the most recent fields in query processing of wireless sensor networks. The methods that use merging of queries often decide about the way of rewriting based on the range overlapping of them. In this paper, we have used a prediction based method for multiple query optimization in environment monitoring applications, which considers the status of the network to decide about the selection of queries to be rewritten together. The experiments demonstrate that usage of prediction based method causes significant reduction in the number of query and the result messages in the network.

REFERENCES

[1] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks," in Proceedings of the CIDR Conference, 2003.

[2] U. Srivastava, K. Munagala, and J. Widom, "Operator Placement for InNetwork Stream Query Processing," in 24th ACM SIGMOD-SIGACT-SIGART symposium on PODS Baltimore, Maryland,USA, 2005.

[3] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in 5th symposium on Operating systems design and implementation, Boston, Massachusetts, USA, 2002, pp. 1-14.

[4] A. Skordylis, N. Trigoni, and A. Guitton, "A Study of Approximate Data Management Techniques for Sensor Networks," in Intelligent Solutions in Embedded Systems, 2006.

[5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TINYDB: An acquisitional query processing system for sensor networks," ACM Transactions on Database Systems (TODS), vol. 30, p. 52, March 2005.

[6] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," SIGMOD Record, vol. 31, p. 10, September 2002.

[7] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao, "Directions in multi-query optimization for sensor networks," in Advances in Pervasive Computing and Networking, 2005, p. 19.

[8] N. Trigoni, Y. Yao, A. Demers, and J. Gehrke, "Multi-query optimization for sensor networks " in DCOSS, 2005.

[9] R. Müller and G. Alonso, "Efficient Sharing of Sensor Networks," in Mobile Adhoc and Sensor Systems (MASS), IEEE International Conference, Vancouver, Canada 2006, pp. 109-118.

[10] R. Müller and G. Alonso, "Shared queries in sensor networks for multi-user support," Department of Computer Science, ETH Zurich February 2006.

[11] H.-Y. Yang, W.-C. Peng, and C.-H. Lo, "Optimizing Multiple In-Network Aggregate Queries in Wireless Sensor Networks " in Advances in Databases: Concepts, Systems and Applications, 12th International Conference on Database Systems for Advanced Application (DASFAA '07) Bangkok, Thailand, 2007.

[12] M. Tang, J. Cao, and N. K. Chilamkurti, "TAMPA: Tabu Search-Based Multiple Queries Optimization for Wireless Sensor Networks," in IEEE Wireless Communications (WiCOM) Shanghai, R.O.C, 2007.

[13] S. Xiang, H. B. Lim, K.-L. Tan, and Y. Zhou, "Two-Tier Multiple Query Optimization for Sensor Networks," in Proceedings of the 27th International Conference on Distributed Computing Systems 2007.

[14] A Behzadan, "Energy Sensitive Query Optimization in wireless sensor networks", Computer Engineering and Information Technology department, MSc thesis,Amirkabir University of Technology, 2008 (persian)

[15] H. Ling and T. Znati, "Similarity Based Optimization for Multiple Query Processing in Wireless Sensor Networks " in 5th IEEE International Conference on Distributed Computing in Sensor Systems, Marina del Rey, CA, USA 2009, pp. 117-130.

[16] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in Proceedings of the Thirtieth international conference on Very large data bases, Toronto, Canada 2004, pp. 588 - 599.