Energy efficient query aggregation and optimization scheme in wireless sensor network

By
Jesun Sahariar Firoz
Student No.: 1009052026
Jesmin Jahan Tithi
Student No.: 0409052065

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000.

October 24, 2010

Abstract

In wireless sensor networks, the main obstacle is power limitation and memory constraint. As communication is the most power consuming task, we need to minimize both the number of communications and concise the amount of data. To achieve these goals, previous works proposed different techniques like query optimization using various heuristics, in-network data aggregation, efficient routing, external storage with historical database. We intend to use all these mechanisms along with some novel ideas to accomplish better energy efficiency. We can infer from logical reasoning that the proposed heuristics give better performance in reducing power consumption.

Keywords: query optimization, data aggregation, energy efficiency, distributed database.
1 Introduction

A sensor network is a distributed sensing network comprised of a large number of tiny devices, each with some computational, storage and communication capability. Such networks can operate in an unattended mode to record detailed information about their surroundings. As these networks scale in size, so will the amount of data they make available.

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, in-network processing, node sleep scheduling, and aggregation strategies. 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. 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. 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. 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. 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.

A historical database can be used for quick query reply which stores the basic query result for a certain amount of time.

The rest of the paper is organized as follows: in section 2 we discuss the related work, in section 3 we focus on our basic query management scheme, section 4 describes the proposed algorithm, section 5 gives the logical basis of better performance of the proposed algorithm and finally we conclude in section 5 where we discuss some future directions.
2 Related works

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 [2]. 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 [10]. 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 [15, 3] 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 [12, 5]. Most studies emphasized efficiently generating alternative plans that maximize shared operations and minimize system cost [7]. Several heuristic algorithms [12, 11] 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 [4], 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 [1] 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.
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 [17], assuming that most queries are snap-shot queries. Similar to above work, a kind of multiquery optimization technique was also presented in [13], 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 [8] 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 [8].

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 [18], 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 [14] 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.
3 Query management scheme

In the query management scheme, there are three aspects that we are going to focus on: first, query optimization, next query aggregation and lastly query dissemination or routing.

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. Moreover, selection predicate also can be used for partitioning purpose. In the query framework, the historical database is a vital part for the users to directly access the sensing data that they are interested in, 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. The data are made invalid after one day. After the time has expired, the data is committed to a permanent database (if needed by the application). 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 as in . Another important aspect of query partitioning is that it is useful for the query manager to evaluate the common subqueries. For example let us consider the query:

\[ Q = \{ \{R_1, R_2, R_3\}, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T \text{ and } V\}, 10s|T \geq 48\} \]. This query can be partitioned based on regions, time, attribute, period and selective predicates.

**Region partitioning:**

\[ [R_1, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T \text{ and } V\}, 10s|T \geq 48], [R_2, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T \text{ and } V\}, 10s|T \geq 48], [R_3, \{t_1 \text{ to } t_2, t_3 \text{ to } t_4\}, \{T \text{ and } V\}, 10s|T \geq 48]. \]

Similarly, other attributes can be used for partitioning a complex query. Techniques proposed in [6] can be used in a similar way for selective predicate partitioning.

As explained in tinydb [9] Query ordering is useful to minimize the amount of data to be sampled from the sensors. For example if a query asked to return data when the temperature exceeds 48 degree and humidity is 45%, and if the normal humidity level is 45% but temperature’s normal value is 25 degree, then it will be optimal to query based on temperature first as that will reduce the data set to be aggregated and the possible number of sensors to be queried can be reduced. Therefore, proper ordering of selective predicate will result in less power consumption of sensor nodes.

Now, we are going to merge the overlapped partitioned queries. 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. Different query merging techniques can be found in [6]. They have done query merging based on region, time, attribute and data rate.

By following all the above mentioned steps, the number of requested queries are minimized by the query manager. This in turn reduces power consumption.

For routing purpose, we distinguish nodes between sensors, relays and clusterheads. First clusterheads are selected among sensors and clusters are formed as leach. Relay nodes resides in the upper layer and used for aggregation and routing purpose. Base station choose the nearest relay node as access point to the region. All successive nodes for query propagation are selected from only relay nodes until the query reach the particular clusterhead reside in the target region. After the clusterhead receive the request, it is responsible to communicate with the sensors under its jurisdiction to collect the data requested. Cluster head aggregates those data and then forwards it to its parent which will be a relay node. The relay node the forwards it upwards to the tree using multi hop routing towards the base station (query manager). If different clusters have results for the same query id, then the relay nodes will aggregate those results based on the query id, and then forward those information upward of the tree. Relay nodes will store some query results for few seconds and if same query reaches more than once, it will return the result without forwarding the query to the relevant clusterheads. Finally, basic query results will be stored in the historical database also.
4 Proposed algorithm

This section gives an overview of the proposed algorithm. The relay nodes are high power nodes. We use relay node based leach algorithm for routing in our algorithm.

1. Users will submit their queries to the query manager implemented in the base station.
2. Query accumulator of query manager will collect queries for a fixed time interval from the user.
3. Query partitioner will partition the collected queries based on time, frequency, region, attribute and selective predicates.
4. Common task evaluator will assess the queries to find out the overlapping subqueries and extract unique basic queries. Then it will assign an ID for each of these basic queries and create a reference table holding pointers to query IDs of each of the interested user involved in the particular basic query.
5. Query manager will scan the historical database to find whether the unique basic queries created by common task evaluator exists in present database. If these basic queries fully satisfy user query requirement, query manager will return the required answer either directly or by merging several basic queries stored previously in the historical database. If partial matching is found, then it reduce the part not in historical database. In case of partial or non-matching query predicates, we go to the next step.
6. An intelligent agent decides query execution ordering based on the selective predicates. That selective predicate which results in less data sampling will be used first for data sampling. This agent decides ordering of sampling data and predicates also.
7. For routing queries from base station to clusterheads, we choose those relay nodes which satisfy shortest distance criteria along the path. For query answers to propagate from clusterheads to base station, intermediate relay nodes are used as in relay node based leach algorithm. These relay nodes, although are not capable of sensing, more powerful than sensor nodes in terms of power, radio transceiver and computation capability. So these nodes act as in-network aggregator of all data received from clusterheads and relay nodes. Also these relay nodes form relay routing tree (RRT) to aggregate and propagate data up to query manager.
8. Query manager, after receiving query result, store it in the historical database.
9. Query manager will aggregate results of the previously partitioned subqueries and send it to the corresponding user.
For more efficiency, we can replicate the local historical database to several locations calculated by using voronoi diagram.

5 Performance Evaluation

Our proposed protocol will give better performance in terms of energy than any other previous algorithm, as we are

1. Minimizing query by using all the criteria based on([6] and [9])

2. Aggregating query results efficiently

3. Routing the data and query by a novel routing protocol. As a variation of relay supported leach is used battery power of sensors are utilized more efficiently than ever.

Minimizing query and aggregating data will reduce number and size of communicated data, thus reduce battery consumption. Efficient routing will reduce communication effort and energy. Optimization in all these sections will save more battery energy and thus will enhance the life time of the whole network.

6 Conclusion

In this paper we discussed an energy efficient query aggregation and optimization scheme in wireless sensor network. For this we have reduced number of queries by using different partitioning and merging criteria so that the resultant query set reduced to a minimal set. We also use in network data aggregation where data are aggregated at cluster heads. Similarly for routing we used relay nodes which further aggregates data. We have make routing efficient by using relay based leach and shortest path minimum energy routing. In future we would like to simulate our algorithm to prove that our algorithm will really perform better than all the previous algorithms. However, its a conjecture that filtering using bloom filter can be used for somewhat better performance. In addition, more than one historical database can be used for replicate data and their placement can be determined by voronoi diagram. These options are left as future directions.

References


