Impact Factor (SJIF): 5.301

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 5, Issue 6, June-2018

Survey of Area query processing using Gray Codes in wireless sensor network

Pratijnya Ajawan[1], Vaidehi Deshpande[2], Priya Kulkarni [3]

[1] *Assistant professorElectronics and communication, kls git*
[2] *Assistant professorElectronics and communication, kls git*
[3] *Assistant professorElectronics and communication, kls git*

***Abstract* — The monitored area will be partitioned into grids, and a unique gray code number will be used to represent a Grid ID (GID), which is an effective way to describe an area .In order to energy-efficiently answering continuous queries, the design of an incremental update method is done here to continuously generate query results. Existing query processing techniques cannot solve the area queries. Centralized processing on Base Station can accomplish area queries namely collecting information from all sensor nodes. A WSN consists of a set of sensor nodes,**

Which are small sensing devices with limited computational resources able to communicate with each other located in their radio range. Network protocols ensure the effectiveness of communication between sensor nodes and provide the foundation for WSN applications. The characteristics of WSNs, including the limited energy supply and computational resources, render the design of WSN algorithms challenging and interesting. Both the Database and Network communities have dedicated considerable efforts to make WSNs more effective and efficient. In this chapter, we survey the problems arisen in practical applications of WSNs, focusing on various query processing techniques over captured

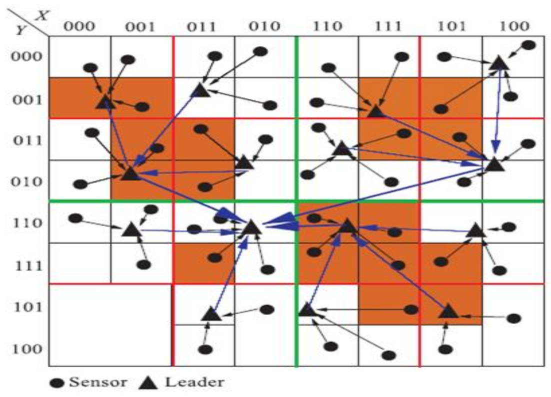
Sensing data.

Keywords- **Area Query; Area Query Processing; Gray Code;Wireless Sensor Networks.**

I. INTRODUCTION

Wireless sensor networks are widely used in many significant applications such as industrial process control, environmental monitoring, habitat monitoring and object tracking. Since users of wireless sensor networks focus on raw sensed values or processed information of the monitored area, sensor nodes are designed to cooperatively sense, collect, and process the raw information of the monitored area, and send the processed information to observers. Sensor-based applications extract different kinds of data via collecting data, processing in-network aggregations, and detecting complicated events. Since sensor nodes have limited resources, novel data management techniques are desired to satisfy application requirements which consider characteristics of wireless sensor networks. Most existing data collection systems are query based ones. Traditional query processing techniques of wireless sensor networks mainly deal with retrieving sensor node locations, sensed values, and aggregating the sensed values. However, in many applications, users expect information about areas of their interests.

According to the requirements of area query applications, we define area query as requests for area information including area locations, sizes of areas, and collected and aggregated data of the areas. Sensor nodes with expected readings and adjacent sensing coverage are divided into the same group. The total coverage area of the sensor nodes in the same group is a possible result area. An area query can retrieve not only sensed values but also specific geographical information compared with traditional queries of wireless sensor networks. Area query is more practicable and useful than traditional queries for some applications, which require geographical
information (Ref. Fig. 1). A common property of area query applications is that the results of these queries are areas and the aggregation values of sensors in these areas. Size of areas can also be query conditions. Furthermore, queries might be run for either the whole network or sub-areas of the network. For different areas, it is possible to use different querying conditions.



II GRAY CODE CONCEPT

Gray code by bit width	
2-bit	4-bit
00 01 11 10	0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000
3-bit	
000 001 011 010 110 111 101 100	

Table:-1.1

The reflected binary code, also known as Gray code after Frank Gray, is a binary numeral system where two successive values differ in only one bit(binary digit). The reflected binary code was originally designed to prevent spurious output from electromechanical switches. Today, Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.
In practice, a "Gray code" almost always refers to a binary-reflected Gray code (BRGC). However, mathematicians have discovered other kinds of Gray codes. Like BRGCs, each consists of a list of words, where each word differs from the next in only one digit (each word has a Hamming distance of 1 from the next word).

2.1 SENSOR DATA

A sensor node has one or more sensors attached that connected to the physical world. Example sensors are temperature sensors, light sensors, or PIR sensors that can measure the occurrence of events (such as the appearance of an object) in their vicinity. Thus each sensor is a separate data source that generates records with several fields such as the id and location of the sensor that generated the reading, a time stamp, the sensor type, and the value of the reading. Records of the same sensor type from different nodes have the same schema, and collectively form a distributed table. The sensor network can thus be considered a large distributed database system consisting of multiple tables of different types of sensors. Sensor data might contain noise, and it is often possible to obtain more accurate results by fusing data from several sensors [12]. Summaries or aggregates of raw sensor data are thus more useful to sensor applications than individual sensor readings [21, 10]. For example, when monitoring the concentration of a dangerous chemical in an area, one possible query is to measure the average value of all sensor readings in that region, and report whenever it is higher than some predefined threshold.

2.2AREA QUERY PROCESSING BASED ON GRAY CODE IN WIRELESS SENSOR NETWORKS

Wireless sensor networks are used in various applications such as industrial process control, object tracking, environmental monitoring, and habitat monitoring. Since users of wireless sensor networks focus on raw sensed values or on processed information of the monitored area, therefore sensor nodes are properly designed to cooperatively sense, collect, and process the raw information of the monitored area, and send the processed information to observers. Sensor-based applications extract different kinds of data by collecting the data, processing in-network aggregations, and detecting complicated events. Since sensor nodes have limited resources, data management techniques are desired to satisfy application requirements which consider characteristics of wireless sensor networks. Most of the existing data collection systems are query based. Traditional query processing techniques of wireless sensor networks mainly deal with retrieving sensor node locations, sensed values, and aggregating the sensed values. However, in many applications, users expect information about areas of their interests. For instance, workers in a coal mine wish to find an area with a high oxygen density to take a break, also this area must be big enough to accommodate several workers.

In Fig. 2 below, all the sensed values of the sensors in regions R1 through R5 reach the expected oxygen density. However, regions R3 and R5 are not big enough for several workers, so R3 and R5 are not acceptable. For this application, the existing methods may only return the locations and sensed values of the sensors which satisfy the oxygen density condition, yet it is meaningless and insufficient. Here, users want to find areas instead of multiple sensor locations as in this case the size of area is also a filtering condition. The traditional methods fail to consider spatial correlation among sensor nodes that is critical for many applications.

Another interesting scenario is an air pollution monitoring application within a city. Users expect the monitoring system to detect regions whose pollution levels reach different thresholds. In reality, the polluted thresholds might be different for different areas.

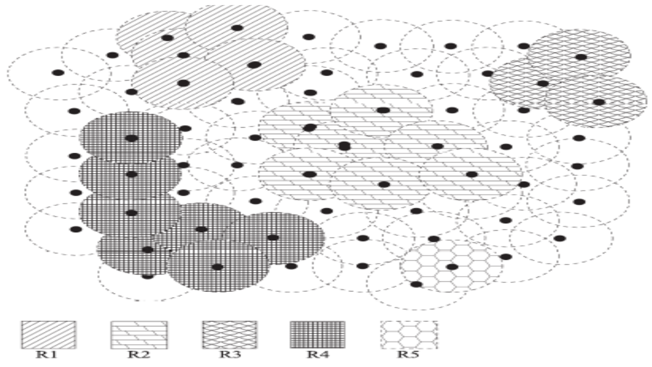


Figure 2:-AREA QUERIES[Reference 1]

For example, the threshold of an industrial area is 70, while that of a residential area is 40. In Fig. 2 above R1 and R4 are in the industrial area, and R2, R3, and R5 are in the residential area. The polluted level of R1 through R5 is 60, 50, 30, 80, and 30, respectively. Usually, the traditional methods apply the same filtering condition for the whole network. In this example, if the threshold 80 is used, polluted area R2 is missed since R2's polluted level actually exceeds the threshold for residential areas while the filtering condition does not indicate this. If the threshold 50 is used, R1 is identified as a polluted area which triggers a false alarm. So the existing methods cannot be used for this kind of applications. These applications concern specific areas and each area have its own specific filtering conditions.

According to the requirements of area query applications, we define area query as requests for Area information including area locations, sizes of areas, and collected and aggregated data of the areas. Sensor nodes with expected readings and adjacent sensing coverage are divided into the same group. The total coverage area of the sensor nodes in the same group is a possible result area. An area query retrieves not only sensed values but also specific geographical information compared with traditional queries method of wireless sensor networks. Area query is more practicable and useful than traditional queries method for some applications that require geographical information. A common or well known property of area query applications is the results of these queries are areas and also the aggregation values of sensors in these areas. Size of areas can also be query conditions. Also, queries might be run for either the whole network or sub-areas of the network. For different areas, we can use different querying conditions. Area query method is a new kind of query in wireless sensor networks. The Base Station can collect data from all sensor nodes and then process area queries in a centralized manner; however, the sensor nodes will drain energy quickly because of the frequent reports needed of the sensed values. The existing techniques of in-network query processing suppress and aggregate sensed values to save energy.

Also, these methods do not consider sensor coverage area as part of the results. Therefore, a new in-network area query processing technique is necessary.

Due to inherent limitations of wireless sensor networks, designing an in-network area query processing mechanism becomes a challenging issue. The first challenge of in-network area query processing is that it is hard to suppress useless data as early as possible. For Example, an area satisfies all the conditions except the size condition. Obviously, this area is not the ultimate expected result. However, we cannot drop this area since it is difficult to decide the boundary of an area locally. Moreover, how to describe an area in-network is another challenge. Because areas are also part of query results, an ideal area description can reduce the amount of the transmitted data and lower the complexity of area size computation, which are two primary considered factors for energy conservation. Here, an energy-efficient in-network area query processing scheme is proposed. The whole network is partitioned into grids, and a gray code is used to represent each grid. A reporting tree is constructed to merge areas and process aggregations. To conserve energy, incremental update techniques are used to process continuous queries.

Thus the contributions can be summarized as follows:

(1) A new area query method is studied here .

(2) A smart area description, Grid ID (GID) list, is used to reduce the size of query results.

(3) An energy-efficient in-network area query processing scheme is proposed. By using GIDs and merging strategies, partial results can be merged to reduce the size of results and useless data can be dropped as early as possible in order to reduce the number of messages.

(4) An incremental update method is addressed to reduce the size of reports for continuous queries.

III AREA QUERY IN WIRELESS SENSOR NETWORKS.

3.1Query Plans

Let us consider an example query that we will use to illustrate the components of a query plan. Consider the query “What is the quietest open classroom in Upson Hall?”.5 Assume that the computation plan for this query is to first compute the average acoustic value of each open classroom and then to select the room with the smallest number. There are two levels of aggregation in this plan: (1) to compute the average value of each qualified classroom, and (2) to select the minimum average over all classrooms. The output of th first level aggregation is the input to the second level aggregation. Users may pose even more complicated queries with more levels of aggregations, and more complex interactions. A query plan decides how much computation is pushed into the network and it specifies the role and responsibility of each sensor node, how to execute the query, and how to coordinate the relevant sensors. A query plan is constructed by flow blocks, where each flow block consists of a coordinated collection of data from a set of sensors at the leader node of the flow block. The task of a flow block is to collect data from the relevant sensor nodes and to perform some computation at the destination or internal nodes. A flow block is specified by different parameters such as the set of source sensor nodes, a leader selection policy, the routing structure used to connect the nodes to the leader (such as a DAG or tree), and the computation that the block should perform. A query plan consists of several flow blocks. Creating a flow block and its associated communication and computation structure (which we also call a cluster) uses resources in the sensor network. We need to expend messages to maintain the cluster through a periodical heart beat message in which the leader indicates that it is still alive; in case the cluster leader fails, a costly leader election process is required. In addition, a cluster might also induce some delay, as it coordinates computation among the sensors in the cluster.

Thus if we need to aggregate sensor data in a region, we should reuse existing clusters instead of creating a new cluster, especially if the data sources are loosely distributed over a larger area, in which case the maintenance cost increases. On the other hand, we should create a flow block if it significantly reduces the data size at the leader node and saves costly transmission of many individual records. It is the optimizer’s responsibility to determine the exact number of flow blocks and the interaction between them. Compared to a traditional optimizer, we would like to emphasize two main differences. First, the optimizer should try to reduce communication cost, while satisfying various user constraints such as accuracy of the query, or a user-imposed maximum delay of receiving the query answers. The second difference lies in the building blocks of the optimizer. Whereas in traditional database systems a building block is an operator in the physical algebra, our basic building block in a sensor database system is a flow block, which specifies both computation and communication within the block.

Query Optimization

In this section we will discuss how to create a good query plan for more complicated queries. Our discussion stays at the informal level with the goal to help us decide what meta-data we need for the optimizer in the systems catalog. We would like to emphasize that creation of the best query plan for an arbitrary query is a hard problem, and our work should be considered as an initial step towards the design and implementation of a full-fledged query optimizer. We leave experimental evaluations of different query plans to section 6, and the design and implementation of a full-fledged optimizer to future work Extension to GROUP BY and HAVING Clauses. Let us consider an aggregate query with

GROUP BY and HAVING clauses. The following query computes the average value for each group of sensors and filters out groups with average smaller than some threshold.

```
(Q1) SELECT D.gid, AVG(D.value)
FROM SensorData D
GROUP BY D.gid
HAVING AVG(D.value)>Threshold
```

There are two alternative plans for this query. We can create a flow block for each group, or we can create a flow block that is shared by multiple groups.To create a separate flow block can aggregate sensor records of the same group as soon as possible, shorten the path length, and allow to apply the predicate of the HAVING clause to the aggregate results earlier,
which saves more communication if the selectivity of the predicate is low. The optimizer should take several
parameters into account to make the best plan. One parameter is the overlap of the distribution of the physical locations of the sensors that belong to the different groups. If sensors that below to a single group are physically close, it is better to create a separate flow block to aggregate them together, since the communication cost to aggregate close-by sensors is usually low. However, if sensors from different groups are spatially interspersed, it is more efficient to construct a single flow block shared by all groups. Joins. The computation part of a flow block does not need to be an aggregate operator. It is possible toadd join operators to our query template and define flow blocks with joins. Joins will be common in applications for tracking or object detection. For example, a user may pose a query to select all objects detected in both regions R1 and R2. The following query has a join operator to connect sensor detections in the two regions.

```
(Q2) SELECT oid
FROM SensorData D1, SensorData D2
WHERE D1.loc IN R1 AND D2.loc IN R2
AND D1.oid = D2.oid
```

Join operators represent a wide range of possible data reductions. Depending on the selectivity of the join, it is possible to either reduce or increase the resulting data size. If the join increases the result size, it is more expensive to compute the join result at the leader instead of having the leader send out the tuples from the base relation. Relevant catalog data to make an informed decision concerns the selectivity of the join and the location of the leader.

Here, an area query processing scheme is proposed, which can handle not only normal queries like selection, aggregation, and threshold-based event detection but also area queries. More importantly, this scheme is energy-efficient since it processes area queries in an in-network manner. Compared with the existing techniques, the remarkable difference is that this scheme supports dynamic sensor grouping which is a remarkable step for area query processing. For dynamic sensor grouping, consideration of not only geographical correlations of sensors sensing coverage is done but also sensed data correlations to derive expected results.

Sensor nodes might be equipped with several sensing components to monitor environment or detect events. Users can specify a query, which describes an event or specific areas of user’s interests.

An area query is defined as follows:

```
SELECT areas and/or aggregation functions
FROM entire sensor network or subareas
.WHERE predicates •
.GROUP BY adjacency •
.HAVING predicates •
.DURATION time  span
EVERY time  interval •
```

SELECT presents query results. It can be areas and some aggregation functions of the sensed values of each area. FROM specifies the queried areas. We can query the whole network or just focus on some particular areas. Users can write complicated query conditions on sensing attributes by the WHERE clause. GROUP BY is used to divide sensor nodes into groups according to the expected readings and adjacent sensing coverage. This is the main difference between traditional query method and area query method. HAVING presents the conditions on the aggregation functions. The requirements of the results areas can be specified by the HAVING clause. DURATION specifies the lifetime of a continuous query. EVERY defines an execution interval, which means the query is continuously executed and the results are returned every time-interval time unit. If a query does not specify the DURATION and EVERY, then it is not a continuous query, and it will be executed only once. A continuous query can also be used to describe events as it will be executed periodically. The coal mine example can be described in terms of the following area query as follows:

```
  SELECT area; area: avg.sensors: oxygen/
  FROM sensors
```

```
 WHERE sensors: oxygen > 80
 GROUP BY adjacency
 HAVING area: size > 50m2
 DURATION 240 hours
 EVERY 60 seconds
```

It depicts that an area is qualified when the oxygen density is greater than 80, and the acreage of this area is greater than 50 m2. This query will be continuously executed for 240 hours. The results will be reported to the user for every 60 seconds. To guarantee the safety of workers, this query continuously reports the results to the user. This query also reports the average oxygen density of each qualified area to user. Most previous aggregation methods provide the aggregated values of some particular properties, but not for each area. Another example, is the air pollution monitoring, that can be given in terms of the following query.

```
SELECT area; area: avg.sensors: pollution  level/
FROM industrial area as R1; residential area as R2
WHERE R1: sensors: pollution  level > 80;
R2: sensors: pollution  level > 50
GROUP BY adjacency
HAVING R1: area: size>1000m2; R2: area: size>100m2
DURATION 72 hours
EVERY 20 minutes
```

In this example, we independently specify the pollution-level threshold for industrial area as 80 and for residential area as 50. Also, in the HAVING clause, the size requirement of the result areas for industrial area is 1000 m2 and for residential area it is 100 m2. Area query allows different WHERE and HAVING conditions for different areas. A scheme which can efficiently process area queries is then necessary. Due to limited computation and energy resources of sensor networks, energy-efficiency is the main optimization goal, and reducing in-network computation complexity should also be a concern.

.

IV. FOOTNOTES

Sensor networks will become ubiquitous, and the database community has the right expertise to address the challenging problems of tasking the network and managing the data in the network. We described a vision of processing queries over sensor networks, and we discussed some initial steps in in-network aggregation, implications on the routing layer, and query optimization.

REFERENCE

[1] Chu D, Deshpande A, Hellerstein J M, Hong W. Approximate data collection in sensor networks using probabilistic models. In: Proceedings of the 22nd International Conference on Data Engineering, 2006: 48.

[2] Silberstein A, Braynard R, Filpus G, Puggioni G, Gelfand A, Munagala K, Yang J. Data-driven processing in sensor networks. In: Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research, 2007.

[3] Wang D, Xu J, Liu J, Wang F. Mobile filter: Exploring migration of filters for error-bounded data collection in sensor networks. In: Proceedings of IEEE 24th International Conference on Data Engineering, 2008: 1483-1485.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. [3], pages 85–97.

[5] M. Calimlim, W. F. Fung, J. Gehrke, D. Sun, and Y. Yao. Cougar Project web page. www.cs.cornell.edu/database/cougar.

[6] S. Ceri and G. Pelagatti. Distributed Database Design: Principles and Systems. MacGraw-Hill (New York NY), 1984.

[7] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In INFOCOM 2000, pages 3–12. IEEE.

[8] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In MOBICOM 1999, pages 263– 270. ACM Press.

[9] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Mining and Knowledge Discovery, 1(1):29–53, 1997