Information Sciences 297 (2015) 1–20

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Aggregate query processing in the presence of duplicates in wireless sensor networks

Jun-Ki Min [a], Raymond T. Ng [b], Kyuseok Shim [c,*]

[a] School of Computer Science and Engineering, Korea University of Technology and Education, Byeongcheon-myeon, Cheonan, Chungnam, Republic of Korea
[b] Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, Canada
[c] School of Electrical and Computer Engineering, Seoul National University, Kwanak, P.O. Box 34, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 12 February 2014
Received in revised form 19 May 2014
Accepted 5 November 2014
Available online 15 November 2014

Keywords:
Query processing
Aggregation
Sliding window
Sensor network

ABSTRACT

Wireless sensor networks (WSNs) have received increasing attention in the past decade. In existing studies for query processing of WSNs, each sensor node measures environmental parameters such as temperature, light and humidity around its location. Instead, in our work, we consider a different type of sensors that detect objects in their sensing regions which may overlap with each other. In WSNs with such sensors, an object may be detected by several sensor nodes and processing of aggregate queries (such as COUNT, SUM and AVERAGE) becomes problematic since an identical object can be considered redundantly.

In this paper, we propose efficient algorithms for processing aggregate queries as well as sliding window aggregate queries in the presence of multiply detected events. To perform de-duplication, our proposed algorithms identify potential duplicates among detected events by communicating with other nodes and perform aggregations as early as possible. In addition, we extend our algorithms for aggregate queries to support time-based sliding windows. By extensive performance study with diverse environments, we show that the energy consumptions of our proposed algorithms are much smaller than those of baseline algorithms.

1. Introduction

A wireless sensor network (WSN) consists of spatially distributed autonomous devices with various sensors, and a powered base station which serves as an access point for users to pose ad hoc queries. Wireless sensor networks have a broad range of applications, such as environment and habitat monitoring [30] that collects meteorological data (e.g., temperature, pressure, and humidity), and combat field surveillance [22] that tracks the movement of personnel or detects potentially hazardous chemicals.

Since sensor nodes have limited battery power, for many applications, it is hard or costly to replace the batteries in the environments of its applications. Thus, minimizing energy consumption of sensor nodes to prolong the network lifetime has been one of the most important research issues [8,20,21]. Typically, sensing and communication dominate the power consumption by orders of magnitude compared to computation and accessing RAM [21].

In existing studies for query processing of WSNs, each sensor node measures environmental parameters such as temperature, light and humidity around its location. However, due to advancements of Micro Electro Mechanical Systems

* Corresponding author. Tel.: +82 2 880 7269; fax: +82 2 871 5974.
E-mail addresses: jkmin@kut.ac.kr (J.-K. Min), rng@cs.ubc.ca (R.T. Ng), shim@kdd.snu.ac.kr (K. Shim).

http://dx.doi.org/10.1016/j.ins.2014.11.021
0020-0255/

technology, the prices of many sensors such as motion detectors, infrared radiation detectors [30], ultra sonic sensors (sonar) and magnetometers [22] have significantly reduced, and their sensing ranges are expanded recently. Thus, for many applications, it becomes cost effective to use a larger number of sensors to increase the robustness of query processing in sensor networks. Therefore, in our work, we consider a type of sensors which detect objects in their sensing regions which may overlap each other. In WSNs with such sensors, an object may be detected by several sensor nodes and processing of aggregate queries (such as COUNT, SUM and AVERAGE) becomes problematic since an identical object can be considered redundantly.

The assumption of disjoint sensing areas is no longer true and particularly problematic for aggregate queries, due to duplicate detections. Moreover, in WSNs, duplicates also arise in situations when the objects move in and out of the sensing regions of multiple sensors in a sufficiently long time. In this case, even if the sensors do not overlap in their sensing regions, de-duplication is required for exact aggregation.

In this paper, we study processing of exact aggregate queries and sliding window aggregate queries in the presence of multiply detected events for WSNs. To the best of our knowledge, our work is the first study for exact in-network aggregation and sliding window aggregation with de-duplication of multiply detected events.

Aggregate Queries: We assume that each object carries a unique identifier (e.g., an animal equipped with a "*collar*" [14]). Specifically, in this paper, we consider an aggregate query Q with the following SQL syntax:

```
Q: SELECT AGG(a_k)
     FROM (
       SELECT AGG_1(S.attr_1) AS a_1, ..., AGG_m(S.attr_m) AS a_m
       FROM Sensor S
       GROUP BY S.ID
     )
     WHERE p(a_1, ..., a_m)
```

In the above query, the key idea is the introduction of the GROUP BY clause in the inner SQL statement. Using the unique object identifiers, de-duplication is achieved by the GROUP BY clause. However, there is no guarantee that all non-ID attributes are the same. The SELECT clause in the inner statement specifies how to obtain the *"representative"* attribute values for each distinct object.[1] Specifically, we ask the query processor to represent the value of every attribute att_i (with $1 \leqslant i \leqslant m$) for each distinct object by applying an aggregation function AGG_i such as MIN, MAX and AVERAGE in the SELECT clause of the inner statement.

To choose the groups generated by the inner statement, we use the WHERE clause in the outer statement. The selection predicate p in the WHERE clause is applied to each group using the representative attribute values of each group. Finally, the aggregation function AGG is applied to the values of the aggregation attribute a_k to obtain the final aggregation result. The following is an example of our aggregate queries.

Example 1. A zoologist is interested in the number of zebras suffering from a disease. If the symptom of the disease is high body temperature, this can be captured by the following query Q_z.

```
Q_z: SELECT COUNT(ZAVG)
     FROM (
       SELECT AVG(Z.body_temperator) AS ZAVG
       FROM Zebra Z
       GROUP BY Z.id
     )
     WHERE ZAVG  ⩾ 38
```

As a zebra may be detected with different body temperatures by different sensors, the inner statement of the query Q_z chooses the average value as the representative body temperature of each zebra. The query Q_z returns the number of zebras with (average) body temperatures above 38 °C.

Towards the goal of minimizing power consumption, we first propose the *LCA* (Least Common Ancestor) algorithm for exact aggregation in the presence of duplicates. We exploit the hierarchical structure of routing trees where sensors are organized in groups with their coordinator nodes that perform de-duplications and partial aggregations. The partially aggregated results are then passed along the routing paths to the base station.

To further reduce communication overhead, we extend *LCA* into the two-phase algorithm called *LCA-EA* (*LCA* with Eager Aggregation). In the first phase, each sensor node identifies potential duplicates by communicating with the other nodes using a variant of bloom filters for detected events. After potential duplicate events are identified, they are sent to coordinators and the second phase applies *LCA* for de-duplication. Meanwhile, for uniquely detected events, early aggregations

[1] We refer to the tuple consisting of every attribute $attr_i$'s (with $1 \leqslant i \leqslant m$) representative value of an object o as the *representative* of o.