Journal of Network and Computer Applications 37 (2014) 323–333

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Bloom filter based processing algorithms for the multi-dimensional event query in wireless sensor networks

Guilin Li [a], Longjiang Guo [b,*], Xing Gao [a], Minghong Liao [a]

[a] School of Software, Xiamen University, Xiamen, China
[b] School of Computer Science and Technology, Heilongjiang University, Harbin, China

ARTICLE INFO

Article history:
Received 22 September 2012
Received in revised form
11 February 2013
Accepted 6 March 2013
Available online 13 March 2013

Keywords:
Multi-dimensional event query
Bloom filter
Approximate algorithm
Wireless sensor networks

ABSTRACT

To solve the multi-dimensional event based query in wireless sensor networks, this paper proposes four bloom filter based query processing algorithms UBP, BBP, SRBP and PBP. The four algorithms proposed can be classified into two classes: two bloom filter based precise algorithms, which are UBP and BBP, and two bloom filter based approximate algorithms, which are SRBP and PBP. By using the bloom filter and introducing the inaccuracy, the communication cost involved by the query processing can be reduced.

For the two precise algorithms UBP and BBP, simulation results show that UBP consumes 51% less energy than BBP on average. UBP is better than BBP on energy consumption. For energy consumption comparison between the approximate algorithms and the precise algorithm UBP, simulation results show that SRBP consumes 18% less energy than UBP on average as while as PBP consume approximately the same energy as UBP on average. For query accuracy comparison between the approximate algorithms and the precise algorithm UBP, simulation results show that the average relative error between UBP and PBP is 14% and the average relative error between UBP and SRBP is 2%. SRBP is better than PBP on energy consumption and query accuracy respectively. UBP and SRBP are two preferred bloom filter based query processing algorithms.

1. Introduction

The emergence of the wireless sensor networks (WSNs) (Li and Liu, 2009; Keung et al., 2010; Kermarrec and Tan, 2010; Dong et al., 2010; Wang et al., 2011) has changed the way people collect information from the physical world. As the WSNs produce a huge amount of data, they can be seen as a new kind of data source and a lot of query processing algorithms (Abadi et al., 2005; Xiang et al., 2007; Yu et al., 2009; Liu et al., 2009; Cheng and Li, 2010) have been proposed to efficiently manage the data produced by the WSNs.

With the development of the hardware design, a sensor node can be equipped with multiple types of sensors (Croosbow, 2008) and detects different kinds of attributes at the same time. The information collected by a sensor node is flexibly composed of multiple different attributes. People send queries to the network looking for the data satisfying some requirement. For instance, a user sends a query to the sensor network distributed in a mine to detect the gas explosion. The query asks for the places where the temperature is above x, pressure is above y and gas density is above z. A tuple of data (x', y', z') detected by a sensor, that satisfies the user's requirement to the temperature, pressure and gas density, is called a multi-dimensional event. The query asking for the places, where the multi-dimensional event happens, is called a multi-dimensional event based query.

There are two kinds of Data Centric Storage (DCS) based schemes to store the multi-dimensional events in the WSNs, which are the composite storage scheme and the separate storage scheme. For the composite storage scheme, a multi-dimensional event is considered as a whole and stored in a particular sensor node of the network. The traditional DCS based algorithms, such as the DIM (Li et al., 2003), the Comb-Needle (Liu et al., 2004) and the Double Ruling (Sarkar et al., 2006), belong to the composite storage scheme. For the separate scheme, however, a multi-dimensional event is considered to be composed of many individual one-dimensional events and each of the one-dimensional events are stored in the different sensor nodes.

A multi-dimensional event is a tuple of values corresponding to different attributes. Each value of an attribute can be considered as a one-dimensional event. The separate storage scheme divides a multi-dimensional event into multiple one-dimensional events and stores each one-dimensional event into a different sensor called the one-dimensional event storage node. For example, the composite storage scheme stores the tuple (x',y',z') in a single node. The separate storage scheme stores x', y' and z' in three

* Corresponding author. Tel.: +86 451 86608887.
E-mail addresses: glli@xmu.edu.cn (G. Li), longjiangguo@gmail.com (L. Guo), gaoxing@xmu.edu.cn (X. Gao), liao@xmu.edu.cn (M. Liao).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.03.003

one-dimensional event storage nodes. The naive method to process a multi-dimensional event based query adopting the separate storage scheme works as follows. The sink sends the query to every one-dimensional event storage node of the multi-dimensional event and collects the results back. The sink gets the final results by intersecting all of the results collected.

The advantage of the composite scheme is that a multi-dimensional event based query can be processed by a single node, while the same query must be processed by several nodes using the separate scheme. The disadvantage of the composite scheme is that it is only suitable for the type of multi-dimensional events with fixed mode. Once determined, contents of the multi-dimensional event seldom change. While a user can easily define new kinds of multi-dimensional events for new applications based on the separate scheme. The algorithms proposed in this paper adopt the separate storage scheme.

Chen et al. (2008) proposed a bloom filter based algorithm to solve the energy waste problem of the separate storage scheme. A data set can be transformed into a bit vector, called the bloom filter of the data set, by a set of hash functions. The bloom filter is very useful to test whether a data belongs to the data set of the bloom filter. The bloom filter based algorithm works as follows. The sink sends a query to the first one-dimensional event storage node. After receiving the query, the first node calculates its local results satisfying the query and transforms its results into a bloom filter. Then the first node sends the query and the bloom filter to the second one-dimensional event storage node. The second node calculates its local results satisfying the user's query and checks whether these results belong to the bloom filter of the first node. Only the results belonging to the bloom filter will be sent back to the first node. The first node calculates the intersection of the two nodes' results and returns the results to the sink. Compared with the naive method, the bloom filter based algorithm can save more energy by only transmitting the bloom filter and the intersection of the results between different one-dimensional event storage nodes. The query processing algorithm given by Chen et al. (2008) only deals with the multi-dimensional event composed of two attributes. This paper first extends the algorithm to process the multi-dimensional event based query composed of more than two attributes. More important, this paper presents two bloom filter based approximate algorithms to solve the multi-dimensional event based query, which can save even more energy than that of the precise algorithms given in Chen et al. (2008).

The contribution of this paper is as follows. First, this paper gives two bloom filter based precise algorithms to process the multi-dimensional event based query with more than two attributes. Second, the paper proposes two approximate algorithms to solve the same problem, which can save more energy than that of the precise algorithms. Third, we validate the performance of the algorithms by extensive simulations.

The rest of this paper is organized as follows. Section 2 shows some related work. The problem description is given in Section 3. In Section 4, two precise and two approximate algorithms are proposed. In Section 5, simulations are given to test the performance of the algorithms proposed in this paper. Section 6 concludes the paper.

2. Related work

A lot of algorithms have been proposed to efficiently manage the data produced by the WSNs.

One of the typical applications of the WSNs is to detect events people are interested in and answer queries about these events. There are mainly two types of schemes to process an event based query, which are the data centric routing (Intanagonwiwat et al., 2000; Madden et al., 2002) and the data centric storage (Ratnasamy et al., 2002; Shenker et al., 2006). The data centric routing scheme stores the events detected by a sensor in the sensor's local storage. An event based query is flooded throughout the network to construct a path from the source node to the sink. The results are transmitted to the user through the path. Such kind of query processing scheme is suitable for continuous query for a certain type of event, because the construction of the path consumes a lot of energy of the sensor nodes. The second kind of scheme to process the event based query is the data centric storage. Instead of storing data in its local storage, according to the type of the detected event, a sensor node stores the event in some place in the network calculated by a global hash function. A query only needs to visit the hashed location to acquire events of a certain type. Compared with the data centric routing scheme, the data centric storage scheme does not involve flooding and is suitable for processing the ad hoc query. Many variants based on the basic idea of DCS have been proposed, such as the DIM, the Comb-Needle, the Double Ruling, etc. The DIM (Li et al., 2003) divides the whole network into a lot of zones and embeds a k–d tree like index in the network. There is only one node in each zone, which acts as the index node of the zone. With the k–d tree like index, events with comparable attribute values are stored nearby and the DIM can fulfill the range query easily. In the Comb-Needle algorithm (Liu et al., 2004), when a sensor detects an event, it distributes the data of the events to the nodes in the vertical direction within h hops around it. If a query is sent to the sensor network along multiple lines in the horizontal direction and the distance between any two neighboring lines is smaller than h hops, the query will be transmitted to the event storage node and get the results. The double ruling algorithm (Sarkar et al., 2006) distributes the data of the events around the network in a circle. The query also traverses the network in a circle. The double ruling algorithm guarantees that the circle of the query can intersect with all circles of the queried events. All of the algorithms above are all based on the composite storage scheme.

3. Preliminaries and problem description

3.1. Bloom filter

Before giving the definition of the problem, we briefly review the basis of bloom filter (BF) (Bloom, 1970; Song et al., 2005; Guo et al., 2011). A BF is a bit vector with l bits, all of which are initially set to 0. The BF facilitates the membership test, which means whether an element x belongs to a finite set $S=\{x_1,x_2,\ldots,x_n\}$. The BF uses a set of k uniform and independent hash functions $h_1,h_2,\ldots,h_k$ to map the set S to the bit vector, which means for each element $x_j\in S$ bloom filter sets the $h_i(x_j)$ th $(1\le i\le k)$ bit of the bit vector to 1. To check whether an item $x\in S$, we check whether all $h_i(x)$ bits are set to 1. If not, x is definitely not a member of S. Otherwise, x is probably a member of S. We call the probability a false positive if an element $x\notin S$ has all $h_i(x)$ bits set to 1.

Assume that a hash function selects each array position with equal probability. The probability that a certain bit is not set to 1 by a certain hash function during the insertion of an element is then $(1-1/l)$. The probability that it is not set to 1 by any of the hash functions is $(1-1/l)^k$. If we have inserted n elements, the probability that a certain bit is still 0 is $(1-1/l)^{nk}$. The probability that it is 1 is therefore $1-(1-1/l)^{nk}$. To test membership of an element that is not in the set, each of the k array positions computed by the hash functions is 1 with a probability as above. The probability of all of them being 1, which would cause the algorithm to erroneously claim that the element is in the set, is given by Eq. (1). Eq. (1) is called the false positive probability