Information Systems 39 (2014) 217–232

Contents lists available at ScienceDirect

Information Systems

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

Supporting historic queries in sensor networks with flash storage

Adam Dou [a], Song Lin [a], Vana Kalogeraki [c], Dimitrios Gunopulos [b,*]

[a] Google, Mountain View, CA, United States
[b] Department of Informatics and Telecommunications, University of Athens, Athens, Greece
[c] Department of Informatics, Athens University of Economics and Business, Athens, Greece

ARTICLE INFO

Available online 24 April 2012

Keywords:
Sensor networks
Flash memories
Indexing sensor data
Continuous queries

ABSTRACT

Many recent sensor devices are being equipped with flash memories due to their unique advantages: non-volatile storage, small size, shock-resistance, fast read access and power efficiency. The ability of storing large amounts of data in sensor devices necessitates the need for efficient indexing structures to locate required information.

The challenge with flash memories is that they are unsuitable for maintaining dynamic data structures because of their specific read, write and wear constraints; this combined with very limited data memory on sensor devices prohibits the direct application of most existing indexing methods.

In this paper we propose a suite of index structures and algorithms which permit us to efficiently support several types of historical online queries on flash-equipped sensor devices: temporally constrained aggregate queries, historical online sampling queries and pattern matching queries. We have implemented our methods using nesC and have run extensive experiments in TOSSIM, the simulation environment of TinyOS. Our experimental evaluation using trace-driven real world data sets demonstrates the efficiency of our indexing algorithms.

1. Introduction

Wireless sensor networks (WSN) have received considerable attention in recent years, deployed in a variety of environments to take measurements that would otherwise be impractical due to hostile environments, remote locations and the extended periods of time required [4,17,22,23,25,28,30].

The recent trend of equipping sensors with flash memories (such as RISE [29] and PRESTO [31]) allows for sensors to store large amounts of data locally. Sensors can now exploit the low energy requirements of processing and data storage by only transmitting the processed data results in response to specific queries. Such in-network storage schemes yield significant energy savings since the communication costs are greatly reduced, prolonging the lifetime of the sensor network.

The ability to store large amounts of raw data necessitates an efficient method of retrieving historical stream data in real-time upon request. The flash memories, however, have many unique characteristics which make the direct application of existing data indexing techniques impractical. Recently, several indexing structures have been proposed for flash-based sensors (such as [5], Presto [31] Capsule [27], B-Flash [32], R-Flash [35], FlashDB [19], Microhash [33]). These are capable of answering simple online aggregate queries such as:

SELECT AGR FROM SENSOR DATA (where AGR = MAX,MIN,AVG,. .)

In real world applications, however, the queries issued by the user are often more complex than the simple online aggregate queries. For example, in many cases we may want to compare the current data with data collected at specific

* Corresponding author. Tel.: +30 210 727 5227; fax: +30 1 909 787 4643.
E-mail address: dg@di.uoa.gr (D. Gunopulos).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.04.002

time in the past. In other cases, we may want to identify patterns or common occurrences. In all such cases, we need efficient techniques to store and index historical data (i.e., data collected in the past) and keep it up-to-date.

Consider, for example, a person studying weather trends. He may be interested in asking temporally constrained queries, such as "*Find the number of sunny days in the month of February*", perhaps to compare with different months.

Another example is the ZebraNet mobile sensor system [25] with GPS enabled devices to gather information about the environment, where the sensor device's flash memory can only hold up to 26 days of collected information. When dealing with such large amounts of data, it is often sufficient to supply approximate answers based on a good sample of sensor data instead of fetching large amounts of the original data to compute exact results [18]. Taking a random sample is a very efficient way to approximate the average, histogram or quantile of the original sensor data. In our work, we are interested in withdrawing a random sample from the sensor data up to any timestamp for data analysis and query approximation. For example, one wants to approximate 90% quantile of all the sensor data until 2006 by withdrawing a random sample from all the sensor data collected up to 2006. In most cases, the *SRAM* of the sensor device is too small (i.e. 3 kB ~ 10 kB) [3] to keep good samples of the sensor data for all the possible time moments. However, we need sophisticated implementations for random sampling if we want the sampling to be done efficiently (see [1] for a comprehensive survey). Examples of the queries we want to answer are the following:

SELECT AGR FROM SENSOR DATA WHERE MONTH = MAY

SELECT RANDOM SAMPLES FROM SENSOR DATA UP TO MAY 2007

WHERE $\text{SAMPLE}_{\text{SIZE}} = K$

The difficulty of realizing indexing structures and samples to efficiently meet the needs of these queries lies in the unique characteristics (wear and delete constraints) of the flash memory and the very limited *SRAM* capacities. We are often dealing with not only large amounts of data, but also large amounts of information required to sufficiently index that data (often more than can be contained within the sensor's *SRAM*). Retrieving this historical data in real-time has been considered in the past to be prohibitively expensive. It is not trivial to maintain dynamic data structures in flash without repeatedly updating pages which contain frequently changing data. For example, trying to maintain a priority heap can cause unacceptable levels of wear in the block containing the root of the heap.

Our contribution: In this paper, we propose a suite of index structures and algorithms which permit us to efficiently support real-time querying of historical data in flash-equipped sensor devices. More specifically, we will address the following queries:

- Temporally Constrained Aggregate Queries: compute the aggregate sensor reading over any time period.
- Historical Online Sample Queries: compute a random sample of the data generated by the sensor up until any time.
- Pattern Matching Queries: find the most similar data pattern to a given query pattern.

By providing answers to these queries we are able to support a much wider range of applications. We have implemented our techniques using nesC [15], the programming language of the TinyOS [14] operating system. Our trace-driven experimental evaluations confirm the efficiency of both the proposed indexing structures and the corresponding query algorithms.

2. Background

The conservation of energy in a sensor network has a major impact on the lifespan of individual sensor devices; this has already been vastly improved by the introduction of flash memories into sensor devices. The relatively low energy requirements of storing data on flash memories have allowed sensor devices to store data locally and only transmit relevant information when requested by specific queries. Although this has resulted in large energy savings, there are still significant inefficiencies caused by the particularities of flash memories which cause suboptimal responses to large classes of queries. To provide an efficient solution to these queries, we investigate the memory architecture of the sensor devices, the distinct characteristics of flash memories and several practical queries in sensor network applications.

In this section we briefly present the memory architecture of the sensor devices, along with the distinct characteristics of the flash memories and several practical queries in sensor network applications.

2.1. Flash-equipped sensors

The memory architecture in a flash-equipped sensor device has many differences from that of traditional computer storage. A flash-equipped sensor device is composed of five major components (shown in Fig. 1):

- *Micro control unit*: the core component of a sensor device, it performs all the data processing and computation
- *Sensor unit*: a sensor device consists of one or more sensor units used to measure various environmental quantities

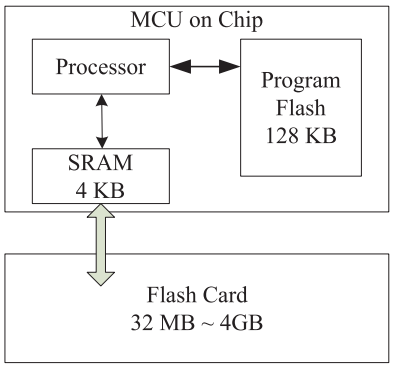

Fig. 1. Memory Architecture of a typical flash-equipped sensor.