Outlier-Aware Data Aggregation in Sensor Networks

Antonios Deligiannakis #1, Vassilis Stoumpos #2, Yannis Kotidis *3, Vasilis Vassalos *4, Alex Delis #5

#*Department of Informatics, University of Athens*
Athens, Greece
1adeli@di.uoa.gr 2stoumpos@di.uoa.gr 5ad@di.uoa.gr

**Department of Informatics, Athens University of Economics and Business*
Athens, Greece
3kotidis@aueb.gr 4vassalos@aueb.gr

***Abstract*— In this paper we discuss a robust aggregation framework that can detect spurious measurements and refrain from incorporating them in the computed aggregate values. Our framework can consider different definitions of an outlier node, based on a specified *minimum support*. Our experimental evaluation demonstrates the benefits of our approach.**

I. Introduction

Recent advances in remote sensing equipment, computing hardware and communication technology have made the creation and deployment of large scale sensor networks easier and cheaper. Their uses in monitoring natural or artificial conditions and processes in diverse physical environments – such as battlefield surveillance, wildlife monitoring, healthcare, traffic monitoring, agriculture, production monitoring – have subsequently multiplied. A lot of recent research has focused on the problem of efficiently processing declarative queries in such networks. The majority of these efforts focuses on answering aggregate queries, which are of great importance to surveillance applications [1], [2] and on enabling *in-network processing* by combining individual sensor readings as they are communicated towards a *base station* through an *aggregation tree*. An equally important line of research addresses the issue of data cleaning of sensor readings [3], [4]. A measurement obtained by a node is only an approximation of the physical quantity observed and is constrained in accuracy and precision by the characteristics of the sensing device. Sensors are also often exposed to severe conditions that adversely affect their sensing devices, thus resulting in readings of low quality. Moreover, sensor nodes often provide imprecise individual readings after a failure, i.e., they tend to *fail dirty* [3]. Thus, data processing applications using sensor networks must deal with information that is at times unreliable and unpredictable.

The goal of our techniques is to provide a resilient query processing platform for aggregate queries over a network consisting of cheap, wireless sensor nodes that are prone to dirty data. This requires identifying potentially multiple "abnormal" readings produced by sensor nodes and removing them from the computation of the aggregate function. In order for our techniques to scale to large sensor networks, our proposed algorithms should follow the in-network paradigm.

In this work, we introduce a query execution model that, together with the aggregates, also recognizes and reports to the user a concise set of readings that are believed to be outliers, along with a set of characteristic values, *i.e., witnesses*, that have been used to derive the requested aggregates. It is important that the user/application is able to control the amount of *support* required on the readings of a node by other nodes in the network. Furthermore, while the computation of outliers is carried out as a side process during query aggregation, we need to be able to derive proper routing paths, based on simple statistics collected during the query processing. These statistics would be utilized in order to periodicaly reorganize the aggregation tree and reduce bandwidth as well as energy consumption.

II. Outlier-Aware Data Aggregation

Similarly to [5], we consider aggregate queries of the form:

```
SELECT AggrFun(s.value)
FROM Sensors s
WHERE cond
SAMPLE PERIOD e FOR t
```

where AggrFun() is a distributive or algebraic function such as MAX,MIN,COUNT,SUM,AVG. It is easy to extend our work to capture GROUP BY queries as well. The period (e) in the above query is the *epoch duration* and determines the frequency at which data is acquired from the sensors. Parameter (t) specifies the life span of the query.

In this paper we extend the in-network computation framework, and define as an outlier a node that can be witnessed by fewer than *MinSupport* other nodes. The *witness test* can be performed through a variety of similarity tests, as described in Section III. Each transmitted witness and outlier value does not necessarily reach the `Root` node that poses the query. These values may be witnessed at some intermediate nodes and removed from the transmitted data. This is the intuition of our algorithm for periodically reorganizing the aggregation tree. If we monitor how often the witness test between pairs of sensor nodes succeeds, then each node can select a parent in the aggregation tree through which it expects to find the most witnesses and relatively nearby, in number of hops.

Consider for example a query that computes the average temperature in the area covered by the sensornet depicted in Figure 1 and that the desired minimum support is 2. For simplicity we assume that the aggregate is collected at node S_1, which acts as the base station in our example. We use x_i

978-1-4244-1837-4/08/$25.00 © 2008 IEEE

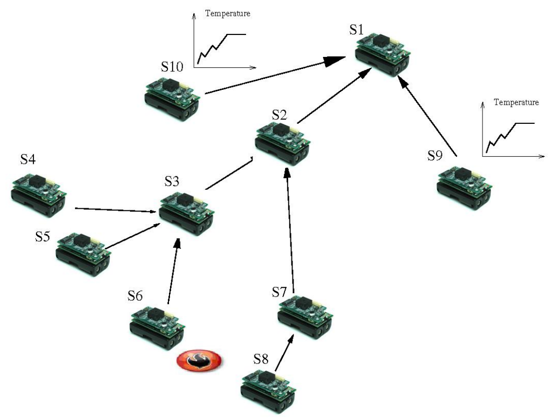

Fig. 1. Sample Aggregation Tree

to denote the temperature readings provided by node S_i. The aggregation tree is also depicted in the figure. A typical way of computing the average value from the temperature readings is for each node to compute the SUM and COUNT functions in its subtree and propagate them to its own parent [5], [6].

In the Figure, nodes S_6, S_7 and S_8 can observe an open fire and, therefore, their readings are expected to be a lot higher (and fluctuate more) than, for example, those of node S_2. When node S_3 receives the values of its children nodes, the readings of node S_6 appear to be suspicious, since no other node in that subtree is aware of the fire. If we decide to reject the reading of S_6 (for instance using majority voting), the monitoring application will lose a crucial observation. Techniques based on smoothing [3], [7] will also obscure the outcome, especially if a lot more nodes are rooted at node S_3.

In our framework, we tentatively put the reading of node S_6 in the list of outliers O_3 communicated by node S_3 to its parent node in the tree S_2. Now let us concentrate on node S_2. This node will receive from the left subtree a pair of aggregate values $(x_3+x_4+x_5, 3)$ from node S_3, an outlier list containing only the value x_6, and a witness list containing one of the values x_3, x_4 or x_5 (depending on which node was selected as the witness). The right-subtree contains nodes S_7 and S_8. Their readings are similar, but these nodes reach only a support of 1, which is less than the desired minimum support of 2. Thus, S_7 includes the readings of S_7 and S_8 in the outlier list transmitted to node S_2. At this point, at node S_2, the nodes S_6, S_7 and S_8 can reach the required support in order to be included in the partial aggregate. Moreover, one of these nodes, will be selected to become a witness in S_2. Finally, we note that the sensor nodes S_9 and S_{10} represent the case of two node that fail dirty. These nodes start reporting abnormal high readings that cannot be justified by any of its neighbors or the values that are provided in the witness list. If the query had specified a minimum support of 1, these two nodes could at some epochs witness each other and, thus, end up including their readings in the reported aggregate. However, this cannot occur in our example with the minimum support value of 2.

We also can see from Figure 1 that an alternative organization of the aggregation tree where both S_6 and S_8 had selected S_7 as their parent node could lead to bandwidth savings at nodes S_3 and S_7. In particular, node S_3 will not have to transmit an outlier to node S_2. Moreover, we note that in node S_7 these three sensors could gain enough support to be included in the computed aggregate and have all three of them replaced by a single witness.

III. Similarity Testing between Nodes

Our algorithm frequently tests whether the recent measurements of two sensor nodes are similar. If this is the case, then each sensor can *witness* the measurements of the other sensor. We will reference this similarity test in this paper as a *witness test*. In our work we have explored the following alternatives:

- **Correlation Coefficient:** If we consider the readings x_k, x_l of sensor nodes S_k, S_l respectively as random variables, the correlation coefficient $r_{k,l}$ is defined as:

$$r_{k,l} = \frac{cov(x_k, x_l)}{\sigma_{x_k}\sigma_{x_l}} = \frac{E(x_k x_l) - E(x_k)E(x_l)}{\sqrt{E(x_k^2) - E^2(x_k)}\sqrt{E(x_l^2) - E^2(x_l)}}$$

where $cov()$, σ and $E()$ stand for the covariance, standard deviation and expected value respectively. The correlation coefficient takes values in the interval [-1,1]. Given a threshold θ provided by the application and communicated to the nodes during the query initialization, the witness test succeeds when $r_{k,l} \geq \theta$.

- **Extended Jaccard Coefficient:** If we consider the readings x_k, x_l of sensor nodes S_k, S_l respectively as vectors and denote their dot product as $x_k \cdot x_l$, the extended Jaccard coefficient $j_{k,l}$ is defined as:

$$j_{k,l} = \frac{x_k \cdot x_l}{\|x_k\|^2 + \|x_l\|^2 - x_k \cdot x_l}$$

Again, given a threshold θ provided by the application and communicated to the nodes during the query initialization, the witness test succeeds when $j_{k,l} \geq \theta$.

- **Regression-Based Approximation:** If we consider the readings x_k, x_l of sensor nodes S_k, S_l respectively as random variables, we may apply approximation techniques to identify the error of approximating x_l given x_k. For example, the work in [8], [9] proposed using a linear regression model for this approximation. Using such a technique, we may determine that the witness test will succeed if the reconstruction maximum/absolute relative error for x_l is below an application-defined threshold (i.e., 2%).

Since the similarity tests cannot be performed simply based on the last received measurement of the sensor nodes, but also require the knowledge of measurements from the recent past, our algorithm maintains in a small cache the latest measurements received by descendant sensor nodes.

IV. Reorganization of the Aggregation Tree

A poor initial construction of the aggregatuion tree could lead to many nodes finding similar measurements (i.e., support) by other sensors only at the `Root` node, or at nodes near the `Root`. This would essentially result in a computation with bandwidth requirements close to those of a SELECT *

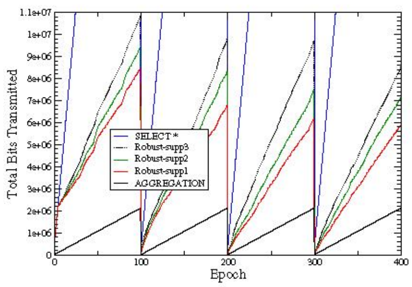

Fig. 2. Bandwidth consumption in synthetic dataset (per 100 epochs)

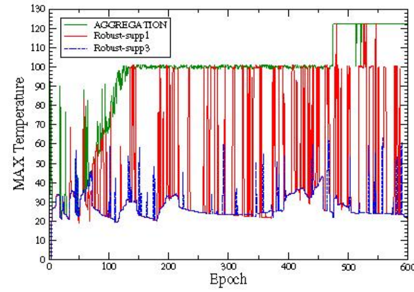

Fig. 3. Computed MAX Temperature, Intel data with noise, Correlation Coefficient

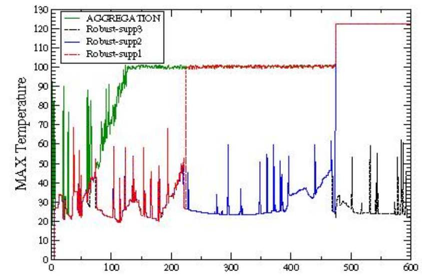

Fig. 4. Computed MAX Temperature, Intel data with noise, Extended Jaccard Coefficient

query. One the other hand, it would seem plausible to route the witnesses and outliers towards the direction where they are expected to be "matched" (witnessed) more quickly by the most outliers or witnesses, received through other parts of the aggregation tree. In our framework we periodically reorganize the aggregation tree by utilizing statistics of the form "*how many times a node S_i has witnessed another node S_j*" in the previous epochs.

V. Experiments

We developed a simulator for testing the algorithms proposed in this paper under various conditions. In all experiments the locations of the sensor nodes were dispersed at random locations over a rectangular area. For the first experiment, we generated a large sensor network of 400 sensor nodes and defined 5 classes of data to control the behavior of the sensors. The readings of nodes that belong to the same class make random walks with different steps, and at the same direction. Each node initially belongs to the default class 0. We then generated 4 events at random locations and assigned all nodes within distance 30 from the centers of the events to belong to the same class (classes 1 to 4). In Figure 2 we show the resulting bandwidth consumption for a minimum support of 1,2 and 3. The aggregation tree reorganization is performed every 100 epochs and its overhead is included in the graphs (we account for this cost only in our method). In the Figures we can see that the aggregation tree gradually improves, as more statistics are collected. Compared to our techniques, the SELECT * case, were evaluation of outliers is performed at the base station, results in up to 7.4 times more transmitted bits and energy consumption.

We also obtained temperature measurements from 48 motes in the Intel Labs data set [10]. In that data set, one of the sensor nodes fails dirty at some point, increasing its temperature until it reaches 122 degrees. In this experiment, we increase the complexity of the data set by: (1) Specifying for each sensor a 6% probability that it will fail-dirty at some point; (2) Each node with probability 0.5% at each epoch obtains a spurious measurement, which we model as a random reading between 0 and 100 degrees. In Figures 3 we show the resulting reported aggregate for this very challenging data set. As we can see, the aggregate computed by pure in-network aggregation quickly becomes meaningless. Our technique with a minimum support of 1 and a witness threshold of 0.7 provides significant improvements, but is still characterized by too many spikes. However, the robust aggregate obtained by a minimum support of 3 (depicted with the blue line) is significantly more accurate and manages to eliminate the spurious readings and the readings of nodes that fail-dirty in all but a few cases. We also examined an alternative technique were we perform the witness test by using the extended Jaccard coefficient [11]. Because the extended Jaccard coefficient is sensitive to the relative difference in the magnitude of the values, in Figure 4 we notice that it performs significantly better, as the readings of nodes that fail-dirty and have reached a large value cannot witness those of functional nodes.

VI. Conclusions

In this paper we presented a robust aggregation framework that can tolerate outlier readings that often arise in sensor network applications. We discussed different definitions of an outlier node, based on a specified minimum support, and considered techniques that alter the aggregation tree in order to minimize the bandwidth and energy drain during the query evaluation.

References

[1] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Processing Approximate Aggregate Queries in Wireless Sensor Networks," *Information Systems*, vol. 31, no. 8, pp. 770–792, 2006.
[2] A. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis, "Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks," *VLDB Journal*, 2004.
[3] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom, "Declarative Support for Sensor Data Cleaning," in *Pervasive*, 2006.
[4] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online Outlier Detection in Sensor Data Using Non-Parametric Models," in *VLDB*, 2006.
[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A Tiny Aggregation Service for ad hoc Sensor Networks," in *OSDI*, 2002.
[6] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *SIGMOD Record*, vol. 31, no. 3, 2002.
[7] S. Jeffery, M. Garofalakis, and M. Franklin, "Adaptive Cleaning for RFID Data Streams," in *VLDB*, 2006.
[8] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Compressing Historical Information in Sensor Networks," in *ACM SIGMOD*, 2004.
[9] Y. Kotidis, "Snapshot Queries: Towards Data-Centric Sensor Networks." in *ICDE*, 2005.
[10] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," in *VLDB*, 2004.
[11] X. Xiao, W. Peng, C. Hung, and W. Lee, "Using SensorRanks for In-Network Detection of Faulty Readings in Wireless Sensor Networks." in *MobiDE*, 2007.