

1 Medians and Beyond: New Aggregation Techniques for Sensor Networks

Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, Subhash Suri

This paper was presented by Jose Reyes Alamo.

1.1 Motivation

With the miniaturization of hardware it has been possible to combine computing, storage and communication capabilities in tiny sensors. This has increased the number of new applications civil and military. Also other phenomena can be sensed other than the normal temperature or sound. Even though a lot of progress has taken place in the development of sensor these devices have constraints like energy, computation, storage and reliability.

A traditional monitoring/detection approach is expensive for a sensor network in terms of communication. Usually in these applications each sensor collects data about the environment and sends it to a base station. In sensor networks communication is three orders of magnitude more expensive than computing in terms of energy consumption, so combining messages and then forwarding has been proposed.

Energy efficient query processing is based on the following observations:

1. Individual sensor values does not hold much value. Usually the interest is in some function over the dataset and not just the particular reading of a particular sensor
2. Extracting all the data out of a sensor network is very inefficient. It is more efficient to gather an aggregate measure such as Average, Sum, Count, Min/Max. This is usually what we want to know about the data and also such aggregate measures reduce the communication and computations costs.

Knowing the data distribution can help in performing more complicated queries like median and quantiles and will help with data analysis

1.2 Background

In this paper a network with n nodes is considered. Sensor reading is assumed to be an integer in the range $[1, \delta]$, where δ is the maximum possible value of the signal. The network contains a

special node as the base station. When a query is initiated, the sensors organized themselves in a spanning tree rooted at the base station node. The only requirement for this network is that there is no loops and no duplicate packets. Reliable links with no packets lost is assumed. It is shown in the paper that with aggregate techniques only approximate functions are possible.

1.3 The Quantile Digest

The quantile digest or q-digest, is a summary structure that support single value queries such as AVERAGE and more complex queries such as HISTOGRAM. This structure has the following properties:

1. Error-Memory Trade-off: The user can decide the appropriate message size and error trade-off depending on the application
2. Confidence Factor: The theoretical worst case error bound applies to only very specific data set that are unlikely to arise in practice
3. Multiple Queries: Once the q-digest query has been completed, the q-digest at the base station contains a lot of information that can be used to get other measures without querying again.

The core idea of q-digests is that it adapts to the data distribution and automatically groups values into variable size buckets of almost equal weight.

1.4 Properties of q-digest

Consists of a set of buckets of different sizes and their associated counts. The set of possible different buckets are chosen from a binary partition of the value $1, \dots, \delta$. The depth of the tree T is $\log(\delta)$. Each node $v \in T$ can be considered a bucket and has a range $[v.min, v.max]$ which defines the position and width of the bucket. Every bucket v has a counter $count(v)$ associated with it.

A q-digest is a subset of these possible bucket and their associated counts. The size of the q-digest is determined by a compression parameter k , and given this k a node must satisfy the following digest property:

$$\begin{aligned} count(v) &\leq \lfloor n/k \rfloor \\ count(v) + count(v_p) + count(v_s) &> \lfloor n/k \rfloor \end{aligned}$$

1.5 Building a q-digest

To construct the q-digest we will merge and reduce the number of buckets in a hierarchical way . We will go over all nodes starting from the bottom and checking if a node violates the

digest property stated in the previous section. Since we are going bottom up, only the second digest property can be violated. For later notational convenience we define a relation Δ_v on the node v as follows:

$$\Delta_v \equiv \text{count}(v) + \text{count}(v_l) + \text{count}(v_r)$$

where, v_l and v_r are the left and right child of node v . If any node v whose child violates the second digest property, these children are merged with node v and its count is set to Δ_v and the children are deleted. The algorithm to execute this hierarchical merge is described as COMPRESS and takes as parameters the uncompressed q-digest Q , the number of readings n and compression parameter k .

1.6 Merging q-digests

The above algorithm showed how the q-digest is built if all the data is available on a single sensor. We need to be able to build the q-digest in a distributed fashion for these approach to be useful in sensor networks. Since merging multiple q-digests is no harder than merging two digests, two q-digests can be merged the following way. The idea is to take the union of the two q-digest and add the counts of buckets with the same range ($[min, max]$). Then, we compress the resulting q-digest. The formal MERGE algorithm takes as parameters 2 q-digest, and the n and k value for each q-digest

1.7 Space Complexity and Error Bound

In this subsection a series of lemmas and theorems are stated that shows the space accuracy and error bounds found when using q-digests

LEMMA 1. A q-digest (Q) constructed with compression parameter k has a size at most $3k$.

LEMMA 2. In a q-digest (Q) created using the compression factor k , the maximum error in count of any node is $\frac{\log(\delta)}{k}n$

LEMMA 3. Given p q-digests Q_1, Q_2, \dots, Q_p , built on n_1, n_2, \dots, n_p values, each with maximum relative error of $\frac{\log(\delta)}{k}$, the algorithm MERGE combines them into a q-digest for $\sum n_i$ values, with the same relative error.

THEOREM 1. Given memory m to build a q-digest, it is possible to answer any quantile query with error ϵ such that $\epsilon \leq \frac{3\log(\delta)}{m}$

1.8 Quantile query

Given a fraction $q \in (0, 1)$, find the value whose rank in sorted sequence of the n values is qn . To perform this query the following strategy is followed: Sort the nodes of q-digest in increasing right end-points (or max values). This list gives the post-order traversal of the nodes in the q-digest. Scan the list adding the counts. When this sum becomes greater than qn , report $v.max$ as the quantile estimate

1.9 Other queries

Other quantities such as inverse quantiles, range and consensus query are possible and the strategies are listed below.

- Inverse Quantile: Given value x , we want to determine its rank in the sorted sequence of input values. In this case, we again make the same sorted list (L), and traverse it from beginning to end. We report the sum of counts of buckets v for which $x > v.max$ as the rank of x .
- Range Query: Find the number of values in the given range $[low, high]$ To do this we perform two inverse quantile queries to find the ranks of low and high, and then take their difference.
- Consensus Query: Given a fraction $s \in (0, 1)$, find all the values which are reported by more than sn sensors. This can be thought of finding a value on which more than certain fraction of sensor agreed. These values are called Frequent items.

1.10 Experimental Evaluation

The following describes the assumptions and the scenario for the simulations in this paper. The simulator takes as input the network topology and the readings of the sensors. A node that acts as a base station sends a query to its children and this query goes all the way to the leaf nodes. Leaf sensors send their values to their parents. Sensors aggregate their children values with their own values in a single package and forwards it to its parent all the way up to the base station. Once the base station has the q-digest the all the queries are performed on the base station.

The scenario assumes that the sensors have a fixed radio range. The Routing Tree required by the simulation is a BFS tree whose root is the base station. A variable area which depends on the number of nodes is also assumed. They had two kinds of data with which they run the algorithm random and correlated. They compared their results with non-aggregated data scheme.

Their simulation shows that for computing the histogram using aggregation the final values are really close in most of the cases that with using the exact values. They also showed that the percent of error is a function of the message size, but with a reasonable size of 200-400b an error of less than 5% can be achieved. Also it is shown that for a network with several sensors, using aggregation the message size can be kept constant and we will have small error while without aggregation as the number of sensors grow so it does the message size. Future work is to relax some of the assumptions made over the network and extend the q-digest structure to multidimensional data.