Ad Hoc & Sensor Wireless Networks, Vol. 0, pp. 1–24
Reprints available directly from the publisher
Photocopying permitted by license only

A Two Tiers Data Aggregation Scheme for Periodic Sensor Networks

JACQUES M. BAHI, ABDALLAH MAKHOUL⋆ AND MAGUY MEDLEJ

FEMTO-ST Institute, University of Franche-Comté
Besançon, France

Received: September 22, 2011. Accepted: May 26, 2012.

The expected lifetime of any wireless sensor network is a critical issue since sensor nodes are powered by small batteries. The propagation of redundant highly correlated data is costly in terms of system performance, and results in energy depletion, network overloading, and congestion. Data aggregation is regarded as an effective technique to reduce energy consumption and prevent congestion. This paper objective is to identify near duplicate nodes that generate similar sets of collected data in periodic applications. We propose a new prefix filtering approach that avoids computing similarity values for all possible pairs of sets. We define a new filtering technique based on the quality of information. To the best of our knowledge, the proposed algorithm is a pioneer in using "sets similarity functions" for data aggregation in sensor networks. To evaluate the performance of the proposed method, experiments on real and synthetic sensor data have been conducted. The analysis and the results show the effectiveness of our method dedicated to sensor networks.

Keywords: Sensor networks; data aggregation; set joins similarity; frequency filtering; real data measurements;

1 INTRODUCTION

Wireless sensor networks have received enormous attention over past few years, due to a wide range of potential applications (environmental, ecological, military, etc). A typical sensor network is expected to consist of a large

⋆ E-mail: abdallah.makhoul@univ-fcomte.fr number of sensor nodes deployed randomly in a large scale. The main functionality of a sensor node is to measure environmental values using embedded sensors, and transmit them to a base station called “sink”. The sensed data needs to be analyzed, which eventually serves to initiate some action. Usually, such nodes have limited power, storage, communication, and processing capabilities, leading to an energy consumption deficiency.

This deficiency becomes important in the case of periodic applications, where sensors monitor a given phenomenon and send notifications and measurements back to the sink at each period p. A significant amount of redundant data is likely delivered to the sink, particularly in case of dense networks, thus, wasting precious bandwidth and energy resources. For this reason aggregation in sensor networks is regarded as an effective technique to reduce energy consumption as well as preventing congestion [14] [2].

An important topic addressed by the sensor networks community over the last several years has been in-network aggregation. The idea is that cumulants, time intervals, or summaries can be computed directly within the network by sensor nodes, endowed with computational power, thus avoiding the expensive transmission of all sensor data to the sink. Previous work was studied the data aggregation as the computation of statistical means and moments, as well as other cumulative quantities that summarize the data obtained by the network. To some extent these concerns can be alleviated by providing tools for filtered aggregation, where only a selected subset of nodes participate in the aggregation, according to the values of some of their sensors, their geographic location, etc.

In this paper we are interested in exploring a new part of the filtering aggregation problem, by focusing on identifying the similarity between sets of data generated by neighboring nodes further to a local processing technique. One of the issues accompanying the growth of sensed data in periodic sensor networks and the need to integrate data from different sensor nodes is the existence of near duplicate sets of data. As sensor nodes are deployed randomly, it is possible that neighboring nodes generate similar sets of data, yet they are not bitwise identical. Our objective is to identify similarities between near sensor nodes, and integrate their sensed data into one record while preserving information integrity.

A quantitative approach in identifying two similar sets of sensed data is by using a similarity function. Such function measures the degree of similarity between two sets and returns a value between 0 and 1. A higher similarity value indicates that the sets are more similar, thus we can treat pairs of sets with high similarity values as duplicates, and send only one set to the sink instead of sending both.

Similarity functions were used in various domains and applications in order to identify near duplicate objects (data). For instance, for Web search engines [12], Web mining applications [3], detecting plagiarism [13], collaborative filtering in data mining [1], *etc*. To the best of our knowledge, we are the first to use these functions for data aggregation in sensor networks. Recently, many studies have proposed new algorithms that define the similarity between objects or records. These algorithms are classified into three categories, inverted index based methods [19], prefix filtering methods [5], and signature based methods [20]. The most of these methods seem to be pretty complex for wireless sensor networks usually generating large amount of candidate pairs, all of which need to be verified by the similarity function.

In this paper, we provide a new prefix filtering method to study the sets similarity in sensor networks. We propose a frequency filtering technique, which exploits the ordering of measurements according to their frequencies. A frequency of a measure is defined by the number of occurrences of this measure in the set. Our method is divided into two phases: the first one is done at the nodes level, where each node compacts its measurements set according to a $link$ function. The second is defined at the aggregator level where the frequency filtering technique will be applied. To evaluate our approach we conducted extensive experimental study using synthetic and real data measurements and we compare our approach to the existing ToD protocol [10,15] for data aggregation in sensor networks. The obtained results show the effectiveness of our method which significantly reduces the number of duplicate data and outperforms existing methods in terms of data accuracy and energy consumption.

The rest of the paper is organized as follows, Section 2 gives an overview on related works reported on data aggregation in sensor networks. The problem statement, the similarity functions and our proposed method are presented in Section 3. Experimental results are given in Section 4. Section 5 concludes the paper with some directions to a future work.

2 PREVIOUS WORK

Data aggregation in wireless sensor networks has been well studied in recent years [2, 4, 23, 28, 30]. It means computing and transmitting partially aggregated data to the end user rather than transmitting raw data in networks to reduce the energy consumption [22]. There are a vast amount of extant works on in-network data aggregation in the literature.

Some of the methods reported recently are query based methods [17, 26]. A query is generated at the sink and then broadcasted through the network. Some nodes just process the query, while others propagate it, receive partial results, aggregate results, and send them back to the sink. Various algorithmic techniques have been proposed to allow efficient aggregation without increasing the message size [7].

Some works, such as [6, 8, 16], use the clustering methods for aggregating data packets in each cluster separately. Among these methods, the LEACH protocol [21, 27]. LEACH operation is divided into two phases: the setup phase and the steady state phase. In the first phase, a predetermined fraction of nodes, elect themselves as cluster heads by comparing a chosen random number with a predefined threshold. In the second phase, cluster heads, aggregate reported data from their cluster member nodes and forward aggregated data to the sink. In [16], the authors propose a self-organizing method for aggregating data based on the architecture CODA (Cluster-based self-Organizing Data Aggregation), based on the Kohonen Self-Organizing Map to aggregate sensor data in cluster. In a first step before deployment, the nodes are trained to have the ability to classify the sensor data. Thus, it increases the quality of data and reduces data traffic as well as energy-conserving. An adaptive data aggregation (ADA) scheme for clustered sensor networks has been proposed in [6]. In this scheme, a time based as well as spatial aggregation degrees are introduced. They are controlled by the reporting frequency at sensor nodes and by the aggregation ratio at cluster heads (CHs) respectively. The function of the ADA scheme is mainly performed at sink node, with a little function at CHs and sensor nodes.

In a tree based network as our presented work, sensor nodes are organized into a tree where data aggregation is performed at aggregators along the tree to arrive to the sink. Tree based data aggregation approaches are suitable for in-network data aggregation. Tan *et al.* [24] have proposed a power efficient data gathering and aggregation protocol (PEDAP). PEDAP is a minimum spanning tree based protocol with main objective to maximize the lifetime of the network in terms of number of rounds, where each round corresponds to aggregation of data sent from different sensor nodes to the sink. The authors in [10, 15] have proposed Tree on DAG (ToD) for data aggregation, a semistructured approach that uses Dynamic Forwarding on an implicitly constructed structure composed of multiple shortest path trees to support network scalability. The key principle behind ToD was that adjacent nodes in a graph will have low stretch in one of these trees in ToD, thus resulting in early aggregation of packets.

Some other methods are also reported for data aggregation in sensor networks. In [9], a learning automate-based data aggregation method in sensor networks where environmental changes cannot be predicted beforehand is provided. In the proposed method, each node in the network is equipped with a learning automaton. These learning automate in the network collectively learn the path of aggregation with maximum aggregation ratio for each node by transmitting its packets toward the sink. The authors in [25] propose a

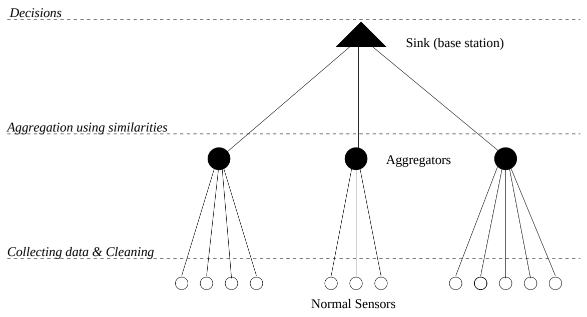


FIGURE 1
Data aggregation scheme

Dynamic Data-Aggregation Aware Routing Protocol (DDAARP). It builds dynamic routes, which improves the cost and quality of the final routing tree. It also reduces the number of messages needed to set up a routing tree, maximize the number of overlapping routes, selects routes with the highest aggregation rate, and performs reliable data aggregation transmission.

We are not aware of previous work that specifically addresses the set joins similarity aggregation problem. However, we have presented some of extant work on general in-network aggregation that we cannot hope to fully survey here. Thus we focus on what is mostly relevant to our approach.

3 DATA AGGREGATION - OUR APPROACH

In this section we present our approach for data aggregation in sensor networks. We consider a tree based network, where sensed data needs to be aggregated on the way to its final destination. Sensor nodes collect information from the region of interest and send it to aggregators. The aggregators can either be special (more powerful) nodes or regular sensors nodes, or mobile agents like robots that traverse the region of interest and collect the data sets. Each aggregator then condenses the data prior to sending it on as presented in Figure 1. Our data aggregation method works in two phases, the first one at the nodes level, which we call local aggregation and the second at the aggregators level. At each period p each node sends its aggregated data set to its proper aggregator which subsequently aggregates all data sets coming from different sensor nodes and sends them to the sink.

3.1 Local aggregation

In periodic sensor networks, we consider that each sensor node i at each slot s takes a new measurement y_{is}. After $p-1$ slots (a period p) the node i

forms a new set of sensed measurements M_i, and sends it to the aggregator. Note that, a sensor node can take different kind of measures (e.g. temperature, humidity, light, *etc*) making of y_{is} a vector instead of a scalar. For the sake of simplicity, in the rest of the paper we shall consider that $y_{is} \in \mathbb{R}$.

It is likely that a sensor node takes the same (or very similar) measurements several times especially when s is too short. In this phase of aggregation, we are interested in identifying duplicate data measurements in order to reduce the size of the set M_i. Therefore, to identify the similarity between two measures, we provide the two following definitions:

Definition 1 *link* function. *We define the link function between two measurements as:*

$$link(y_{is_1}, y_{is_2}) = \begin{cases} 1 & \text{if} \quad \|y_{is_1} - y_{is_2}\| \le \delta, \\ 0 & \text{otherwise}. \end{cases}$$

where δ is a threshold determined by the application. Furthermore, two measures are similar if and only if their $link$ function is equal to 1.

Definition 2 Measure's frequency. *The frequency of a measurement y_{is} is defined as the number of the subsequent occurrence of the same or similar (according to the $link$ function) measurements in the same set. It is represented by $f(y_{is})$.*

Using the notations defined above we present the local aggregation algorithm which is run by the nodes themselves (see Algorithm 1). For each new sensed measurement (at each slot), a sensor node i searches for similarities of the new taken value. If a similar measurement is found, it deletes the new one while incrementing the corresponding frequency by 1.

Algorithm 1 Aggregation at the nodes level.

```
Require: new measure $y_{is_k}$, set of previous measures $M_i$.
Ensure: searching for similarities in $M_i$.
1: for every measure $y_{is_l} \in M_i$ do
2:     if ($link_{is}(y_{is_k}, y_{is_l}) = 1$ then
3:         $f(y_{is_l}) \leftarrow f(y_{is_l}) + 1$
4:         delete $y_{is_k}$
5:     else
6:         add $y_{is_k}$ to the set $M_i$
7:         $f(y_{is_k}) \leftarrow 1$
8:     end if
9: end for
```

At the end of the period p, each node i possesses a local aggregated set M_i. The second step is to send it to the aggregator which in his turn aggregates the data sets coming from different sensor nodes.

3.2 Aggregation using similarity functions

At this level of aggregation, each aggregator has received k sets of measurements and their frequencies. The idea here is identifying all pairs of sets whose similarities are above a given threshold t. For this reason we use a similarity function which measures the degree of similarity between the two sets and returns a value in [0, 1]. A higher similarity value indicates that the sets are more similar. Thus we can treat pairs of sets with high similarity value as duplicates and reduce the size of the final data to send to the sink.

Similarity Functions

A variety of similarity functions have been used in the literature such as hamming distance, overlap threshold and Jaccard similarity [1, 5, 20].

For two sets M_i and M_j:

- Overlap similarity is defined as: $O(M_i, M_j) = |M_i \cap M_j|^{\star}$.
- Jaccard similarity is defined as: $J(M_i, M_j) = \frac{|M_i \cap M_j|}{|M_i \cup M_j|}$.
- Hamming distance[†] is defined as: $H(M_i, M_j) = |(M_i - M_j) \cup (M_j - M_i)|$.

In this paper we will focus on the Jaccard similarity. It is one of the most widely accepted function because it can support many other similarity functions [20].

In our application, two given sets M_i and M_j are considered similar if and only if:

$$J(M_i, M_j) \geq t$$

where t is a threshold given by the application itself. This equation can be transformed as:

$$J(M_i, M_j) \geq t \Leftrightarrow O(M_i, M_j) \geq \alpha \qquad (1)$$

where, $\alpha = \frac{t}{1+t}.(|M_i| + |M_j|)$.

* In this paper, we do not normalize the overlap similarity to [0, 1].
[†] The notion of distance is a closely related concept to similarity functions.

Jaccard Similarity for sensor networks

In order to study the similarity functions for data aggregation in sensor networks, we define a new function for overlapping "$\cap_s$" between two sets of measurements as follows:

Definition 3 Overlap function. *Consider two sets of measurements M_1 and M_2, then we define:*
$M_1 \cap_s M_2 = \{(y_1, y_2) \in M1 \times M2 / link(y_1, y_2) = 1\}$;

For instance, we consider the following example:

Example 1. *Consider* 2 *sets of measurements:*

$$M_1 = \{y_{11}, y_{12}, y_{13}, y_{14}, y_{15}\}$$
$$M_2 = \{y_{21}, y_{22}, y_{23}, y_{24}, y_{25}\}$$

Such that:
$M_1 \cap_s M_2 = \{(y_{12}, y_{21}), (y_{13}, y_{22}), (y_{14}, y_{23}), (y_{15}, y_{24})\} \Rightarrow |M_1 \cap_s M_2| = 4.$

To evaluate the similarity between two sets we obtain:

$$J(M_i, M_j) \geq t \Leftrightarrow |M_i \cap_s M_j| \geq \alpha = \frac{t}{1+t}.(|M_i| + |M_j|) \qquad (2)$$

Prefix filtering technique

After defining the similarity function for sensor networks, we introduce prefix filtering technique that avoid computing similarity values for all possible pairs of measurement sets. However, a naïve method to compute the similarity of the received data (sets) is to enumerate and compare every pair of sets. This method is obviously prohibitively expensive for large data sets (such the case of sensor networks), as total number of comparison is $O(n^2)$. This is without considering the size of the sets.

In this paper we will use a prefix filtering method that allows the reduction of the number of comparisons between sets. Several approaches to traditional similarity join between sets are based on the prefix filtering principle [1, 5]. The intuition is that if two sets are similar, some fragments of them should overlap with each other. An inverted index maps a given measurement m to a list of identifiers of sets that contain m_i such that $link(m_i, m) = 1$. After inverted indices for all measures in the set are built, we can scan each one, probe the indices using every measure in the set M, and obtain a set of candidates; merging these candidates together gives us their actual overlap with the current set M; final results can be extracted by removing sets whose overlap with M is less than $\lceil \frac{t}{1+t}.(|M_i| + |M_j|)\rceil$(Equation 1).

The main problem of such approach is that the inverted lists of some sets can be very long. These long inverted lists incur significant overhead for building and accessing them. In addition, computing the actual overlap by probing indices essentially requires memorizing all pairs of sets that share at least one measure, a number that is often prohibitively large. To avoid this problem we propose an approach based on prefix filtering algorithm with defined measurement order and while taking into account their frequencies.

Our approach is based on the intuition that if two data sets are similar, some measurements of them should overlap with each other, and especially the ones that have *higher frequencies values*.

Definition 4 Ordering $\mathcal{O}$. *We define an ordering $\mathcal{O}$ which arranges the measurements of a given set by the decreasing order of their frequencies.*

Then we formalize this intuition by the following $Lemma$ inspired from [5]:

Lemma 1. *Consider two sets of sensor measures M_i and M_j, such that their elements are sorted in the order of $\mathcal{O}$. Let the p-$prefix$ be the first p elements of M_i. If $|M_i \cap_s M_j| \geq \alpha$, then the $(|M_i| - \alpha + 1)$-$prefix$ of M_i and the $(|M_j| - \alpha + 1)$-$prefix$ of M_j must share at least one element.*

Proof. Lemma 1 can be proven similarly to the lemma of page 6 in [5].

Our algorithm (cf 2) is then designed as follows: first we built inverted indices on measures that only appear in the prefix of each set; then we generate for each set M_i a set of candidates that may be similar with it. This is done by merging sets identifiers returned by probing the inverted indices for elements in the prefix of each set; finally we verify the similarity of each candidate and if it is higher than a given threshold we eliminate the set which has lower cardinality.

A subtle technical issue is that the prefix of a set depends on the sizes of the other set to be compared and thus cannot be determined before hand. The solution is to index the longest possible prefixes for a set M_i. It can be shown that we only need to index a prefix of length $|M_i| - \lceil t.|M_i|\rceil + 1$ for every set M_i to ensure the prefix filtering based method does not miss any similarity result. The major benefit of this approach is to minimize the

Algorithm 2 Aggregation at the aggregators level.

```
Require: $R$ is a multi set of sets of measures, each set is sorted by the ordering $\mathcal{O}$, a
    Jaccard similarity threshold $t$
Ensure: $S$ is a multi set of measurements sets after suppression of similar sets of $R$
 1: $S \leftarrow \emptyset$
 2: $I_i \leftarrow \emptyset$ ($1 \leq i \leq$ total number of measures)
 3: for each set $A \in R$ do
 4:     $X \leftarrow$ empty map from set id to int
 5:     $sumComFreq \leftarrow$ empty map from set id to int
 6:     $p \leftarrow |A| - \lceil t \times |A| \rceil + 1$
 7:     for $i \leftarrow 1$ to $p$ do
 8:         $w \leftarrow A[i]$
 9:         if ($I_{w_s}$ exists such that $link(w, w_s) = 1$) then
10:             for each $((B, l),\ f(B[l])) \in I_{w_s}$ do
11:                 $X[B] \leftarrow X[B] + 1$
12:                 $sumComFreq[B] \leftarrow sumComFreq[B] + f(B[l])$
13:             end for
14:             $I_{w_s} \leftarrow I_{w_s} \cup \{A, i\}$
15:         else
16:             create $I_w$
17:             $I_w \leftarrow I_w \cup \{A, i\}$
18:         end if
19:     end for
20:     frequencyFilter($A$, $X$, $sumComFreq$)
21: end for
22: return $S$
```

number of candidates for each set. For instance let us consider the following example:

Example 2. *Consider a collection of* 4 *sets ordered based on $\mathcal{O}$, and the jaccard similarity threshold of $t = 0.8$:*

$$\begin{aligned} M_1 &= \{\underline{y_{11}}, y_{12}, y_{13}\} \\ M_2 &= \{\underline{y_{21}, y_{22}}, y_{23}, y_{24}, y_{25}\} \\ M_3 &= \{\underline{y_{31}, y_{32}}, y_{33}, y_{34}, y_{35}\} \\ M_4 &= \{\underline{y_{41}, y_{42}}, y_{43}, y_{44}, y_{45}\} \end{aligned}$$

Prefix length of each set x is calculated as $|x| - \lceil t.|x| \rceil + 1$. Elements in the prefixes are underlined and indexed. We suppose that all the following links are equal to 1: $\{link(y_{11}, y_{42}), link(y_{22}, y_{31}), link(y_{32}, y_{41})\}$. Consider the set M_4, its candidates of similarity are the sets returned by inverted lists *of measures y_{41} and y_{42} (from the prefixes values only). Hence M_4 candidates are M_1 and M_3.*

Next we introduce our filter method dedicated to sensor networks and based on measurements frequencies.

3.3 Frequency filtering

The sensor network is composed of a large number of sensor nodes which are densely deployed and very close to each other. Neighboring sensor nodes almost catch similar data on periodic basis. Our intuition leads us to compare two sets of data based not only on prefix filter approach but combine it with frequency filtering principle. The frequencies assigned to the measurements in the first phase can be used in several ways to further reduce the candidates set size. Let us consider the sets in example 1:

Example 3. *The 2 sets are sorted based on $\mathcal{O}$, and the jaccard similarity threshold of $t = 0.8 \Rightarrow \alpha = 5$:*

$$M_1 = \{\underline{y_{11}, y_{12}}, y_{13}, y_{14}, y_{15}\}$$
$$M_2 = \{\underline{y_{21}, y_{22}}, y_{23}, y_{24}, y_{25}\}$$

If we refer to the equation 2, we observe that these 2 sets do not meet the overlap constraint of $|M_1 \cap_s M_2| \geq 5$, hence M_2 is not in the final result as similar to M_1. However, since they share a common element (y_{12}, y_{21}), in their prefixes, the previous filtering method selects M_2 as candidate of M_1.

However, if we look at the frequencies of the common measurements in the prefixes, we can observe that y_{21} has the highest frequency in M_2 which is not the case of y_{12} in M_1. Furthermore, y_{11} is the measurement that has the highest frequency in M_1 and it does not appear in the prefixes of M_2. Therefore, if we compute the sum of frequencies of the common elements between the two sets and the sum of the frequencies of elements non in common we deduce that $f(y_{11}) \geq f(y_{12})$ then we can safely prune M_2 from the set of candidates of M_1. It means that the number of measurements taken and are in the two sets is less than the number of the rest.

Naturally to optimize the number of candidates of a given set, we use the sum of frequencies of common elements. We are convinced that if the sum of the frequencies of the measurements in common is greater than the sum of the remains, the two sets have a high probability to be similar.

Algorithm 3 frequencyFilter(A, X, $sumComFreq$)

```
Require: $p$-$A$ and $p$-$B$ is the sets of prefixes for $A$ and $B$ respectively
Ensure: Candidates of $A$ /($J(A, B) \geq t$)
 1: $\alpha = \frac{t}{1+t}.(|A| + |B|)$
 2: for each $B$ such that $X[B] > 0$ do
 3:     $Candidate \leftarrow FALSE$
 4:     if $X[B] = min(|A|, |B|)$ then
 5:         $Candidate \leftarrow TRUE$
 6:     else
 7:         if $sumComFreq[B] \geq sumTotFreq[B] - sumComFreq[B]$ then
 8:             $Candidate \leftarrow TRUE$
 9:         end if
10:     end if
11:     if $Candidate$ then
12:         if $(|A \cap_s B| \geq \alpha)$ then
13:             $S \leftarrow S \cup \{A\}$
14:             Delete $B$ in later testing
15:         end if
16:     end if
17: end for
```

We now formally state the frequency filtering principle as:

Consider two sets of sensor measurements M_i and M_j, such that their elements are sorted according to the ordering $\mathcal{O}$. Let the $p\text{-}prefix$ p-M, be the first p elements of the set M. Then, M_j is candidate of M_i if:

- $|p\text{-}M_i \cap_s p\text{-}M_j| = min(|p\text{-}M_i|, |p\text{-}M_j|)$,
  or
- $\sum_{k=1}^{|p\text{-}M_i \cap_s p\text{-}M_j|}(f(y_{jk} \in p\text{-}M_j) \geq \sum_{k=1}^{|p\text{-}M_j - p\text{-}M_i|}(f(y_{jk} \in p\text{-}M_j)$

Our idea is to combine prefix filtering approaches with our frequency filtering principle. Algorithm 2 describes our method to find similar sets of measures. It takes as input a collection of sets sorted according to the ordering $\mathcal{O}$, then it scans sequentially each set, selects the candidates that intersects with its prefix (line 6) and accumulates the overlap and the sum of frequencies in hash maps X and $sumComFreq$ respectively (lines 10 - 11).

Algorithm 3 is designed to apply the frequency filtering principle in order to verify whether the actual overlap between A and candidates B in the current candidate set, $\{B|X[B] > 0\}$, meets the frequency conditions. To be noted that we have already accumulated in $X[B]$ the amount of overlaps that occur in the prefixes of A and B. After finding the candidates of a set A, then we verify if they meet the similarity constraint by applying the Jaccard similarity function. If it is the case we add A to the final set S which the aggregator sends it to the sink, and we eliminate B for later testing.

4 PERFORMANCE EVALUATION

To verify our suggested approach, we conducted multiple series of simulations using a custom *C#* based simulator. The objective was to confirm that our prefix frequency filtering technique (PFF) can successfully achieve intended results for data aggregation in sensor networks with random distribution of sensor nodes. We performed several runs of the algorithm and we noticed experiment results for synthetic and real sensor data measurements. We make a series of simulations on synthetic and real data measurements to analyze the impacts of different features and thresholds defined in PFF. In addition, we also compare our approach with the ToD protocol proposed in [10, 15] using real sensor data.

4.1 Experiments on synthetic data measurements

We performed a variety of experiments using synthetic sensor measurements, shown in the below diagram. As part of this experiment, we considered a network formed of 150 sensor nodes. The nodes assignment to aggregators was not uniform, for instance 25 nodes were assigned to the first aggregator, 50 to the second one and 75 to the third. The aggregators are chosen in function of their distance to the sink, we chose the closest to the sink. In each experimental run, every node takes randomly 24 measurements, corresponding to 24 slots and one period p. The measurements of each set were drawn uniformly at random from real number interval of $size = 7.5$. Our data generation is similar to the one used in [29]. After each slot, a node applies Algorithm 1 and at the end of the period p it forms its set of measurements and sends it to its aggregator. Once the aggregator receives all the sets coming from nodes it applies the frequency filtering technique and then sends the aggregated set of sets to the sink. All the results presented below are the average of 25 runs.

In these simulations we tackled the algorithm performance using the following parameters: the number of sensor nodes assigned per aggregator, the threshold δ, which defines the $link$ function between two measurements, and the threshold of the Jaccard similarity function t; and we evaluated the following metrics:

- The percentage of data sent to the aggregator. After the first aggregation stage, we calculated the percentage of the number of measurements sent to the aggregator regarding the total number of measurements taken by the nodes while varying the threshold δ.
- The percentage of the aggregated sets. After applying the frequency filter by the aggregator, we noted the number of duplicate sets that are deleted and not sent to the sink.
- Data accuracy: represents the measures loss rate. It is a evaluate of measures taken by the source nodes and did not received at the base station (sink). It is defined also as the aggregation error.

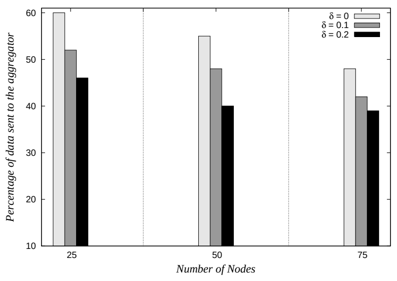


FIGURE 2
Percentage of total data sent to the aggregator

Local aggregation

In these series of simulations, we computed the percentage of the measurements sent by the nodes to the aggregators after local aggregation. We varied the threshold $delta$ between 0 and 0.2[‡] . The results are shown in Figure 2.

We noticed this percentage decreases when δ increases, which means that when we stretch the similarity constraint we eliminate more similar measurements. The goal of this stage is to reduce the number of measurements sent by nodes. The experimental results show that at the minimum and for 25 nodes we reduce the size of the initial set by 40%. Note that in this percentage we have calculated the added values of frequency of each measure.

Aggregation using PFF

In this section we focus on data sets aggregation process which can be performed at an aggregator node aiming to reduce the amount of data traveling towards the sink.

Percentage of Aggregated Sets: Figure 3, Figure 4 and Figure 5 show the percentage of similar sets which are not sent to the sink with varying similarity and $link$ thresholds from 0.8 to 1 and from 0 to 0.2 respectively.

Several observations are eminent: **a)** The percentage of deleted sets (not sent) grows and the algorithm generates more candidate pairs when the similarity threshold decreases. **b)** The percentage of deleted sets increases when the threshold δ decreases. **c)** The percentage of deleted sets increases with the number of nodes assigned to the aggregator. Obviously, when the number of nodes increases it is more likely to find similar measurements and sets, especially neighboring nodes. These results show clearly the effectiveness of our suggested algorithm in finding and eliminating redundant data and data sets. For instance, it can eliminate until 30% out of 50 sets sent to the aggregator,

[‡] These values was chosen according to the interval size (an error between 0 and 5%)

in case of $\delta = 0$ and $t = 0.8$. These results are very interesting and prominent in the field of sensor networks.

Data accuracy: Figure 6, Figure 7 and Figure 8 show the percentage of measurements that, at the end of the aggregation process, did not arrive neither their similar values to the sink. This results represents the aggregation error of PFF. The percentage of information loss is computed at the end of

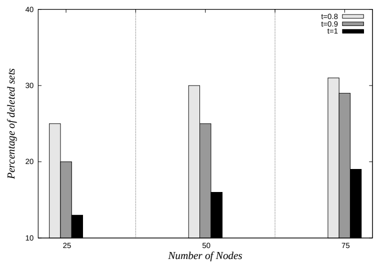


FIGURE 3
Percentage of deleted sets, $\delta = 0$

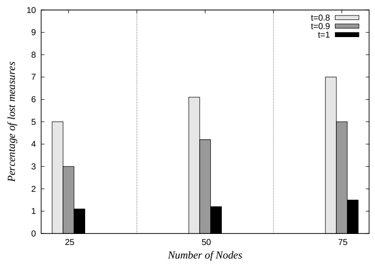


FIGURE 6
Percentage of lost measures, $\delta = 0$

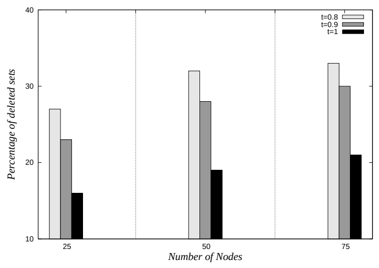


FIGURE 4
Percentage of deleted sets, $\delta = 0.1$

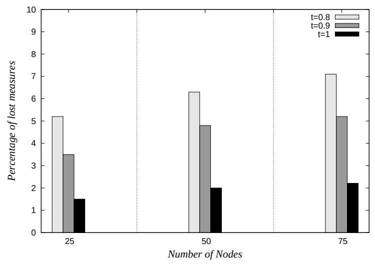


FIGURE 7
Percentage of lost measures, $\delta = 0.1$

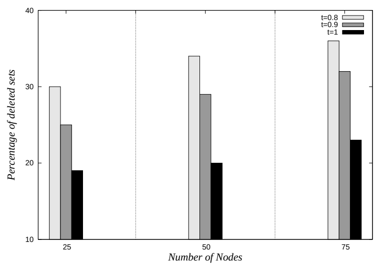


FIGURE 5
Percentage of deleted sets, $\delta = 0.2$

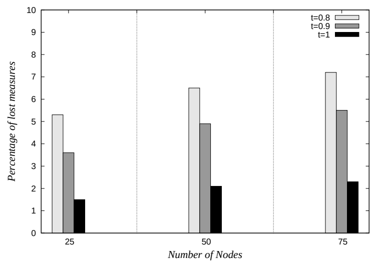


FIGURE 8
Percentage of lost measures, $\delta = 0.2$

every simulation run. All considered schemes are efficient since the information integrity is somehow fully preserved. For instance, for $\delta = 0$ and $t = 1$ usually all the taken measures appearing in the final set arrived to the sink.

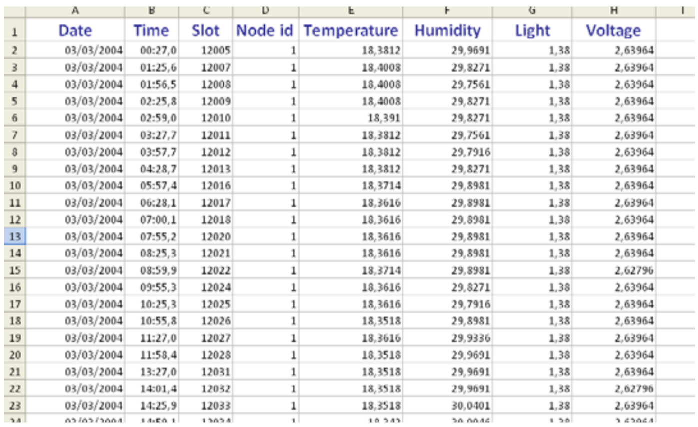
4.2 Experiments on real data measurements

To validate the obtained results in the previous section we performed experiments on real sensor data measurements. We exploited data collected from 46 sensors deployed in the Intel Berkeley Research lab. Mica2Dot sensors with weather boards collected timestamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform. In our experiments, we used a file that includes a log of about 2.3 million readings collected from these sensors. Figure 9 shows a screen capture of this file. Temperature is in degrees Celsius. Humidity is temperature corrected relative humidity, ranging from 0-100%. Light is in Lux. Voltage is expressed in volts [18].

In our experiments, we are interested in two sensors measurements: the temperature and the humidity fields since the variation is more frequent than the light and the voltage. Each node reads an average of 1600 measurements per day and per field. The sensor readings are generated based on intervals. According the varation of measurements, we choose to vary the threshold delta between 0.01 and 0.07. All the results presented below are the average of 7 days readings.

Local aggregation

At the first tier the data is filtered on periodic basis where the period is the interval between each reading. Our experiments generate the results as



	A	B	C	D	E	F	G	H	I
1	Date	Time	Slot	Node id	Temperature	Humidity	Light	Voltage	
2	03/03/2004	00:27,0	12005	1	18,3812	29,9691	1,38	2,63964	
3	03/03/2004	01:25,6	12007	1	18,4008	29,8271	1,38	2,63964	
4	03/03/2004	01:56,5	12008	1	18,4008	29,7561	1,38	2,63964	
5	03/03/2004	02:25,8	12009	1	18,4008	29,8271	1,38	2,63964	
6	03/03/2004	02:59,0	12010	1	18,391	29,8271	1,38	2,63964	
7	03/03/2004	03:27,7	12011	1	18,3812	29,7561	1,38	2,63964	
8	03/03/2004	03:57,7	12012	1	18,3812	29,7916	1,38	2,63964	
9	03/03/2004	04:28,7	12013	1	18,3812	29,8271	1,38	2,63964	
10	03/03/2004	05:57,4	12016	1	18,3714	29,8981	1,38	2,63964	
11	03/03/2004	06:28,1	12017	1	18,3616	29,8981	1,38	2,63964	
12	03/03/2004	07:00,1	12018	1	18,3616	29,8981	1,38	2,63964	
13	03/03/2004	07:55,2	12020	1	18,3616	29,8981	1,38	2,63964	
14	03/03/2004	08:25,3	12021	1	18,3616	29,8981	1,38	2,63964	
15	03/03/2004	08:59,9	12022	1	18,3714	29,8981	1,38	2,62796	
16	03/03/2004	09:55,3	12024	1	18,3616	29,8271	1,38	2,63964	
17	03/03/2004	10:25,3	12025	1	18,3616	29,7916	1,38	2,63964	
18	03/03/2004	10:55,8	12026	1	18,3518	29,8981	1,38	2,63964	
19	03/03/2004	11:27,0	12027	1	18,3616	29,9336	1,38	2,63964	
20	03/03/2004	11:58,4	12028	1	18,3518	29,9691	1,38	2,63964	
21	03/03/2004	13:27,0	12031	1	18,3518	29,9691	1,38	2,63964	
22	03/03/2004	14:01,4	12032	1	18,3518	29,9691	1,38	2,62796	
23	03/03/2004	14:25,9	12033	1	18,3518	30,0401	1,38	2,63964	
24	03/03/2004	14:59,1	12034	1	18,342	30,0046	1,38	2,63964	

FIGURE 9
Extraction from real sensor data

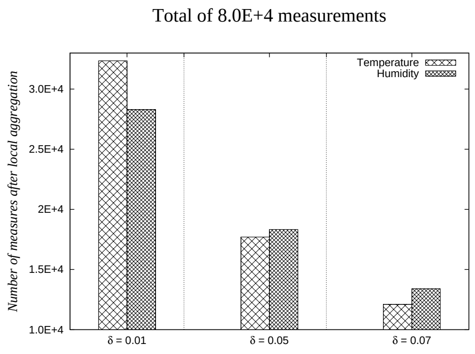


FIGURE 10
Percentage of total data sent to the aggregator

represented in Figure 10. We noted the number of the measurements sent by the nodes to the aggregators after local aggregation. The size of the set contains also the values of the frequencies of each measurements. We considered that the size of each frequency value is equal to the size of a sensor reading. In one day, the 46 sensor nodes collect around $8.0E + 4$ measurements by field. It is well noticed that the size of data improves strongly depending on the threshold δ. It decreases when δ increases. We note that these results meet the results obtained from random measurements presented above.

Aggregation using PFF technique

In this section we show the impact of our frequency filtering technique on data aggregation in sensor networks. As before, we calculated the percentage of aggregated sets as well as the information integrity.

Figures 11 to 13 show the percentage of number of sets not sent to the sink depending on similarity and $link$ thresholds that vary between 0.8 to 0.9 and between 0.01 to 0.07 respectively. For instance, we noticed that the size of the data set sent to the aggregator is reduce by around 30% in case of $\delta = 0.07$ and $t = 0.8$.

The obtained results validated by real and synthetic data measurements, show clearly the effectiveness of our filtering technique in finding and eliminating redundancy and comparing between two sets of data.

We evaluate the impact of data accuracy by calculating the total number of measurements in sets not sent to the sink (the aggregation error). Figures 14 to 16 show the percentage of measurements not received by the sink. These results demonstrate clearly that our approach conserves the information integrity. Therefore, we can consider that our approach decreases the amount of redundant data forwarded to the sink and performs an overall lossless process.

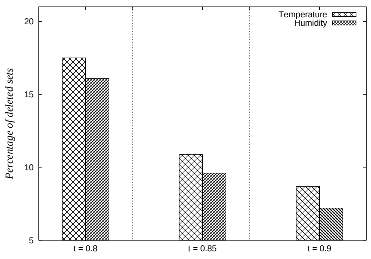


FIGURE 11
Percentage of deleted sets, $\delta = 0.01$

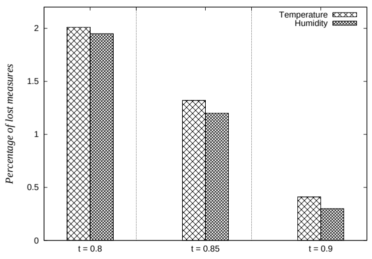


FIGURE 14
Percentage of lost measures, $\delta = 0.01$

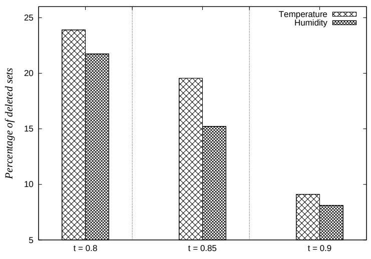


FIGURE 12
Percentage of deleted sets, $\delta = 0.05$

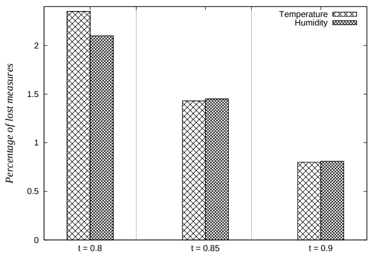


FIGURE 15
Percentage of lost measures, $\delta = 0.05$

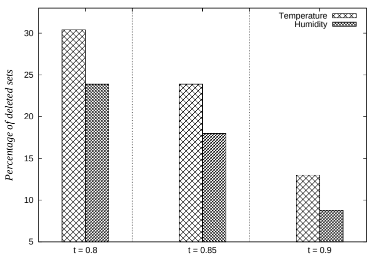


FIGURE 13
Percentage of deleted sets, $\delta = 0.07$

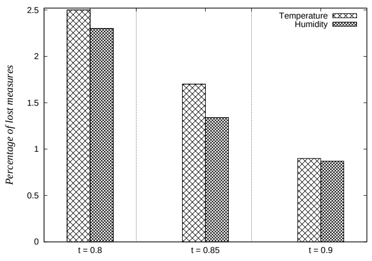


FIGURE 16
Percentage of lost measures, $\delta = 0.07$

In summary, the results obtained from real data sets of sensor reading were qualitatively similar to the one obtained from a random scheme of data measurements.

4.3 PFF vs ToD aggregation protocols

In these expirements we compare our approach (PFF) to the ToD protocol proposed in [10, 15]. In ToD, the network is divided into square cells, where each cell can contain a number of sensor nodes. Then these cells are grouped into clusters called F-clusters (first cluster). All nodes in F-clusters send their data to F-aggregators (cluster heads). This structure is called F-tree. A second clustering layer (s-clusters) is built. It must interleave with F-clusters so it can cover adjacent cells in different F-clusters. In the ToD approach, the authors present data as events while in our simulations these data are sensor readings. Thus, we consider every new reading as an event-like [10].

As our real data sensor network consists of 46 nodes, we use ToD in a one dimensional Network as explained in [10] and we only divide the network into two F-clusters. When a node takes a new measurement it creates a packet containing the new measure, the node's id and the order of the reading. It searches in a waiting queue if this packet can be aggregated with an older one or not. In our simulations, at the end of each period, each source node using ToD protocol sends its packets to the aggregators, while the nodes using our PFF algorithm send sets of measures. In these simulations we evaluated the performance of the protocols using the following parameters: the number of sensor measurements taken by all nodes and the threshold of the Jaccard similarity function t. The threshold δ is fixed to 0.07. The aggregation function used for the ToD protocol is the same used for PFF based on the link function (cf section 3.1). We employed three metrics for the comparison evaluation of PFF and ToD.

- Percentage of received measures: it represents how effective a protocol is in aggregating data. It is the number of measures received by the sink over the number of measures taken by all nodes.
- Data accuracy: as defined before it represents the aggregation error.
- Overall energy dissipation: it is the total energy dissipation of the entire network. To evaluate the energy consumption of our approach we used the same radio model as discussed in [11]. In this model, a radio dissipates $Eelec = 50nJ/bit$ to run the transmitter or receiver circuitry and $\beta_{amp} = 100pJ/bit/m^2$ for the transmitter amplifier. The equations used to calculate transmission costs and receiving costs for a k-bit message and a distance d are: $E_{Tx}(k, d) = E_{elec} \times k + \beta_{amp} \times k \times d^2$ and $E_{Rx}(k) = E_{elec} \times k$, respectively.

Percentage of received measures and data accuracy

Figures 17 and 18 show the percentage of received measures over the total number of measures taken by all nodes for temperature and humidity fields respectively. These experiments permit to show how well the two protocols aggregate and reduce redundant data. PFF performs better than ToD in terms of data aggregation because of its ability to compare sets of data instead of single packets. In other words, PFF reduces the number of redundant data

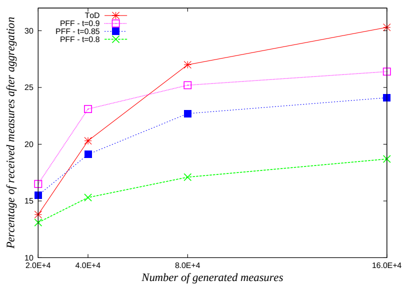


FIGURE 17
Received measures (Temperature)

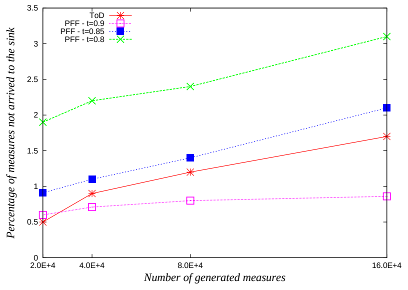


FIGURE 19
Data accuracy (Temperature)

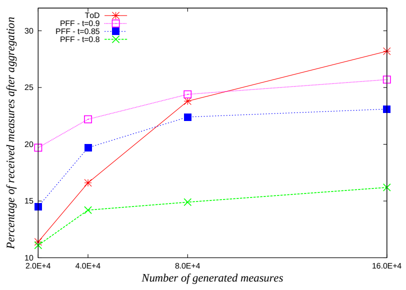


FIGURE 18
Received measures (Humidity)

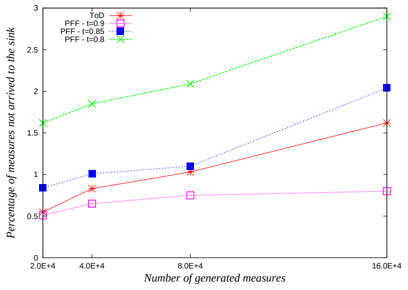


FIGURE 20
Data accuracy (Humidity)

traveling into the networks better than ToD especially when the number of readings increases (the case of periodic networks). We also notice that the percentage of received packets remains almost unchangeable while increasing the sensor readings.

Figures 19 and 20 depict the results of the aggregation error for temperature and humidity fields respectively. This metric is an important performance index, and the high rate of measures loss will greatly impact the use of the data. The obtained results show that both protocols have good performances regarding the aggregation error. As expected, when we increase the threshold t of the similarity function we reduce the measures loss rate. For instance, we can notice that PFF outperforms ToD in terms of data accuracy for $t = 0.9$.

Overall energy dissipation

The overall energy dissipation is the total energy consumption of the entire network. Note that in these simulations we focus on the consumption induced by communication (transmission/reception) and thus the results presented below do not include the amount of energy consumed by data processing, memory access, etc.

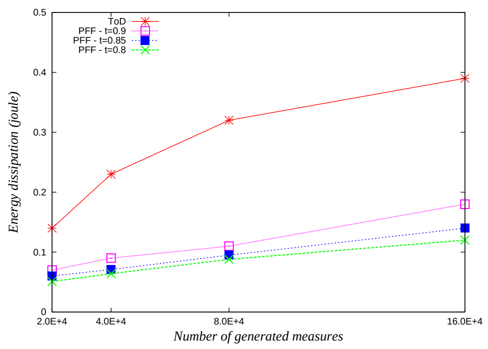


FIGURE 21
Total energy dissipation

Figure 21 shows the results for total energy consumption obtained while varying the total number of sensor readings. The Figure shows that the overall energy dissipation for different protocols increases as the number of readings increases. We notice that ToD does not consume too much, but does not scale well as the number of readings increases. For all the values of the threshold *t* tested, PFF always outperforms the ToD protocol in total energy dissipation. This is because the packet-packet comparison used in ToD instead of data sets in PFF generates more transmissions in the network. Furthermore, the packet construction in ToD contains additional information required for the aggregation which is not the case in PFF. To conclude, PFF uses a factor of 1.5 to 2.5 less overall energy than ToD as shown in Figure 21.

5 CONCLUSION AND PERSPECTIVES

Data aggregation is an important technique to save communication bandwidth and increase network life time for data collection in wireless sensor networks. In this paper we proposed a new method based on sets similarity functions and prefix filtering technique for data aggregation in periodic sensor networks. In periodic networks, nodes are likely to send multiple correlated data to the sink, thus causing the propagation of redundant information throughout the network which in turn leads to both a waste of energy resources and bandwidth, and increase in network congestion. The objective of our aggregation approach is to reduce the number of redundant data sent to the final user while preserving the data integrity. We have developed a new frequency based prefix filtering technique that avoids computing similarity values for all possible pairs of sets. We used the Jaccard similarity function to estimate the similarity between sets of data measures. It was shown through simulations on synthetic and real data measurements that the proposed method reduces drastically the redundant sensor measures while preserving information integrity. Therefore, it saves energy and improves the overall network lifetime.

For future work, we will consider other type of filtering technique. To optimize the number of the generated candidates, we plan to develop a new suffix filter algorithm beside the prefix filtering approach proposed in this paper. Our goal is to use additional filtering method that prunes erroneous candidates that survive after applying the prefix and frequency filtering technique.

Furthermore, we wish to extend our approach to other commonly used similarity measures, like overlap or cosine similarities and especially "Hamming distance".

ACKNOWLEDGMENT

Authors acknowledge the Intel Berkeley Research lab as well as his staff for the hard work they have done to collect real sensor measurements, and for allowing them to use it in the present research study.

REFERENCES

[1] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. (2007). Scaling up all pairs similarity search. *16th international conference on World Wide Web, WWW'07*, pages 131–140.

[2] A. Boulis, S. Ganeriwal, and M. B. Srivastava. (2003). Aggregation in sensor networks: an energy - accuracy tradeoff. *Elsevier Ad-hoc Networks Journal (special issue on sensor network protocols and applications)*, 1(2-3):317–331.

[3] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systemsl*, 29(8-13):1157–1166.

[4] Supriyo Chatterjea, Tim Nieberg, Nirvana Meratnia, and Paul Havinga. (2008). A distributed and self-organizing scheduling algorithm for energy-efficient data aggregation in wireless sensor networks. *Transactions on Sensor Networks (TOSN)*, 4(4):1550–4859.

[5] Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. (2006). A primitive operator for similarity joins in data cleaning. *22nd International Conference on Data Engineering (ICDE'06)*, page 5.

[6] Huifang Chen, Hiroshi Mineno, and Tadanori Mizuno. (2009). Adaptive data aggregation scheme in clustered wireless sensor networks. *Computer Communications*, 31(15):3579–3585.

[7] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. (2005). Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. *2005 ACM SIGMOD International Conference on Management of Data*, pages 25–36.

[8] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. (2005). Prolonging the lifetime of wireless sensor networks via unequal clustering. *Proceedings of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks*.

[9] Mehdi Esnaashari and M. R. Meybodi. (2010). Data aggregation in sensor networks using learning automata. *Wireless Networks*, 16(3):687–699.

[10] Kai-Wei Fan, Sha Liu, and Prasun Sinha. (2008). Dynamic forwarding over tree-on-dag for scalable data aggregation in sensor networks. *IEEE Transactions on Mobile Computing*, 7(10):1271–1284.

[11] Wendi Rabiner Heinzelman, Anantha Ch, and Hari Balakrishnan. (2000). Energy-efficient communication protocol for wireless microsensor networks.

[12] Monika Henzinger. (2006). Finding near-duplicate web pages: a large-scale evaluation of algorithms. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291.

[13] Timothy C. Hoad and Justin Zobel. (2003). Methods for identifying versioned and plagiarized documents. *Journal of the American Society for Information Science and Technology*, 54(3):203–215.

[14] B Krishanamachari, D Estrin, and S Wicker. (2002). The impact of data aggregation in wireless sensor networks. *22nd International Conference on Distributed Computing Systems Workshops*, pages 575 –578.

[15] Prakash G L, Thejaswini M, S H Manjula, K R Venugopal, and L M Patnaik. (2009). Tree-on-dag for data aggregation in sensor networks. *World Academy of Science, Engineering and Technology*, 37.

[16] SangHak Lee and TaeChoong Chung. (2005). Data aggregation for wireless sensor networks using self-organizing map. *Artificial Intelligence and Simulation Lecture Notes in Computer Science*, pages 508–517.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. (2002). Tag: A tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev*, 36(SI):131–146.

[18] Samuel Madden. http://db.csail.mit.edu/labdata/labdata.html.

[19] Sunita Sarawag and Alok Kirpal. (2004). Efficient set joins on similarity predicates. *2004 ACM SIGMOD international conference on Management of data*, pages 743–754.

[20] Sunita Sarawag and Alok Kirpal. (2006). Efficient exact set-similarity joins. *32nd international conference on Very large data bases, VLDB'06*, pages 918–929.

[21] R.C. Shah and J.M Rabaey. (2002). Energy aware routing for low energy ad hoc sensor networks. *2002 IEEE Wireless Communications and Networking Conference. WCNC2002.*, pages 350–355.

[22] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. (2003). Tina: A scheme for temporal coherency-aware in-network aggregation. *3rd ACM international workshop on Data engineering for wireless and mobile access*, pages 69–76.

[23] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. (2004). Medians and beyond: New aggregation techniques for sensor networks. *2nd international conference on Embedded networked sensor systems*, pages 239–249.

[24] H. O. Tan and I. Korpeoglu. (2003). Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Record*, 32(4):66–71.

[25] Leandro Villa, Azzedine Boukerche, Regina Borges de Araujo, and Antonio Loureiro. (2010). Highly dynamic routing protocol for data aggregation in sensor networks. *2010 IEEE Symposium on Computers and Communications (ISCC)*, pages 496–502.

[26] Yingqi Xu, Wang-Chien Lee, Jianliang Xu, and G. Mitchell. (2006). Processing window queries in wireless sensor networks. *22nd International Conference on Data Engineering. ICDE '06*, page 70.

[27] O. Younis and S. Fahmy. (2005). An experimental study of routing and data aggregation in sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, page 8 pages.

[28] Bo Yu, Jianzhong Li, and Yingshu Li. (2009). Distributed data aggregation scheduling in wireless sensor networks. *IEEE, INFOCOM2009*.

[29] Yan Yu. (2005). Scalable, synthetic, sensor network data generation. *Doctoral Dissertation, University of California, Los Angeles*.

[30] Yanfei Zheng, Kefei Chen, and Weidong Qiu. (2010). Building representative-based data aggregation tree in wireless sensor networks. *Mathematical Problems in Engineering*, 2010:11 pages.