Frequency Filtering Approach for Data Aggregation in Periodic Sensor Networks

Jacques M. Bahi, Abdallah Makhoul and Maguy Medlej
Computer Science Laboratory (LIFC)
University of Franche-Comté
Rue Engel-Gros, 90016 Belfort, France
{jacques.bahi, abdallah.makhoul, maguy.medlej}@univ-fcomte.fr

***Abstract*—This paper presents an energy-efficient technique for data aggregation in periodic sensor networks. We investigate the problem of finding all pairs of nodes generating similar data sets such that similarity between each pair of sets is above a threshold t. We provide a frequency filtering approach to solve this problem. Our experiments demonstrate that our algorithm outperforms existing prefix filtering methods in reducing energy consumption.**

***Index Terms*—periodic sensor networks; data aggregation; similarity functions; frequency filtering; optimization;**

I. Introduction

In this paper we present a hierarchical multilevel data aggregation scheme for periodic sensor networks. It aims to optimize the volume of data transmitted thus saving energy consumption and reducing bandwidth on the network level. A first level in-sensor process is done by the nodes themselves. Instead of sending each sensor node's raw data to a base station, the data is cleaned periodically by the sensor node itself before sending it to an aggregator node for a second level of aggregation. At this level, we are interested in exploring a new part of the filtering aggregation problem, by focusing on identifying the similarity between data sets generated by neighboring nodes and sent to the same aggregator. Our objective is to identify similarities between near sensor nodes, and integrate their captured data into one record while preserving information integrity.

Most of the extant work [1] has 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. Other works, are query based methods [2]. A query is generated at the sink and then broadcasted through the network. Some nodes are selected to 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 [3]. Some works, such as [4], use the clustering methods for aggregating data packets in each cluster separately. In [5], a learning automaton-based data aggregation method in sensor networks is provided. In our previous work [6], we have shown that existing prefix filtering methods are very complex and not suitable for sensor networks and we proposed a heuristic based on the frequency ordering. In this paper we provide a new prefix filtering method to study the sets similarity in sensor networks. We propose frequency filtering optimization 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 defined at the first aggregation level. We show through the result that our approach offers significant data reduction by eliminating in-network redundancy and sending only necessary information to the sink.

978-1-4673-0269-2/12/$31.00

The rest of the paper is organized as follows, Section II describes our periodic data aggregation scheme, the local aggregation level and our proposed frequency filtering techniques. Experimental results are given in Section III. Section IV concludes the paper with some directions to a future work.

II. Periodic Data Aggregation

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.

A. Local aggregation

In periodic sensor networks, we consider that each sensor node i at each slot s takes a new measurement y_{is}. Then node i forms a new set of captured measurements M_i with period p, and sends it to the aggregator. 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:

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

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 $f(y_{is})$.

For each new sensed measurement (at each slot), a sensor node i searches for the similar measure already captured. If a similar measurement is found, it deletes the new one while incrementing the corresponding frequency by 1. For more details of this algorithm see [6].

At the end of the period p, each node i will possess 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.

B. Frequency filtering aggregation

At this level of aggregation, each aggregator has received k sets of measurements and their frequencies. The idea here is to identify 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 set that will be sent to the sink. Researchers can find various similarity functions [7]. The similarity function selection primarily depends on the application domain. In this paper we will focus on the Jaccard similarity, a commonly used function to define similarity between two sets M_i and M_j as:

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

where "$\cap_s$" is a new function for overlapping defined as:

Definition 3: Consider two sets of measurements M_1 and M_2, then we define:

$M_1 \cap_s M_2 = \{(y_1, y_2) \in M1 \times M2 / link(y_1, y_2) = 1\}$;

1) Sets similarity computation: In this section we provide techniques for computing the similarity between the received sets. A naïve method to compute the similarity of 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 $O(n^2)$.

To reduce the number of comparisons between sets a prefix filtering method has been proposed. Several approaches for traditional similarity join between sets are based on the prefix filtering principle [8] [6]. This method is based on the intuition that if all sets of measures are sorted by a global ordering, some fragments of them must share several common tokens with each other in order to meet the threshold similarity. 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).

This intuition is formalized by the following $Lemma$ inspired from [9]:

Lemma 1: Consider two sets of sensor measures M_i and M_j, such that their elements are ordered by a global defined ordering. 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 [9]. ■

To ensure the prefix filtering based approach does not miss any similarity set result, as shown in Lemma 1 we need a prefix of length $|M_i| - \lceil t.|M_i| \rceil + 1$ for every set M_i [6]. The prefix filtering algorithm for finding similarity sets takes as input a collection of datasets coming from different sensor nodes already sorted according to a defined ordering. It scans sequentially each set M_i, selects the candidates that intersects with its prefix. Afterwards, M_i and all its candidates will be verified against the jaccard similarity threshold to finally return the set of correct similar measurements sets. For more details of the prefix filtering algorithm refere to [6].

Prefix filtering algorithm helps prune out unfeasible sets of measures, however, in practice the number of non-similar sets surviving after this technique is still quadratic growth [10]. Following the prefix filtering, many optimization methods [10], [11] were proposed to prune out further the unfeasible non-similar sets. A trade-off of these prefix filtering optimizations is that usually require more computational efforts which is unsuitable by heavy resources sensor networks. In our approach, we provide some optimizations for prefix filtering techniques based on measures frequency while taking into account this trade-off.

2) Frequency filtering approach: Let us introduce some definitions and notations which will be the basis of what follows. In periodic sensor networks, two data sets are similar if their measurements overlap with each other, and especially the ones having *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.

For two similar measures m_i and m_j such that $link(m_i, m_j) = 1$, we denote $f_{min}(m_i, m_j) = Min(f(m_i), f(m_j))$ the minimum value of the frequency of these measures.

Definition 5 ($f_s(M_i, M_j)$): Consider two sets of measures M_i and M_j, we define

$f_s(M_i, M_j) = \sum_{k=1}^{O_s(M_i, M_j)} (f_{min}((m_i, m_j) \in M_i \cap_s M_j))$.

In this paper, we consider that all sensor nodes operate with the same sampling rate, and every node captures τ measures with each period p. Thus we can deduce that for every received set M_i from node i we have: $\sum_{k=1}^{|M_i|} (f(m_k \in M_i) = \tau$.

Using the Jaccard similarity function, two sets M_i and M_j are similar if and only if: $O_s(M_i, M_j) \geq \alpha$ where $\alpha = \frac{t}{1+t}.(|M_i|+|M_j|)$ (Equation (1)). Supposing that the sets were sent to the aggregators without applying the first aggregation phase and without computing measures frequencies, thus we can observe that:

$$|M_i| = |M_j| = \tau \text{ and } f_s(M_i, M_j) = O_s(M_i, M_j). \quad (2)$$

Hence, from Equation (1) and Equation (2) we can deduce that:

$$M_i \text{ and } M_j \text{ are similar iff: } f_s(M_i, M_j) \geq \frac{2 \times t \times \tau}{1+t}. \quad (3)$$

Lemma 1 states that the prefixes of two sets of measures must share at least one measure in order to satisfy the prefix filtering condition (PFC). Nevertheless, in sensor networks this condition is easily satisfied. In this section, we will present an extension of the prefix filtering technique making the PFC condition more difficult to be satisfied.

Lemma 2: Assume that all the measures in the sets M_i and M_j are ordered according to the global ordering $\mathcal{O}$. Let the $p\text{-}prefix$ be the first p elements of M_i. If $f_s(M_i, M_j) \geq \frac{2 \times t \times \tau}{1+t}$, then $f_s(p\text{-}M_i, p\text{-}M_j) \geq \sum_{k=1}^{|p\text{-}M_i|}(f(m_k \in p\text{-}M_i)) - \frac{1-t}{1+t} \times \tau$.

Proof: We denote by $p\text{-}M_i$ the prefix of the set M_i and $r\text{-}M_i$ the set of reminder measures where $M_i = \{p\text{-}M_i + r\text{-}M_i\}$. We have:

$$\begin{aligned} f_s(M_i, M_j) &= f_s(p\text{-}M_i, M_j) + f_s(r\text{-}M_i, M_j) \\ &= f_s(p\text{-}M_i, p\text{-}M_j) + f_s(p\text{-}M_i, r\text{-}M_j) + \\ &\quad f_s(r\text{-}M_i, M_j) \\ &\cong f_s(p\text{-}M_i, p\text{-}M_j) + f_s(r\text{-}M_i, M_j) \\ &\leq f_s(p\text{-}M_i, p\text{-}M_j) + \sum_{k=1}^{|r\text{-}M_i|}(f(m_k \in r\text{-}M_i)) \end{aligned}$$

In the second line we can omit the term $f_s(p\text{-}M_i, r\text{-}M_j)$ because we have assumed that it is negligible compared to the other terms in the equation. Indeed, if the two sets are similar then the measures having highest frequencies must be in the prefix set and not in the reminder, which means that the overlapping between the $p\text{-}M_i$ and $r\text{-}M_j$ is almost empty. From the above equations and equation (3)(similarity condition) we can deduce:

$$\frac{2 \times t \times \tau}{1+t} \leq f_s(p\text{-}M_i, p\text{-}M_j) + \sum_{k=1}^{|r\text{-}M_i|}(f(m_k \in r\text{-}M_i)) \quad (4)$$

From the following equation:

$$\sum_{k=1}^{|p\text{-}M_i|}(f(m_k \in p\text{-}M_i)) + \sum_{k=1}^{|r\text{-}M_i|}(f(m_k \in r\text{-}M_i)) = \tau \quad (5)$$

We obtain:

$$f_s(p\text{-}M_i, p\text{-}M_j) \geq \sum_{k=1}^{|p\text{-}M_i|}(f(m_k \in p\text{-}M_i)) - \frac{1-t}{1+t} \times \tau \quad (6)$$

The lemma is proved. ■

Algorithm 1 describes our method to find similar sets of measures based on the frequency filtering approach. It is a hybrid solution, where we integrate our frequency condition presented in Lemma 2 to the prefix filtering approach.

Algorithm 1 Frequency-filtering based algorithm.

Require: Set of measures' sets $M = \{M_1, M_2 ... M_n\}$, t, τ.
Ensure: All pairs of sets (M_i, M_j), such that $J(M_i, M_j) \geq t$.

```
S ← ∅
I_i ← ∅ (1 ≤ i ≤ total number of measures)
for each set M_i ∈ M do
    p ← |M_i| − ⌈t × |M_i|⌉ + 1
    Fs ← empty map from set id to int
    sumFreq ← 0
    for k ← 1 to p do
        sumFreq ← sumFreq + f(m_k ∈ p-M_i)
    end for
    for k ← 1 to p do
        w ← M_i[k]
        if (I_{w_s} exists such that link(w, w_s) = 1) then
            for each Measurement (M_j[l]), f(M_j[l]) ∈ I_{w_s} do
                Fs[M_j] ← Fs[M_j] + f_min(M_i[k], M_j[l])
            end for
            I_{w_s} ← I_{w_s} ∪ {M_i}
        else
            create I_w
            I_w ← I_w ∪ {M_i}
        end if
    end for
    for each M_j such that Fs[M_j] > sumFreq − ((1−t)/(1+t)) × τ do
        if O_s(M_i, M_j) ≥ α then
            (S ← {(M_i, M_j)})
        end if
    end for
end for
return S
```

III. Experimental Results

To evaluate our approach, we conducted multiple series of simulations using the discrete event simulator OMNET++ [12]. The objective of these simulations is to confirm that our frequency filtering technique can successfully achieve desirable results for data aggregation in periodic sensor networks. Therefore, in our simulations we used real readings collected from 45 sensor nodes deployed in the Intel Berkeley Research Lab [13]. Every 31 seconds, sensors with weather boards were collecting humidity, temperature, light and voltage values. Each node senses an average of 90000 values of each measurement per day and per field. For the sake of simplicity, in this paper we are interested in one field of sensor measurements: the temperature [1]. We performed several runs of the algorithms (an average of 15 runs).

In the first series of simulations, each node runs local aggregation with every new reading. Then, it generates an aggregated set composed of non-similar measurements assigned by their frequencies. We varied the threshold δ and the number

[1] the others are done by the same manner

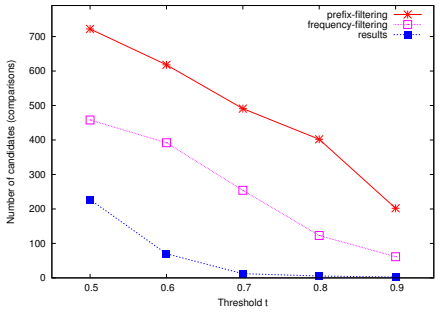

Fig. 1. $\tau = 2.E + 03$

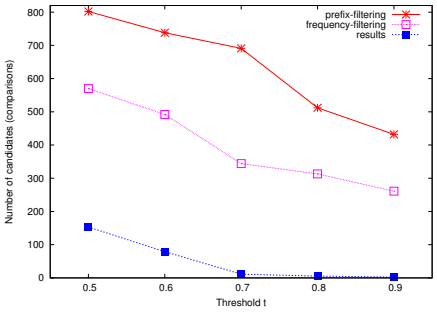

Fig. 2. $\tau = 4.E + 03$

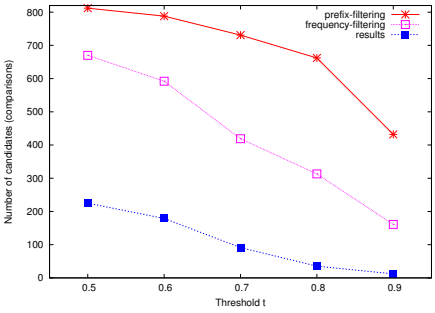

Fig. 3. $\tau = 6.E + 03$

of total readings τ and we computed the cardinality of each generated set. The results are shown in Figure 4.

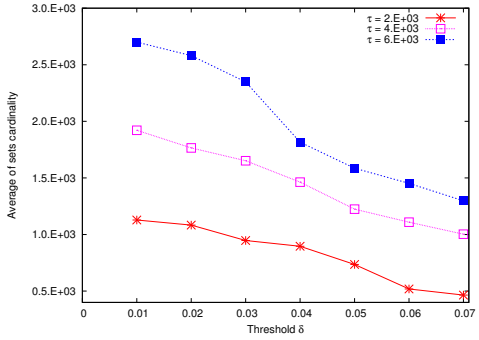

Fig. 4. Sets cardinality after first aggregation

We noticed the cardinality 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 least w reduced the sent set by 30%.

We compared the number of candidates (number of comparisons) generated respectively by our frequency filtering technique, the prefix filtering algorithm and the results obtained after applying the Jaccard similarity function. In these series of simulations we varied the parameter τ (the size of datasets) while fixing the threshold δ to 0.05. The obtained curves are shown in figures 1, 2 and 3. Observing the obtained result, we divide them into two groups according to τ. First, consider the results on small dataset where $\tau = 2.E + 03$ as shown in figure 1. We notice that, when the similarity threshold increases from 0.7 to 0.9, the number of comparisons of the frequency filtering and the prefix filtering becomes closer. In the next step, consider another group of datasets with big size ($\tau = 4.E + 03$ and $\tau = 6.E + 03$). The obtained results are shown in 2 and 3. We observe that when the sets cardinality increases the number of candidates also increases.

IV. Conclusion and Future Work

In this paper we proposed a new frequency filtering approach and several optimizations using sets similarity functions to find similar data sets generated by sensor nodes. The objective of our approach is to reduce the number of redundant data sent to the end user while preserving the data integrity. It was shown through simulations on real data measurements that our method reduces drastically the redundant sensor measures and outperforms the existing prefix filtering approaches. Thus, it saves energy and improves the overall network lifetime. A direction for future work is to develop a new suffix frequency filter algorithm beside the frequency 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.

References

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

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

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

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

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

[6] Jacques Bahi, Abdallah Makhoul, and Maguy Medlej. Data aggregation for periodic sensor networks using sets similarity functions. *IWCMC 2011, 7th IEEE Int. Wireless Communications and Mobile Computing Conference*, pages 559–564, 2011.

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

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

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

[10] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient similarity joins for near duplicate detection. *Proceeding of the 17th international conference on World Wide Web*, pages 131–140, ACM 2008.

[11] Chuan Xiao, Wei Wang, Xuemin Lin, and Haichuan Shang. Top-k set similarity joins. *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 916–927, 2009.

[12] OMNeT++. http://www.omnetpp.org/.

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