Energy Efficient Filtering Techniques for Data Aggregation in Sensor Networks

Hassan Harb[a,b], Abdallah Makhoul[a], Samar Tawbi[c]
, and Oussama Zahwe[b]
[a]*Univ. Bourgogne Franche-Comté, FEMTO-ST Institute/CNRS, Belfort, France*
[b]*Department of Computer Science, American University of Culture and Education (AUCE), Beirut, Lebanon*
[c]*Department of Computer Science, Lebanese University, Beirut, Lebanon*
Emails: hassan.moustafa_harb@univ-fcomte.fr, abdallah.makhoul@univ-fcomte.fr

***Abstract*—Minimizing latency is a major issue for data aggregation in wireless sensor networks (WSNs). Hence, the proposed algorithms must achieve the minimum delay in data delivery while decreasing the energy consumption. In this paper, we propose a new version of the prefix frequency filtering technique (PFF) proposed by [1], which aims to minimize aggregation latency. PFF finds similar sets of data generated by nodes in order to reduce redundancy in data over the network, thus, nodes consume less energy. While in the enhanced version of the PFF technique, called PPSFF, we propose a positional filtering that exploits the order of readings both in the prefix and the suffix of a set and leads to upper bound estimations of similarity scores. Experiments on real sensor data show that our enhancement can significantly improve the latency of the PFF technique without affecting its performance.**

***Keywords*—Wireless sensor networks, periodic applications, data aggregation, similarity functions, prefix-suffix filtering.**

I. Introduction

Nowadays, wireless sensor networks (WSNs) found their way into a great number of application areas. In such networks, each sensor node monitors a given phenomenon and sends notifications and readings back to the sink at each period p [2]. Subsequently, in-network data aggregation is essential for periodic applications where data generated by sensor nodes is usually redundant and highly correlated, particularly in case of dense networks. The main objective of in-network aggregation is to eliminate redundancy and minimizes the number of transmissions to the sink, thus saving energy.

Subsequently, data latency is an important factor in WSNs that depends on applications. Some applications, such as permafrost monitoring [3], require periodic and fast data delivery over long periods. Therefore, in-network data aggregation must be completed within a minimum delay in order to deliver aggregated data to the sink as fast as possible [4], [5], [6]. In result, latency and energy efficiency are key elements to evaluate the performance of an in-network data aggregation technique. In WSNs, in-network data aggregation has been largely studied by research community [7], [8], [9], [10], [11], [12], [13]. In the most of the proposed techniques, networks are based on two-tier tree-based or single-hop clustering networks where data aggregation are performed at the sensor nodes and an intermediates nodes called, "aggregators" or cluster-heads (CHs) (see Fig. 1). The authors in [11] propose a two-level sensor fusion-based event detection technique for the WSN. In the first level, each sensor node is responsible for deciding whether an event has been occurred, using a feed forward neural network (FFNN) or Naive Bayes classifier. In the second level, at CH or aggregator, a fusion algorithm is proposed to reach a consensus among individual detection decisions made by sensor nodes. In [12], the authors propose a two-stage data aggregation model dedicated to underwater WSNs. After processing data at each sensor, the authors use Euclidean and Cosine distances at the aggregator level to reduce packet size and minimize data redundancy.

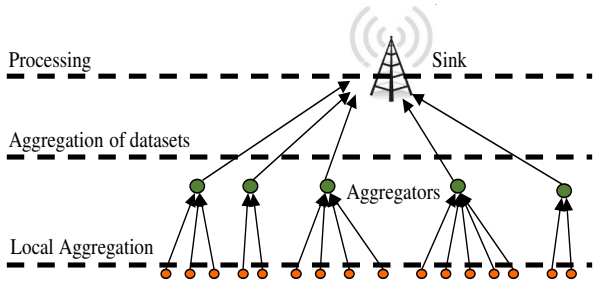


Fig. 1. Data aggregation scheme.

Recently, the authors in [1] study a new area within filtering aggregation problem, the Prefix-Frequency Filtering (PFF) technique. Further to a local processing at sensor node level, PFF uses Jaccard similarity function at aggregator's level to identify similarities between near sensor nodes and integrate their sensed data into one record. However, PFF stays a hard technique for the aggregator because it needs costly computations to search the similarity between all received sets and thus affecting data latency factor. In this paper, our objective is to enhance PFF by adding additional filters, based on prefix data indexing and suffix filtering, in order to reduce the number of comparisons between data sets and thus improving the data latency.

The rest of the paper is organized as follows. Section II recalls the PFF technique proposed in [1]. Section III presents our enhancing version of PFF technique with additional data filtering. Experimental results based on real data readings are exposed in Section IV. Section V concludes this paper and provides directions for future work.

II. PFF Technique Recalls

This section recalls the data aggregation scheme proposed in [1] and that will be enhanced in this paper. The proposed method works at two levels: sensor nodes and aggregators.

A. First Level: Local Aggregation

In this tier of aggregation, the idea is to identify similar readings captured by a sensor during a period p. In periodic applications, each sensor S_i captures a vector of τ readings, R_i=$\left[r_{i_1}, r_{i_2}, \ldots, r_{i_\tau}\right]$, during each period then sends it to the aggregator. In order to reduce the size of R_i, a similarity function, called $link$, between readings and a frequency of a reading are defined as follows:

Definition 1: ($link$ function): We define the $link$ function between two readings as:

$$link(r_{i_j}, r_{i_k}) = \begin{cases} 1 & \text{if } |r_{i_j} - r_{i_k}| \leq \delta, \\ 0 & \text{otherwise} \end{cases}$$

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

Definition 2: (Reading's frequency): The frequency of a reading r_i is defined as the number of the subsequent occurrence of the same or similar (according to the $link$ function) readings in the same vector. It is represented by $f(r_i)$.

For each new captured reading, S_i searches for similarities of the new taken reading. If a similar reading is found, it deletes the new one and increments the corresponding frequency by 1, else it adds the new reading to the set and initialize its frequency to 1 (See Algorithm 1 proposed in [1] for more details). Therefore, S_i will transform the initial vector of readings, R_i, to a set of readings, R'_i, associated to their corresponding frequencies as follows: R'_i=$\{(r'_{i_1}, f(r'_i)), (r'_{i_2}, f(r'_{i_2})), \ldots, (r'_{i_k}, f(r'_{i_k}))\}$ before sending it to the aggregator.

B. Second Level: PFF Aggregation

At the end of each period, the aggregator will receive n sets of readings with their frequencies coming from different nodes. The objective in this aggregation step is to identify all pairs of sets whose similarities are above a given threshold t. For this reason, the Jaccard similarity function is used which returns a value in $[0, 1]$. Thus, if two reading sets have similarity more than t then, the aggregator deletes one of them before sending them to the sink. The Jaccard similarity function between two sets R'_i and R'_j, generated respectively by S_i and S_j, can be formulated as follows [1]:

$$J(R'_i, R'_j) \geq t \Leftrightarrow |R'_i \cap_s R'_j| \geq \alpha = \frac{2 \times t \times \tau}{1+t} \qquad (1)$$

In order to prevent to enumerate and compare every pair of sets which has a $O(n^2)$ number of comparisons, the authors in [1] propose to use a prefix frequency filtering approach (PFF). After ordering readings by decreasing order of their frequencies, e.g. order $\mathcal{O}$, PFF defines a prefix of length $|R'_i| - \lceil t.|R'_i|\rceil + 1$ for every set R'_i. Then, it scans sequentially each set R'_i, selects the candidates (may be similar) that intersects with its prefix. Afterwards, R'_i and all its candidates will be verified against the Jaccard similarity threshold to finally return the set of correct similar readings sets. In practice, the number of non-similar sets surviving after PFF is still quadratic growth. Therefore, in this paper, we are looking to add more additional filters in order to prune out further the unfeasible non-similar sets thus, enhancing the data latency of PFF.

III. Position-based Prefix-Suffix Frequency Filtering technique (PPSFF)

In this section, we present our enhanced version of PFF called Position-based Prefix-Suffix Frequency Filtering technique (PPSFF). PPSFF can reduce the number of candidates (comparisons) in PFF, thus improving the data latency, by implementing two additional filters: positional and suffix.

A. Positional Filtering

In this section, we propose to insert a new filter, called positional filtering, which it can be applied over the prefix of each set, e.g. $|R'_i| - \lceil t.|R'_i|\rceil + 1$. The positional filtering has an objective to estimate an upper bound of overlap between prefixes during the searching for similar readings. The intuition of positional filtering is formalized by the following lemma inspired from [14]:

Lemma 1: Consider two sets of sensor readings R'_i and R'_j, such that their prefixes are ordered by increasing order of their values. let $p\text{-}prefix$ be the first p elements of R'_i. let reading $(w_i, f(w_i)) = (p\text{-}R'_i[i], f(p\text{-}R'_i[i]))$, w_i partitions $p\text{-}R'_i$ into the left partition $p\text{-}R'_{il}(w_i) = p\text{-}R'_i[1 \ldots (i-1)]$ and the right partition $p\text{-}R'_{ir}(w_i) = p\text{-}R'_i[(i+1) \ldots p]$. Based on [1], two sets are candidates if $f_s(p\text{-}R'_i, p\text{-}R'_j) \geq (\sum_{k=1}^{|p\text{-}R'_i|} f(r'_k)) - \frac{1-t}{1+t} \times \tau$ where $r'_k \in p\text{-}R'_i$; thus for every $(w_i, w_j) \in (p\text{-}R'_i \cap_s p\text{-}R'_j)$ with frequency $f_{min}(w_i, w_j)$ where $w_j \in p\text{-}R'_j$ and $link(w_i, w_j) = 1$: $f_s(p\text{-}R'_{il}(w_i), p\text{-}R'_{jl}(w_j)) + f_{min}(w_i, w_j) + min\big(\sum_{k_1=1}^{|p-R'_{ir}|} f(r'_{k_1}), \sum_{k_2=1}^{|p-R'_{jr}|} f(r'_{k_2})\big) \geq \big(\sum_{k_3=1}^{|p\text{-}R'_i|} f(r'_{k_3})\big) - \frac{1-t}{1+t} \times \tau$ where $r'_{k_1} \in p\text{-}R'_{ir}$, $r'_{k_2} \in p\text{-}R'_{jr}$, $r'_{k_3} \in p\text{-}R'_i$, $p\text{-}R'_{jl}(w_j) = p\text{-}R'_j - \{(w_j, f(w_j) - f_{min}(w_i, w_j))\} - p\text{-}R'_{jr}$ if $w_j \leq w_i$ or $p\text{-}R'_{jr}(w_j) = p\text{-}R'_j - \{(w_j, f(w_j) - f_{min}(w_i, w_j))\} - p\text{-}R'_{jl}$ if $w_j > w_i$.

Proof: We denote by $p\text{-}R'_i$ and $p\text{-}R'_j$ the prefixes of the sets R'_i and R'_j respectively. We also consider $(w_i, w_j) \in (p\text{-}R'_i \cap_s p\text{-}R'_j)$, where $w_i \in p\text{-}R'_i$ and $w_j \in p\text{-}R'_j$. Then, we have: R'_i and R'_j are similar:

$$\Rightarrow f_s(p\text{-}R'_i, p\text{-}R'_j) \geq \sum_{k=1}^{|p\text{-}R'_i|} f(r'_k) - \frac{1-t}{1+t} \times \tau = A$$

$$\Rightarrow f_s(p\text{-}R'_{il}, p\text{-}R'_j) + f_s(p\text{-}R'_{ir}, p'\text{-}R'_j) \geq A, \quad \text{where } p'\text{-}R'_j = p\text{-}R'_j - f_s(p\text{-}R'_{il}, p\text{-}R'_j)$$

$$\Rightarrow f_s(p\text{-}R'_{il}, p\text{-}R'_{jl}) + f_{min}(w_i, w_j) + f_s(p\text{-}R'_{il}, p\text{-}R'_{jr}) + f_s(p\text{-}R'_{ir}, p'\text{-}R'_j) \geq A$$

we have: $f_s(p\text{-}R'_{il}, p\text{-}R'_{jr}) = 0$, because $p\text{-}R'_i$ and $p\text{-}R'_j$ are ordered by increasing order of their values

$$\Rightarrow f_s(p\text{-}R'_{il}, p\text{-}R'_{jl}) + f_{min}(w_i, w_j) + f_s(p\text{-}R'_{ir}, p'\text{-}R'_j) \geq A$$

$$\Rightarrow f_s(p\text{-}R'_{il}, p\text{-}R'_{jl}) + f_{min}(w_i, w_j) + f_s(p\text{-}R'_{ir}, p'\text{-}R'_{jl}) + f_s(p\text{-}R'_{ir}, p\text{-}R'_{jr}) \geq A, \quad \text{where } p'\text{-}R'_{jl} = p\text{-}R'_{jl} - f_s(p\text{-}R'_{il}, p\text{-}R'_{jl})$$

we have: $f_s(p\text{-}R'_{ir}, p'\text{-}R'_{jl}) = 0$, for the same reason above

$\Rightarrow f_s(p\text{-}R'_{il}, p\text{-}R'_{jl}) + f_{min}(w_i, w_j) + f_s(p\text{-}R'_{ir}, p\text{-}R'_{jr}) \geq A$

since:

$$f_s(p\text{-}R'_{ir}, p\text{-}R'_{jr}) \leq min\Big(\sum_{k_1=1}^{|p-R'_{ir}|} f(r'_{k_1}), \sum_{k_2=1}^{|p-R'_{jr}|} f(r'_{k_2})\Big),$$

where $r'_{k_1} \in p - R'_{ir}$ and $r'_{k_2} \in p - R'_{jr}$

$$\Rightarrow f_s(p\text{-}R'_{il}, p\text{-}R'_{jl}) + f_{min}(w_i, w_j) + min\Big(\sum_{k_1=1}^{|p-R'_{ir}|} f(r'_{k_1}), \sum_{k_2=1}^{|p-R'_{jr}|} f(r'_{k_2})\Big) \geq A$$

The lemma is proved. ■

B. Suffix Filtering

This section exploits, in addition to the prefix of a set, the suffix of this set. As well as to eliminate the non-similar sets prior to the similarity computation. Let us define the suffix of a set R'_i, $s\text{-}R'_i$ as follows:

Definition 3: Suffix of the set R'_i, $s\text{-}R'_i$. We denote by $s\text{-}R'_i$ the remainder readings of the set R'_i after excluding the prefix $p\text{-}R'_i$. We compute the length of $s\text{-}R'_i$ as $\lceil t \times |R'_i| \rceil - 1$.

Our intuition of suffix filtering is that two sets of readings are similar if their common readings in their suffixes are greater than a calculated bound B as presented in the following lemma.

Lemma 2: Assume that all readings in the sets R'_i and R'_j are ordered according to the ordering $\mathcal{O}$. If $f_s(R'_i, R'_j) \geq \frac{2\times t\times\tau}{1+t}$, then $f_s(s\text{-}R'_i, s\text{-}R'_j) \geq B = \frac{2\times t\times\tau}{1+t} - \sum_{k=1}^{|p\text{-}R'_j|} f(r'_k)$, where $r'_k \in p\text{-}R'_j$.

Proof: We denote by $p\text{-}R'_i$ the prefix of the set R'_i and $s\text{-}R'_i$ the set of remainder readings where $R'_i = \{p\text{-}R'_i + s\text{-}R'_i\}$.

We have: $f_s(R'_i, R'_j)$

$= f_s(p\text{-}R'_i, R'_j) + f_s(s\text{-}R'_i, R''_j)$,

where $R''_j = R'_j - f_s(p\text{-}R'_i, R'_j)$

$= f_s(p\text{-}R'_i, p\text{-}R'_j) + f_s(p\text{-}R'_i, s\text{-}R'_j) + f_s(s\text{-}R'_i, R''_j)$

$\cong f_s(p\text{-}R'_i, p\text{-}R'_j) + f_s(s\text{-}R'_i, R''_j)$

$\leq f_s(p\text{-}R'_i, p\text{-}R'_j) + f_s(s\text{-}R'_i, p'\text{-}R''_j) + f_s(s\text{-}R'_i, s\text{-}R'_j)$,

where $p'\text{-}R''_j = p\text{-}R'_j - f_s(p\text{-}R'_i, p\text{-}R'_j)$

$\leq f_s(s\text{-}R'_i, s\text{-}R'_j) + \sum_{k=1}^{|p\text{-}R'_j|} f(r'_k)$, where $r'_k \in p\text{-}R'_j$

■

In the third line we omitted $f_s(p\text{-}R'_i, s\text{-}R'_j)$ 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 readings having highest frequencies must be in the prefix of the set and not in the remainder. From the above equations and Equation (1) (similarity condition) we can deduce:

$$f_s(s\text{-}R'_i, s\text{-}R'_j) \geq \frac{2\times t\times\tau}{1+t} - \sum_{k=1}^{|p\text{-}R'_j|} f(r'_k), \text{where } r'_k \in p\text{-}R'_j \quad (2)$$

The lemma is proved.

Usually, the length of the suffix is greater than the prefix. Thus, the calculation of the common readings in suffixes is more complex, especially in WSNs where collected sets can consist of ten hundreds or thousands readings. Therefore, in order to reduce the overhead of this calculation, we propose to divide the suffixes of sets into two partitions and then find a reading where at this position a similarity upper bound uB for the overlapping f_s is estimated and checked against the similarity threshold. As soon as the check is failed we can stop the suffix condition computation early. In the rest of this section we will consider that readings in each suffix are ordered following the increasing order of their values.

1) Set's partition: Given a reading w and a set of readings R'_i, the goal of this phase is to divide R'_i into two partitions (R'_{il}, R'_{ir}) such that the first one contains all readings less than w and the second one contains all readings greater than w. This partition is shown in Algorithm 1.

Algorithm 1 Set's Partition Algorithm.

Require: A set of readings R'_i and a reading w.
Ensure: Two subsets of R'_i (R'_{il} *and* R'_{ir}).

```
R'_il ← ∅, R'_ir ← ∅
Search for w or its similar in R'_i
if w is found at position k of the set then
    R'_il ← the first k − 1 readings of R'_i
    R'_ir ← the remainder readings from k to |R'_i|
else
    Find the position k of the first reading in R'_i such that
    this readings is greater than w
    R'_il ← the first k − 1 readings of R'_i
    R'_ir ← the remainder readings from k to |R'_i|
end if
```

2) Upper Bound Computation: After the partition of the sets and in order to reduce the overhead calculation of the overlapping f_s at the suffix level, our proposed technique finds an upper bound uB for f_s that allows stopping the suffix condition computation early. Given a reading w (in random or deterministic) the partition algorithm divides $s\text{-}R'_i$ and $s\text{-}R'_j$ into $s\text{-}R'_{il}$, $s\text{-}R'_{ir}$ and $s\text{-}R'_{jl}$, $s\text{-}R'_{jr}$ respectively. Then, as readings are ordered by increasing order of their values, we can deduce that: $(s\text{-}R'_{il} \cap s\text{-}R'_{jr} = \emptyset)$ and $(s\text{-}R'_{ir} \cap s\text{-}R'_{jl} = \emptyset)$. Hence, the upper bound of f_s can be computed as:

$$f_s(s\text{-}R'_i, s\text{-}R'_j) \leq min\Big(\sum_{k_1=1}^{|s-R'_{il}|} f(r'_{k_1}), \sum_{k_2=1}^{|s-R'_{jl}|} f(r'_{k_2})\Big) + min\Big(\sum_{k_3=1}^{|s-R'_{ir}|} f(r'_{k_3}), \sum_{k_4=1}^{|s-R'_{jr}|} f(r'_{k_4})\Big)$$

where $r'_{k_1} \in s\text{-}R'_{il}, r'_{k_2} \in s\text{-}R'_{jl}, r'_{k_3} \in s\text{-}R'_{ir}$ and $r'_{k_4} \in s\text{-}R'_{jr}$.

If this upper bound is violated, thus both sets are not candidates. This operation can be repeated k times following a recursive manner as shown in Algorithm 2, where k[1] can be defined prior to PSN deployment.

Algorithm 2 Upper Bound Algorithm.

Require: Two Sets of readings R'_i and R'_j, B.
Ensure: The upper bound of overlapping between R'_i and R'_j (uB).

1: $m \leftarrow \lceil |R'_i|/2 \rceil$
2: $w \leftarrow R'_i[m]$ // the reading of R'_i at position m
3: partition(R'_i, w)
4: partition(R'_j, w)
5: $uB \leftarrow \min(\sum_{k_1=1}^{|s\text{-}R'_{il}|} f(r'_{k_1}), \sum_{k_2=1}^{|s\text{-}R'_{jl}|} f(r'_{k_2}))+ \min(\sum_{k_3=1}^{|s\text{-}R'_{ir}|} f(r'_{k_3}), \sum_{k_4=1}^{|s\text{-}R'_{jr}|} f(r'_{k_4}))$
6: **if** $uB < B$ **then**
7: return uB
8: **else**
9: $uB_l \leftarrow$ Upper bound (R'_{il}, R'_{jl}, B)
10: $uB \leftarrow uB_l + \min(\sum_{k_3=1}^{|s\text{-}R'_{ir}|} f(r'_{k_3}), \sum_{k_4=1}^{|s\text{-}R'_{jr}|} f(r'_{k_4}))$
11: **if** $uB \geq B$ **then**
12: $uB_r \leftarrow$ Upper bound $(R'_{ir}, R'_{jr}, B - uB_l)$
13: return $uB_l + uB_r$
14: **else**
15: return uB
16: **end if**
17: **end if**

Based on the complexity of prefix filtering calculated on [15], we can found the complexity of the suffix filtering in similar manner and we can say that it is equal, in worst case, to the prefix filtering ($O(\Theta_\ell)$).

C. PPSFF Algorithm

In this section we present our Position-based Prefix-Suffix Frequency Filtering (PPSFF) algorithm, which integrates the position filter to the prefix following by the suffix filter (studied before). Algorithm 3 describes our technique to find similar sets of readings based on the PPSFF. It is a hybrid solution, where we integrate our position and suffix conditions presented in lemmas 1 and 2 to the PFF technique proposed in [1].

When PPSFF identifies all pairs of similar data sets, the aggregator deletes redundant data sets sent from neighboring sensors in order to reduce the amount of data transmitted to the sink while conserving the integrity of information. Algorithm 4 shows how the aggregator selects the data sets to be sent to the sink among the pairs of similar received sets. For each similar pair of set, the aggregator chooses the one having the highest cardinality (line 3), then it adds it to the list of sets to be sent to the sink (line 4). After that, it removes all pairs of similar sets that contain R'_i or R'_j from the set of pairs (which means it will not check them again). Finally, the aggregator assigns to each set its frequency (line 6).

[1]in this article k is equal to 2.

Algorithm 3 PPSFF Algorithm.

Require: Set of reading sets $R' = \{R'_1, R'_2, \ldots, R'_n\}$, t.
Ensure: All pairs of sets (R'_i, R'_j), such that $J(R'_i, R'_j) \geq t$.

- Insert after line 2 in PFF Algorithm [1]
for each set $R'_i \in R'$ **do**
$p\text{-}R'_i \leftarrow$ sort($p\text{-}R'_i, |p\text{-}R'_i|$), the prefix of R'_i is sorted by increasing order of the readings
end for

- Insert after line 15 in PFF Algorithm [1]
$ubound \leftarrow Fs[R'_j] + min(\sum_{n=k+1}^{|p\text{-}R'_i|} f(R'_i[n]), \sum_{n=l+1}^{|p\text{-}R'_j|} f(R'_j[n]))$
if $ubound < sumFreq - \frac{1-t}{1+t} \times \mathcal{T}$ **then**
$F_s[R'_j] \leftarrow 0$
else
line 16 in PFF Algorithm [1]
end if

- Replace line 22 in PFF Algorithm [1] with
for each R'_j such that $F_s[R'_j] > 0$ **do**
compute the bound B
compute uB following suffix filtering technique
if $uB \geq B$ **then**
lines 23 to 25 in PFF Algorithm [1]
end if
end for

Algorithm 4 Selecting Sets Algorithm.

Require: All pairs of sets (R'_i, R'_j), such that $J(R'_i, R'_j) \geq t$.
Ensure: List of selected sets, L.

1: $L \leftarrow \emptyset$
2: **for** each similar pair of sets (R'_i, R'_j) **do**
3: Consider $|R'_i| \geq |R'_j|$
4: $L \leftarrow L \cup \{R'_i\}$
5: Remove all pairs of sets containing one of the two sets R'_i and R'_j
6: $f(R'_i) = number\,of\,removed\,pairs + 1$
7: **end for**

IV. Experimental Results

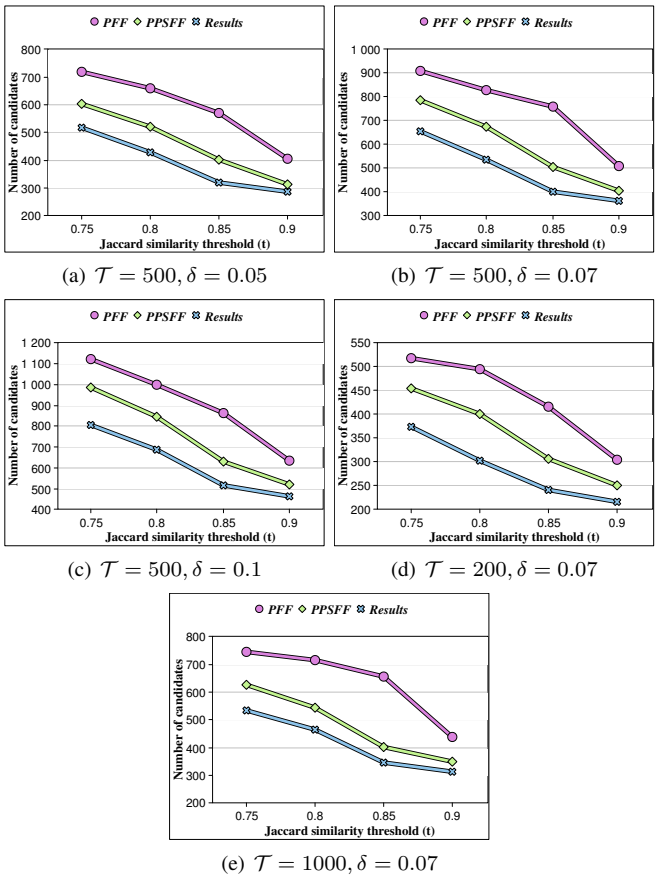
In this section, we present the experimental results which show the effectiveness of PPSFF compared to PFF technique proposed in [1]. We have implemented both techniques based on a Java simulator. We used the publicly available Intel Lab dataset which contains data collected from 54 Mica2Dot sensors deployed in the Intel Berkeley Research Lab [16]. Each sensor collects new readings of humidity, temperature, light and voltage once every 31 seconds. For the sake of simplicity, in this paper we are interested in one field of sensor readings: the temperature[2]. Each node reads periodically real readings while applying the first aggregation phase. At the end of this step, each node sends its set of readings/frequencies to an aggregator node which in its turn applies filtering techniques over theses sets. In our experiments, we have varied δ to 0.05,

[2]the others are done by the same manner.

0.07 and 0.1, and we varied the period size, e.g. $\mathcal{T}$, from 500 to 1000 and t is varied from 0.75 to 0.9.

A. Number of Candidates / Comparisons

In this section, we compare the number of candidates (number of compared sets) generated respectively using PPSFF technique, PFF technique and the results obtained after applying the Jaccard similarity function (the real number of similar sets). The obtained results in Fig. 2 (a to e) show that our PPSFF technique can successfully reduce, when varying the threshold t, the number of comparisons in all cases compared to PFF. Subsequently, the suffix filtering helps prune from 12% to 39% infeasible candidates that remain after applying PFF.



(a) $\mathcal{T} = 500, \delta = 0.05$ (b) $\mathcal{T} = 500, \delta = 0.07$

(c) $\mathcal{T} = 500, \delta = 0.1$ (d) $\mathcal{T} = 200, \delta = 0.07$

(e) $\mathcal{T} = 1000, \delta = 0.07$

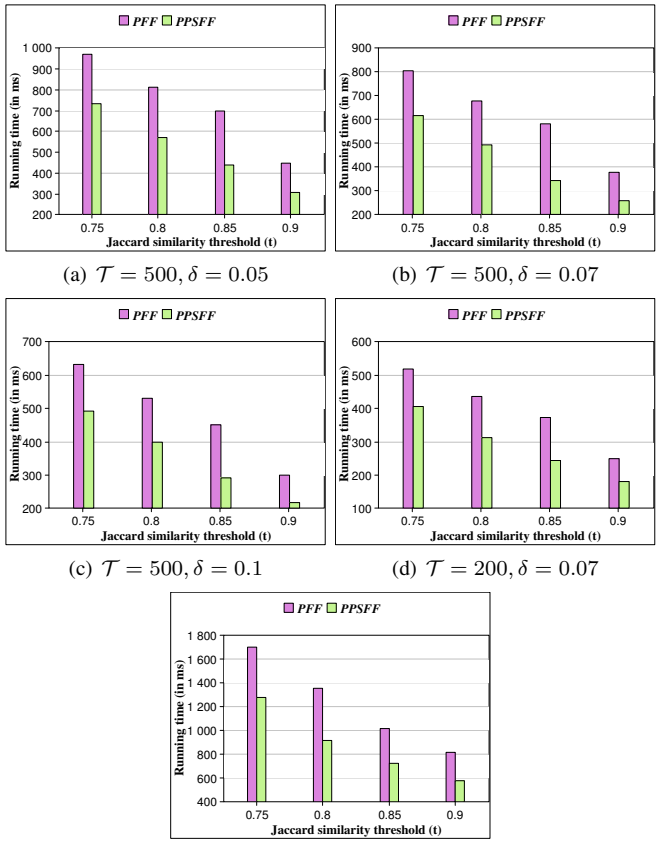
Fig. 2. Sets comparison.

Based on the results in Fig. 2, several observations are eminent:

1) both techniques generate less candidate pairs by increasing similarity threshold t. This happens because the similarity between sets decreases when t increases.
2) the number of candidates generated using PPSFF becomes closer to the final results when t increases. This is because, when t increases the length of the prefix decreases thus its corresponding suffix length's increases. Therefore, the suffix filtering can found more unsimilar readings between suffixes of two sets.
3) the number of erroneous candidates deleted by the PPSFF technique increases when δ threshold decreases. This is because, the cardinality of a set increases when δ decreases, thus the length of the suffix increases. Therefore, the suffix filtering used in PPSFF will find more erroneous candidates to be deleted.

B. Data Latency / Running Time

In this section, we compare the execution time required for both techniques, PFF and PPSFF, in order to aggregate data (Fig. 3). In PFF, the execution time is highly dependent on the prefix filtering and the number of generated candidates. Consequently, PPSFF outperforms PFF in terms of execution time for two reasons: first, the positional filtering used in PPSFF can minimize the computation of similarity between readings in prefixes of the sets; second, the number of candidates is reduced in PPSFF thanks to the suffix filtering which leads to minimize the time required to verify these candidates. The obtained results in Fig. 3 show that the additional filters used in PPSFF can successfully minimize, up to 41%, the execution time for PFF for all similarity thresholds.



(a) $\mathcal{T} = 500, \delta = 0.05$ (b) $\mathcal{T} = 500, \delta = 0.07$

(c) $\mathcal{T} = 500, \delta = 0.1$ (d) $\mathcal{T} = 200, \delta = 0.07$

(e) $\mathcal{T} = 1000, \delta = 0.07$

Fig. 3. Running time.

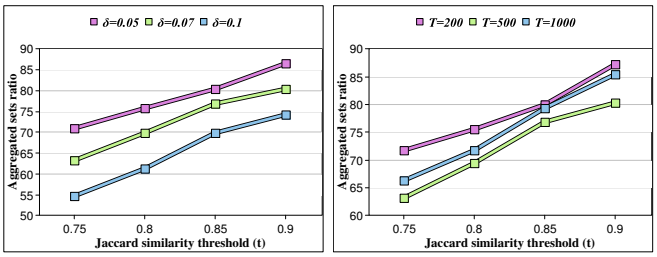
Several observations can be made based on the results shown in Fig. 3:

1) the running time for both techniques, PPSFF and PFF, decreases when the similarity threshold t increases, because the candidates number's and the final results decrease when t increases (see results in Fig. 2).
2) the PPSFF technique is more efficient when the threshold δ decreases, except for $\delta = 0.9$. This is due to

two reasons: first, when δ decreases the length of sets prefixes' increases therefore the positional filtering can optimize more the overhead of similarity computation between prefixes of the sets; second, the number of erroneous candidates deleted by the PPSFF technique increases when δ decreases (see observations for Fig. 2). Otherwise, for $\delta = 0.9$, the positional filtering has no effect because the length of the prefix becomes very short.

C. Aggregated Sets Ratio

In Fig. 4, we show the results for the aggregated sets ratio which represents the percentage of readings sets' received by the sink, at each period, over the total number of sets generated by all sensor nodes for the temperature field. These experiments permit to show how well aggregation techniques reduce redundant data. In both cases, PPSFF and PFF, we have the same aggregated sets ratio which means that we find, at the end of the similarity computation, the same number of readings sets' to be sent to the sink. The obtained results show that both techniques can reduce 13% and up to 45% of sets sent to the sink. Therefore, PPSFF can be considered as an efficient data aggregation technique that eliminates redundant data forwarded to the sink within a minimum delay for data delivery. We can also notice that the aggregated sets ratio increases when the similarity threshold (t) increases or the threshold δ decreases. This confirm the good behavior of PPSFF which eliminates redundant data sets while preserving the accuracy of sent data.



(a) $\mathcal{T} = 500$ (b) $\delta = 0.07$

Fig. 4. Aggregated sets ratio.

V. Conclusion and Future Work

In this paper, we proposed an efficient data aggregation technique called PPSFF with low level data latency dedicated to periodic sensor applications. The objective of our technique is to reduce the number of redundant data sent to the end user while preserving the data integrity. We have implemented two novel filtering technique, e.g. position and suffix algorithms. We used the additional filtering methods to accelerate the calculation and prune erroneous candidates that survive after applying the prefix frequency filtering technique. We show through experiment results that our technique reduces the number of comparisons and outperforms existing prefix filtering techniques in terms of data latency.

As future work, we will adapt our proposed technique to take into consideration reactive periodic sensor networks, where sensor nodes operate with different sampling rate. In periodic applications the dynamics of the monitored condition or process can slow down or speed up; and to save more energy the sensor node can adapt its sampling rates to the changing dynamics of the condition or process.

Acknowledgements

This project has been performed in cooperation with the Labex ACTION program (contract ANR-11-LABX-0001-01).

References

[1] J. Bahi, A. Makhoul, and M. Medlej, "A two tiers data aggregation scheme for periodic sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 21, no. (1-2), pp. 77–100, 2014.

[2] H. Harb, A. Makhoul, R. Tawil, and A. Jaber, "Energy-efficient data aggregation and transfer in periodic sensor networks," *IET Wireless Sensor Systems*, vol. 4, no. 4, pp. 149–158, 2014.

[3] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "Permasense: investigating permafrost with a wsn in the swiss alps," *Proc. Of the 4th workshop on Embedded networked sensors (EmNets'07), New York, USA*, pp. 8–12, 2007.

[4] M. Bagaa, Y. Challal, A. Ksentini, A. Derhab, and N. Badache, "Data aggregation scheduling algorithms in wireless sensor networks: Solutions and challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1339–1368, 2014.

[5] M. Bagaa, M. Younis, D. Djenouri, A. Derhab, and N. Badache, "Distributed low latency data aggregation scheduling in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 3, p. Article No. 49, 2015.

[6] J. M. Bahi, A. Makhoul, and M. Medlej, "An optimized in-network aggregation scheme for data collection in periodic sensor networks," *Ad-hoc, Mobile, and Wireless Networks - 11th International Conference, ADHOC-NOW 2012, Belgrade, Serbia, July 9-11, 2012. Proceedings*, pp. 153–166, 2012.

[7] D. Wang, R. Xu, X. Hu, and W. Su, "Energy-efficient distributed compressed sensing data aggregation for cluster-based underwater acoustic sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2016, no. 2016, p. 14 pages, 2016.

[8] G. Li and Y. Wang, "Automatic arima modeling-based data aggregation scheme in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 85, pp. 1–13, 2015.

[9] A. Tripathi, S. Gupta, and B. Chourasiya, "Survey on data aggregation techniques for wireless sensor networks," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 7, 2014.

[10] P. Zou and Y. Liu, "A data-aggregation scheme for wsn based on optimal weight allocation," *Journal Of networks*, vol. 9, no. 1, pp. 100–107, 2014.

[11] M. Bahrepour, N. Meratnia, and P. Havinga, "Sensor fusion-based event detection in wireless sensor networks," *6th Annual International Mobile and Ubiquitous Systems: Networking & Services*, pp. 1–8, 2009.

[12] K. T.-M. Tran, S.-H. Oh, and J.-Y. Byun, "Well-suited similarity functions for data aggregation in cluster-based underwater wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2013, no. Article ID 645243, p. 7 pages, 2013.

[13] H. Harb, A. Makhoul, D. Laiymani, A. Jaber, and R. Tawil, "K-means based clustering approach for data aggregation in periodic sensor networks," *IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2014, Larnaca, Cyprus, October 8-10, 2014*, pp. 434–441, 2014.

[14] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for similarity joins in data cleaning," *Proc. Int. Conf. on Data Engineering (ICDE'06), Atlanta, GA*, pp. 1–5, 2006.

[15] J. Wang, G. Li, and J. Feng, "Can we beat the prefix filtering? an adaptive framework for similarity join and search," *Proc. of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*, pp. 85–96, 2012.

[16] S. Madden, "http://db.csail.mit.edu/labdata/labdata.html," *Available Online*, 2004.