A Distributed Processing Technique for Sensor Data Applied to Underwater Sensor Networks

Mohamad Mortada[a,1], Abdallah Makhoul[a,2], Chady Abou jaoude[c,3], Hassan Harb[c,3], David Laiymani[a,5]
[a]*FEMTO-ST Institute/CNRS, the DISC department, Univ. Bourgogne Franche-Comté, Belfort, France*
[c]*Ticket lab, Antonine University, Baada, Lebanon*
Emails: [1]reda.mrtada@gmail.com, [2]abdallah.makhoul@univ-fcomte.fr, [3]chady.aboujaoude@ua.edu.lb,
[4]hassanharb@auce.edu.lb, [5]david.laiymani@univ-fcomte.fr

***Abstract*—Wireless sensor networks (WSN) present a low cost solution to enhance our lives. They allow a large variety of applications. One of the major challenges faced by WSN is that of energy saving. A well known efficient way to reduce energy consumption is data reduction. It consists in reducing the amount of data sensed and transmitted to the sink. Consequently, sensor data communication should be minimized in order to increase network lifetime. In this paper, we propose an energy-efficient data reduction technique based on a clustering architecture. Our objective is to identify and study data similarities at both sensor and cluster-head (CH) levels. At the first level, each sensor sends a set of representative points to the CH at each period, instead of sending the raw data. When data points are received by the CH, it uses the Euclidean distance in order to eliminate redundant data generated by neighboring sensor nodes, before sending them to the sink. To validate our approach, we applied our techniques on real underwater sensor data and we compared them with other existing data reduction methods. The results show the effectiveness of our technique in terms of improving the energy consumption and the network lifetime, without loss in data fidelity.**

***Index Terms*—Underwater Sensor Networks; Periodic Applications; Euclidean Distance; Data Aggregation, Real Sensor Data.**

I. Introduction

Wireless sensor networks (WSN) provide a low cost solution to enhance our lives. They allow a large variety of surveillance applications (medical, environmental, smart city, etc.). Their main advantages are fast, easy deployment and low maintenance cost. One of the major challenges faced by wireless sensor networks is that of energy saving. Indeed, data transmission consumes most of the available sensor's energy [1]. Furthermore, the periodic data collection in surveillance applications, produces a huge amount of data, which are usually redundant [2]. Then, the transmission of such amount of data is very expensive in terms of energy. In this way, reduction of sensed data becomes an efficient way to reduce energy consumption in WSNs.

In this paper, our aim is to propose a new distributed and low complex sensor data processing technique. We adopt a cluster based network's topology, where, sensor data are processed and aggregated at intermediate nodes, i.e. called Cluster-Heads (CHs), before sending them to the sink. This architecture has been proved to be efficient in terms of scalability and energy saving. Therefore, we studied a two data reduction levels technique. It aims at optimizing the volume of transmitted data at each cluster by achieving aggregation at both sensor nodes and CH levels in a periodically way. At the first level, each sensor node transforms its set of collected data to a reduced set of representative points. Then, it sends the set of points to its CH at the end of each period. After receiving the sets of points from all its sensors, the CH searches the similarity between each pair of data points coming from two sensors, with a technique based on the Euclidean distance concept. This phase allows to eliminate the redundancy between the received set of points. Finally, each CH sends the set of final data points to the sink. It is important to notice, that from the representative set of point the sink will be able to reconstruct the whole set of data with minimal errors.

To evaluate our approach, we choose to apply our techniques to underwater sensor networks. We believe that monitoring the aquatic environments is becoming a requirement for offering a better understanding of marine life. In such networks, the main objective is to monitor and observe the different kinds of aquatic environments, then periodically send the collected data to the end user for analysing and studying purposes. Furthermore, although the acoustic technology used in underwater ensures a long distance data communication [3], it consumes most of the available sensor's energy which is usually a limited battery power. For all these reasons, we chose real underwater sensor data to be the test-bed for our proposal. In addition, we compared our techniques with existing data reduction methods. We show how the effectiveness of our approach in reducing data and saving energy while guaranteeing high level of information integrity.

The rest of the paper is organized as follows: Section II gives an overview on related works reported on data aggregation in UASNs. Section III presents our sensor data processing technique, where each sensor node computes its representative data vector. In section IV we provide a multi-sensor data aggregation technique at the cluster head level. In Section V we detail the simulations we have conducted on real underwater readings data with a discussion of the obtained results. Finally, we conclude our paper and provide our directions for future work in Section VI.

II. Related works

In the literature, one can find various data reduction approaches based on in-network processing [4]–[6], data compression or data prediction methods [7]–[10]. They are based mainly on algorithms like least mean square [8], [10] and Kalman Filter [7], [9]. Although these approaches predict sensed values and allow efficient data reduction, however they present several disadvantages. They are computationally complex, sometimes they generate communication overhead, and the sink may need some transmissions to detect failures.

To reduce the amount of data transmitted in the network, other techniques have been proposed and consist in data aggregation. Data aggregation aims at eliminating redundancy in data collected and minimizing the number of transmissions, thus saving the overall network energy. Recently, the majority of the proposed data aggregation techniques have been built with the clustering scheme which has been proven to be an efficient way in terms of scalability and data traffic [5].

In [11], [12], the authors study the data aggregation in UASNs as a compression scheme for data generated in each cluster. The authors in [11] and [13] propose two data aggregation schemes, namely block diagonal matrix and block upper triangular matrix, for cluster-based UASNs inspired by the Distributed Compressed Sensing (DCS) technique . The main objective of such schemes is to generate RIP-preserving (Restricted Isometric Property) measurements of sensor readings by taking multi-hop underwater acoustic communication cost into account. Finally, a distributed compressed sensing reconstruction algorithm, called DCS-SOMP, is adopted to recover raw sensor readings at the fusion centre.

Some works in data aggregation in UASNs, such as [14], are based on the formation of clusters and the selection of cluster-heads. The authors in [14] propose a data aggregation round-based clustering scheme in order to reduce the transmission of redundant data in UASNs. The proposed scheme works in rounds where each round consisting in four main phases: initialization, cluster-head selection, clustering, and data aggregation. By applying some mechanisms in each round, the proposed scheme reduced the energy consumption in the network and minimized data redundancy, while still guaranteeing data accuracy. In [15], the authors propose EBDSC, a distributed Energy-Balanced Dominating Set-based Clustering scheme, to extend the network lifetime by balancing energy consumption among different nodes. In EBDSC, a node becomes a cluster head candidate if it has the longest lifetime among its neighbours.

Other data aggregation and collection studies, such as [16], [17], have been proposed. In these works the computation of statistical means and moments summarize the data obtained by the UASNs. In [16], the authors propose an analytical model group-based sensor network in order to monitor the accurate amount of pollution that is deposited on the seabed. The objective is to study the effects produced by feed loss in the marine fish cages and its environment impact. After searching the best location to place the sensor nodes, the proposed model determines the amount of food that is wasted while it measures the amount of generated deposits. The authors in [17] propose to design a fuzzy based clustering and aggregation technique for UWSN. In this technique the parameters such as the residual energy, the distance to sink, the node density, the load and the ilnk quality are considered as input to the fuzzy logic. Based on the output of fuzzy logic module, appropriate cluster heads are elected and act as aggregator nodes.

Finally, works in [18], [19] are dedicated to periodic applications in sensor networks. The authors use some similarity functions to aggregate data generated in the networks. The main objective is to eliminate redundancy and reduce the size of data transmitted thus, optimizing the energy consumption and reducing overload on the network level. Further to a local processing at the sensor node level, the authors in [18] propose a prefix frequency filtering (PFF) technique at the CH level. PFF uses Jaccard similarity function to identify similarities between near sensor nodes at each period and integrates their sensed data into one record. Then, several versions of PFF, i.e. PSFF [20] and KPFF [21], have been proposed in order to optimize the data latency. On the other hand, the authors in [19] use distance functions, such as Euclidean and Cosine, at the CH level to build an efficient underwater network by reducing packet size and by minimizing data redundancy. However, although the proposed techniques eliminate the similar data, some redundancy still remain in the final data sets sent to the sink.

In this paper, we propose a new less complex data reduction and aggregation technique suitable for low resources sensor networks. In this technique, data aggregation are performed at both sensor and CH levels where we transform the raw data to a set of representative data points after eliminating redundancy among them. Compared to the existing techniques, our technique is more efficient to reduce the redundancy among raw data and thus, to preserve the energy in the network.

III. Sensory Data Processing

In this section we present our sensor data processing technique, to be executed by sensor nodes in order to find the representative data points. First, we present the network's topology that we consider in our approach.

A. WSN topology

In this paper, we consider a cluster-based architecture for the network. The cluster-based architecture is based on grouping sensor nodes into clusters, and assigning for each cluster a super node (intermediate node), the cluster head (CH). The CH is elected after the network deployment and can be changed dynamically during the network lifetime. It can be a regular node or a specific more powerful one. Data transmission between sensor nodes and their appropriate CH is based on single-hop communication as presented in Fig. 1. In this work, we use the periodic data collection approach, in which each sensor node sends periodically (period p) its data to the appropriate CH, which in his turn sends them to the sink. Then, we propose an energy efficient technique which performs data reduction at sensor node and CH levels.

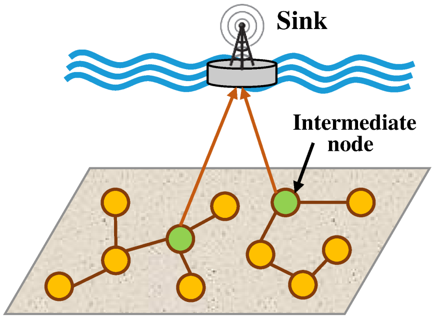


Fig. 1. WSN clustering based topology

B. *Similar sensor data searching*

In periodic applications, each sensor node collects a vector of readings in each period before sending it to the CH as follows: $R_i = [r_1, r_2, \ldots, r_{\tau-1}, r_\tau]$ where τ is the total number of readings captured during a period p. Mostly, readings collected from the sensor in each period, i.e. in R_i, are redundant depending on how the monitored conditions vary. Thus, searching data redundancy in each sensor becomes necessary in order to reduce the number of reported readings and to save sensor's energy. Hence, our objective, in this section, is to reduce the size of R_i by searching the existing similarities between the readings in R_i. First, we define a threshold δ for the similarity between readings in R_i. Two readings are considered redundant if their difference is less than the defined threshold as follows:

$$|| \, r_i - r_j \, || \leq \delta \tag{1}$$

where r_i and $r_j \in R_i$ and δ is a user defined value.

Here, we propose to transform the vector R_i to a reduced set of points, where each two points represent a line. The points are chosen among the readings in R_i where the readings between any two points, i.e. along a line, are considered as redundant. For this purpose, we use the Euclidean distance in order to search the number of lines for each data vector R_i. In the next sub-sections, we detail the computation of the Euclidean distance between two data vectors and then, we describe how we transform R_i to a set of points.

1) Computation of the Euclidean distance: According to equation 1, a reading r_i is considered similar to another reading r_j within a specified error δ. Consequently, the data vector R_i will be considered similar to another vector R_j after searching similar readings. Therefore, the threshold δ should be taken into account when searching the Euclidean distance between two data vectors. Hence, we propose to integrate δ in the computation of the Euclidean distance threshold t_d as shown in the following lemma:

Lemma 1: Assume two vectors of data R_i and R_j with the same size τ. R_i and R_j are considered similar if the Euclidean distance between them is less than a defined threshold t_d as follows:

$$Ed(R_i, R_j) \leq t_d = \delta \times \sqrt{\tau} \tag{2}$$

Proof 1: Let two vectors of data R_i and R_j. The Euclidean distance between them is calculated as follows:

$$E_d(R_i, R_j) = \sqrt{\sum_{i=1}^{\tau} (r_i - r_j)^2} \leq t_d, \quad \text{where } r_i \in R_i \text{ and } r_j \in R_j$$

Consider that R_i is similar to R_j after searching similarity readings in R_i. Thus , we have:

$$\begin{aligned} E_d(R_i, R_j) = \sqrt{\sum_{i=1}^{\tau} (r_i - r_j)^2} &\leq \sqrt{\sum_{i=1}^{\tau} \delta^2} \\ E_d(R_i, R_j) = \sqrt{\sum_{i=1}^{\tau} (r_i - r_j)^2} &\leq \sqrt{\tau \times \delta^2} \\ E_d(R_i, R_j) = \sqrt{\sum_{i=1}^{\tau} (r_i - r_j)^2} &\leq \delta \times \sqrt{\tau} \end{aligned}$$

Thus, R_i and R_j should be considered similar only if the Euclidean distance between them does not exceed $t_d = \delta \times \sqrt{\tau}$. The lemma is proved.

2) Searching for representative data points: After collecting its data vector R_i at each period, the sensor S_i computes a set of representative points for R_i to send to the CH instead of sending the whole data vector. Each point is represented by the pair $(index,\ R_i[index])$ where $index$ indicates the index of a reading in R_i, i.e. between 1 and τ, and $R_i[index]$ its corresponding value. Algorithm 1 describes how S_i can find the minimum number of points that represent R_i by applying iteratively the Euclidean distance. The process starts by defining a line corresponding to the first and the last points in R_i: $(start,\ R_i[start])$ and $(end,\ R_i[end])$ respectively. Then, it calculates the Euclidean distance between the points belonging to this line and their corresponding data in R_i, i.e. in this case all data in R_i (line 2). If the calculated distance is less the distance threshold t_d (line 3) then, the points are considered as final points and they are added to the list of representative points P_i (line 4). Otherwise, i.e. the distance is greater than the threshold, the distance between the indexes of the two points is divided by two and new indexes are calculated (line 6 and 7). Then, the process is restarted over the points of the new indexes. Finally, the process is repeated until all the points are added to the list of representative points.

During this phase and in addition to the list of representative points, the sensor S_i calculates the radius, e.g. D_i, of its collected data vector R_i. D_i is defined as the Euclidean distance between the collected data, e.g. R_i, and the origin centre in $\mathbb{R}^\tau$ as shown in equation 3. The objective of the radius is to help the CH in computing the similarities between each sensor and its neighbouring nodes (see next section).

Algorithm 1 Local Aggregation Recursive Algorithm.

```
Require: Vector of readings: $R_i$, distance threshold: $t_d$, start index: $start = 1$, end index: $end = \tau$.
Ensure: List of representative points of $R_i$: $P_i$.
$P_i \leftarrow \emptyset$; // list of empty points
$E_d = \sqrt{\sum_{index=start}^{end}\left[\frac{(index-start)\times(R[end]-R[start])}{end-start} + R[start] - R_{index}\right]^2}$
if $E_d \leq t_d$ then
    $P_i \leftarrow P_i \cup \{(start, R[start])\} \cup \{(end, R[end])\}$
else
    $P_i \leftarrow P_i \cup Local\_Aggregation(R, \frac{t_d}{\sqrt{2}}, start, \frac{start+end}{2})$
    $P_i \leftarrow P_i \cup Local\_Aggregation(R, \frac{t_d}{\sqrt{2}}, \frac{start+end}{2}, end)$
end if
return $P_i$
```

$$D_i = \sqrt{\sum_{i=1}^{\tau} r_i^2}, \text{ where } r_i \in R_i \tag{3}$$

Finally, each sensor node S_i sends its list of representative points P_i and its radius D_i to the CH, at the end of each period. In the next section, we describe how the CH will aggregate the data coming from its member nodes before to send them to the sink.

IV. Multi-sensor data similarity searching

At the end of each period, the CH receives the sets of points with their corresponding radiuses coming from its member nodes. The objective is then to identify all pairs of member nodes that generate redundant sets in order to eliminate duplication before sending them to the sink. In the previous section, we considered that two sets are similar if the Euclidean distance between them is less than the threshold t_d. However, applying the Euclidean distance for every pair of sets is very expensive in terms of computation since it generates $O(n^2)$ number of comparisons, where n is the number of received sets. In addition, the computation will be more complex for large data sets as in the case of dense sensor networks. Therefore, in order to reduce the number of comparisons, it is mandatory to search the pairs of redundant sets. This search will be performed in two phases. In the first phase we compute a list of pairs which are "candidates" to be similar. A pair is candidate if it satisfies some conditions and it means that the two sets composing this pair may be similar. However, a pair is not candidate means that it is for sure not similar. To ensure the similarity of candidate pairs, we need a verification phase. This verification is necessary since different sets of points coming from different sensor nodes may be of different size (see sub-section IV-B).

A. The candidate pairs generation phase

In this phase, each CH computes the pairs of sets (set of points or vectors) which are "candidate to be similar". Our intuition is that if the distance between the radiuses of two sets of points is less than the threshold t_d then, the Euclidean distance between the two sets of points is candidate to be less than t_d. Therefore, in our work, we prove that two sets of points P_i and P_j are candidates if and only if the distance between their corresponding radiuses is less than t_d as shown in the following lemma:

Lemma 2: Consider two sets of points P_i and P_j with their corresponding radiuses D_i and D_j respectively. Assume that R_i and R_j are the initial data vectors of P_i and P_j respectively. Thus, if the Euclidean distance between R_i and R_j is less than the distance threshold t_d then, the distance between their corresponding radiuses should be also less than t_d. Therefore:

$$P_i \text{ and } P_j \text{ are considered candidates} \iff |D_i - D_j| \leq t_d$$

Proof 2: This lemma can be simply demonstrated based on the proof of lemma 1.

Algorithm 2 describes how each CH searches the set of candidates for each sensor. It takes as input a collection of data points sets with their radiuses coming from different sensor nodes. It scans sequentially each set of points P_i (line 2) and selects the candidates based on the lemma 2 (line 5). Afterwards, P_i and all its candidates will be verified according to the Euclidean distance threshold (line 7 and sub-section 4.2). Finally the algorithm returns a list of n-uplet where for each points sets P_i are associated the sensors S_j "candidate" to be similar to S_i.

Algorithm 2 Candidate Pairs Searching Algorithm.

```
Require: Sets of points: $P = \{P_1, P_2, \ldots, P_n\}$, Set of radius: $D = \{D_1, D_2, \ldots, D_n\}$, distance threshold: $t_d$.
Ensure: List of data points sets with duplicated data sensors for each one.
 1: $S \leftarrow \emptyset$
 2: for each set of points $P_i \in P$ do
 3:     $F_i \leftarrow S_i$; // list of similar sensors to $S_i$
 4:     for each set of points $P_j \in P$ such that $j > i$ do
 5:         if $|D_i - D_j| \leq t_d$ then
 6:             // $P_i$ and $P_j$ are candidates
 7:             if $Euclidean\_Distance(P_i, P_j) \leq t_d$ then
 8:                 $F_i \leftarrow F_i \cup \{S_j\}$
 9:                 $P \leftarrow P - \{P_j\}$ // remove $P_j$ from $P$
10:             end if
11:         end if
12:     end for
13:     $S \leftarrow S \cup \{(P_i, F_i)\}$
14: end for
15: return $S$
```

B. Candidate pairs' verification

As previously exposed, two sets of points P_i and P_j in a candidate pair are considered similar if their distance is less than the distance threshold t_d (line 7 in Algorithm 2). However, P_i and P_j can have different sizes, i.e. number of points. This property makes the computation of the Euclidean distance not trivial.

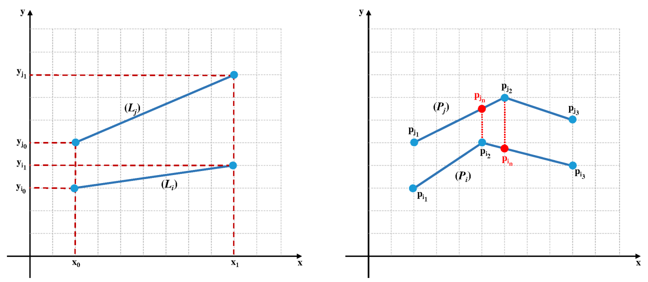
In this section, we propose an improved version of the Euclidean distance in order to compute the distance between two sets in a candidate pair. The improved version calculates the Euclidean distance based on the lines formed by the points in the sets. For clarity reason, we first describe how we calculate the Euclidean distance between two lines formed by two points, then we generalize our method to all points in the two sets.

Let consider a line L_i defined by two points $p_0(x_{i_0}, y_{i_0})$ and $p_1(x_{i_1}, y_{i_1})$ where $x_{i_0} \leq x_i \leq x_{i_1}$ and $y_{i_0} \leq y_i \leq y_{i_1}$. Thus, the equation of L_i can be calculated as follows:

$$(L_i) : y_i = a_i \times x + b_i \tag{4}$$

where $a_i = \dfrac{y_{i_1} - y_{i_0}}{x_{i_1} - x_{i_0}}$ and $b_i = y_{i_0} - a_i \times x_{i_0}$.

Thus, if we have two lines L_i and L_j with the same values for x_{i_0} and x_{j_0} (see Fig. 2(a)), then we use the following lemma to compute the Euclidean distance between them.



(a) lines with same sizes

(b) lines with different sizes

Fig. 2. Computation of Euclidean distance.

Lemma 3: Consider two lines L_i and L_j where L_i is defined by $\{(x_0, y_{i_0})(x_1, y_{i_1})\}$ and L_j is defined by $\{(x_0, y_{j_0})(x_1, y_{j_1})\}$. Assume that, the equation of L_i is $y_i = a_i \times x + b_i$ and the equation of L_j is $y_j = a_j \times x + b_j$. Thus, the Euclidean distance between L_i and L_j is:

$$E_d(L_i, L_j) = \sqrt{a_{ij}^2 \times \frac{q \times (q-1) \times (2q-1)}{6} + q \times b_{ij}^2 + a_{ij} \times b_{ij} \times q \times (q-1)} \tag{5}$$

where $a_{ij} = a_i - a_j$, $b_{ij} = b_i - b_j$ and $q = x_{i_1} - x_{i_0}$

Proof 3: Consider two lines $L_i = \{(x_0, y_{i_0})(x_1, y_{i_1})\}$ and $L_j = \{(x_0, y_{j_0})(x_1, y_{j_1})\}$. Then, assume that $y_i = a_i \times x + b_i$ and $y_j = a_j \times x + b_j$ are the equations of L_i and L_j respectively. Thus, the Euclidean distance between L_i and L_j is:

$$E_d(L_i, L_j) = \sqrt{\sum_{k=x_0}^{x_1} y_{ij}^2},$$

where $y_{ij} = a_{ij} \times k + b_{ij}$, $a_{ij} = a_i - a_j$ and $b_{ij} = b_i - b_j$

$$E_d(L_i, L_j) = \sqrt{\textstyle\sum_{k=x_0}^{x_1} (a_{ij} \times k + b_{ij})^2}$$

$$E_d(L_i, L_j) = \sqrt{\textstyle\sum_{k=x_0}^{x_1} (a_{ij}^2 \times k^2 + b_{ij}^2 + 2 \times a_{ij} \times b_{ij})}$$

$$E_d(L_i, L_j) = \sqrt{\textstyle\sum_{k=x_0}^{x_1} (a_{ij} \times k + b_{ij})^2}$$

$$E_d(L_i, L_j) = \sqrt{\textstyle\sum_{k=x_0}^{x_1} (a_{ij}^2 \times k^2 + b_{ij}^2 + 2 \times a_{ij} \times b_{ij})}$$

$$E_d(L_i, L_j) = \sqrt{\textstyle\sum_{k=x_0}^{x_1} (a_{ij}^2 \times k^2) + \sum_{k=x_0}^{x_1} b_{ij}^2 + \sum_{k=x_0}^{x_1} (2 \times a_{ij} \times b_{ij} \times k)}$$

$$E_d(L_i, L_j) = \sqrt{a_{ij}^2 \times \tfrac{q \times (q-1) \times (2q-1)}{6} + q \times b_{ij}^2 + 2 \times a_{ij} \times b_{ij} \times \tfrac{q \times (q-1)}{2}}$$

$$E_d(L_i, L_j) = \sqrt{a_{ij}^2 \times \tfrac{q \times (q-1) \times (2q-1)}{6} + q \times b_{ij}^2 + a_{ij} \times b_{ij} \times q \times (q-1)}$$

The lemma is proved.

Based on the lemma 3, the Euclidean distance is calculated between two lines with the same size, i.e. the same values for x-axis points. However, data sets received by the CH may contain different number of points or the lines may have different lengths. Let consider a simple example of two sets P_i and P_j (Fig. 2(b)). Each set contains three points while the lines have different lengths (Fig. 2(b)). Then, in order to make equal the lengths of their lines, we propose to insert two points p_{i_n} and p_{j_n} to P_i and P_j respectively. It is important to notice that, the x-axis of p_{i_n} (respectively p_{j_n}) is the same to those of p_{j_2} (respectively p_{i_2}) while the y-axis of p_{i_n} (respectively p_{j_n}) can be calculated from the equation of line formed by $\{p_{j_1}, p_{j_2}\}$ (respectively $\{p_{i_1}, p_{i_2}\}$). Finally, the Euclidean distance between P_i and P_j is calculated based on the distance between the three pairs of lines in P_i and P_j.

Algorithm 3 describes the computation of the Euclidean distance between two sets of points P_i and P_j. For every line formed by two successive points in P_i (line 2), the CH searches the corresponding one in P_j. In the case that the two lines have the same length, the CH calculates directly the Euclidean distance based on the equation 5 (line 8). Otherwise, it inserts a new point along the line that has the greatest length (lines 10-13) in order to make equal the length of the two lines before calculating the Euclidean distance between them (lines 15-17). Finally, the CH calculates the sum of distances between every pair of lines in P_i and P_j (line 19).

V. Evaluation and Simulation Results

To evaluate the performance of our technique, we conducted multiple series of simulations using a custom Java based simulator. In these simulations, we used real data collected from the Argo project [22]. Argo collects data about salinity and temperature via more than 3000 sensors distributed over the oceans. In our simulations, we focus on data sensed by 180 sensors deployed in the Indian ocean over an area of $5000 \times 5000\,m$. Sensors are deployed in the upper $2000\,m$ of depth and collect periodically salinity and temperature readings. For the sake of simplicity, we are interested in this paper in one field of sensor readings: the salinity[1]. We divided the network into two clusters: CH$_1$ with 60 sensors and CH$_2$

[1] the temperature field can be processed in the same manner.

Algorithm 3 Euclidean Distance Computation Algorithm.

```
Require: $P_i = \{p_{i_1}, p_{i_2}, \dots, p_{i_{n_i}}\}$, $P_j = \{p_{j_1}, p_{j_2}, \dots, p_{j_{n_j}}\}$,
        $p_k = (x_k, y_k)$.
Ensure: Euclidean distance between $P_i$ and $P_j$: $E_d(P_i, P_j)$.
$distance = 0$
for each points $\{p_i, p_{i+1}\} \in P_i$ do
    if $\{p_j, p_{j+1}\} \in P_j$ exists such that $x_i = x_j$ and $x_{i+1} = x_{j+1}$ then
        // find equations of the two lines based on equation 4
        $(L_{i,i+1}) : y_i = a_i \times x + b_i$
        $(L_{j,j+1}) : y_j = a_j \times x + b_j$
        // calculate the Euclidean distance between lines based on equation 5
        $e_d = [E_d(L_{i,i+1}, L_{j,j+1})]^2$
    else
        consider $x_{i+1} < x_{j+1}$
        // create a new point then add it to $P_j$
        $p_{j_{new}} = \{x_{i+1}, y_j(x_{i+1})\}$
        insert $p_{j_{new}}$ just before $p_{j+1}$ in $P_j$
        // find equations of the two lines based on equation 4
        $(L_{i,i+1}) : y_i = a_i \times x + b_i$
        $(L_{j,j_{new}}) : y_j = a_j \times x + b_j$
        $e_d = [E_d(L_{i,i+1}, L_{j,j_{new}})]^2$
    end if
    $distance = distance + e_d$
end for
return $\sqrt{distance}$
```

with 120 sensors. Thus, data collected by the sensors are sent to their appropriate CHs which are located geographically at the centre of the clusters. We compare our results to those obtained with the technique proposed in [19], which we will refer as EuDi, used for UASNs. We choose to compare our works to EuDi because the two architectures are the same and because the results obtained by EuDi are well positioned in the state of the art.

In our simulations, we evaluated the performance using the following parameters:

- the period size, τ, takes the following values: 128, 256, 512 and 1024.
- the similarity between two readings, δ, takes the following values: 0.001, 0.005, 0.01 and 0.025.

A. Percentage of data sent periodically from sensors to CH

Fig. 3 shows the percentage of data sent from each sensor to its CH at each period, after transforming its raw data into a set of points. Compared to the EuDi technique, these results show that, by sending its set of points, a sensor can reduce the amount of transmitted data by 60% and up to 97% . Therefore, our technique can successfully minimize the data transmission to the CH by eliminating redundancy among sensor's raw data. Furthermore, we observe that sensor nodes send less data when δ or τ increases. This can be explained by the fact that when the similarity between collected readings increases, the number of representative points is also be reduced.

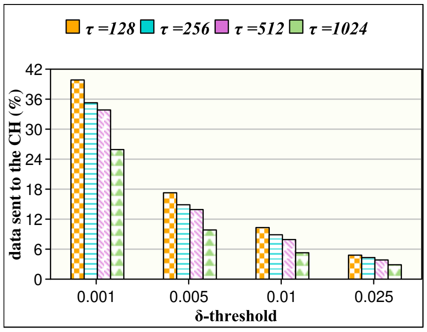


Fig. 3. Percentage of data sent periodically from each sensor to CH_1.

B. Number of iterations required to compute a set of points

Fig. 4 presents the average number of iterations required at each period to find the final set of representative points obtained in algorithm 2. It is important to recall that a high number of iterations can increase the complexity of the proposed algorithm as well as the data latency at the sensor. The obtained results show that, the number of iterations in our technique is almost less than 20, except when $\delta = 0.001$ where it exceeds 40 iterations. We think that these results are suitable for the most kinds of sensors where the parameters values should be determined by the decision makers depending on the application requirements.

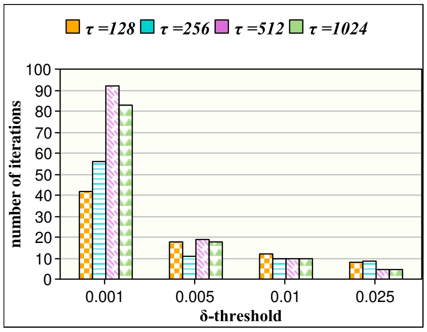


Fig. 4. Average number of iterations required to compute a set of points.

C. Number of candidates / comparisons

Fig. 5 shows the number of compared sets without applying our technique (i.e. with naïve comparisons between every pair of sets), the number of candidates generated by our technique and the results obtained after applying the Euclidean distance function (the real number of similar sets).We fixed the period size to 1024 and we varied δ as shown in the figure. We notice that, the number of comparisons in our technique is largely minimized compared to the naïve comparison. This is due to the lemma 2 which prune out the infeasible non-similar data sets and limits the number of comparisons to the candidates sets. Moreover, it is important to notice that the number of comparisons in our technique is more minimized when δ increases, thus, the number of candidates tend to match the exact number of similar sets.

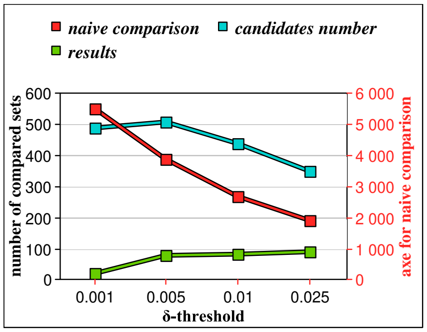
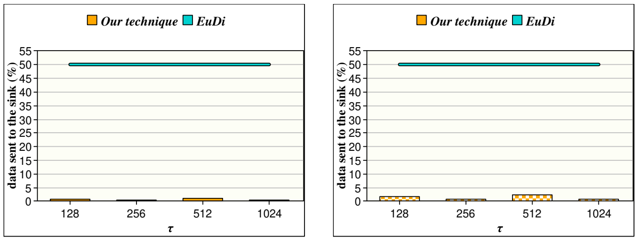


Fig. 5. Data sets comparison at CH_1, $\tau = 1024$.

D. Percentage of final data sent to the sink

Fig. 6 shows the percentage of final data sent from each cluster to the sink after aggregating data at sensor and CHs levels. We fixed the value of δ to 0.005 while we varied τ from 128 to 1024. The obtained results show that the data collected in each cluster have been largely reduced using our technique and compared to EuDi, for all values of τ. For instance, using our technique, the percentage of data sent from CH_1 and CH_2 does not exceed, in the worst case, 4% of the whole collected data. Otherwise, EuDi technique can reduce, in the best case, up to 50% the data collected in each cluster. Therefore, our technique can efficiently minimize the overload in the network and send only the useful information to the sink, without loss of the data integrity.



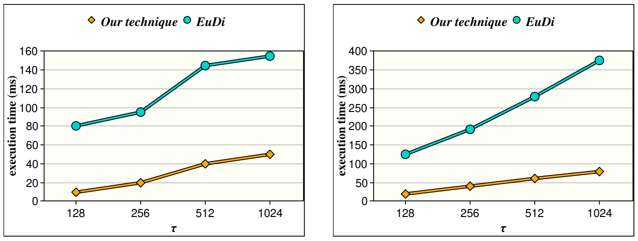
(a) CH_1

(b) CH_2

Fig. 6. Percentage of final data sent to the sink, $\delta = 0.005$.

E. Aggregation process time at the CH

In this section, we compare the execution time required for the aggregation process at the CHs level when fixing δ to 0.005 and varying the period size (Fig. 7). The test machine consisted in a standard Core i5 (1.6Ghz) laptop running Windows 7 operating system with 4Gb of RAM. Clearly this configuration appears to be more powerful than a typical CH. Nevertheless, the aim of these tests is to evaluate and compare our technique. In this way, compared to EuDi the obtained results show that our technique can accelerate the aggregation process from 3 (when the period size is big) to 20 times (when the period size is small). This is due to two reasons: first, the Euclidean distance is applied, in our technique, over the candidate pairs only while it is applied, in EuDi, over all pairs of sets; second, the Euclidean distance is calculated, in our technique, between the representative points while it is calculated, in EuDi, between all the readings of two sets. Therefore, we can deduce that our technique can minimize the data latency at the CHs and deliver data to the sink in a faster way.



(a) CH_1

(b) CH_2

Fig. 7. Aggregation process time at CHs, $\delta = 0.005$.

F. Energy consumption

In this section, we study the energy consumption of our technique at both sensors and CHs levels, for different values of δ and τ. For this, we use the energy model described in [2]. In Fig. 8, we show the energy consumption in each sensor when fixing δ and varying τ every time. The obtained results show that our technique greatly outperforms EuDi in terms of preserving the energy in the sensors in all tested cases. As we can see, the energy consumption in the sensor can be minimized, using our technique, up to 96% compared to the EuDi technique. Such energy optimization is obtained since the amount of data transmitted has been largely reduced in our technique (see Fig. 3).

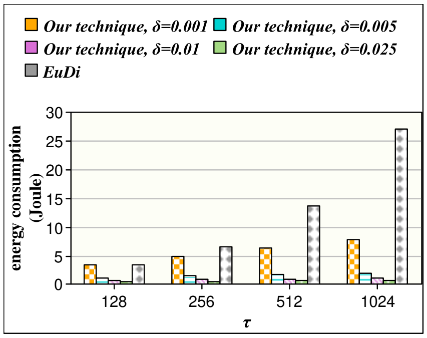
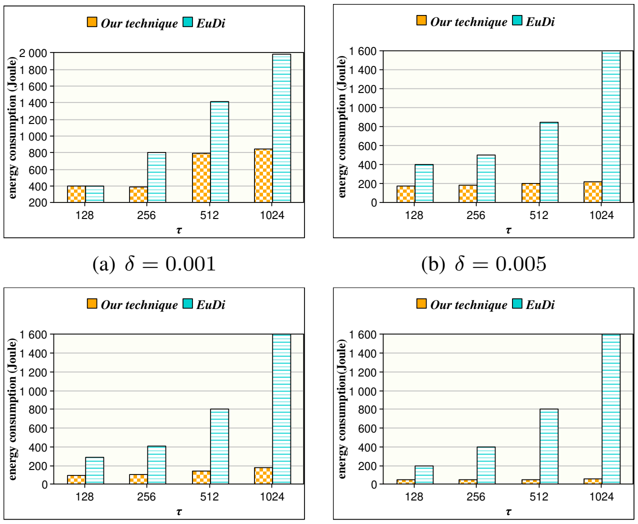


Fig. 8. Energy consumption in each sensor.

The energy consumption at the CH level, i.e. CH_2, using our technique and EuDi is presented in Fig. 9. The results are dependent on, first, the amount of data received from the sensor members (Fig. 3) and, second, the amount of final data sent to the sink after eliminating the redundancy among them (Fig. 6). We can show clearly that our technique significantly reduces, e.g. up to 91%, the energy consumption in CH_2 compared to EuDi. Therefore, our technique can be considered as a very efficient technique since the energy consumption is highly reduced in the network while the information integrity is fully preserved at the sink node.



(a) $\delta = 0.001$

(b) $\delta = 0.005$

(c) $\delta = 0.01$

(d) $\delta = 0.025$

Fig. 9. Energy consumption in CH_2.

VI. Conclusions and future work

In this paper, we have proposed an energy-efficient data reduction technique dedicated to surveillance applications. After reducing the size of the collected data, each sensor node sends a set of representative points to the CH at the end of each period. When these sets arrive to the CH, this last uses the Euclidean distance to eliminate redundant data generated by neighbouring sensor nodes, before sending it to the sink. Comparing to other existing data aggregation techniques, simulation results on real sensor data show the effectiveness of our technique in terms of energy consumption, while still keeping a high quality of the collected data. We have two major directions for our future work. First, we plan to improve the polygonisation technique to achieve the search of minimum line number corresponding to a given data curve. Thus, sensor nodes will conserve more energy and network lifetime will be extended. Second, we seek to adapt our technique to take into consideration *reactive* periodic sensor networks, where sensor nodes operate with different sampling rate.

Acknowledgement

his work has been supported by the EIPHI Graduate School (contract ”ANR-17-EURE-0002”) and partially funded with support from the Hubert Curien CEDRE programme, Project RESEAUCORP, Number: 40283YK.

References

[1] V. Raghunathan, S. Ganeriwal, and M. Srivastava, “Emerging techniques for long lived wireless sensor networks,” *IEEE Communications Magazine*, pp. 108–114, 2006.

[2] H. Harb, A. Makhoul, and R. Couturier, “An enhanced k-means and anova-based clustering approach for similarity aggregation in underwater wireless sensor networks,” *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5483–5493, 2015.

[3] N. Javaid, M. Jafri, Z. Khan, N. Alrajeh, M. Imran, and A. Vasilakos, “Chain-based communication in cylindrical underwater wireless sensor networks,” *Journal of Sensors*, vol. 15, no. 2, pp. 3625–3649, 2015.

[4] H. Harb and A. Makhoul, “Energy-efficient sensor data collection approach for industrial process monitoring,” *IEEE Trans. Industrial Informatics*, vol. 14, no. 2, pp. 661–672, 2018.

[5] H. Harb, A. Makhoul, S. Tawbi, and R. Couturier, “Comparison of different data aggregation techniques in distributed sensor networks,” *IEEE Access*, vol. 5, pp. 4250–4263, 2017.

[6] H. Harb, A. Makhoul, D. Laiymani, and A. Jaber, “A distance-based data aggregation technique for periodic sensor networks,” *ACM TOSN*, vol. 13, no. 4, pp. 32:1–32:40, 2017.

[7] B. Ali, N. Pissinou, and K. Makki, “Approximate replication of data using adaptive filters in wireless sensor networks,” *2008 3rd International Symposium on Wireless Pervasive Computing*, pp. 365–369, 2008.

[8] J. Ankur, C. Edward, and W. Yuan-Fang, “Adaptive stream resource management using kalman filters,” *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pp. 11–22, 2004.

[9] R. Abdolee and B. Champagne, “Diffusion lms algorithms for sensor networks over non-ideal inter-sensor wireless channels,” *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–6, 2011.

[10] S. Mirco, B. Klemens, and B. Erik, “Processing continuous join queries in sensor networks: A filtering approach,” *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pp. 267–278, 2010.

[11] 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.

[12] H. Lin, W. Wei, P. Zhao, X. Ma, R. Zhang, W. Liu, T. Deng, and K. Peng, “Energy-efficient compressed data aggregation in underwater acoustic sensor networks,” *Wireless Networks Journal*, pp. 1–13, 2015.

[13] D. Wang, R. Xu, and X. Hu, “Energy-efficient data aggregation scheme for underwater acoustic sensor networks,” *Proceedings of the 10th International Conference on Underwater Networks & Systems (WUWNET ’15)*, p. Article No. 44, 2015.

[14] K. Tran and S.-H. Oh, “Uwsns: A round-based clustering scheme for data redundancy resolve,” *International Journal of Distributed Sensor Networks*, vol. 2014, no. 2, pp. 1–6, 2014.

[15] X. Kui, J. Wang, S. Zhang, and J. Cao, “Energy balanced clustering data collection based on dominating set in wireless sensor networks,” *Journal of Ad Hoc & Sensor Wireless Networks*, vol. 24, no. 3-4, pp. 199–217, 2015.

[16] J. Lloret, M. Garcia, S. Sendra, and G. Lloret, “An underwater wireless group-based sensor network for marine fish farms sustainability monitoring,” *Telecommunication Systems*, vol. 60, no. 1, pp. 67–84, 2015.

[17] N. Goyal, M. Dave, and A. Verma, “Fuzzy based clustering and aggregation technique for under water wireless sensor networks,” *International Conference on Electronics and Communication Systems (ICECS)*, pp. 1–5, 2014.

[18] A. Bahi, J.and 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.

[19] K. Tran, O. Hyun, 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, p. 7 pages, 2013.

[20] H. Harb, A. Makhoul, R. Tawil, and A. Jaber, “A suffix-based enhanced technique for data aggregation in periodic sensor networks,” *10th IEEE Int. Wireless Communications and Mobile Computing Conference (IWCMC 2014)*, pp. 494–499, 2014.

[21] H. Harb, A. Makhoul, A. Laiymani, D.and Jaber, and R. Tawil, “K-means based clustering approach for data aggregation in periodic sensor networks,” *10th IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WIMOB 2014)*, pp. 434–441, 2014.

[22] ARGO, “Argo project,” *http://www.argo.ucsd.edu/index.html*, 2000.