World Wide Web (2020) 23:1197–1216
https://doi.org/10.1007/s11280-019-00767-w

Fault tolerant data transmission reduction method for wireless sensor networks

Gaby Bou Tayeh[1] · Abdallah Makhoul[1] · Jacques Demerjian[2] · Christophe Guyeux[1] · Jacques Bahi[1]

Received: 24 October 2018 / Revised: 2 September 2019 / Accepted: 5 December 2019 / Published online: 11 February 2020

Abstract
Several theoretical studies have clearly demonstrated that the Dual Prediction Mechanism (DPM) remains the most efficient technique for data reduction in Wireless Sensor Networks (WSNs). In real world, the deployed sensor nodes suffers from packet loss and even failures which renders the DPM unreliable, since it requires flawless synchronization between the source (sensor node) and the destination (Sink). In this paper, we introduce a Fault Tolerant Data Transmission Reduction (FTDTR) technique consisting of three main components: DPM-based transmission reduction, synchronization and packet loss detection, and finally reconstruction of missing data. Our method was evaluated on real-world data sets collected at our laboratory and compared to three recent prediction-based data reduction approaches. The results were promising in quality of the replicated measurements and transmission reduction.

Keywords Wireless sensor networks · Data estimation · Data reduction · Data reconstruction · Energy saving

1 Introduction

Wireless sensor network (WSN) refers to a group of spatially and largely dispersed and dedicated sensors for monitoring and recording of the physical conditions of the environment and organizing the collected data at a central location also referred to as a Sink. WSN continue to attract the attention of academia, equipment and chip manufacturing industries and services providers that can offer a variety of user-specific applications or multi-features applications using WSN.

This article belongs to the Topical Collection: *Special Issue on Smart Computing and Cyber Technology for Cyberization*
Guest Editors: Xiaokang Zhou, Flavia C. Delicato, Kevin Wang, and Runhe Huang

✉ Gaby Bou Tayeh
gaby.bou_tayeh@univ-fcomte.fr

[1] Femto-St Institute, UMR 6174 CNRS, Université de Bourgogne Franche-Comté, Besançon, France

[2] LaRRIS, Faculty of Sciences, Lebanese University, Fanar, Lebanon

However, sensor devices have a limitation in memory, energy, and processing capabilities. Therefore, several approaches have been proposed to reduce the energy consumption of these nodes. Since radio communication is the dominant factor of energy consumption in WSN, the most effective approach is the reduction of data transmission between the sink and the node. One of the most commonly used technique to reduce radio communication is the dual prediction mechanism [21, 23, 26, 32]. However, while all of these approaches have been proven to be very effective in reducing the amount of data reported to the sink, their efficiency is countered by an increase in complexity. Moreover, the proposed approaches are very sensitive to data loss which renders the dual prediction mechanism obsolete.

In this paper, we present an alternative technique that lends itself to be simple yet robust, and more effective in terms of prediction accuracy and data reduction. Our approach exploits the fact that sensor data changes smoothly over time, therefore we use the prediction model proposed in [27] to forecast future readings. We also coupled this technique with a data reconstruction algorithm [16] that exploits both temporal smoothness and spatial correlation among different sensed features in order to estimate missing values. To evaluate and compare our approach with state of the art data reduction methods, we conducted multiple experimentation on real environment data collected at the offices of our laboratory. The main contributions of this research work include:

– Extending our previously proposed data reduction algorithm by integrating a mechanism that can identify missing data and maintain a synchronized communication link between the source node and the Sink (which is mandatory for a successful implementation of the DPM technique).
– Adoption and adaptation of a data reconstruction algorithm in order to recover missing data that failed to reach the Sink.

The rest of the paper is organized as follow. In Section 3 the data transmission reduction algorithm is explained. In Section 4 the method used to identify wrong predictions is presented. Section 5 illustrates and explains the data reconstruction algorithm. The experimental results are presented in Section 6. In Section 7 the energy consumption, scalability and prediction delay of the proposed method compared to other recent works is discussed. Finally, Section 8 concludes our work and the intended future work is outlined.

2 Related work

Several approaches for a prediction-based data transmission reduction have been proposed in the literature. In the following subsections, we list and discuss the most common techniques, and a summary of the discussion is provided in Table 1.

2.1 Regression, neural networks models, and machine learning

In [11], the authors proposed to build a prediction model using kernel linear regression, where the node transmits only the model coefficient instead of transmitting real data. In [15], the authors adopted the Auto-Regressive Integrated Moving Average (ARIMA) as a forecasting model. In [5], the authors suggested combining both ARIMA and a Neural Network based prediction model into a single hybrid model to cover the limitation of ARIMA in adapting to non-linear changes. In [4], the authors proposed a hybrid model consisting of an Adaptive Neuro Fuzzy Interference System(ANFIS) and ARIMA in order to detect and predict structural damages of wind turbines.

The mathematical/statistical models become obsolete and unreliable rapidly due to the changing nature of sensor data, therefore periodic updates are needed that requires a sufficient amount of data being transmitted to the sink in order to reconstruct a new updated model. This would severely limit the efficiency of such proposals.

As for machine learning methods, the authors in [19] have proposed an IoT devices classification method to help end users choosing a device that has a sufficient computational, memory and energy resources to perform the processing tasks required by machine learning algorithms. The authors in [3] presented a survey on different and diverse machine learning methods for wireless sensor networks including methods for data reduction such as clustering, detection of spatial-temporal correlation, compressive sensing, and aggregation, etc.

However, the majority of sensor device are tiny with very limited computational and memory resources. Machine learning methods are not designed to operate on such resource-constrained devices. Therefore their usage for WSNs remains limited to few capable sensor devices.

2.2 Spatial-temporal correlation and clustering approaches

The exploitation of temporal and spatial correlation among sensor nodes for relationship detection [28] and clustering [17, 22, 29] of highly correlated nodes in order to reduce data transmission has been widely researched in the literature.

The authors in [22] proposed a semantic clustering model based on a fuzzy inference system. The latter is used to establish the relationships of the semantic neighborhood in order to group semantic neighbors into semantic clusters. The aim of this clustering technique is to reduce energy consumption and improve the quality of the data. The author in [14] proposed an algorithm that first creates multi-dimensional clusters of heterogeneous sensors based on their sensed values rather than the physical distance (clustering based on Euclidean distance). Then, an Ant Colony Optimization algorithm is proposed to select the optimal subset of sensors nodes from the formed clusters that can provide the user with the most representative information according to its requested query. The authors in [24] proposed an approach that computes the correlation among the deployed sensors nodes, it then attaches each sensor node with a companion, which is the one that correlates the most with. Once every sensor has been assigned a companion, either the sensor or the companion is set into sleep mode and its values while sleeping are predicted by its activated companion. The sleeping sensors are chosen in such a way that the leaf nodes can still route their information to the sink through the sensor hierarchy. Finally, a re-validation phase is performed after a specific period of time, where all the sleeping sensor are re-activated to update the correlation.

The approaches relying on clustering of sensor nodes or computing the spatial-temporal correlation between data reported by different nodes to set up sleep schedules are prone to problems similar to the one discussed for mathematical/statistical models. The changing nature of sensor data requires a constant update of the clusters and the relationship rules between sensors. Forming clusters and computing spatial-temporal correlation are very complex tasks in terms of computation and requires a sufficient amount of collected data to be constructed.

2.3 Adaptive sampling

In [18] the authors proposed a technique named Dual Prediction with Cubic adaptive sampling (DPCAS). They adopted an approach that merges an exponential time series predictive model with a TCP CUBIC congestion adaptive sampling technique. This approach enables the sensor to reduce the number of transmissions using the predictive model and adapt as well its sampling rate using the TCP CUBIC congestion technique. In [33], taking in to account both the system and the application context levels, the sampling rate of the sensor node is adjusted. For example, the energy available for harvesting represents the context of the system. One of the used criteria to set the node's maximum sampling rate is this availability. The application context is reflected by the user's request, where feedback from the system performing specific rules is used to optimally set sensor nodes sampling rates. In [13] in the context of Industrial Process Monitoring the authors propose three distinct data collection and adaptive sample techniques. One uses ANOVA, while the second uses a similarity sets and the third uses distance functions in order to capture similarities in the collected data and adapt the sampling rate accordingly. in [7] proposed an event-sensitive adaptive sampling and low-cost monitoring (e-Sampling) scheme. where each sensor has short and recurrent bursts of high sampling rate in addition to a low sampling rate. Depending on the analysis of the frequency content of the signal, each sensor can autonomously switch between the two sampling speed.

Adaptive sampling techniques will be effective as long as the average sampling rate of the sensor is kept at a low level during its operational lifetime. If the sampled data shows low temporal correlation, the sampling rate will be kept at a maximum level and eventually, the computational cost of the sampling rate adaptation algorithm will surpass the reduction it offers in transmission cost since such algorithms tend to be complex in general. Another disadvantage of lowering the sampling rate is the possibility of missing important information that could have been collected if the sensor was sensing data at a faster paced.

2.4 DPM based adaptive filters

The authors in [23] proposed the implementation of a dual prediction mechanism based on the Least Mean Squares (LMS) adaptive filter. However, in such models, there is a step size parameter (μ) that balances between the speed of convergence of the filter and its stability. If it is not chosen optimally, the model will lose its efficiency. Since the value of this parameter is fixed by the user, it remains uncertain if such a model can be effectively implemented in a real world deployment. Therefore, the authors in [26] proposed the usage of Hierarchical Least Mean Squares (HLMS) adaptive filter as a dual prediction mechanism, which is a multi-level LMS filter that has been proven to achieve faster and more stable convergence, since the step size parameter is re-calculated at each iteration according to the newly inputted data. Aiming to replace the step size parameter, the author in [32] proposed an approach called the Optimal Step Size LMS (OSSLMS), where μ is replaced by a few math equations that try to minimize the mean-square derivative at each iteration.

All the previously mentioned approaches [23, 26, 32] perform well in term of data reduction. However, the main drawback is that they are complex, and it remains questionable if they can be implemented on a sensor node that has a very limited computational power.

2.5 Other DPM based approaches

Other DPM based linear prediction models were also introduced. A “naive” approach was proposed in [2]. Instead of building a prediction model, the predicted value is considered to be exactly the same as the last received/transmitted one. The authors wanted to demonstrate that sometimes simple models can be as effective as complex ones. Indeed, in their experimentation, they outperformed at most of the time LMS, ARIMA, and the fixed-Weighted Moving Average (WMA) approaches. A Derivative Based Prediction (DBP) was also introduced in [21]. It’s less complex than the adaptive-filter based approaches. The goal is to find the best line interpolating a window of data of size m, by connecting the mean value of the first and last l measures in this window. Once this “optimal” slop is found, the predicted values are considered to follow the same direction as the latter.

2.6 Compression and aggregation

The authors in [8] Proposed Dynamic Message List (DMLDA) Technique Based Data Aggregation. This method is based on the technique of data clustering and provides data aggregation in real-time. A special data structure called a dynamic list is the cornerstone of this mechanism. This list is used for storing history messages before transmission in every filtering node. Therefore older messages are used instead of period delays. In [30] a rearranging algorithm which resorts the sensor nodes on the Sink is constructed using raw signal processing and signal reconstruction. This system improves signal sparseness by reducing the number of measurements necessary for reconstruction. The result is a low compression sampling rate that, in turn, reduces the irrelevant communication traffic. The authors in [6] proposed a cluster-based quality-aware adaptive data compression scheme, that takes into account the application’s data quality and it also limits information loss by using adaptive clustering and novel coding algorithm.

Compression and aggregation can effectively reduce the amount of transmitted data and they could be integrated with all the previously mentioned approaches as a complementary data reduction phase to further reduce transmission. However, the usage of such techniques becomes problematic for delay sensitive applications.

2.7 Reconstruction of missing data

All the previously proposed methods consider loss-free communication links, which is not the case in a real-world distributed system. Data loss is a major issue that can reduce the performance of these techniques dramatically. Therefore, a mechanism that can identify missing data coupled with an algorithm that can reconstruct them is mandatory for a realistic application of the dual prediction mechanism. Several methods have been proposed in the literature that exploit spatial correlation along with temporal smoothness to recover missing data in WSN [9, 10, 20, 31]. However, due to the sizes of the collected datasets that are massively large and containing multiple dimensions, these methods are extremely expensive in terms of computation complexity. Therefore, we propose an approach inspired from [16] that captures the correlations between multiple coevolving time sequences, by identifying automatically a few hidden variables from the large dataset and mining their dynamics to impute missing values with low reconstruction errors.

Table 1 A comparison between different data reduction approaches

Machine Learning and Neural Networks	- Frequent periodic model updates - Requires a computationally exhaustive training phase - Not designed to work on resources constrained sensor nodes
Spatial-temporal correlation	- Frequent periodic model updates - Requires a computationally exhaustive training phase
Adaptive sampling	- Possibility of missing sudden events/information if the sampling rate was too low.
Compression and Aggregation	- Limited efficiency - Problematic for delayed sensitive applications
DPM based approaches	- Requires a faultless synchronization between the nodes and the Sink
FTDTR	- Ensures a synchronized communication - low prediction delay - light in term of complexity - preserves the quality of the data

Non of the aforementioned data reduction approaches provides a fully efficient scheme that is able to maintain a reliable and loss free communication link between the node and the Sink. Therefore, we present in this paper a Fault Tolerant Data Transmission Reduction (FTDTR) technique that estimates data without any prior knowledge about the statistical properties of the sensed data, nor a set of global parameters that control the performance of the prediction model. It also lends itself to be light, robust, and requires a very small memory space. Moreover, we have adopted and adapted the data reconstruction method proposed in [16] and we integrated it into our transmission reduction technique through a mechanism that can identify and flag missing data. We developed our technique with several goals in mind: better prediction accuracy, less energy consumption, less computational cost, small memory footprint, and minimum prediction delay.

3 Our DPM based Transmission Reduction (TR) algorithm

Let us first describe our data transmission reduction method [27] which exploits the temporal correlation among sensed measurements in order to limit the number of radio transmissions performed by each sensor node. As mentioned earlier, both the sensor and the sink need to build and maintain simultaneously a prediction model capable of forecasting future readings within a predefined error margin. This is achieved by following a few simple steps, described hereafter.

The moment when a sensor is activated, it transmits to the Sink the first two collected measurements, x_0 and x_1 at time t_0 and t_1. Afterward, they calculate the value C_Temp that is supposed to represent the temporal correlation among the already collected and the upcoming measurements as shown in (1). Moreover, the Sink and the sensor need to store in their memories the last communicated value which is referred to as LC (in this case x_1), that will be used later to update the prediction model when it is needed.

$$C\_Temp_0 = x_1 - x_0 \tag{1}$$

Once C_Temp_0 is calculated, the sensor is no longer required to transmit to the sink any collected measurement at any time k as long as the predicted value $\hat{x}_k$ as shown in (2) is accurate.

$$\hat{x}_k = \hat{x}_{k-1} + C\_Temp_0 * \alpha \tag{2}$$

The parameter α is a rectification value that can range between 0 and 1. Its role is to harmonize the predictions with the real sensed readings. This is achieved by applying a dynamic penalty on C_Temp based on an error and a model accuracy feedback. By doing so, we aim to reduce the bias in the temporal correlation value C_Temp. A more detailed explanation on how α is automatically calculated is provided in the next subsection.

As mentioned earlier, as long as the prediction is accurate (the difference between the real sensed value and the predicted one does not exceed a user predefined error threshold $emax$), the sensor discards the real sensed value and does not transmit it to the Sink. In its turn, if the Sink does not receive any value at a given time where it is supposed to receive one, it acknowledges that the prediction outputted by the same shared model is accurate. However, if a prediction $\hat{x}_n$ at a given time n is not accurate, the sensor is required to transmit the real sensed value x_n to the Sink in order to simultaneously update the prediction model (C_Temp). This is done by subtracting LC from the current communicated reading x_n and dividing the results by the number of successful predictions N that preceded the update phase as shown in (3). Finally, LC is set equal to x_n until the next update occurs.

$$C\_Temp_n = \frac{x_n - LC}{N}. \tag{3}$$

3.1 Updating α

In this section, we will briefly explain how the rectification value α is calculated automatically. The interested reader is referred to [27] for a more detailed explanation.

During the update phase, and in order to tune α for the next prediction phase, we need to know first how did the model perform in terms of prediction accuracy and operation time. Therefore, both the sink and the node are required to calculate a value called the Accuracy Factor (AF). This is done by subtracting the value of the communicated measurement x_n from the erroneous prediction $\hat{x}_n$ that triggered the update, and dividing the result by the prediction horizon N (number of successful predictions that preceded the update). Then, a percentage value (P) of how much the value of AF is compared with $emax$ is calculated. The smaller is the absolute value of AF the more accurate was C_Temp_n, therefore, the latter must be updated relatively to the value of P. If AF is negative, this means that C_Temp_n must be increased in order to fit better the upcoming data, therefore, α is increased by $P\%$. In contrast, if AF is positive this means that C_Temp_n must be decreased by decreasing α by $P\%$. In this way, we are automatically tuning alpha according to a model accuracy feedback extracted from the last produced error and the prediction horizon. This will lead to have a more accurate model, thus a longer prediction horizon and fewer transmissions.

A few rules are required to take into consideration in order to prevent any possible deadlock when updating α : AF is very small compared to emax (e.g. 10% of emax), this means that the error is very small, and α is almost optimal. Thus, it should remain unchanged. AF exceeds $emax$, in order to prevent α from having a negative value, it should be reset to 0.5. AF remains positive for multiple successive adjustments, α could start to deviate to 0. If this is the case, α should be reset to 0.5.

The Algorithm 1 below illustrates the proposed method that is implemented on the sensor.

Algorithm 1 Transmission reduction.

```
 1: Read $x_0$ and $x_1$
 2: Transmit $x_0$ and $x_1$ to Sink
 3: $C\_Temp_0 \leftarrow x_1 - x_0$
 4: $\alpha \leftarrow 0.5$
 5: $LC \leftarrow x_1$
 6: while $Energy \neq 0$ do
 7:     Read $x_k$ at time $k$
 8:     $\hat{x}_k \leftarrow \hat{x}_{k-1} + C\_Temp_k \times \alpha$
 9:     if $|x_k - \hat{x}_k| \geq emax$ then
10:         Send $x_k$ to Sink
11:         $C\_Temp_k \leftarrow \frac{x_k - LC}{N}$
12:         $LC \leftarrow x_k$
13:         $AF \leftarrow \frac{\hat{x}_k - x_k}{N}$
14:         if $AF \geq emax$ then
15:             $\alpha \leftarrow 0.5$
16:         end if
17:         if $AF \leq \frac{10 \times emax}{100}$ then
18:             Do not update $\alpha$
19:         end if
20:         $P \leftarrow \frac{AF \times 100}{emax}$
21:         $\alpha^{new} \leftarrow \alpha - \frac{P \times \alpha}{100}$
22:         if $alpha \approx 0$ then
23:             $\alpha \leftarrow 0.5$
24:         end if
25:         $\hat{x}_k \leftarrow x_k$
26:     else
27:         $N = N + 1$;
28:     end if
29: end while
```

4 Identifying wrong predictions

Let us assume a scenario where a sensor fails to report a reading to the sink during the adaptation phase (where the prediction model update procedure is launched). The sensor will not know that the sink did not receive it, and it will use this reading to update its model. However, the sink considers that its prediction is within the error budget, therefore no update is needed. Hence, the prediction models on both sides will lose synchronization and start outputting different values. Therefore, we propose a solution that is based on an acknowledgment mechanism between the sensor and the sink. Consider that a sensor transmits a reading to the Sink, instead of switching immediately to the adaptation phase, the sensor must wait for an acknowledgment indicating that the reported value has been well received. As long as the sensor has not yet received an acknowledgment, it must keep reporting readings to the sink. This method ensures that both the sink and the node update their models simultaneously. Moreover, a sequence number is sent with each reading. If the sink detects a jump in sequence numbers, it flags the corresponding measurements as missing, which allows the reconstruction algorithm to identify and reconstruct them. Yet this is not sufficient to cover all potential failures, the batteries may deplete or the sensor could crash due to a software failure. Therefore, a sensor must operate in rounds, where each round is divided into several time slots. At the beginning of each time slot, a sensor must send the current reading even if it is within the error budget. Hence, if the sink does not receive a reading at the beginning of a given time slot, it will consider that the sensor has crashed, and all the future estimations will be replaced by a "NaN" value (flagged as missing). The Algorithm 2 is implemented on the Sink, it illustrates how the latter can detect missing values. An instance of this algorithm is initialized for each sensor node

Algorithm 2 Detecting missing data.

```
1: Receive $x_0$ and $x_1$
2: $C\_Temp_0 \leftarrow x_1 - x_0$
3: $LC \leftarrow x_1$
4: $N \leftarrow 1$
5: while $Energy \neq 0$ do
6:     if $x_k$ is received with sequence number $SN$ then
7:         Send an acknowledgment to the sensor
8:         if SN > 1 then
9:             for i=1:SN do
10:                $x_{k-i} \leftarrow$ "NaN"
11:            end for
12:        end if
13:        $C\_Temp_k \leftarrow \frac{x_k - LC}{N}$
14:        Update $\alpha$
15:        $LC \leftarrow x_k$
16:        $\hat{x}_k \leftarrow x_k$
17:    else
18:        $\hat{x_k} \leftarrow \hat{x}_{k-1} + d \times \alpha$
19:        $N \leftarrow N + 1$
20:    end if
21: end while
```

5 Reconstruction of missing data

At the end of the sensing period, all missing data (NaN values) must be reconstructed. To achieve this, we have adopted and adapted the method proposed in [16]. This method exploits both temporal smoothness and spatial correlation among data sequences in order to estimate the values of missing measurements. Let us consider a time sequence X with duration T in m dimensions, where m is equal to the number of sensors in the monitoring area. This sequence X contains all the data reproduced by the sink, and it also includes "NaN" values indicating that a reading is missing. The goal of this algorithm is to reconstruct these missing readings, by observing values of the missing sensor and other ones at neighboring time ticks.

$$X = \begin{bmatrix} x_1^1 & x_1^2 & \dots & NaN & \dots & x_1^T \\ x_2^1 & x_2^2 & \dots & x_2^T & \dots & x_2^T \\ \dots & NaN & \dots & & & \\ \dots & & & & & \\ x_m^1 & x_m^2 & \dots & NaN & \dots & x_m^T \end{bmatrix}$$

Let us denote the observed part as X_o, and the missing part as X_m. A probabilistic model (Figure 1) is built to estimate the expectation of missing values conditioned by the observed part $\mathbb{E}[X_m|X_o]$. A set of latent variables denoted Z_n are calculated using a belief propagation system. These latent variables model the dynamic and hidden patterns of the observed sequence. Moreover, the latent variables Z_n are assumed to be time-dependent with the value at time tick t is determined by the value at time tick $t-1$ using a linear mapping F. In addition, linear projection matrix G from the latent variables Z_n to the data sequence for each time tick, is assumed to represent the spatial correlation among different dimensions. Once the latent variables are calculated, they are used as input for an EM iterative algorithm [25] in order to find the best-fit parameters (such as G and F) for the data reconstruction probabilistic model.

Figure 1 illustrates this probabilistic model used to estimate missing values at a given time tick. For instance, the figure shows two missing values from two different sensor nodes at time tick 3. These values can be estimated using the linear projection matrix G and the estimated latent variable Z_3 at time tick 3. A detailed explanation of how the parameters of the model and the latent variables are calculated can be found in [16].

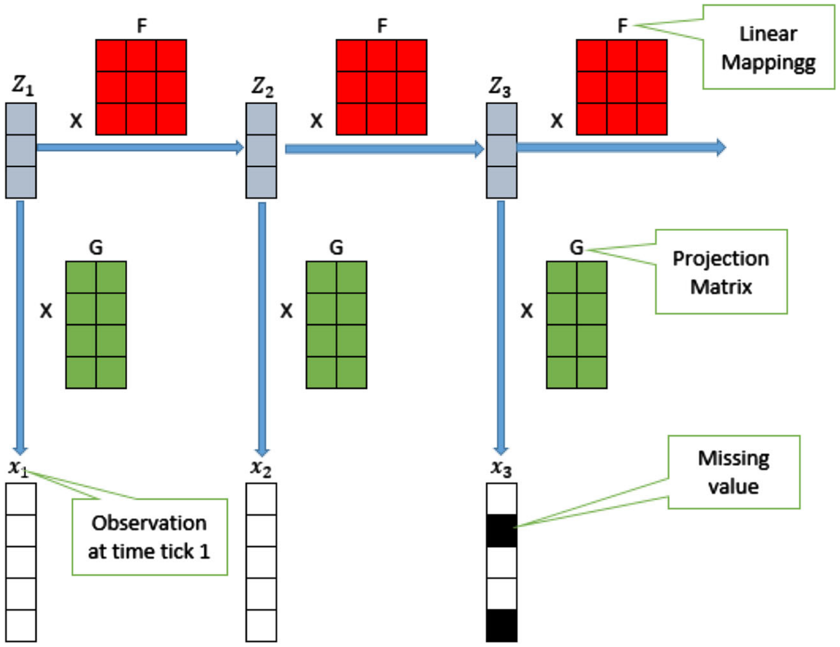


Figure 1 Probabilistic model

The size of the latent variables set can vary between 1 and m. There is an optimal value for the number of latent variables that can render the reconstruction algorithm more accurate. In [16] this number was fixed during the experiments to 15. However, in order to adapt the reconstruction algorithm to our needs we have calculated the optimal value for the latent variables using the following method:

- We first divide the collected data set into two subsets, a training subset and a validation one. The values belonging to either one of the sets are chosen randomly. The values are equally divided between the two sets and one value can only belong to one of the two sets.
- In the training subset the values that belong to validation subset are set to "NaN". For instance if x_i^j where $i \in [1, m]$ and $j \in [1, T]$ is selected to be in the validation set, the value of x_i^j in the training set will be set to "NaN"
- For each value of n ranging from 1 to m we run the reconstruction algorithm and we give it as input the training set and we set the number of latent variables equal to n. Then we calculate the Root Mean Square Error (RMSE) between the predicted values and their corresponding matches in the validation set.
- For each value of n, we select a random training and a validation set and we run the reconstruction algorithm 10 different times. Finally, we average the RMSE of the 10 runs and we chose the value of n that returned the lowest RMSE.

6 Experimental results

In this section, we present the experimentation we have conducted so as to demonstrate the efficiency and the robustness of our proposed approach compared to three state of the art linear data reduction algorithms (OSSLMS, HLMS, and DBP). Our goal is to maximize the data suppression ratio while maintaining a low computation cost and ensuring a loss-free communication link between the source node and the Sink via synchronization and data recovery.

Twenty data sets containing each 300,000 readings of temperature, humidity, and infrared data have been used for simulation. These datasets were collected by twenty Crossbow TelosB nodes that have been deployed in our laboratory. A measurement was taken every 30 seconds and transmitted to a central Sink node called SG1000 connected to a laptop machine. Figure 2, shows the geographical distribution of the sensor nodes that where deployed in our lab.

The temperature readings are smooth, where neighboring measurements vary only slightly. However, humidity readings are not as smooth, a noticeable variation can be observed, but they are still easily predictable since they follow a specific trend. Last but not least, infrared data does not follow any specific upward or downward trend and the data are basically either irregular and highly varying or stable and barely changing. These three types of data can show a weakness if existed in any of the 4 algorithms being compared in handling a specific type of upcoming readings.

6.1 Transmission reduction comparison

In this section, we evaluate the performance of our transmission reduction algorithm compared to the OSSLMS, HLMS, and DBP approaches. The simulation has been conducted on the previously described data set using a custom WSN simulator built in MATLAB [1] and

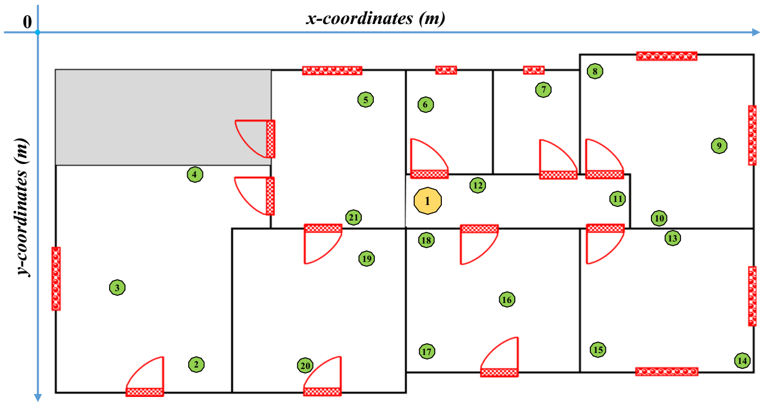


Figure 2 The geographical distribution of the sensor nodes

the following experimentation settings were used: The error threshold $emax$ was set to ± 0.1 for temperature and humidity, and ± 1 for Infrared. The number of sub-filters (m2) and the size of each one of them (m1) for the HLMS algorithm, were set to 2 and 3 respectively. The size of the OSSLMS filter was set to 5. The number of edge points l, the learning window m, the relative error for all environmental features, and the time tolerance ϵ_T, were set to 3, 6, 5%, and 2 respectively. Finally, we would like to note that the simulation was repeated 10 different times and the results presented in the this section are the average results of the 10 simulations.

The final averaged results of the 20 sensor nodes are listed in Table 2. All the proposed algorithms performed well in terms of data suppression. However, our data reduction method outperformed the other approaches in two out of three environmental features, and OSSLMS has outperformed our method in one feature.

As shown in Table 2, all of the four algorithms have approximately the same Suppression Ratio (SR) for temperature data, except for DBP that has an SR that is around "2%" lower than the others. The reason is that temperature data are very smooth and the variations in neighboring measured values are small. Thus, a linear prediction algorithm is very efficient in keeping up with these small changes. For humidity data, OSSLMS has adapted itself better and achieved the best suppression ratio of 94.1% among the three other approaches, including ours that achieved an SR of 93.6%. Finally, when we tested the algorithms on highly varying infrared data, our approach was significantly better in reducing the number of radio communication. For instance, our SR was 5.6%, 13.1%, and 29.7% greater than OSSLMS, HLMS, and DBP respectively.

Table 2 Suppression ratio of transmitted data

	Supression ratio (%)			
	TR	OSSLMS	HLMS	DBP
Temperature	**99.8**	99.7	99.6	97.7
Humidity	93.6	**94.1**	90.3	82.7
Infrared	**93.2**	87.6	80.1	63.5

Bold is used to highlight the best result

6.2 Complexity comparison

The complexity of the data reduction algorithm is as important as its efficiency. An optimal algorithm is the one who has the highest suppression ratio and the lowest complexity. Sensor nodes in WSN have limited energy resources as well as limited computation power and small memory size. Therefore, for a realistic implementation, a data reduction algorithm must respect these constraints. In this section we will compare the complexity of each algorithm by breaking down each one of them into a series of mathematical operations and counting their number at each iteration.

OSSLMS has achieved the best suppression ratio in one out of two features and was ranked second for the other two. However, despite its efficiency in suppressing transmissions, this algorithm has a very high complexity of order O(n^3). At each iteration this algorithm requires: $4N + 4N^2$ multiplications, $2N + 2N^2$ additions, N subtractions, 1 division, $2N + \frac{N^2-N}{2}$ swap operations, and $\frac{2N^3+N^2+2N^2+N}{6} - \frac{N^2-N}{2}$ operation to compute the pseudo-inverse of a matrix (N is the size of the adaptive LMS filter). HLMS ranks second in terms of complexity and it has achieved the third best suppression ratio. For HLMS the following operations are required at each iteration: $4m_2m_1$ multiplications, $2m_2m_1+3m_2+m_1$ additions, $m_2 + 1$ subtractions, and $m_2 + m_1$ divisions, where m2 represents the number of the LMS sub-filters, and m1 represents the length of each LMS filter. The complexity of this algorithm is than of order $O(n^2)$. DBP However, is linear of order O(n) when an update is needed since it requires only 2(l-1) additions, 2 divisions and 1 subtraction to readjust the model, and when no readjustment is needed, only 1 addition is required to calculate the prediction. Therefore during prediction DBP has a constant complexity of order O(1).

It seems that the higher the complexity of the algorithm, the better its efficiency. However, our method is the least complex one and achieves the best results. When no adjustment is needed for the model, 1 addition is required to calculate the prediction ($\alpha * C\_Temp$ is computed once during readjustment only). When an adjustment is needed, 2 subtractions, 4 divisions, and 3 multiplications are required to update the model. Thus our algorithm has a constant complexity of O(1).

7 Energy consumption, scalability and prediction delay

7.1 Energy consumption

Most of the energy consumed by a sensor node is generally related to four main tasks, namely, sampling, processing, and radio transmission. Therefore, in order to estimate the energy consumed by each algorithm we will be using the following model:

$$E_{node} = E_{sampling} + E_{processing} + E_{radio} \tag{4}$$

$E_{sampling}$ is the energy required to transform a physical signal into an digital one. This value is calculated using (5). Where I_{sens} is the total current for the sensing activity, T_{sens} is the total duration of the the latter, V is the supply voltage and b is the number of bits in the sensed packet when transformed to digital.

$$E_{sampling} = bVI_{sens}T_{sens} \tag{5}$$

$E_{proccessing}$ is the energy consumed by the CPU to perform a certain number of instructions. it is calculated using (6). Where N is the number of cycles required at each iteration of the algorithm, b is the the number of bits to be processed, C is the average capacitance switched per cycle, I is the leakage current, n_p is a constant that depends on the processor, V_t is the thermal voltage, and f is the sensor frequency.

$$E_{Processing} = bNCV^2 + bV(Ie\frac{V}{n_p V_t})(\frac{N}{f}) \tag{6}$$

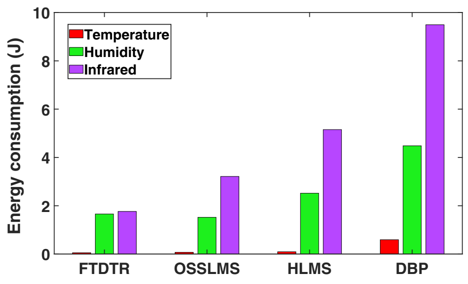
Finally E_{radio} is the energy required to transmit a b bits packet for a distance d. It could be calculated using (7). Where E_{elec} is the energy required to transmit electronics, E_{amp} is the energy consumed by the power amplifier, and n is the distance based path loss exponent (n = 2 for free space fading, and n = 4 for multi-path fading)

$$E_{radio} = bE_{elec} + bd^n E_{amp} \tag{7}$$

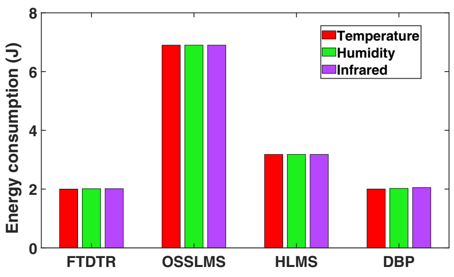
This energy model is inspired from the one discussed in [12] and all the base values used in this simulation for all the previously described parameters can be found in the cited paper.

Figure 3 shows the energy consumed by the transmission, and processing activities, in addition to the overall energy consumed by both activities combined. the sensing activity has been excluded since it is the same for all algorithms and it would not affect the results. Figure 3a shows that the energy consumed by the transmission activity is directly related to the number of data transmitted by the node. The obtained results are proportional to the suppression ratio results in Section 3.

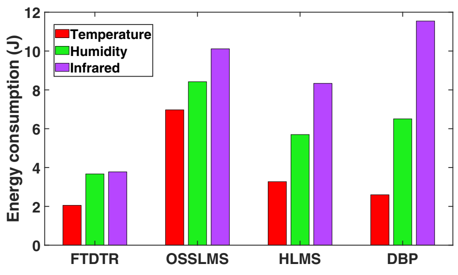
At every iteration, each algorithm performs a number of CPU cycles in order to execute the required instructions. The more complex is the algorithm the more CPU cycles are required to execute it which implies more computational energy consumption. Figure 3b shows the energy consumed by the processing activity. The results are aligned with the complexity of the algorithms discussed in the previous section. For instance, OSSLMS with a complexity of O(n^3) has consumed the most energy, followed by HLMS, DBP, and FTDTR.



(a) Transmission activity



(b) Processing activity



(c) Transmission + Processing

Figure 3 Energy consumption

If we only take into consideration the energy consumed by data transmission, it seems that OSSLMS can outperform FTDTR with humidity data and holds second place with temperature and infrared data. However, when the computational energy consumption is considered, OSSLMS falls to the fourth place behind HLMS and even DBP with temperature and humidity data. Since our algorithm has a constant complexity and it can effectively reduce data transmission, the results show that FTDTR has consumed the least energy compared with OSSLMS, HLMS and DBP.

7.2 Scalability and prediction delay

The scalability of the Dual-Prediction scheme, whether it is FTDTR, OSSLMS, HLMS, DBP or others greatly depends on two main factors. The computational power of the Sink and its memory capacity. Therefore, the more complex an algorithm is, and the more memory space it requires, the fewer nodes the Sink can handle simultaneously. The advantage that our proposal provides is that it requires a small memory footprint and it is not complex, yet, it is robust and efficient. For Instance, Figure 4 shows the time each algorithm needs to produce a prediction (Prediction delay). The shown results are the aggregate sum of 10000 predictions. As we can see, OSSLMS takes significantly more time to produce a prediction than its counterparts, this is due to its high complexity. Since the CPU of the Sink will be blocked for a longer duration in order to produce a prediction, this would affect negatively the number of nodes a sink running OSSLMS can handle simultaneously. In the case of FTDTR the prediction delay is almost negligible. Thus the CPU and the memory will be liberated rapidly which gives more space to the Sink to handle a larger number of nodes.

Eventually, Sensor nodes can be grouped into clusters and each group can be assigned to a different Sink. The number of sensors belonging to the same group is influenced by the complexity of the running algorithm and the computational capacity of the Sink. A large number of possible settings could be taken into consideration. Therefore in future work, a comprehensive study on the scalability of our proposal will be conducted.

7.3 Data loss/Communication error

In this section, we will compare the different approaches in a scenario where data loss occurs randomly when a sensor is trying to send a measurement to the Sink. The percentage

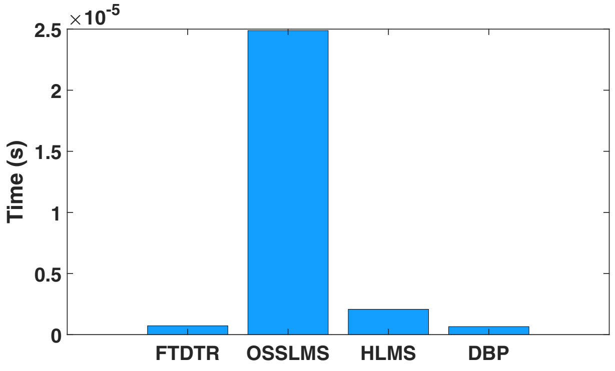


Figure 4 Prediction delay comparison

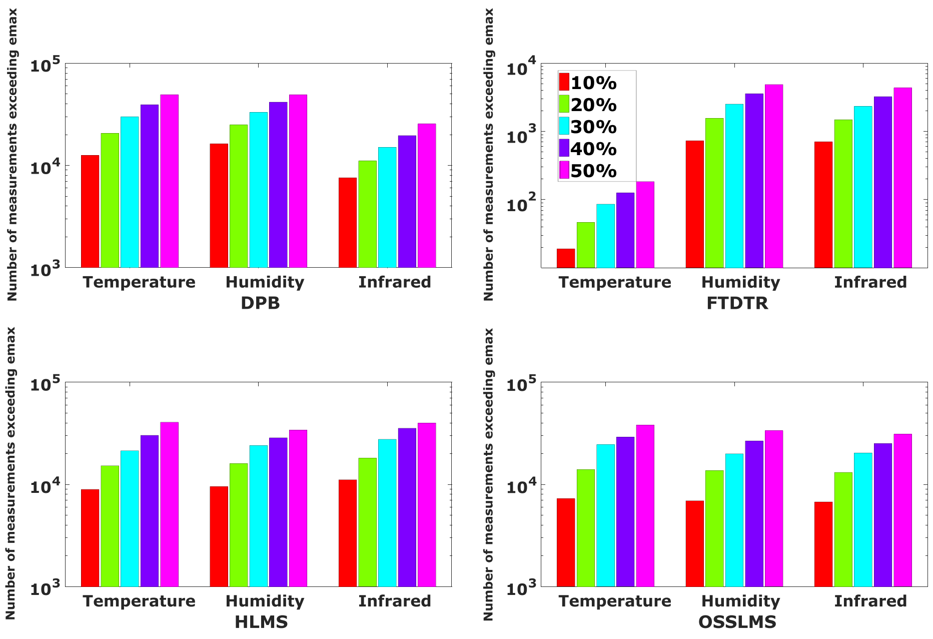


Figure 5 Number of measurements surpassing the error threshold

of a sensor failing to transmit its readings can vary from 10% in an ideal environment and when there is low collision and overload in the network, up to 50% in harsh environments and a jammed network [34, 35]. We have evaluated the performance of each of the four methods with a transmission failure possibility ranging from 10% to 50% based on the Bernoulli distribution, where the probability of failed transmission p is varied within the range of $[0.1 - 0.5]$, and the probability of a successful transmission q $(1 - p)$ within $[0.5 - 0.9]$.

Figure 5 shows the number of temperature, humidity, and infrared data exceeding the error threshold *emax* when different missing possibilities are considered. With a data loss detection mechanism, FTDTR was able to limit the number of wrong estimations by keeping

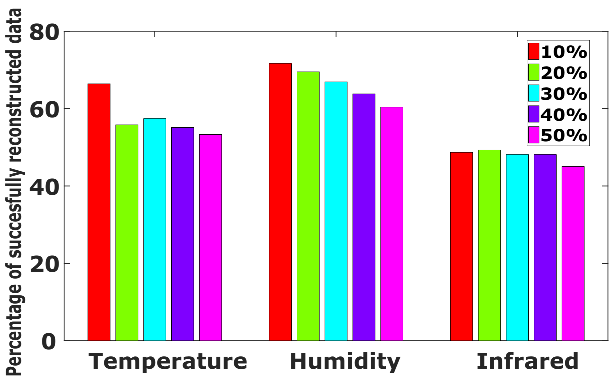


Figure 6 Percentage of successfully reconstructed data

Table 3 RMSE of data exceeding emax

	Temperature				Humidity				Infrared			
	FTDTR	OSSLMS	HLMS	DBP	FTDTR	OSSLMS	HLMS	DBP	FTDTR	OSSLMS	HLMS	DBP
10%	**0.103**	0.16	0.22	0.639	**0.13**	0.41	1.88	0.44	**2.3**	3.21	2.84	10.65
20%	**0.104**	0.188	0.238	0.849	**0.15**	0.29	2.15	0.47	**2.7**	3.39	4.39	15.52
30%	**0.104**	0.218	0.356	1.233	**0.14**	0.28	2.424	0.763	**2.8**	3.45	4.85	15.28
40%	**0.111**	0.242	0.782	1.025	**0.14**	0.312	2.937	0.834	**2.8**	4.06	5.42	18.46
50%	**0.110**	0.252	3.04	1.10	**0.16**	0.472	3.374	0.838	**3.3**	4.41	6.82	22.31

Bold is used to highlight the best result the model at the sensor and the sink synchronized. Thus, only the readings that failed to reach the sink and were flagged as “NaN” values by the later are considered to exceed the error threshold. Oppositely, the number of estimations exceeding the error threshold for other approaches is far greater than the number of measurements that failed to be reported. The reason is that the readings that fail to reach the sink are used by the sensor to adjust the model, thus leaving the sink with an outdated one that produces wrong estimations, while the sensor is producing correct ones.

For each environmental feature (Temperature, humidity, and infrared) the twenty data sets reproduced by the sink corresponding to the twenty deployed sensor node are passed to the reconstruction algorithm (DynnaMMO) in order to fill the blank values flagged as “NaN”.

Figure 6 shows the average percentage of the successfully reconstructed data. The reconstruction success rate can range between 45% and 72% according to the number of missing readings and on the temporal smoothness and spatial correlation of the data set. The reconstruction of a missing reading at time t is considered to be unsuccessful if the difference between the values of the reconstructed measurement and the real one is greater than the maximum error tolerance ($|\hat{x}_t - x_t| > emax$). For a missing probability varying from 10% to 50%, Table 3 shows the Root Mean Square Error (RMSE) of the measurements that have been unsuccessfully reconstructed by FTDTR and the data exceeding $emax$ for OSSLMS, HLMS, and DBP.

The results show that our method has the lowest RMSE for all environmental features and for all missing probabilities. Moreover, for temperature and humidity data, the RMSE’s are very close to $emax$ (0.1). Therefore, when we increased $emax$ to 0.2, the reconstruction success rate for a 50% miss probability reached 99.6% and 94.3% for temperature and humidity data respectively. For infrared data when we increased $emax$ to 4 instead of 1 (which is still an acceptable error for most applications) the reconstruction success rate increased to 91%.

The obtained results are in line with what has been previously emphasized, showing that the proposal outperforms existing works in maintaining high-quality estimations when a data loss scenario occurs.

8 Conclusion and future work

In this research work, we have demonstrated that our proposed method is better at reducing the number of transmissions from the node to the sink compared with other approaches existing in the literature. Moreover, we take into consideration communication error, links failures, and battery depletion in order to prevent the loss of synchronization between the sink and the node, by applying an appropriate mechanism that identifies and reconstruct missing data.

However, the data reconstruction algorithm is taking as input estimated values that deviate from the real readings by a margin of error, which reduces its efficiency. In future work, we aim to improve our data transmission reduction technique to decrease the root mean square error of the estimated values, in order to provide the reconstruction algorithm with more accurate information.

Acknowledgements This work is partially funded by the EIPHI Graduate School (contract “ANR-17-EURE-0002”), the France-Suisse Interreg RESponSE project, and the Lebanese University Research Program (Number: 4/6132).

References

1. Matlab simulator. https://github.com/BouTayehGaby/Matlab-Simulator- - -Fault-Tolerant-Data-Transmission-Reduction
2. Aderohunmu, F.A., Paci, G., Brunelli, D., Deng, J.D., Benini, L., Purvis, M.: An Application-Specific Forecasting Algorithm for Extending Wsn Lifetime. In: 2013 IEEE International Conference on Distributed Computing in Sensor Systems, pp. 374–381 (2013)
3. Alsheikh, M.A., Lin, S., Niyato, D., Tan, H.P.: Machine learning in wireless sensor networks: algorithms, strategies, and applications. IEEE Commun. Surv. Tutorial. **16**(4), 1996–2018 (2014)
4. Alves, M.M., Pirmez, L., Rossetto, S., Delicato, F.C., de Farias, C.M., Pires, P.F., dos Santos, I.L., Zomaya, A.Y.: Damage prediction for wind turbines using wireless sensor and actuator networks. J. Netw. Comput. Appl. **80**, 123–140 (2017)
5. Askari Moghadam, R., Keshmirpour Mehrnaz, e.M.A., Sahibuddin, S., Ahmad, R., Mohd Daud, S., El-Qawasmeh, E.: Hybrid ARIMA and Neural Network Model for Measurement Estimation in Energy-Efficient Wireless Sensor Networks, pp. 35–48. Springer, Berlin (2011)
6. Basheer, A., Sha, K.: Cluster-based quality-aware adaptive data compression for streaming data. J. Data Inf. Qual. **9**(1), 2:1–2:33 (2017)
7. Bhuiyan, M.Z.A., Wu, J., Wang, G., Wang, T., Hassan, M.M.: e-sampling: Event-sensitive autonomous adaptive sensing and low-cost monitoring in networked sensing systems. ACM Trans. Auton. Adapt. Syst. **12**(1), 1:1–1:29 (2017). https://doi.org/10.1145/2994150
8. Du, T., Qu, Z., Guo, Q., Qu, S.: A high efficient and real time data aggregation scheme for wsns. Int. J. Distrib. Sens. Netw. **11**(6), 261381 (2015)
9. Gao, Z., Cheng, W., Qiu, X., Meng, L.: A missing sensor data estimation algorithm based on temporal and spatial correlation. Int. J. Distrib. Sen. Netw., pp. 178:178–178:178 (2016)
10. Gruenwald, L., Yang, H., Sadik, M.S., Shukla, R.: Using data mining to handle missing data in multi-hop sensor network applications. Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 9–16 (2010)
11. Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., Madden, S.: Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. In: Third International Symposium on Information Processing in Sensor Networks, pp. 1–10 (2004)
12. Halgamuge, M.N., Zukerman, M., Ramamohanarao, K., Vu, H.L.: An estimation of sensor energy consumption. Progress Electromagn. Res. **12**, 259–295 (2009)
13. Harb, H., Makhoul, A.: Energy-efficient sensor data collection approach for industrial process monitoring. IEEE Trans. Indust. Inf. **14**(2), 661–672 (2018)
14. Lemos, M., Rabêlo, R., de Carvalho, C., Mendes, D., Costa, V., et al.: An energy-efficient approach to enhance virtual sensors provisioning in sensor clouds environments. Sensors **18**(3), 689 (2018)
15. Li, G., Wang, Y.: Automatic arima modeling-based data aggregation scheme in wireless sensor networks. EURASIP Journal on Wireless Communications and Networking (1), 85. https://doi.org/10.1186/1687-1499-2013-85 (2013)
16. Li, J., McCann, J., Pollard, N., Faloutsos, C.: Dynammo: Mining and summarization of coevolving sequences with missing values. ACM SIGKDD, pp. 527–534. (CMU-RI-TR-) (2009)
17. Liu, X., Liu, Y., Xie, Q., Li, L., Li, Z.: A potential-based clustering method with hierarchical optimization. World Wide Web **21**(6), 1617–1635 (2018)
18. Monteiro, L.C., Delicato, F.C., Pirmez, L., Pires, P.F., Miceli, C.: Dpcas: Data Prediction with Cubic Adaptive Sampling for Wireless Sensor Networks. In: Au, M.H.A., Castiglione, A., Choo, K.K.R., Palmieri, F., Li, K.C. (eds.) Green, Pervasive, and Cloud Computing, pp. 353–368. Springer International Publishing, Cham (2017)
19. Neto, A.R., Soares, B., Barbalho, F., Santos, L., Batista, T., Delicato, F.C., Pires, P.F.: Classifying Smart Iot Devices for Running Machine Learning Algorithms. In: 45 Seminário Integrado De Software E Hardware 2018 (SEMISH 2018), vol. 45. SBC, Porto Alegre (2018)
20. Pan, L., Gao, H., Li, J., Gao, H., Guo, X.: Ciam: an Adaptive 2-In-1 Missing Data Estimation Algorithm in Wireless Sensor Networks. In: 19Th IEEE International Conference on Networks (ICON), pp. 1–6 (2013)
21. Raza, U., Camerra, A., Murphy, A.L., Palpanas, T., Picco, G.P.: Practical data prediction for real-world wireless sensor networks. IEEE Trans. Knowl. Data Eng. **27**(8), 2231–2244 (2015)
22. Rocha, A.R., Pirmez, L., Delicato, F.C., Rico Lemos, S.antos., I., Gomes, D.G., de Souza, J.N.: Wsns clustering based on semantic neighborhood relationships. Comput. Netw. **56**(5), 1627–1645 (2012)
23. Santini, S., Römer, K.: An adaptive strategy for quality-based data reduction in wireless sensor networks. In: Proceedings of the 3rd International Conference on Networked Sensing Systems, pp. 29–36 (2006)

24. Sarkar, C., Rao, V.S., Prasad, R.V., Das, S.N., Misra, S., Vasilakos, A.: Vsf: an energy-efficient sensing framework using virtual sensors. IEEE Sens. J. **16**(12), 5046–5059 (2016)
25. Shumway, R.H., Stoffer, D.S.: An approach to time series smoothing and forecasting using the em algorithm. J. Time Ser. Anal. **3**(4), 253–264 (1982)
26. Tan, L., Wu, M.: Data reduction in wireless sensor networks: a hierarchical lms prediction approach. IEEE Sens. J. **16**(6), 1708–1715 (2016)
27. Tayeh, G.B., Makhoul, A., Demerjian, J., Laiymani, D.: A New Autonomous Data Transmission Reduction Method for Wireless Sensors Networks. In: 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM), pp. 1–6 (2018)
28. Wang, R., Ji, W., Song, B.: Durable relationship prediction and description using a large dynamic graph. World Wide Web **21**(6), 1575–1600 (2018)
29. Wen, G., Zhu, Y., Cai, Z., Zheng, W.: Self-tuning clustering for high-dimensional data. World Wide Web **21**(6), 1563–1573 (2018)
30. Wu, H., Wang, J., Suo, M., Mohapatra, P.: A holistic approach to reconstruct data in ocean sensor network using compression sensing. IEEE Access **PP**(99), 1–1 (2017)
31. Wu, H., Xian, J., Wang, J., Khandge, S., Mohapatra, P.: Missing data recovery using reconstruction in ocean wireless sensor networks. Comput. Commun. **132**, 1–9 (2018)
32. Wu, M., Tan, L., Xiong, N.: Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications. Inf. Sci. **329**(Supplement C), 800–818 (2016)
33. Yang, J., Tilak, S., Rosing, T.S.: An Interactive Context-Aware Power Management Technique for Optimizing Sensor Network Lifetime. In: SENSORNETS, pp. 69–76 (2016)
34. Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, 1-13. ACM, New York (2003)
35. Zong, C., Yang, X., Wang, B., Liu, C.: Minimal explanations of missing values by chasing acquisitional data. World Wide Web **20**(6), 1333–1362 (2017)