Adaptive Sensing Algorithm for IoT Applications with Data and Temporal Accuracy Requirements

Tayeb Lemlouma

Adaptive Sensing Algorithm for IoT Applications with Data and Temporal Accuracy Requirements

Tayeb Lemlouma
Univ Rennes, CNRS, IRISA
Lannion, France
Tayeb.Lemlouma@irisa.fr

***Abstract*—This paper proposes an adaptive sensing algorithm for long-term IoT applications. The objective is to satisfy data and temporal accuracy requirements while prolonging the lifetime of battery-powered devices with energy-hungry transmission modules. The algorithm is based on the *Send-on-Delta* (SoD) technique combined with a GM(1,1) prediction and considers a moving temporal window and outliers removal. Numerical results show the superiority of our algorithm with respect to a linear approximation. The effectiveness of the proposal is demonstrated in terms of adaptability, accuracy, and reduction of data transfer. This is of particular relevance for applications requiring long sensing periods and high sampling rate.**

***Index Terms*—adaptive sampling, prediction, grey model, sensing energy, Internet of Things (IoT), wireless sensor networks**

I. Introduction

Several IoT applications rely on battery-operated devices that regularly sense and transmit data to a central node. To prolong lifetimes of IoT applications, it is important to identify a sampling and transfer strategy that considers the accuracy requirements of sensed data within a limited temporal threshold and which is more efficient than the conventional periodic sampling method. Much effort has been devoted to sensing strategies to reduce the usage of IoT resources and extend the lifetime of devices. Sampling, data and communication compressions are the pillars of such strategies [1], [2]. Context-aware and event-based sensing are relevant to reduce the transmissions between sensors and sink nodes. These strategies only consider the situations where a change in the global context or the tracked data is detected [3], [4].

Major strategies consider the following conditions: (a) the difference between the sensed value and the last sample sent is higher than a given threshold (*Send-on-Data* or SoD strategy) [5], (b) the expected value exceeds the previously sent value by at least a threshold (*Send-on-Prediction* or SoP strategy) [1]. Two extensions of SoD are the *Send-on-Energy* and *Send-on-Area* strategies where the considered difference between current and last values are the energy or the cumulative integral of differences respectively [1], [2]. Proposed strategies intended to be performed by IoT devices, should simplify the computing operations and memory usage and consider the hardware and software limitations of these devices. Therefore, in this paper, we aim to avoid the recourse to computing-intensive models such as learning strategies with reinforcement learning, Neural and Recurrent Neural Networks [6], [7] in favour of less complex and energy-hungry ranging approach suitable for the limitations of IoT devices and networks. To accommodate a majority of use cases, we consider raw data sequences provided by IoT sensor devices without any assumption about the raw data distribution [8]. We present an extensive evaluation conducted to assess the performance of the proposed prediction, outliers removal and the benefit of temporal windows. We compare the performance of the proposed algorithm with Kulau et al. [9] in terms of accuracy and transmissions to satisfy the application requirements. We also provide insights into the minimal energy saved for different IoT technologies. To the best of our knowledge, our proposed algorithm is the first dedicated to IoT devices that comprehensively complies with accuracy and energy requirements with the presented degree of efficiency and performances.

The remainders of this paper are organised as follows. Section II discusses a Bollinger Bands based approach used to reduce nodes sampling. Section III presents the system model, followed by Section IV which describes our adaptive sensing and transfer algorithm. Section V evaluates our approach on a reference real-world dataset (generated by an IoT device that we have implemented) and provides a comparison with the Bollinger Bands approach. Section VI concludes the work.

II. Bollinger Bands-based approach: dyn_{vd}

Kulau et al. [9] uses the Bollinger Bands (BB) to reduce the sampling rate. They adapt the BB with the *vertical distances dynamic estimation*: dyn_{vd}. BB performances were already evaluated in [8], [10] and in [9] with a small real-world time series (1 day and $\approx$ 1 month) at a high sample rate (0.3Hz). For n acquired samples, $dyn_{vd}(t)$ is the mean of differences between raw values $x(i)$ and estimated values $\widehat{x}(i)$ using a linear approximation $\widehat{x}$: $dyn_{vd}(t) = \frac{k}{n} \cdot \sum_{i=0}^{n-1} |\widehat{x}(t-i) - x(t-i)|$, where k is a factor used to control the width between the upper and the lower bands of series. $\widehat{x}$ is calculated using the raw values x_{t_0} and x_{t_0-n+1} sensed by the IoT sensor device. The new determined duration between two sensed and transmitted data (i.e. the new time slot) is $\delta(t) = \frac{\delta_{max}}{1+dyn_{vd}(t)^{\varphi}}$, where δ_{max} is a fixed maximum waiting period and φ is used to weight $dyn_{vd}(t)$. Overall, when the observed data show a linear dynamics, next values can be approximated with a time slot that depends on the observed linearity.

III. System model

For the sake of clarity, we focus on a typical IoT application where a battery-powered device must periodically, every $\delta(t)$ time slot, carry out certain tasks and transmit the result to a sink node or central controller node (CC). Such a process can be the sampling of the environmental parameters relying on sensors. In addition to the temporal requirement ($\delta(t)$), the IoT application may expect an error threshold of ϵ that the values transmitted by the device should not exceed. The CC with high computation performance can control the device's activity for instance by imposing some particular settings.

We model the transmitted values of the IoT device as a time series $X^{(0)} = \{x^{(0)}(1),\ x^{(0)}(2),\ ..,\ x^{(0)}(n)\}$, where $x(t)$ represents the value sensed by the device at time t. With a conventional periodic sampling method, the data points of $X^{(0)}$ are sensed and transmitted every $\delta(t)$ time slot.

A. Prediction

Instead of requiring a systematic data transmission every new sensed values, raw data can be approximated (predicted) to lighten the tasks of the IoT device and reduce its energy and resources consumption. Hence, prolonging the lifetime of the application and avoid multiple battery replacements. We propose the adaptation of the Grey Model (GM) first-order one variable prediction, named GM(1,1), to predict future values $\widehat{x}(t)$. GM theory shows great performances in systems with uncertainty and poor or incomplete data [11]. In our context, the GM (1, 1)-based prediction is achieved as follows. Given the time series $X^{(0)}$, a new sequence $X^{(1)}$ is calculated with the accumulated generating operation (AGO): $x^{(1)}(k) = \sum_{i=1}^{k} x^{(0)}(i), \quad 1 \leq k \leq n$. The generated mean sequence $Z^{(1)}$ is then derived from $X^{(1)}$: $z^{(1)}(k) = \frac{1}{2}(x^{(1)}(k)+x^{(1)}(k-1))$, ($2 \leq k \leq n$). The first order differential equation of GM (1, 1) is defined by: $x^{(0)}(k) + az^{(1)}(k) = b$, ($2 \leq k \leq n$). Thus, the whitening equation is:

$$\frac{dx^{(1)}}{dt} + ax^{(1)} = b \tag{1}$$

Then, a and b parameters can be estimated with : $[a, b]^T = \left(B^T B\right)^{-1} B^T Y$, where:

$$Y = \begin{bmatrix} x^{(0)}(2) \\ x^{(0)}(3) \\ \vdots \\ x^{(0)}(n) \end{bmatrix}, B = \begin{bmatrix} -z^{(1)}(2) & 1 \\ -z^{(1)}(3) & 1 \\ \vdots & \vdots \\ -z^{(1)}(n) & 1 \end{bmatrix}$$

According to (1), $X^{(1)}$ at time k is:

$$\widehat{x}^{(1)}(k+1) = \left[x^{(0)}(1) - \frac{b}{a}\right]e^{-ak} + \frac{b}{a} \tag{2}$$

Consequently, to determine the next predicted value of X^0 at time $(k+1)$, we can use:

$$\widehat{x}^{(0)}(k+1) = \left[x^{(0)}(1) - \frac{b}{a}\right]e^{-ak}(1 - e^{a}) \tag{3}$$

B. Outliers Removal

The prediction of a next value $\widehat{x}^0(t)$ depends on previous values of X^0. Although the GM prediction smoothes the randomness using AGO, it is important to remove outliers from initial data especially with a small size of X^0 and during the lifetime of the IoT application. To be adapted to the characteristics of IoT devices, we perform the outliers calculation using the Welford's online algorithm and Chebyshev inequality [12], [13]. We accordingly set the non-outliers range to [$\bar{X}^0$ - ($k \cdot \sigma$), $\bar{X}^0$ + ($k \cdot \sigma$)] with k=5 and σ is the standard deviation which is computed incrementally as follows:

$$\bar{X}_n = \frac{(n-1)\cdot \bar{X}_{n-1} + x(n)}{n} = \bar{X}_{n-1} + \frac{x(n) - \bar{X}_{n-1}}{n} \tag{4}$$

$\bar{X}_n$ is the mean of X^0 calculated each time a new data $x(n)$ becomes available, $\bar{X}_{n-1}$ is the previous mean of X^0. The value of σ is then calculated with: $\sigma_n^2 = \frac{S_n}{n}$, where S_n is recursively computed: S_n = S_{n-1} + M_n with M_n = $(x(n) - \bar{X}_{n-1})\cdot(x(n) - \bar{X}_n)$. The k value depends on the concentration of data around the mean of X^0. According to the Chebyshev's inequality, if X is a random variable with finite expected value μ and a variance σ^2, then $P(|X - \mu| \geq k \cdot \sigma) \leq \frac{1}{k^2}$. With k=5, such probability is lower or equal to 0.04.

IV. Proposed algorithm

Our sampling and transfert strategy for IoT devices and CC nodes is depicted in algorithms 1, 2 and 3. We use the following notation and requirements:
δ_{tmax} : the device is required to sense and return a value at most every δ_{tmax} time slot *Requirement* (1).
ϵ : the error of each returned value must not exceed ϵ if compared to the raw value *Requirement* (2).
w: the size of the prediction temporal window W which is the input time series used in the prediction and outliers removal.

Algorithm 1: Helper Function

Input: (W, $mode$, w_0) ≡ current temporal window, current sampling mode, size of W
Output: Sampling mode

```
function selectSamplingMode (W, mode)
    if |W| = w0 then
        if mode = SoD then
            mode ← prediction;
            W_p ← W;  ▷ start the prediction with a copie of W
        else
            mode ← SoD;
            W_p ← ∅
        end
        W ← ∅
    end
end function
```

Unlike algorithm 2 which is performed uninterruptedly every δ_{tmax} by the CC, algorithm 3 is carried out by the

Algorithm 2: Central Controller

Input: (ϵ, δ_{tmax}) $\equiv$ data accuracy and temporal threshold requirements

Output: Adaptive sensing

1: $mode \leftarrow SoD$;
2: $w \leftarrow w_0$; ▷ w_0: size of the prediction window
3: **if** $(bat(t)$ *or* $\widehat{bat}(t)) \leq \delta_{bat}$ **then**
4: Return bat_alert
5: **else**
6: selectSamplingMode (W, $mode$, w_0);
7: **if** *mode* = *prediction* **then**
8: $W_p \leftarrow outlier\_removal(W_p)$;
$\widehat{x}(t) \leftarrow predicted\_value(W_p)$;
9: $W_p \leftarrow W_p \frown \widehat{x}(t)$; ▷ append $\widehat{x}(t)$
10: Return $\widehat{x}_(t)$;
11: **else**
12: Return $x_{lastReceived}$;
13: **end**
14: **end**
15: **upon** *reception of* $M \equiv (x(t), bat(t))$*:*
16: send (ϵ, $\delta_{tmax}, mode, w$);
17: **if** *timestamp(M(x(t))) - 1 = timestamp*($x_{lastReceived}$) **then**
18: $W \leftarrow W \frown x(t)$; ▷ append $x(t)$ to W, $|W| \leq w$
19: **else**
20: $W \leftarrow x(t)$;
21: **end**
22: **if** *mode = prediction* **then**
23: $W_p \leftarrow W_p \frown x(t)$; ▷ consider the received $x(t)$ in the prediction, $|W_p| \leq w$
24: **end**
25: $x_{lastReceived} \leftarrow \widehat{x}(t)$;
26: update (W, $bat(t)$);
27: Return $x_{lastReceived}$;
28: **end**

IoT device every time it wakes up before sleeping (line 32). When the sleeping time is up, the device performs the algorithm again and so on. In the first active cycle and before the first communication with the CC, the variables are not initialized. For instance, the mode (*prediction* or SoD) is not initialized. Line 12 (algorithm 3) guarantees a default SoD mode. Similarly, line 4 guarantees a default ϵ before the first setting received from the CC. Line 5 transmits the first sensed value. Lines 2 in algorithm 3 and 3 in algorithm 2 are used by the CC to distinguish a flat battery from the absence of data due to the data transfer reduction. The message mentioned in line 17 (algorithm 3) is received from the CC (line 16 in algorithm 2) as an answer to the *transmit* message (line 16, algorithm 3). In algorithm 3, the received δ_{tmax} is updated and used in the current active cycle.

Lines 2–11 in algorithm 1 allow determining the sensing mode to be applied by the IoT device and the CC. The idea is while the variation of raw data is within a tolerated error threshold of ϵ (i.e. $x(t) - x_{lastSent} \leq \epsilon$), it is better to keep the SoD mode and avoid the prediction calculations and unnecessary transmissions from the IoT device to the CC. However, when there is a significant variation in raw values, the prediction becomes more interesting. Moreover, when there are many outliers, the last transmitted value ($x_{lastSent}$) can be affected and being itself outlier. Hence, SoD becomes inefficient. Note that in this case, considering an older value (i.e. a value prior to the encountered outlier) as $x_{lastSent}$ is risky for the required accuracy.

With a high fluctuations of data during a temporal window W, the sampling and transfer is based on the prediction while the new prediction is within the ϵ threshold. When the prediction deviates too far from real data during W, the sampling mode becomes SoD. This is an adaptive method to the fluctuation of raw data. It is a flexible way suitable for the tracked values dynamics. Note that the moving window W includes the latest consecutive values with a size of w_0 (Lines 17–24 in algorithm 2). Sensed values with a variation higher than ϵ are sent with line 16 (algorithm 3) to continuously follow the data evolution and adapt the sampling mode. The same sampling mode (SoD or prediction) is applied by both the device and the CC. This is important because it ensures that the returned values by the CC are the same as the sensor node. As there is no systematic communication between the device and the CC, lines 19, 28 and 30 (algorithm 3) ensure a persistent storage available in next wake-up cycles of the IoT device. Lines 15–32 in algorithm 3 ensure that the CC returns sensed values that satisfy requirements of ϵ and δ_{tmax}.

V. Performance Evaluations

At data point t, let A_t be the real data that can be sensed by the IoT device and F_t the forecast value. We consider the following metrics in the evaluation of our approach:

- The percentage error (PE or FE): $pe_t = 100 \cdot (A_t - F_t)/A_t$
- The Mean Forecast Error (MFE): MFE= $\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)$, ($n$: the size of the original data)
- The Mean Absolute Deviation (MAD or Mean Absolute Error –MAE): MAD= $\frac{1}{n}\sum_{t=1}^{n}|A_t - F_t|$
- The mean absolute percentage error (MAPE): MAPE = $\frac{1}{n}\sum_{t=1}^{n}|pe_t|$
- The mean absolute percentage forecast accuracy (FA): FA = $\frac{1}{n}\sum_{t=1}^{n}100 - |pe_t|$ (or 100−MAPE)
- The mean absolute scaled error (MASE) for j predictions: MASE = $\frac{1}{j}\sum_{t=1}^{j}|\frac{A_t - F_t}{\frac{1}{n-1}\sum_{i=2}^{n}|A_i - A_{i-1}|}|$

To validate our approach, we use the data-trace illustrated in Fig. 1 as a reference for the prediction. Data represent the sensed values (outdoor temperature) as transmitted by an IoT device implemented using a WiFi-enabled ESP8266-12E board with a DS18B20 digital temperature sensor. A step-by-step implementation is available online on http://1do.me/Uk.

A. Prediction with outliers removal and temporal window

We evaluate our prediction, with and without outliers removal, during the three periods P1, P2, and P3 of the reference

Algorithm 3: IoT sensor node

Input: $(\epsilon, \delta_{tmax}, mode, w) \equiv$ data accuracy, temporal threshold, sensing mode, temporal window

Output: Adaptive sensing

$x(t) \leftarrow sense\_value(t)$;
$bat(t) \leftarrow sense\_battery(t)$;
if $\epsilon = null$ **then**
  $\epsilon \leftarrow default\_value$;
  transmit $(x(t), bat(t))$;
end
selectSamplingMode $(W, mode, w_0)$;
if *mode* $=$ *prediction* **then**
  $W_p \leftarrow outlier\_removal(W_p)$;
  $\widehat{x}(t) \leftarrow predicted\_value(W_p)$;
  $W_p \leftarrow W_p \frown \widehat{x}(t)$; ; ▷ append $\widehat{x}(t)$
  $\Delta \leftarrow |x(t) - \widehat{x}(t)|$;
else
  $\Delta \leftarrow |x(t) - x_{lastSent}|$;
end
if $\Delta > \epsilon$ **then**
  transmit $(x(t), bat(t))$;
  receive $(\epsilon, \delta_{tmax}, mode, w)$;
  update (δ_{tmax});
  save $(\epsilon, \delta_{tmax}, mode, w)$;
  if *timestamp(x(t)) - 1 = timestamp($x_{lastReceived}$)* **then**
    $W \leftarrow W \frown x(t)$; ▷ $|W| \leq w$
  else
    $W \leftarrow x(t)$
  end
  if *mode = prediction* **then**
    $W_p \leftarrow W_p \frown x(t)$;
  end
  save (W);
  $x_{lastSent} \leftarrow x(t)$;
  save $(x_{lastSent})$;
end
sleep (δ_{tmax});

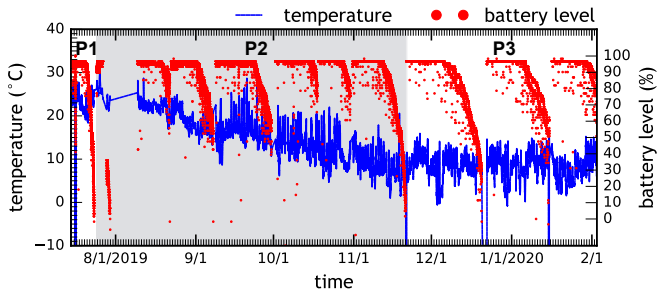


Fig. 1: Reference data: outdoor temperature in °C and battery level, ≈ 7 months of measurement (from July 14, 2019 to Feb. 2, 2020), 49463 samples with 3 continuous periods (P1, P2 and P3) at the following sampling/transfer rates: 0.0167 Hz (i.e. every 60s), 0.0033 Hz, and 0.0017 Hz respectively.

TABLE I: Algorithm's prediction with and without outliers removal (sampling rate: 0.0167 Hz, fixed window W=$|X^{(0)}|$

Prediction algorithm	MAPE	MASE
Prediction without O.R.	6.28%	5.43
Prediction with O.R.	5.98%	4.81

data with different sampling and transfer rates. As observed in Fig. 2 and 3, the predicted values are tracking favourably the trend of real data during P1 and P2. This observation is confirmed by the low values of the MAPE and MASE metrics (Table I and II) with a better performance when applying outliers removal (O.R.), e.g. in P1, 5.98% of MAPE with O.R. against 6.28% without O.R. Of course, this depends on the number of outliers present in the tracked data. As expected, the prediction shows better results regarding the accuracy (FA) and the mean error (MFE and MAD) when the sample rate is low (e.g. accuracy of 94,02% at a sample rate of 0.016Hz against 81.31% at a sample rate of 0.0033Hz).

The bad prediction accuracy observed in P3 (-18,31% of FA) can be explained by the strong variability of real values due to the low sampling rate. Another reason is the low values of sensed data in P3 (remember that the PE is a percentage of the real value). This is confirmed by the high value of MAD that accumulates the mean absolute value of the prediction error. Indeed, in P3, the MAD is 2,48 while during a comparable period in term of duration (i.e. P1) the MAD is much smaller with a value of 1,44.

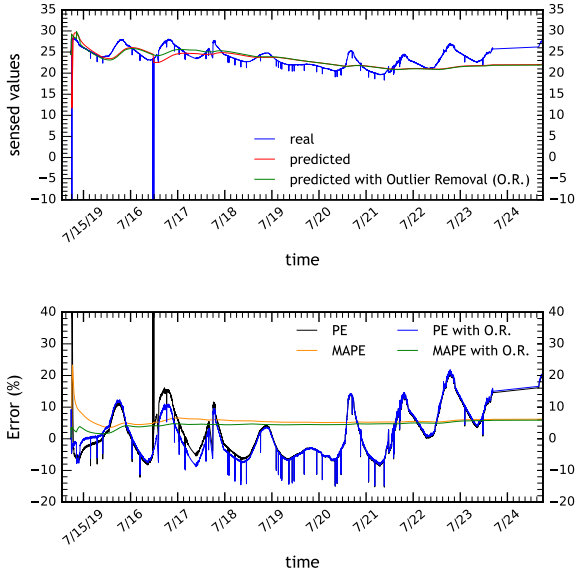


Fig. 2: Prediction evaluation (instant PE and cumulative MAPE) with and without outliers removal on data during P1 (9777 samples, sample rate: 0.0167 Hz).

For a better match between predicted and real values and to mitigate the effects of data variability, we consider a moving prediction window W and evaluate the resulting performance (Fig. 4). Only the last values sensed within the size of W

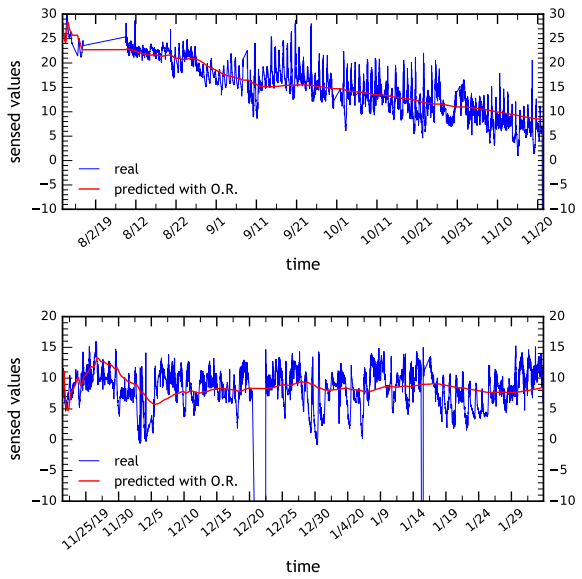


Fig. 3: Predictions during P2 (0.0033Hz) and P3 (0.0017Hz), $|X^{(0)}| \in \{29233, 10455\}$ samples.

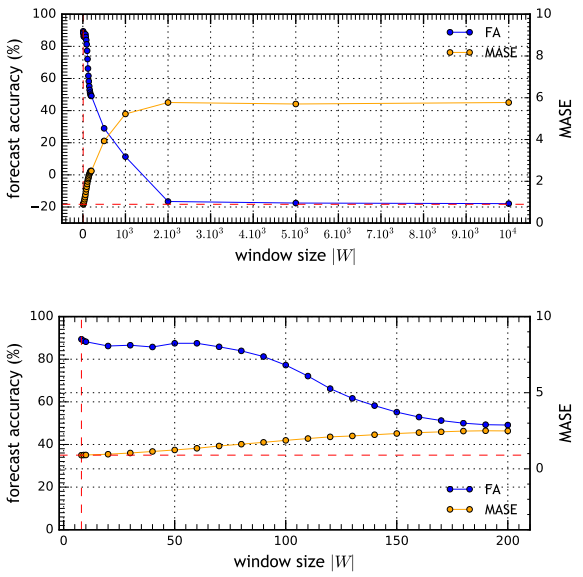


Fig. 4: Predictions evaluation (FA and MASE) with outliers removal and variable prediction window $|W|$={10, 20, .., 200, 500, 1000, 2000, 5000, 10.000} during P3 (10455 samples).

TABLE II: Algorithm's prediction with O.R. and W=$|X^{(0)}|$

Sample rate	FA	MFE	MAD	MASE
0.0167Hz (Period 1)	94.02%	0.48	1.44	4.81
0.0033Hz (Period 2)	81.03%	0.02	1.95	8.16
0.0017Hz (Period 3)	-18.31%	0.05	2.48	5.85

are considered in the prediction (Fig. 4). Table III shows the significant improvement of the prediction in terms of FA and MASE during P3 when using a window size $|W|$=8. For the same period, the good match between predicted and real values is observed in Fig. 5. Table IV shows a further performance improvement in P1/P2/P3 when compared to the prediction without the moving window (Table II). For instance, with $|W|$=8, the accuracy in P1 surges from 94.02% to 99.68%.

B. Comparison of the algorithm performance and dyn_{vd}

Unlike our approach which provides an easy way to satisfy the requirements of data accuracy with a temporal threshold,

TABLE III: Prediction with O.R. and variable window (P3).

$\|W\|$	FA	MASE	$\|W\|$	FA	MASE	$\|W\|$	FA	MASE
8	90.95%	0.96	9	88.61%	0.91	10	88.19%	0.91
20	86.19%	0.96	30	86.56%	1.04	40	85.73%	1.14
50	87.47%	1.24	100	77.24%	1.87	500	28.96%	3.93
			1000	11.32%	5.22			

TABLE IV: Predictions with O.R. and $|W|$=8 (P1, P2, P3).

Period	$\|W\|$	W duration (s)	FA	MAD	MASE
1	8	480	99.68%	0.18	0.59
2	8	2400	98.18%	0.24	0.99
3	8	4800	90.95%	0.41	0.96

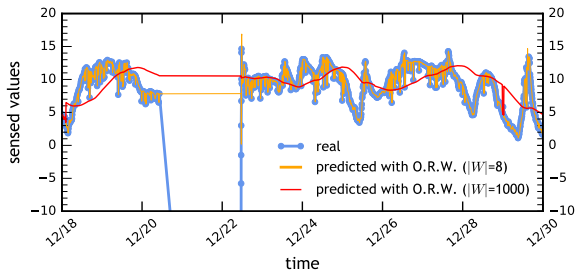
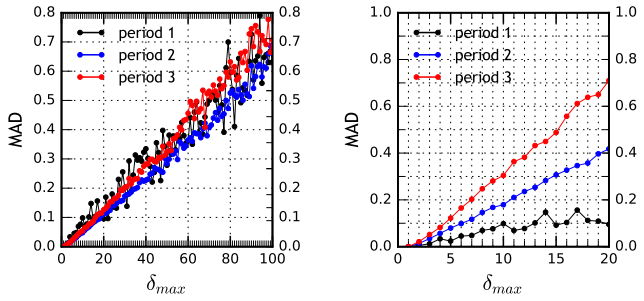


Fig. 5: An extract (Dec. 18, 2019 – Dec. 30, 2019) from the prediction applied during P3 with O.R and $|W|$=8.

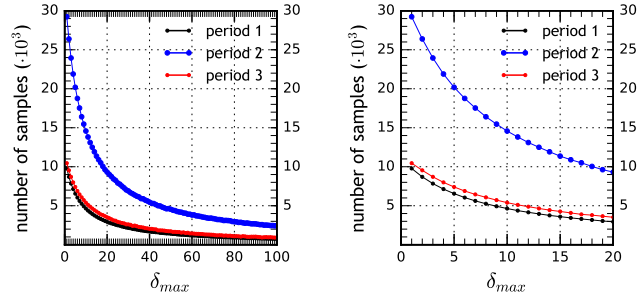
i.e. (ϵ, δ_{tmax}), it is not obvious with the dyn_{vd} approach (Section II) to find the optimal setting that guarantees the (ϵ, δ_{tmax}) desired requirements. To ensure a fair comparison, we evaluate dyn_{vd} within a large dataset (periods 1–3) by varying the parameters of the estimation function and identify the best setting of dyn_{vd} so that we can compare it with our approach (Fig. 6). As stated in [9], the variation of the buffer size n used in dyn_{vd} has no significant impact on the error $\epsilon = |\widehat{x}(t) - x(t)|$. Therefore, we set n to 8 to provided the best performances of dyn_{vd} as evaluated in [9]. This selected setting allows to use the same storage space of the IoT device as in our approach (in our prediction $|W|$=8).

To be able to perform the performances evaluation with a reference real-world dataset and evaluate the forecast accuracy of dyn_{vd}, the time slot $\delta(t)$ used in dyn_{vd} must be at least the value of the required temporal threshold (δ_{tmax}). This will allows the IoT device to sense or linearly approximate the data each δ_{tmax} time slot. This implies:



(a) MAD ($\delta_{max} \in [1..100]$ slots)

(b) MAD ($\delta_{max} \in [1..20]$ slots)



(c) Samples ($\delta_{max} \in [1..100]$ slots)

(d) Samples ($\delta_{max} \in [1..20]$ slots)

Fig. 6: Mean absolute deviation error and samples transmissions of dyn_{vd} (with n=8, k=1, φ=2) according to δ_{max}.

$$\delta(t) = \frac{\delta_{max}}{1 + dyn_{vd}(t)^{\varphi}} \geq \delta_{tmax} \quad (5)$$

According to (5) and with a perfect linearity of the observed data (i.e. $dyn_{vd}(t)^{\varphi}$=0), $\delta(t)$ is the same as δ_{max}. Conversely, when data are not linear, $dyn_{vd}(t)^{\varphi}$ grows significantly and $\delta(t)$ approaches 0. In the latter case, dyn_{vd} can not be evaluated within a reference data except if $\delta(t)$ is greater than or equal to the original time slot of the reference data. All in sum, the following requirement must be satisfied: $\delta_{max} \geq \delta_{tmax}$ and $\delta(t) \geq \delta_{reference\_data}$. When the new time slot $\delta(t)$ is exactly equal to the required time slot δ_{tmax}, the data is sensed and transmitted to the CC, hence no approximation is applied. When $\delta(t)$ is greater than δ_{tmax}, all the required values at times $t = i \cdot \delta_{tmax} \leq \delta(t)$, where i is an integer value, are approximated using the dyn_{vd}. If there are plenty of approximated values, the number of transmissions is reduced but the risk of errors increases. Note that with dyn_{vd}, to update the linear approximation function and identify the next time slot $\delta(t)$, all the values of the buffer used in dyn_{vd} (i.e. the n values) are sensed. Since these values can not be approximated, they must be transmitted every time to the CC.

The evaluation illustrated in Table IV shows the good quality of the proposed prediction. However, the FA metric does not guarantee the satisfaction of *Requirement* (2) with a reduced number of transmitted values. For this reason, in Fig. 7, the satisfaction of *Requirement* (2) is evaluated according to the required ϵ and the number of transmissions that ensures the satisfaction of the ϵ by all the returned values. Our combination of the prediction and SoD avoids unnecessary calculations related to the prediction. Requirement (1) is satisfied by the algorithm's design (line 32 in algorithm 3). In Fig. 7, the performance of our algorithm is compared with the best setting of the dyn_{vd}. Indeed, for each new period, we select a new δ_{max} that satisfies the ϵ requirement with the smallest number of transmissions, i.e. the higest value of δ_{max} (Fig. 6.b and 6.d). Thanks to the performed evaluation illustrated in Fig. 6, the selected δ_{max} enables dyn_{vd} to either sense or approximate data every δ_{tmax}. While the proposed algorithm ensures the error of returned values will never exceed ϵ (lines 15–31 in algorithm 3), the MAD measurement of dyn_{vd} (Fig. 6) indicates that the overall mean error is less than ϵ. However, some values can violate *Requirement* (2).

The results of Fig. 7 demonstrate the superiority of our approach. For the same requirement, our approach outperforms dyn_{vd} by requiring a smaller number of transmissions from the IoT device to the CC whatever the value of ϵ. Moreover, with the input (ϵ, δ_{tmax}) requirements, the same adaptive algorithm is applied for different periods without a need to adjust the setting to face the fluctuation of tracked data. The important number of transmissions is observed during P2. This is explained by the fact that P2 is the longest period with an important number of samples: 29233. Although P1 and P3 have approximately the same number of samples, we observe that for the same value of ϵ, P3 requires more sampling than P1. This is due to the fluctuation of raw data in P3 (Fig. 3) and the small sampling rate (0.0017Hz).

Table V confirms the best performance of the proposed algorithm in comparison to the dyn_{vd} strategy. While ensuring the desired precision for all the returned values, the algorithm succeeds to provide a high reduction of the number of transmissions if compared to the periodic sampling and transfer method which makes the proposed approach very interesting in terms of bandwidth and energy-saving for IoT sensor devices. For instance, with a guaranteed error threshold of 0.5, the algorithm saves 84.29% of the IoT device transmissions. This performance becomes more beneficial for energy-efficient communications, particularly with unreliable network links. Indeed, in networks with a high network packet error rate (PER), saving only one transmission prevents additional retransmissions for all the related packets (application data and network control), thus the consumption of energy is further reduced.

C. Energy saving

With the same assumptions of the realistic energy comparison detailed in [14] without PER and based on the energy consumption model of [15], Table VI summarizes the energy saving of our algorithm during periods 1–3 for different technologies and a device operating at 3.3V with a typical current consumption of 70mA in normal operation mode.

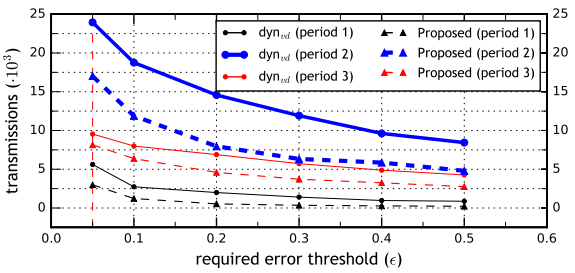


Fig. 7: Required transmissions with dyn_{vd} (with different δ_{max}) and the proposed algorithm according to the error requirement $\epsilon \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ during P1 (δ_{tmax}=60 s), P2 (δ_{tmax}=300 s), and P3 (δ_{tmax}=600 s).

TABLE V: Transfer reduction by the proposed algorithm and dyn_{vd} with a fixed δ_{max} within P1, P2 and P3.

	dyn$_{vd}$		**proposed algorithm**	
ϵ	# samples[a] (with δ_{max}[b])	reduction	# samples[a]	reduction
0.05	44632 (2)	9.77 %	28148	43.09 %
0.1	37019 (4)	25.16 %	19489	60.60 %
0.2	31695 (6)	35.92 %	13081	73.55 %
0.3	26112 (9)	47.21 %	10393	78.99 %
0.4	22138 (12)	55.24 %	9386	81.02 %
0.5	19224 (15)	61.13 %	7771	84.29 %

[a]the number of transmitted samples by the approach.
[b]the best possible (i.e. highest) value of δ_{max} satisfying ϵ for the 3 periods.

Note that the saved amount of energy is undervalued and simplified since we only consider the transmission reduction of small application packets (a minimal packet structure without network control and security overhead). Furthermore, we only consider the *Tx/Rx* states of the IoT device (without *Idle* and *Sleep* states) with known power consumption patterns and timing constraints [14]. In real-world usages, the expected amount of saved energy is much higher. As depicted in Table V, it can be easily inferred that the percentage of energy-saving is at least 84.29% of the energy required by the sensor to transfer data using the conventional periodic sampling.

TABLE VI: Energy saving with a required ϵ of 0.5, sample rates: 60 s (P1), 300 s (P2), 600 s (P3).

Technology	**P$_{TX}$/P$_{RX}$ Min-Energy (mW)**	**Data rate**	**Min-Energy Saving (J)**[a]
Bluetooth Low Energy (BLE)	24.11 / 19.26	2 Mb/s	842.80
802.15.4	24.11 / 19.26	250 kb/s	842.80
SIGFOX	147 / 39	1000 b/s	1080.66
LoRa Classe A	419.6 / 44.06	11 kb/s	1543.71

[a]Energy = $\sum_{s=Tx/Rx} P_s \cdot t_s$, P_s: power consumption in state s, t_s: time spent in states Tx/Rx [14] [15]. Here, $t_s = \sum_{i=1}^{n} t_i$, where t_i is the time required to transmit one acknowledged packet, and n= (size of one sample)/(max data packet size). n set at one (i.e. data without fragmentation) and $t_{TX/RX}$ at 40ms for one data sample.

VI. Conclusion

In this paper, we proposed a new adaptive algorithm for reducing the sample transfer rate of IoT sensor nodes. We combined a GM(1,1) based prediction with outliers removal and moving temporal window to guarantee data and temporal accuracy. The results of this study reveal that relying exclusively on prediction in sampling and transfer reduction for IoT sensor devices can not guarantee the satisfaction of application requirements in terms of maximal error threshold and temporal accuracy. For this reason, we believe that without any assumption on the distribution of tracked data, the proposed combination of SoD and prediction will give, for several real-world use cases, the best results regarding the reduction of transfer rate, energy-saving and strict satisfaction of accuracy requirements. This is beneficial for long-term IoT application especially with devices using energy-hungry transmission modules and requiring long periods of sensing at a high sampling rate. Extensive experimental results demonstrated the effectiveness of our approach for IoT applications if compared to linear approximations.

References

[1] C. Santos, J. A. Jimnez, and F. Espinosa, "Effect of Event-Based Sensing on IoT Node Power Efficiency. Case Study: Air Quality Monitoring in Smart Cities," *IEEE Access*, vol. 7, pp. 132577–132586, 2019.

[2] M. Razzaque and S. Dobson, "Energy-Efficient Sensing in Wireless Sensor Networks Using Compressed Sensing," *Sensors*, vol. 14, pp. 2822–2859, 2 2014.

[3] S. Trimpe and D. Baumann, "Resource-Aware IoT Control: Saving Communication Through Predictive Triggering," *IEEE Internet of Things Journal*, vol. 6, pp. 5013–5028, June 2019.

[4] H. Mshali, T. Lemlouma, and D. Magoni, "Adaptive Monitoring System for e-Health Smart Homes," *Pervasive and Mobile Computing*, vol. 43, pp. 1 – 19, 2018.

[5] M. Miskowicz, "Send-On-Delta Concept: An Event-Based Data Reporting Strategy," *Sensors*, vol. 6, no. 1, pp. 49–63, 2006.

[6] F. Li, A. Shinde, Y. Shi, J. Ye, X. Li, and W. Song, "System Statistics Learning-Based IoT Security: Feasibility and Suitability," *IEEE Internet of Things Journal*, vol. 6, pp. 6396–6403, Aug 2019.

[7] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-Based Computation Offloading for IoT Devices With Energy Harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 1930–1941, Feb 2019.

[8] D. Zucchetto, C. Pielli, A. Zanella, and M. Zorzi, "Random Access in the IoT: An Adaptive Sampling and Transmission Strategy," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[9] U. Kulau, J. van Balen, S. Schildt, F. Bsching, and L. Wolf, "Dynamic Sample Rate Adaptation for Long-Term IoT Sensing Applications," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 271–276, Dec 2016.

[10] C. Pielli, D. Zucchetto, A. Zanella, and M. Zorzi, "An Interference-Aware Channel Access Strategy for WSNs Exploiting Temporal Correlation," *IEEE Transactions on Communications*, vol. 67, pp. 8585–8597, Dec 2019.

[11] F. Gao, "Application of Improved Grey Theory Prediction Model in Medium-Term Load Forecasting of Distribution Network," in *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, pp. 151–155, Sep. 2019.

[12] C. R. de Sá, "Variance-Based Feature Importance in Neural Networks," in *Discovery Science* (P. Kralj Novak, T. Šmuc, and S. Džeroski, eds.), (Cham), pp. 306–315, Springer International Publishing, 2019.

[13] Z. Kun, W. C. rong, and W. Cong, "Adaptive Threshold Background Modeling Algorithm based on Chebyshev's Inequality," *Computer Science*, vol. 40, no. 4, pp. 287–291, 2013.

[14] É. Morin, M. Maman, R. Guizzetti, and A. Duda, "Comparison of the Device Lifetime in Wireless Networks for the Internet of Things," *IEEE Access*, vol. 5, pp. 7097–7114, 2017.

[15] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. S. J. Pister, "A Realistic Energy Consumption Model for TSCH Networks," *IEEE Sensors Journal*, vol. 14, pp. 482–489, Feb 2014.