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

Tayeb Lemlouma

To cite this version:

Tayeb Lemlouma. Adaptive Sensing Algorithm for IoT Applications with Data and Temporal Accuracy Requirements. 25th IEEE Symposium on Computers and Communications (ISCC), Jul 2020, Rennes, France. 10.1109/ISCC50000.2020.9219699 . hal-02900275

HAL Id: hal-02900275
https://hal.archives-ouvertes.fr/hal-02900275
Submitted on 16 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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-Delta 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 \( \tilde{x}(i) \) using a linear approximation \( \tilde{x} \): 

\[
dyn_{vd}(t) = \frac{k}{n} \cdot \sum_{i=0}^{n-1} |x(t-i) - \tilde{x}(t-i)|, \quad \text{where} \quad k \quad \text{is a factor used to control the width between the upper and the lower bands of series.} \quad \tilde{x} \quad \text{is calculated using the raw values} \quad x_{t_0} \quad \text{and} \quad x_{t_0-n+1} \quad \text{sensed by the IoT sensor device}. \quad \text{The new determined duration between two sensed and transmitted data (i.e. the new time slot)} \quad \delta(t) = \frac{\delta_{max}}{1+dyn_{vd}(t)}, \quad \text{where} \quad \delta_{max} \quad \text{is a fixed maximum waiting period and} \quad \varphi \quad \text{is used to weight} \quad dyn_{vd}(t). \quad \text{Overall, when the observed data show a linear dynamics, next values can be approximated with a time slot that depends on the observed linearity.}
\]
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 \( \epsilon \) 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 \( X^{(0)} = \{x^{(0)}(1), x^{(0)}(2), \ldots, x^{(0)}(n)\} \), where \( x(t) \) represents the value sensed by the device at time \( t \). With a conventional periodic sampling method, the data points of \( X \) 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 \( \hat{x}(t) \). 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 \( X^{(1)}: z^{(1)}(k) = \frac{1}{Z^{(1)}}(x^{(1)}(k) + x^{(1)}(k-1)), \) \( (2 \leq k \leq n) \). 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:

\[
\frac{dx^{(1)}}{dt} + ax^{(1)} = b
\]  

Then, \( a \) and \( b \) parameters can be estimated with:

\[
[y^{(0)}] = [B]^{-1} Y, \quad \text{where:}
\]

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

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

\[
\hat{x}^{(1)}(k + 1) = [x^{(0)}(1) - \frac{b}{a}]e^{-ak} + \frac{b}{a}
\]  

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

\[
\hat{x}^{(0)}(k + 1) = [x^{(0)}(1) - \frac{b}{a}]e^{-ak} (1 - e^{a})
\]

B. Outliers Removal

The prediction of a next value \( \hat{x}^{(0)}(t) \) depends on previous values of \( X^{(0)} \). Although the GM prediction smooths 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 \([\bar{x}^{(0)} \cdot (k \cdot \sigma), \bar{x}^{(0)} + (k \cdot \sigma)] \) with \( k=5 \) and \( \sigma \) is the standard deviation which is computed incrementally as follows:

\[
\bar{x}^{(0)} = \frac{(n-1) \cdot \bar{x}_{n-1} + x(n)}{n} = \bar{x}_{n-1} + \frac{x(n) - \bar{x}_{n-1}}{n} (4) \]

\( \bar{x}^{(0)} \) 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 \( \sigma \) is then calculated with: \( \sigma^2 = \frac{S_n}{n} \), where \( S_n \) is recursively computed: \( S_n = S_{n-1} + M_n \), with \( M_n = (x(n) - \bar{x}_{n-1} - (x(n) - \bar{x}_{n-1})) \) \( k=5 \) and \( \sigma \) is the standard deviation which is computed incrementally as follows:

\[
W_e = \frac{1}{(n-1)(n)} \sum_{i=1}^{n} (x_i - \bar{x}^{(0)}_n)^2 \]  

\( W_e \) is the standard deviation which is computed incrementally as follows:

\[
\sigma_{n+1} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x}^{(0)}_n)^2} \]

According to (4), 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: \((W, \text{mode, } w_0) \equiv \text{current temporal window, current sampling mode, size of } W \)

Output: Sampling mode

1: function selectSamplingMode \((W, \text{mode})\)
2: if \(|W| = w_0\) then
3: if \(\text{mode} = \text{SoD}\) then
4: mode \(\leftarrow\) prediction;
5: \(W_p \leftarrow W; \) \(\triangleright\) start the prediction with a copule of \( W \)
6: else
7: mode \(\leftarrow\) SoD;
8: \(W_p \leftarrow \emptyset\)
9: end
10: \(W \leftarrow \emptyset\)
11: end
12: end function

Unlike algorithm 2 which is performed uninterruptedly every \( \delta_{t_{\text{max}}} \) by the CC, algorithm 3 is carried out by the
Algorithm 2: Central Controller

Input: (ε, δtm max) ≡ data accuracy and temporal threshold requirements

Output: Adaptive sensing

1: mode ← SoD;
2: w ← w0; \( \triangleright w_0 \): size of the prediction window
3: if (bat(t) or bat(t)) ≤ δbat then
4: \ Return bat_alert
5: \ else
6: selectSamplingMode (W, mode, w0);
7: if mode = prediction then
8: \( W_p \leftarrow \) outlier_removal(W p);
9: \( \hat{x}(t) \leftarrow \) predicted_value(W p);
10: \( W_p \leftarrow W_p \triangleq \hat{x}(t) \); \( \triangleright \) append \( \hat{x}(t) \)
11: \ Return \( \hat{x}(t) \);
12: \ else
13: \ Return \( x_{lastReceived} \);
14: \ end
15: \ upon reception of M ≡ \( (x(t), bat(t)) \):
16: send (ε, δtm max, mode, w);
17: if \( \) timestamp(M(x(t))) - 1 = timestamp(\( x_{lastReceived} \)) then
18: \( W \leftarrow W \triangleq x(t) \); \( \triangleright \) append \( x(t) \) to \( W \), \( |W| \leq w \)
19: \ else
20: \( W \leftarrow x(t) \);
21: \ end
22: if mode = prediction then
23: \( W_p \leftarrow W_p \triangleq x(t) \); \( \triangleright \) consider the received \( x(t) \) in the prediction, \( |W_p| \leq w \)
24: \ end
25: \( x_{lastReceived} \leftarrow \hat{x}(t) \);
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 \( \delta_{tm max} \) 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 \( \delta_{tm max} \).

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):
  \[
  MFE = \frac{1}{n} \sum_{t=1}^{n} (A_t - F_t), \quad (n: \text{the size of the original data})
  \]
- The Mean Absolute Deviation (MAD or Mean Absolute Error –MAE): \( \text{MAD} = \frac{1}{n} \sum_{t=1}^{n} |A_t - F_t| \)
- The mean absolute percentage error (MAPE): \( \text{MAPE} = \frac{1}{n} \sum_{t=1}^{n} |pe_t| \)
- The mean absolute percentage forecast accuracy (FA): \( FA = \frac{1}{n} \sum_{t=1}^{n} 100 - |pe_t| \text{ or } (100 - \text{MAPE}) \)
- The mean absolute scaled error (MASE) for j predictions:
  \[
  \text{MASE} = \frac{1}{J} \sum_{t=1}^{n} |x_{j,t} - A_t - F_t| \]

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://Ido.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: \((\epsilon, \delta_{t_{\max}}, \text{mode}, w)\) data accuracy, temporal threshold, sensing mode, temporal window

Output: Adaptive sensing

1. \(x(t) \leftarrow \text{sense\_value}(t)\);
2. \(bat(t) \leftarrow \text{sense\_battery}(t)\);
3. if \(\epsilon = \text{null}\) then
4. \(\epsilon \leftarrow \text{default\_value}\);
5. transmit \((x(t), bat(t))\);
6. end
7. selectSamplingMode \((W, \text{mode}, w_0)\);
8. if \(\text{mode} = \text{prediction}\) then
9. \(W_p \leftarrow \text{outlier\_removal}(W_p)\);
10. \(\tilde{x}(t) \leftarrow \text{predicted\_value}(W_p)\);
11. \(W_p \leftarrow W_p \bowtie \tilde{x}(t)\); \(\triangleright \text{append } \tilde{x}(t)\)
12. \(\Delta \leftarrow |x(t) - \tilde{x}(t)|\);
13. else \(\Delta \leftarrow |x(t) - x_{\text{lastSent}}|\);
14. end
15. if \(\Delta > \epsilon\) then
16. transmit \((x(t), bat(t))\);
17. receive \((\epsilon, \delta_{t_{\max}}, \text{mode}, w)\);
18. update \((\delta_{t_{\max}})\);
19. save \((\epsilon, \delta_{t_{\max}}, \text{mode}, w)\);
20. if \(\text{timestamp}(x(t)) - 1 = \text{timestamp}(x_{\text{lastReceived}})\) then
21. \(W \leftarrow W \bowtie x(t)\); \(\triangleright |W| \leq w\)
22. else \(W \leftarrow x(t)\)
23. end
24. if \(\text{mode} = \text{prediction}\) then
25. \(W_p \leftarrow W_p \bowtie x(t)\);
26. end
27. save \((W)\);
28. \(x_{\text{lastSent}} \leftarrow x(t)\);
29. save \((x_{\text{lastSent}})\);
30. end
31. sleep \((\delta_{t_{\max}})\);
32. end

Fig. 1: Reference data: outdoor temperature in °C and battery level, \(\approx 7\) 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.

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 \(W\)

<table>
<thead>
<tr>
<th>Prediction algorithm</th>
<th>MAPE</th>
<th>MASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction without O.R.</td>
<td>6.28%</td>
<td>5.43</td>
</tr>
<tr>
<td>Prediction with O.R.</td>
<td>5.98%</td>
<td>4.81</td>
</tr>
</tbody>
</table>

Table I: Algorithm’s prediction with and without outliers removal (sampling rate: 0.0167 Hz, fixed window \(W = |X^{(0)}|\)).
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, i.e. ($\epsilon, \delta_{t_{max}}$), it is not obvious with the $dyn_{vd}$ approach (Section II) to find the optimal setting that guarantees the ($\epsilon, \delta_{t_{max}}$) 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 $\epsilon = |\hat{x}(t) - x(t)|$. 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 ($\delta_{t_{max}}$). This will allows the IoT device to sense or linearly approximate the data each $\delta_{t_{max}}$ time slot. This implies:
According to (5) and with a perfect linearity of the observed data (i.e. $dy_{vd}(t)^2 = 0$), $\delta(t)$ is the same as $\delta_{max}$. Conversely, when data are not linear, $dy_{vd}(t)^2$ grows significantly and $\delta(t)$ approaches 0. In the latter case, $dy_{vd}$ 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: $\delta_{max} \geq \delta_{t_{max}}$ and $\delta(t) \geq \delta_{reference\_data}$. When the new time slot $\delta(t)$ is exactly equal to the required time slot $\delta_{t_{max}}$, the data is sensed and transmitted to the CC, hence no approximation is applied. When $\delta(t)$ is greater than $\delta_{t_{max}}$, all the required values at times $t = i$·$\delta_{t_{max}} \leq \delta(t)$, where $i$ is an integer value, are approximated using the $dy_{vd}$. If there are plenty of approximated values, the number of transmissions is reduced but the risk of errors increases. Note that with $dy_{vd}$, to update the linear approximation function and identify the next time slot $\delta(t)$, all the values of the buffer used in $dy_{vd}$ (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 $\epsilon$ and the number of transmissions that ensures the satisfaction of the $\epsilon$ 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 $dy_{vd}$. Indeed, for each new period, we select a new $\delta_{max}$ that satisfies the $\epsilon$ requirement with the smallest number of transmissions, i.e. the higest value of $\delta_{max}$ (Fig. 6.b and 6.d). Thanks to the performed evaluation illustrated in Fig. 6, the selected $\delta_{max}$ enables $dy_{vd}$ to either sense or approximate data every $\delta_{t_{max}}$. While the proposed algorithm ensures the error of returned values will never exceed $\epsilon$ (lines 15–31 in algorithm 3), the MAD measurement of $dy_{vd}$ (Fig. 6) indicates that the overall mean error is less than $\epsilon$. 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 $dy_{vd}$ by requiring a smaller number of transmissions from the IoT device to the CC whatever the value of $\epsilon$. Moreover, with the input ($\epsilon$, $\delta_{t_{max}}$) 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 $\epsilon$, 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 $dy_{vd}$ 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.
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, 0.6 rates: 60 s (P1), 300 s (P2), 600 s (P3).

<table>
<thead>
<tr>
<th>Technology</th>
<th>P_{TX}/P_{RX} (mW)</th>
<th>Data rate</th>
<th>Min-Energy Saving (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth Low Energy (BLE)</td>
<td>24.11 / 19.26</td>
<td>2 Mbit/s</td>
<td>842.80</td>
</tr>
<tr>
<td>802.15.4</td>
<td>24.11 / 19.26</td>
<td>2.5 Mbit/s</td>
<td>842.80</td>
</tr>
<tr>
<td>SIGFOX</td>
<td>147 / 39</td>
<td>1000 bps</td>
<td>1080.66</td>
</tr>
<tr>
<td>LoRa Classe A</td>
<td>419.6 / 44.08</td>
<td>11 kbps</td>
<td>1543.71</td>
</tr>
</tbody>
</table>

*Energy = \sum n \cdot t_s \cdot P_s \cdot t_i \cdot P_t \cdot t_i, \quad P_t \cdot t_i = \text{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}/t_{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