Toward a Standard Time Series Representation for IoT based on CBOR Templates

Sebastian Molina Araque, Ivan Martinez, Georgios Papadopoulos, Nicolas Montavont, Laurent Toutain

Toward a Standard Time Series Representation for IoT based on CBOR Templates

Sebastian Molina Araque[1], Ivan Martinez[2], Georgios Z. Papadopoulos[3], Nicolas Montavont[3], Laurent Toutain[3]

IMT Atlantique, IRISA, France
[1]*juan.molina-araque@imt-atlantique.fr* [2]*ivan-marino.martinez-bolivar@imt-atlantique.fr*
[3]*firstname.lastname@imt-atlantique.fr*

***Abstract*—Nowadays, we are facing an era where the Internet of Things (IoT) is growing intensively and IoT devices are being deployed massively. At this point, interoperability with the information system is a major issue for the accelerated device deployment. Most of the time, IoT information is presented as Time Series (TS). However, no standardized representation format has emerged, while the majority of the studies in the literature focus on the compression, processing, or prediction of TS. In this paper, we propose a versatile format for TS representation based on the Concise Binary Object Representation (CBOR). This representation exploits CBOR to be compact thanks to the introduction of deltas to represent measurements, tags to represent variables, and templates to transform our TS representation into the data format required by information systems. Our preliminary results show that the data collected by IoT devices can be regrouped compactly. Indeed, for the cases evaluated we have proven that the total amount of data to be sent can be reduced between 76% and 96% when compared against JavaScript Object Notation (JSON). Finally, by lowering the amount of data transmitted while keeping the same amount of information delivered, our approach has the potential to extend the battery life of IoT devices and their useful life.**

***Index Terms*—Internet of Things, IoT, Time Series, Interoperability, CBOR, JSON.**

I. Introduction

In 2020 the Internet of Things (IoT) market has grown by 23.1%, and is projected to grow up to 25.4% per year between 2021-2028 [1]. Besides, some other studies forecast 50 billion devices at the end of the decade [2]. This implies a massive deployment of IoT devices, which can only be sustained if the management operation is handled efficiently. Moreover, considering the large IoT deployment and the considerable amount of data that collects the IoT devices, the integration into an information system will become more challenging.

Since the devices are currently chosen and incorporated into IoT systems during the conception phase rather than afterward, altering an application may require changing the device. On the other hand, if new devices are introduced to the market, the application must be modified to support them.

The majority of IoT data measurements are presented in TS format. Once again, this representation depends on the application or the environment in which the system is employed [3]. Currently, the academic and industrial communities are focusing rather on compressing or processing the data produced, and to the best of our knowledge, there is no compact representation of TS. Indeed, we have observed that the works from the literature are rather concentrated on exploiting the TS than efficiently representing it. For instance, in [4], the authors propose a framework to improve the analysis of the information produced by IoT sensor devices. This framework adds contextual and historical information to the initial TS information. Next, in [5], the authors present a new compression method of TS for IoT.

In this paper, we propose a versatile format for TS representation based on CBOR [6]. This format is compact and is the first step to decoupling the sensor data format from the application data format. Our approach uses CBOR characteristics to represent measurements using: the difference of the subsequent values (deltas) in the TS and a Tag Number (TAGN) with a tree format in CBOR to identify where the values of the variables will be in the TS. Then, we introduce Variable-based TS (VTS) and Measurements-based TS (MTS) templates, where the representation of TS is grouped by variables or measurements, respectively, and additional information such as the time stamp, sensor identifiers, and sensor precision, is represented with metadata. These templates will transform our proposed TS representation into the data format required by any information system. The performance evaluation demonstrates that the actual data sent by IoT devices can be reduced by between 76% and 96% compared against JSON. Furthermore, when networks such as LPWAN or Wi-Fi are used, the metadata represents an additional 2% of the total amount of information sent.

II. Technical Background

This section provides a brief description of the most used representation formats on the Internet. We will start with JSON, then we describe the binary data format CBOR, which is the base of our work. Finally, we give an overview of SenML; this format allows to represent sensor measurements and device parameters in a simple form.

A. JavaScript Object Notation (JSON)

JSON, defined in [7], is a data interchange format that defines a set of rules for the representation and serialization of structured data. It is a widely used data serialization format over the internet due to three main reasons: (i) it is easy to understand and write by humans, (ii) compared

to XML, it is a lightweight data format, and (iii) it uses conventions commonly used in programming languages while being language-independent.

JSON defines certain notations to represent structured data. It is built by a collection of "name/value" pairs, and it can contain an ordered list of values. Moreover, only objects ({}), arrays ([]), strings, numbers, 'true', 'false', and 'null' can be represented as values. Listing 1 shows an example:

```
{    'name': 'value',
     'array':[1,2,3,4],
     'send': true }
```

Listing 1. A JSON example.

B. Concise Binary Object Representation (CBOR)

CBOR was designed by the Internet Engineering Task Force (IETF) and is specified in [6]. It is a binary data serialization format based on JSON. Furthermore, CBOR was created with the following set of goals [8]:

i) CBOR must be able to represent the most common data formats in internet standards without any ambiguity.
ii) The code for an encoder/decoder must be compact.
iii) Data must be able to be decoded without a schema description.
iv) The serialization must be reasonably compact.
v) Must apply to constrained nodes and high-volume applications.
vi) All JSON data types must be supported.
vii) The format needs to be extensible.

Previous work has compared the size of messages serialized between CBOR and JSON, and CBOR shows a size of around 26% less in Bytes than JSON [9]. As an example, consider the following array `[1,[2,3],[4,5]]` [6], JSON needs 15 bytes to represent it, while as seen in Listing 2, CBOR only needs 8 bytes. This represents a reduction of 53%.

```
83       # array(3)
   01    # unsigned(1)
   82    # array(2)
      02 # unsigned(2)
      03 # unsigned(3)
   82    # array(2)
      04 # unsigned(4)
      05 # unsigned(5)
```

Listing 2. The CBOR representation of [1,[2,3],[4,5]] in Hexadecimal.

Note that this reduction is not optimized in CBOR for floating numbers, 0.1 is coded in 3 bytes in JSON and up to 9 bytes in CBOR. Therefore, using CBOR correctly and with all its capabilities, it is a potential tool to reduce considerably the bytes sent by IoT devices.

C. Sensor Measurement List (SenML)

SenML is defined in [10]. It was created to encode sensor measurements into the media type by processors with limited capabilities, while also allowing a server processing the data to save and decode many sensor measurements reasonably efficiently. The design goal is to be able to send simple sensor measurements in small packets from a large number of constrained devices. An example of this data model with JSON syntax is:

```
[{   'n':'urn:dev:ow:10e2073a01080063',
     'u':'Cel','v':23.1  }]
```

Listing 3. A single SenML Record example.

When it is needed, and if one message contains multiple measurements, it can be represented as "base values" in SenML, which allows optimization re-using values per measurement. An example with base values as 'base time (bt)' and 'base unit (bu)' is shown below:

```
[{   'bn': 'urn:dev:mac:0024befffe804ff1/',
     'bt': 1276020076,          'bu': 'A'},
   { 'n':'voltage',    'u':'V', 'v':120.1},
   { 'n':'current',    't':-2,  'v':  1.5},
   { 'n':'current',    't':-1,  'v':  1.6},
   { 'n':'current',    't':0,   'v':  1.7}  ]
```

Listing 4. A SenML Record example with base values.

III. Related Work & Problem Statement

A. Literature Review of Binary Formats

Serialization formats are typically utilized for information transmission between IoT devices [11]. It refers to the process of translating data structures or object states into a format that can be transmitted and reconstructed later. This operation becomes critical for because it can improve the limited capabilities and life cycle of end devices at the same time [12]. Other binary formats found in the literature are:

i) Messagepack: a concise, widely implemented counted binary serialization format, similar in many properties to CBOR, although somewhat less regular. While the data model can be used to represent JSON data, MessagePack has also been used in many Remote Procedure Call (RPC) applications and for long-term storage of data [6].
ii) Binary JSON (BSON): a data format that was developed for the storage of JSON-like maps (JSON objects) in the MongoDB database. Its major distinguishing feature is the capability for an in-place update, which prevents a compact representation [6].

There have been several binary formats with various objectives in addition to the two just stated. These objectives were typically not expressly expressed, though they may occasionally be inferred from the context in which the format was first used. Thus, alternative binary formats with comparable objectives include Protocol Buffers designed by Google, PSON, Smile, and Message Services Data Transmission (MSDTP) [13], [14].

B. Problem Statement

Currently, reporting measurement is use-case oriented in a really simple and trivial manner over IoT devices. To the best of our knowledge, there is no standard representation for TS in IoT. Therefore, having such a representation may be beneficial since it breaks the significant dependency between IoT devices and cloud applications, which makes IoT devices independent from the network or application.

Thus, this paper introduces an architecture with a middleware in charge of regrouping in a compact form the TS information produced by IoT devices. Then, we introduce two

approaches to improve a template proposed in a current draft from the IETF [15], as well as a tree representation to indicate the position of the variables and deltas to represent the values in a compact manner. Finally, we compare the results of our work against the JSON in conjunction with CBOR.

IV. The Proposed Architecture

Fig. 1 illustrates our vision of the IoT architecture on IoT networks. We propose to have a middleware between the data transmission process and the processing information process. This allows us to focus on the production of interoperable datasets, which in turn enables a seamless integration of IoT devices into different information systems without modifying the actual code of the device.

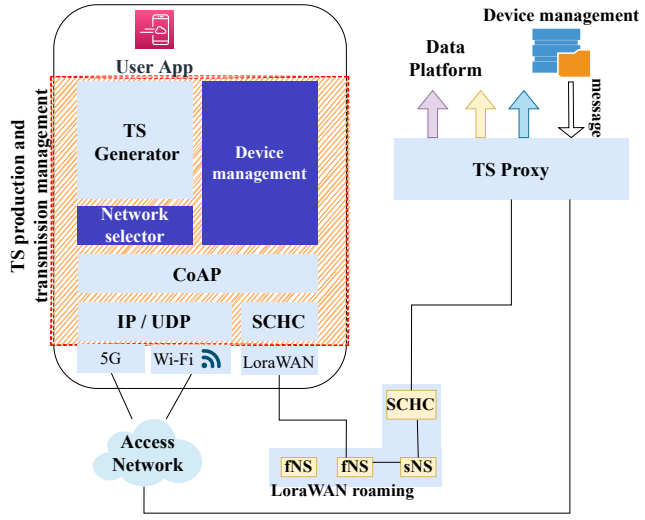

Fig. 1. The proposed IoT architecture.

The device user application produces all the data obtained from the measured values. Then, below the user application is placed the middleware. The primary function is to compactly group the data generated by the sensor using a common TS representation; this is the main contribution of this paper. Furthermore, this middleware is responsible for selecting the interface through which the data will be sent, as well as controlling the volume of data and the frequency of each measurement depending on the interface, band restrictions, and the type of measurement.

For Wi-Fi and 5G networks, these data will be sent over IP/UDP/CoAP to a proxy.

In order to meet the LoRaWAN frame size requirements and to provide reliability, header compression, fragmentation and re-transmissions is assured by using the Static Context Header Compression (SCHC) protocol [16][1]. Access networks such as 5G, Wi-Fi, or LoRaWAN can be selected to send the compact TS representation. However, all of them will converge into a TS proxy, present at the right in Fig. 1. This part of the architecture will be in charge of reconstructing the information sent by the device. Then, it will transform from a TS standard representation into the data representation required by the data platform or cloud application, which will mostly be reconstructed to a specific structure such as SenML with a JSON format.

[1]It must be noted, that the use of SCHC does not imply a full IP/UDP/CoAP protocol stack implementation, but since SCHC relies on compression rules, the device can process these rules directly to limit the footprint [17].

V. TS Representation with CBOR Templates

Thanks to the extensibility of CBOR and with the help of the definition of a new tag, [15] it sees the need to define variables within CBOR to reduce the number of bytes that are sent [15]. Indeed, the proposed CBOR template allows for handling variables and, thus, transforming the data representation, e.g., from JSON to CBOR. Moreover, [15] defines a CBOR template as a CBOR data item containing one or more variables. These variables are represented as a CBOR data item containing a specific identifier, i.e., CBOR tag 42.

A. Static-based Time-Series (STS)

Below, we present the STS [15] proposition through the following example:

```
{ 'name":  'Carsten Bormann',
  'place': 42(0) }
```

Listing 5. An example in [15]

where:

- **42**: The Tag Number (TAGN), which indicates the variable identifier.
- **(0)**: Indicates the position where the value of the variable is, in this case, the first position of a CBOR array.

When the template from Listing 5 undergoes substitution, with the variable 0 set to the value “Bremen”, this will result in the data item as it is depicted in Listing 6:

```
{  'name':  'Carsten Bormann',
   'place': 'Bremen'  }
```

Listing 6. CBOR variable substituted [15].

This example shows that the template from [15] (i.e., Static-based Time-Series (STS)) handles only static variables. That means that if the devices require to send 100 values from the same variable, they will need 100 variables in the template. Consider that we want to represent the JavaScript Object Notation (JSON) structure presented in Listing 7, which has information related to a temperature and humidity sensor transmitting three measurements at the same time:

```
[ {'n':'temperature','u':'Cel', 'v': 32},
  {'n':'humidity',   'u':'%RH', 'v': 20},
  {'n':'temperature','u':'Cel', 'v': 31},
  {'n':'humidity',   'u':'%RH', 'v': 21},
  {'n':'temperature','u':'Cel', 'v': 30},
  {'n':'humidity',   'u':'%RH', 'v': 22} ]
```

Listing 7. A JSON structure that will be employed in the rest of the paper.

In this case, the corresponding STS template would be:

```
[ {'n':'temperature','u':'Cel','v':42(0)},
  {'n':'humidity',   'u':'%RH','v':42(1)},
  {'n':'temperature','u':'Cel','v':42(2)},
  {'n':'humidity',   'u':'%RH','v':42(3)},
  {'n':'temperature','u':'Cel','v':42(4)},
  {'n':'humidity',   'u':'%RH','v':42(5)} ]
```

Listing 8. Example of Listing 7 as a template with CBOR variables.

Then, the CBOR array to be sent undergoes the substitution is: [32, 20, 31, 21, 30, 22]. However, as previously mentioned, STS is a static template, which means, if this template receives a different CBOR data item with a different amount of measurements (e.g., an array with only 4 measurements [32, 20, 31, 21]), then the template is expecting more values (6 variables) and would not be able to make the substitution.

B. Tree Formatting

In this paper, we propose a tree representation inside the TAGN, indicating where the values of the variables will be found in the CBOR data item. This allows the reduction of the amount of data sent by IoT devices and the introduction of non-static templates in a further section. Let us now consider the example presented in Listing 9, this listing contains nested arrays in four different levels to explain the tree representation:

```
[[0, 1, 2, 3], [4, 5, [6, 7, 8, 9]], [10, 11]]
```

Listing 9. Example with nested arrays to explain tree representation.

The tree representation of the previous array in Listing 9 is depicted in Fig. 2.

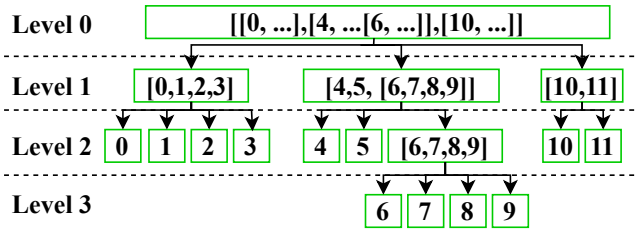

Fig. 2. Tree representation in levels of Listing 9.

In this case, If it is necessary to select any value according to the new representation of the TAGN proposed, the following values can be selected:

i) Consider the tag represented as TAGN [0,1]: It refers to the first array of data (TAGN [**0**,1]), in this case ([0,1,2,3]), and the second position of that array (TAGN [0,**1**]). Thus, the referring value is: **1**
ii) Consider another tag represented as TAGN [1,2,3]: It is referring to the second array of data (TAGN [**1**, 2, 3]), in this case ([4,5[6,7,8,9]]), inside that array is referring to the third position (TAGN [1, **2**, 3]) which is another array ([6,7,8,9]), and finally the value present in the fourth position (TAGN [1, 2, **3**]) which is: **9**.
iii) Finally, adding the value **'true'** to the TAGN representation. This value means that all the values present in the array will be selected. Thus, consider the tag represented as TAGN [1, 2, true]: Then, it corresponds to all the values present in the array placed on [1, 2]:

```
- [ 1, 2, 0]: 6,     - [ 1, 2, 1]: 7
- [ 1, 2, 2]: 8,     - [ 1, 2, 3]: 9
```

Listing 10. Values selected with TAGN [1, 2, true].

C. Delta Between Measurements

In this section, we introduce the use of deltas in CBOR templates.

Consider the example of Listing 7. Here, we propose that the devices will transmit only the difference of the subsequent measurements, by doing that we reduce the amount of data sent by IoT devices.

Then, the CBOR data item for the STS template required to reproduce Listing 7 is: [32, 20, 31, 21, 30, 22]. It uses 10 bytes when transforming with CBOR. Thus, the difference present in each value for temperature is -1 (32-1 = 31, 31-1 = 30) and humidity is +1 (20+1 = 21, 21+1=22), then only the first value is needed as a reference. Therefore, the result array using the delta between measurements is: [32, 20, -1, 1, -1, 1]. Thus, only 8 bytes are used against 10. When using deltas to represent measurements is more significant when the values are higher(e.g., [535,537,539,538] uses 13 bytes while [535,2,2,-1] only uses 7 bytes).

In the rest of the paper, all examples will be expressed in deltas as described in this section.

D. Variable-based TS (VTS)

Having a representation with CBOR variables and templates in Section V-A reduces the amount of data sent by IoT devices. However, it is limited by knowing the exact number of data items or variables to be replaced.

In this section, we introduce a new value to the TAGN specification, which allows differentiating when it is not possible to know how many values are going to be sent inside a CBOR data item. Thus, it is possible to represent TS data items with a template and an array of information as follows:

```
[{'n':'temperature','u':'Cel','v':TAGN[0,true]},
{  'n':'humidity',   'u':'%RH','v':TAGN[1,true]}]
```

Listing 11. VTS template.

Note that it is possible to identify the CBOR variables present in the CBOR data item thanks to the tag TAGN, and the rest of the value indicates the position where the value will be found following the structure: TAGN([v1, v2, .., vn]). Where:

i) TAGN is the tag number.
ii) v1, v2, and vn, are the values to specify where it is possible to find the values for each variable:
   a) If 'true', it corresponds to the value to select all the items on the substitution item.
   b) If it is a number, it corresponds to the specific position or array where the value is.

Thus, continuing with the structure of the template proposed, the CBOR data item needed to represent the JSON structure presented in Listing 7 must be:[[32, -1, -1], [20, 1, 1]][2], and the result when the values are replaced is:

[2]Review Section V-B and V-C to understand how the template works and how the CBOR data item was built

```
[ {'n':'temperature','u':'Cel', 'v':32},
  {'n':'humidity',   'u':'%RH', 'v':20},
  {'n':'temperature','u':'Cel', 'v':31},
  {'n':'humidity',   'u':'%RH', 'v':21},
  {'n':'temperature','u':'Cel', 'v':30},
  {'n':'humidity',   'u':'%RH', 'v':22} ]
```

Listing 12. Final result for VTS with information replaced.

This template allows us to send more or fewer values if needed thanks to the 'true' presented in the TAGN in Listing 11. Thus, this value allows to send two values for humidity and two for temperature with the following the CBOR data item [[32, -1], [20, 1]][2], and when the values from the CBOR data item are replaced, the result is:

```
[ {'n':'temperature','u':'Cel', 'v':32},
  {'n':'humidity',   'u':'%RH', 'v':20},
  {'n':'temperature','u':'Cel', 'v':31},
  {'n':'humidity',   'u':'%RH', 'v':21} ]
```

Listing 13. Result for VTS sending only two values.

Notice that two arrays are present in the last two examples, one for each variable (temperature and humidity) independent of the number of measurements sent. Then this approach is ordered by variables as shown in Fig. 3.

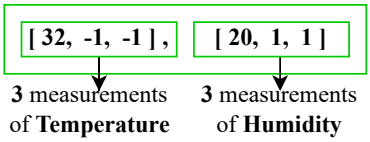

Fig. 3. Explanation for the CBOR data item required in VTS.

E. Measurements-based TS (MTS)

This section introduces MTS as a way to order TS by measurements. Thus, each measurement inside the CBOR data item is differentiated by an array, and each internal array is composed of the values of the variables presented in the template. In order to continue with the structure present in Listing 7 the following CBOR template is needed:

```
[{'n':'temperature','u':'Cel','v':TAGN[true,0]},
 {'n':'humidity',   'u':'%RH','v':TAGN[true,1]}]
```

Listing 14. MTS template.

Thus, the following CBOR data item is required to represent the JSON data values: [[32, 20], [-1,1], [-1,1]][2], in this case the value 'true' over TAGN takes the values 0,1,2,3,4[2] referring to all arrays present in the CBOR data item. Obtaining the same result as in VTS. Fig. 4 depicts the CBOR data item grouped by measurements with 3 arrays and 2 values per measurement.

With MTS it is also possible to send more or less values depending on the requirements. By employing the same example as in VTS, if only two values for humidity (20,21) and two values for temperature (32,31) are required to transmit, then, the CBOR data item would be: [[32, 20], [-1, 1]].

Contrary to VTS, two arrays of data are present, instead, in each intern array the values for each measurement are present, and the number of arrays represents the number of measurements taken.

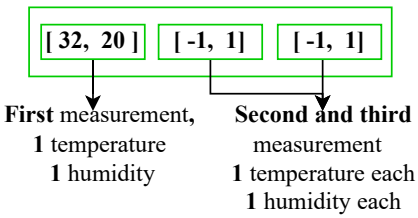

Fig. 4. Explanation for the CBOR data item required in MTS.

F. Metadata

The examples presented until now only show a simple payload, however, IoT information is much richer. In addition to the actual measurement of the variable, some use-cases may need additional information such as: (i) the time stamp, (ii) sensor identifiers, (iii) sensor precision, etc. To handle this additional information, we propose to add what we call 'Metadata'.

To store this metadata we propose to have a map type ({}) in CBOR. It will be present each time a message is created. It allows us to differentiate between the metadata and the measurements sent. Thus, the metadata format in MTS and VTS templates will be represented as follows:

```
[{METADATA},[32,20],[-1,1],[-1,1]]
```

Listing 15. Representation of metadata in MTS.

```
[{METADATA},[32,-1,-1],[20,1,1]]
```

Listing 16. Representation of metadata in VTS.

Furthermore, the metadata would be present by pairs as an object in the JSON format (name/value pairs) and using base values as in SenML.

```
{  metadata1: value1,
   ...
   metadataN: valueN }
```

Listing 17. Representation of metadata.

In the following subsections, we will explain the types of metadata proposed.

1) Time Stamps: Time stamps will be represented by using two values: (i) a Base Time (bt) with the TAGS 0 and 1 as in [6] and (ii) the Difference Time (dt) representing the difference in seconds between each measurement based on 'bt'. If bt is not present in the metadata, the reception time will be considered instead by default. As for dt, 60 seconds will be set by default. An example of the time represented in the metadata is:

```
{  bt: 1654070400,   dt: -1, }
```

Listing 18. Base time and difference time in metadata.

Listing 18 represents that the first measurement present in each template was taken on Wednesday, 01 June 2022 at 10:00:00, and the following measurements were taken one second before, Wednesday, 01 June 2022 09:59:59 for the second measurement, Wednesday, 01 June 2022 09:59:58 for the third measurement, etc.

2) Precision: Precision refers to the decimal part of the floating-point number. Consider 1.258 and 1.25, both are floating numbers the former has a precision of 3, while the latter is 2.

Floating-point numbers are not optimized in CBOR, instead, an integer value would be preferred since it needs less bytes to represent a value. Hence, if the precision can be indicated, the decoder can interpret and represent the correct values.

Thus, 1.258 will be represented as the integer 1258 with a Base Precision (bp) of 3. If bp is present in the metadata all the variables will have the same precision, and all the other values for precision will be ignored. However, if bp is not present in the metadata the value is 2 by default.

```
{ bp: 3 }
```

Listing 19. Base precision for all variables.

On the contrary, if all values do not have the same precision, a precision value per measurement must be present. Thus, the abbreviation per each precision of each variable would be:

```
{ bN: value }
```

Listing 20. Base precision for a specific variable.

Where N is the number of the variable that needs to be modified. Each variable is identified by the tag and how they are organized in the template. As seen in Section V, for both templates, VTS and MTS, the temperature is the first variable, and humidity is the second one. Accordingly to this, the abbreviation 'b0' modifies the precision of the temperature, and 'b1' modifies the precision of the humidity.

In the next example, the precision of the humidity is modified to 1, and as bp is not present, the precision for the temperature is 2:

```
{ b1: 1 }
```

Listing 21. Example, base precision for the second variable (humidity).

Thus the final result replacing the values in Listing 11 or 14 with the metadata in Listing 21 is:

```
[ {'n':'temperature','u':'Cel', 'v':0,32},
  {'n':'humidity',   'u':'%RH', 'v':2,0},
  {'n':'temperature','u':'Cel', 'v':0,31},
  {'n':'humidity',   'u':'%RH', 'v':2,1},
  {'n':'temperature','u':'Cel', 'v':0,30},
  {'n':'humidity',   'u':'%RH', 'v':2,2} ]
```

Listing 22. Final result with a precision of one in humidity and two as bp.

When more precision values than variables are present in the metadata they are ignored, and only the ones who apply for each variable are considered. Furthermore, metadata depends on the application where the IoT device is being used (and even the sensor).

VI. Performance Evaluation

To evaluate the performance of our proposals, we first compare the JSON example presented in Listing 7 when transformed into CBOR, STS, MTS, and VTS, then (ii) we take a real-world data set[3] and perform the same transformation, and (iii) finally, we assess the impact of including metadata in each message sent.

1) JSON Representation: Consider the JSON representation example depicted in Listing 7:

```
[ {'n':'temperature','u':'Cel', 'v':32},
  {'n':'humidity',   'u':'%RH', 'v':20},
  {'n':'temperature','u':'Cel', 'v':31},
  {'n':'humidity',   'u':'%RH', 'v':21},
  {'n':'temperature','u':'Cel', 'v':30},
  {'n':'humidity',   'u':'%RH', 'v':22} ]
```

Listing 23. A JSON representation to be transformed into CBOR, STS, MTS, and VTS.

To represent all of the data in this JSON, the CBOR representation requires 139 bytes, whereas the STS template requires 17 bytes. Following that, as shown in Fig. 5, using the VTS and MTS approaches, it is possible to further reduce by approximately 93% (from 139 bytes to 10 bytes) and 92% (from 139 bytes to 11 bytes), respectively. The STS proposal reduces the 88%, which is a significant value at the expense of a non-flexible template.

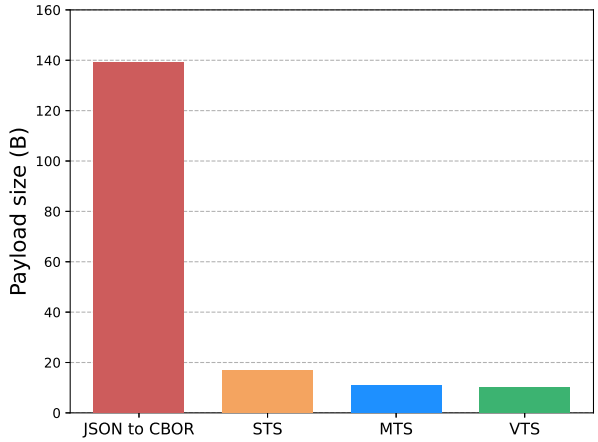

Fig. 5. The gain in bytes per CBOR template.

2) Performance with real-world data: To test our hypothesis with real data, we used data set 3 from the repository[3], which contains the first 100 measurements of an electrophoresis painting plant with 7 sensors sensing every 10 s. This yields 700 measurements.

In this case, even though the STS template is static, it can be used, but at the expense of a large template, because it would require 700 variables in the template to replace all of the data. VTS and MTS, on the other hand, just need a template with seven variables and the associated TS representation. As a result, the amount of this data in JSON format serialized with CBOR is 9294 bytes, compared to 6273 bytes for STS, 2141 bytes for VTS, and 2228 bytes for MTS, a reduction of 32%, 77% percent, and 76%, respectively.

3) Impact of added Metadata: Up until this point, we have only considered the payload size required to represent the information in the two previous subsections. Here, we will look at hypothetical situations to determine how adding

[3]Time series dataset for CBOR templates, Github: https://github.com/sebasmol96/Time-Series-data-set-for-CBOR-templates

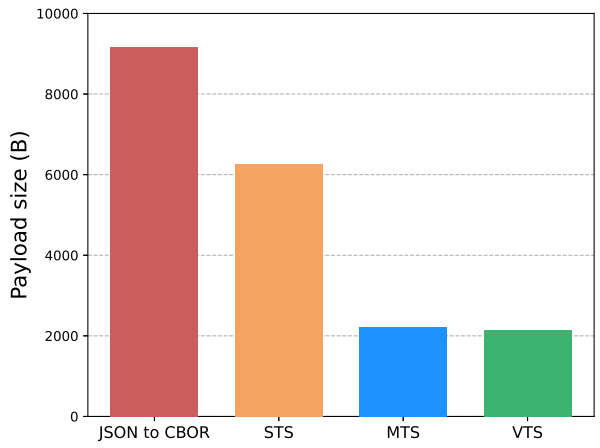

Fig. 6. Payload size per CBOR template for the electrophoresis painting plant.

metadata to each transmission will impact the performance if we consider the data set from Subsection VI-2.

The Metadata to be sent inside each packet transmission will be: {bt: 1593982800, dt:10, bp:3} which represents 17 bytes in CBOR. Thus, we consider 3 cases. One is a LoRaWAN Network working with a Spread Factor (SF) of 7 and with the maximum allowed payload size of 222 bytes, then a Wi-Fi network with a Maximum transmission unit (MTU) of 1468 bytes, and finally an LTE/5G network with an MTU of 1428 bytes.

Table I presents the number of packets that would be sent per Access Network, as well as the total size in bytes needed to represent the metadata and its percentage compared to the value required to transmit all the payload of 9294 bytes.

TABLE I
IMPACT OF ADDED METADATA WITH DIFFERENT ACCESS NETWORKS.

		LoRaWAN	Wi-Fi	LTE/5G
VTS	Packets number	10	2	2
	Size in Bytes	170	34	34
	Percentage (%)	1.9	0.4	0.4
MTS	Packets number	11	2	2
	Size in Bytes	187	34	34
	Percentage (%)	2.1	0.4	0.4

To summarize, the metadata varies between 2.1% and 0.4% depending on the access network selected to transmit the data. Furthermore, for STS template it can not be represented and, for JSON transformed into CBOR is not needed.

VII. Conclusions and Future Work

IoT devices are being deployed massively, and in order to integrate these devices into information systems, interoperability is required. Besides, IoT devices have strong limitations as processing power, available memory, battery life, and a strong dependency with cloud applications. This paper proposes a compact TS representation with templates based on CBOR. The proposed templates, VTS and MTS, have shown that the data collected by IoT devices can be regrouped in a compact manner. Moreover, it reduces between 76% and 96% the total amount of data that would be sent when compared to JSON.

The results of this proposal have practical implications. First, reducing the data size sent by the devices is directly reflected in better battery life. Next, a standard format for TS breaks the strong dependency between cloud applications and IoT devices since it can be transformed into any data representation expected by the data platform. As for future work, we plan to review the impact of the metadata on the performance of our templates using real deployments, and complex data structures.

References

[1] F. B. Insights, "Internet of Things (IoT) Market Size, Share & COVID-19 Impact Analysis, By Component (Platform, Solution & Services), By End-Use Industry (BFSI, Retail, Government, Healthcare, Manufacturing, Agriculture, Sustainable Energy, Transportation, IT & Telecom, Others), and Regional Forecast, 2021-2028," accessed: March 2022.

[2] C. Dave Evans, "The Internet of Things. How the Next Evolution of the Internet Is Changing Everything (white paper)," accessed: January 2022.

[3] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2020.

[4] K. Kenda, B. Kazic, E. Novak, and D. Mladenić, "Streaming Data Fusion for the Internet of Things," *MDPI AG Sensors*, vol. 19, no. 8, pp. 1424–1450, 2019.

[5] D. Blalock, S. Madden, and J. Guttag, "Sprintz: Time Series Compression for the Internet of Things," *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, vol. 2, no. 93, pp. 1–23, 2018.

[6] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," IETF, RFC 8949, December 2020.

[7] E. T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," IETF, RFC 8259, December 2017.

[8] E. Ingles-Sanchez, D. Garcia-Carrillo, G. Z. Papadopoulos, N. Montavont, and A. F. S. Gomez, "Adaptation of EAP-NOOB Method for LoRaWAN with LO-CoAP-EAP and CBOR," in *Proceedings of the IEEE Global Internet of Things Summit (GIoTS)*, 2020.

[9] Ivanovo, "Comparison of JSON Like Serializations – JSON vs UBJSON vs MessagePack vs CBOR," http://zderadicka.eu/comparison-of-json-like-serializations-json-vs-ubjson-vs-messagepack-vs-cbor/, accessed: January 2022.

[10] C. Jennings, Z. Shelby, C. Bormann, J. Arkko, and A. Keranen, "Sensor Measurement Lists (SenML)," IETF, RFC 8428, August 2018.

[11] F. Pereira, R. Correia, P. Pinho, S. I. Lopes, and N. B. Carvalho, "Challenges in Resource-Constrained IoT Devices: Energy and Communication as Critical Success Factors for Future IoT Deployment," *MDPI AG Sensors*, vol. 20, no. 22, pp. 6420–6449, 2020.

[12] D. Tomaszuk, R. Angles, Ł. Szeremeta, K. Litman, and D. Cisterna, "Serialization for Property Graphs," in *Springer. Beyond Databases, Architectures, and Structures. Paving the Road to Smart Data Processing and Analysis*, 2019, pp. 57–69.

[13] J. J. C.-G. Álvaro Luis, Pablo Casares and M. A. Patricio, "PSON: A Serialization Format for IoT Sensor Networks," *MDPI AG Sensors*, vol. 21, no. 13, pp. 4559–4576, 2021.

[14] J. C. Viotti and M. Kinderkhedia, "A Survey of JSON-compatible Binary Serialization Specifications," *arXiv Computing Research Repository (CoRR)*, vol. 2, 2022.

[15] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR) Tag for CBOR Templates," IETF, DRAFT 02, July 2018.

[16] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and J. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation," IETF, RFC 8724, April 2020.

[17] C. Gomez, A. Minaburo, L. Toutain, D. Barthel, and J. C. Zuniga, "IPv6 over LPWANs: Connecting Low Power Wide Area Networks to the Internet (of Things)," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 206–213, 2020.