Journal of ELECTRICAL ENGINEERING, VOL 71(2020), NO4, 281–285

COMMUNICATIONS

Internet of things - nonstandard data compression

Ivan Sokol[1], **Peter Hubinský**[1]

JSON is probably one of the most popular data formats in the Internet of Things (IoT). It has gained popularity among developers due to its easy readability. The disadvantage of this format is the size of the resulting document and the associated bandwidth requirements. In order for us to be able to transfer such data securely, it is necessary to secure it before transferring it. Often, this data is compressed before it is encrypted. This article deals with a non-standard method of data compression, which reduces the computational demands on the device side. It should be noted that many of these devices are powered by batteries with limited capacity.

K e y w o r d s: internet of things, IoT, JSON, data compression

1 Introduction

The Internet of Things is becoming a part of our daily lives, whether we like it or not. We meet it everywhere.

In 2017, Gartner [1] published an estimated number of IoT devices connected to the Internet for the coming years. According to these estimates, there should be around 20.5 billion such devices in 2020. SAP's [2] estimates from 2019 speak about 30 billion devices in 2020. In 2025, there should be as high as 75 billion devices. In an article from 2018 [3], analysts assume that in 2022, the number of IoT devices connected to the Internet will be at the level of 50 billion, in contrast to the originally assumed 21 billion.

Thousands of new devices connect to the network every day. We have smart appliances and entire households, buildings, cities, but also transport, agriculture or health care. For this to work correctly, communication between devices is required. Regardless of all forecasts, we are faced with the question of how to transfer and process the amount of data generated by these devices. At the same time, it should be noted that IoT devices use the same transmission routes for communication purposes as our computers or IP TV.

HTML [4] is the standard language of web pages. IoT devices usually use JSON [5] or XML [6] data formats for the data transmission. The advantage of the above languages is that they are easily readable and writable for people. The disadvantage is the size of the created documents. From the point of view of data transfer, these data formats are unprofitable. In the case of HTML, Google try to solve the problem of the amount of transferred data. In their project [7] they try to reduce the volume of transmitting data in HTML format by compressing it. On the other hand, the authors of the article Schema Based XML Compression [8] focus on the compression of XML data format.

In this article we present a non-standard method of the data compression in JSON format with focus on IoT devices.

2 JSON - JavaScript object notation

JSON is probably the most popular data format used by IoT devices. It is a lightweight data-interchange format independent of the platform. The language data structure is defined by standard ECMA 262. [9]

The data is in JSON presented as an object. The object represents a set of pairs in the form

keyword:value

enclosed in curly braces { **object** }.

The value can take one of the allowed formats:

object, array, number, string, "true", "false", "null"

As an example, take the following message:

```
{
    id: 12f8-0860-535d-f00b,
    type: switch,
    name: bathroom,
    status: off,
    date: 2020-04-20 18:56:26,
}
```

[1] Slovak University of Technology, Faculty of Electrical Engineering and Information Technology, Ilkovičova 3, 81219 Bratislava, Slovakia, peter.hubinsky@stuba.sk

DOI:10.2478/jee-2020–0038, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

Table 1. Description of the values

Key word	Value	Alt. value	Change by	Format	Parameters
id	12f8-0860-535d-f00b		fix	text	max. 20 char
type	switch		fix	text	max. 20 char
name	bathroom	define by customer	customer	text	max. 20 char
status	off	on/off	device	text	max. 3 char
date	2020-04-20 18:56:26	valid date / time	device	date-time	date format

3 Data compression standard methods

The message generated by IoT devices can range in size from tens of bytes to several kilobytes. We are going to focus on messages with a size of max. hundreds of bytes, where standard algorithms are not very effective. The proposed solution is usable for a message of any size.

3.1 Removal of unnecessary characters

The size of the message in our example is 134 bytes. In the present form it is easily readable. We can reduce the size of the message by removing unnecessary characters, like space, tab or a new line. Then, the message looks like this:

{"id": "12f8-0860-535d-f00b", "type" :"switch",
"name": "bathroom", "status": "off", "date":
"2020-04-20 18:56:26"},

After this operation the size of the message is 107 bytes. The data reduction is about 20%.

3..2 Standard compression algorithms

Compression algorithms are divided into two basic groups: lossy and lossless. In the case of data transmission of IoT devices, it is necessary to use a lossless compression method in order to restore the data to its original form for further processing. For this reason we have used in our test the GZIP and BZIP2 libraries in default setting. The compression results of the presented message were

GZIP	108 characters
BZIP2	132 characters.

The results of the compression are poor in this case. On the other hand, we must keep in mind how these algorithms work. Each output message includes a dictionary created during the compression process. As the size of the input data increases, the compression ratio improves significantly. With a message size of about 1000 bytes, the size of the compressed message reaches about 25% of the original message.

4 Proposed solution

In the following lines, we are going to look at a diametrically different way of data compression in the IoT area. This method is based on the fact that the services of IoT are built on the client-server architecture. The basic idea of the proposed data compression is to separate the dictionary from the data itself in the process of its transfer. The dictionary is built on the application side.

The solution is focused on speed and minimum energy consumption on the equipment side. Many IoT devices are often powered by batteries with limited capacity. A computationally intensive operation is building the dictionary on the application side.

4.1 Building the main dictionary

IoT devices send the status information to the application running on the server. The range of IoT devices is wide, but we still work with a relatively small number of keywords. According to iot.schema.org [10], the number of keywords used in the IoT world is circa 2000. The range of values that these keywords can acquire is much wider, but their spectrum is relatively narrow. We can divide this data into three categories:

- messages defined by the manufacturer *eg* manufacturer identification, device type, device id,
- fixed messages defined during configuration *eg* facility, name, location,...
- dynamic values temperature, speed, binary states

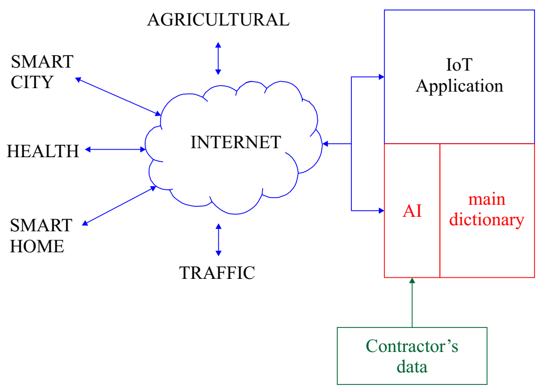

Fig. 1. Building a dictionary mode 1

The dictionary is built in the AI (Artificial Intelligence) module in two modes. In the first mode, the data supplied by the producer of the devices are imported into this module. This data must contain at least the following information:

- the list of the keywords
- the description of the values
- the process of changing values

For our device, the information from the vendor could look like Tab. 1.

The second way is to analyze the data coming from individual devices in the AI module where the learning process takes place. All received data go through an extensive analysis. The results of the analysis are used in the next level of learning, especially in the case of dynamically changing values.

In the learning process, keywords are continuously identified, as well as the values they can acquire. In the first step, the table of the relationship is created.

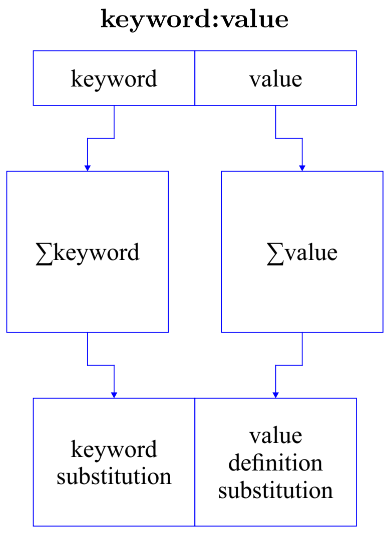

Fig. 2. Building a dictionary mode 2

The keyword group presents a list of all key words used in the communication. Afterwards, these can be replaced by a unique identifier in the compression process. We use a fixed string length to encode keywords. The fixed length provides a fast and energy-efficient data processing during its compression and decompression. In our case we are working with a 2-byte length substitution, see Tab. 2.

Table 2. Keyword substitution

Key word	Substitution
id	#AQ
type	#DE
name	#Me
status	#R5
date	#dK

# is not part of the substitution string

In the value fork, the data analysis process takes place in relation to the keywords. The result of this process can look like in Tab. 3.

Table 3. Result of value analysis process for the keyword status

Key word	Value	Format	Dictionary value	Substitution
status	on	text	ON	#gD
status	off	text	OFF	#3E
status	true	text / logic	ON	#AD
status	false	text / logic	OFF	#4u
status	1	num / logic	ON	#AP
status	0	num / logic	OFF	#4E

These values presented are illustrative and capture the fact that the status keyword appears in several combinations with a value within the application. However, it acquires only 2 predefined values on the selected device. A deeper analysis will lead us to the point where only 2 substitution strings will be used for the status keyword.

Using the previous substitution strings, we can rewrite "status": "off" into

$$\#R5\{\#gD,\#3E,\#AD,\#4u,\#AP,\#4E\}.$$

A more complicated situation occurs when a numeric value, date, or text transmit is required. How to approach this type of variables is solved by the AI module in the data analysis process. Depending on the analysis result, a number transfer may be required in a binary form, datum in unix timestamp format, etc. In this case, the default length of the transmitted chain (2 bytes) is changed. In order to be able to recover such data, we need to extend the dictionary by a variable which defines how to work with the given data type.

The fixed value is directly compressed replaced by a predefined string based on an existing dictionary. In the case of a dynamic variable, the sent string consists of predefined strings extended with necessary data based on the results of the data structure analysis. In this case, the machine learning algorithm recognizes that it is a date and chooses the appropriate way to insert this information into the compressed message ,see Tab. 4.

Table 4. Variable classification

Key word	Substitution	Length
id	#AQ	fixed
type	#DE	fixed
name	#Me	fixed
status	#R5	dynamic
date	#dK	dynamic

For example #dK{str_length}{value} presents datum compression.

In the final stage, we get a dictionary for all devices connected to the system. In robust application, it can be hundreds of thousands to millions of records. The basic idea is to separate the dictionary from the transmitted message, which will reduce its size. To achieve this, it is necessary to upload the created dictionary to the IoT device. It is pointless to upload the main dictionary to each IoT device. For this reason, we will take the position of have to need. Only that part of the dictionary that is relevant to the customer's device is uploaded to the device.

4.1 Building the personal dictionary

A dictionary created for individual customers involves only the part of the main dictionary necessary for their devices. The main dictionary is remapped to the user dictionary. In this step we have the option of further compression. While the main dictionary uses 2, 3 or more bytes for compression, 1 or 2 bytes will suffice on the customer's side, compare Tab. 5.

Table 5. Remapping the main dictionary to the user dictionary

	Main dictionary			User dictionary
	Key_word	Value	Subst	Subst
status:on	#R5	#gD	#R5gD	#uR
status:off	#R5	#3E	#R53E	#F5
...				

In addition to data compression, the proposed solution offers us the possibility to easily identify an attempt to manipulate the data. The customers dictionary represents a reduced number of acceptable strings on the application side. Any unauthorized, unexpected string indicates an attempt to manipulate the data. Based on this, it is possible to generate a new dictionary on the application side and send it to the suspicious device.

5 Simulation

The described solution is under development. The proposed algorithm was tested on a Smart Home simulator. The simulator generated the behavior of 10 devices in the home during 24 hours. The generated messages were in JSON format. The simulation ran in 3 modes:

- each device sent a separate message every minute
- a message was sent every minute and contained data for all 10 devices
- the message was sent only if the status of the device had changed

In the test, we compared the results of message compression by the standard method with the proposed solution. GZIP and bzip2 libraries which were used, operated in default mode. The data was compressed into a zip format. The test results are shown in Tab. 6 to Tab. 9.

Table 6. A message per device

Description	IoT msg	Gzip	bzip2	AI compr.
Messages	14400	14400	14400	14400
Size total	1844517	1531731	1531759	244800
Size average	128	106	106	17
Compression	100%	83%	83%	13%

Table 7. A message for all 10 devices

Description	IoT msg	Gzip	bzip2	AI compr.
Messages	267	267	267	267
Size total	53711	32957	33001	6942
Size average	201	123	124	26
Compression	100%	61%	61%	13%

Table 8. A message after the change of the device status

Description	IoT msg	Gzip	bzip2	AI compr.
Messages	267	267	267	267
Size total	53711	32957	33001	6942
Size average	201	123	124	26
Compression	100%	61%	61%	13%

Table 9. Legend

Messages	Number of generated messages
Size total	The total size of the generated messages
Size average	The average size of 1 message
Compression	The size of the compressed message in% of the original message

6 Conclusion

The test with 10 devices allowed us to work with the substitution of 1 byte when remapping the dictionary. In all cases, the resulting value of data compression of the proposed solution is maintained at 13%. However, in reality it can be assumed that 2 bytes substitution will be used. In that case, we would be around 25%. We achieved this level of compression by standard methods of data compression for 10 devices.

Based on these tests, this algorithm appears to be very efficient. In the following steps it is necessary to build the dictionaries and verify them in real operation.

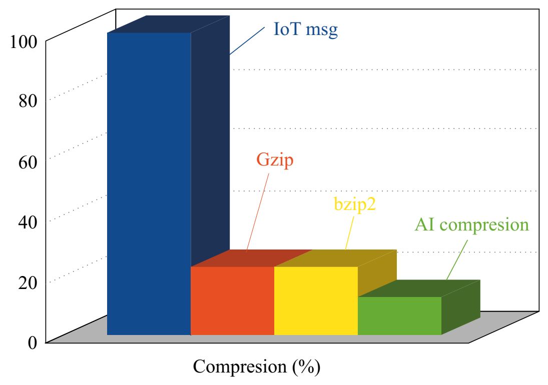

Fig. 3. A message for one device per minute

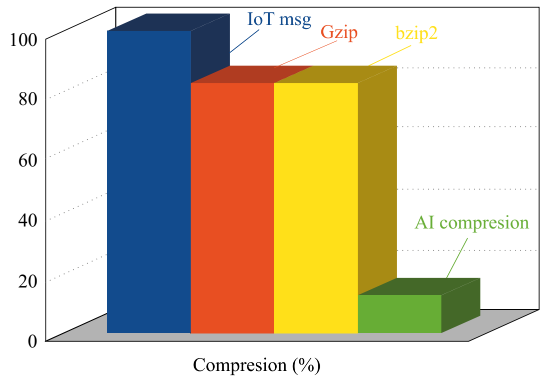

Fig. 4. A message for all 10 devices per minute

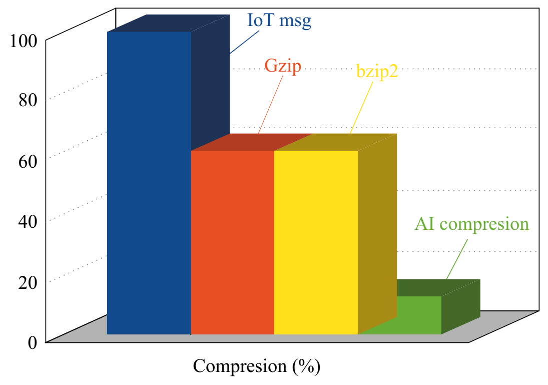

Fig. 5. A message after the change of the device status

References

[1] "Gartner Says 8. 4 Billion Connected 'Things' Will Be in Use 2017, Up 31 Percent From 2016", *Gartner*, 2020, [Online] https://www. gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016. [Accessed: 20-Jul-2020].

[2] "IoT-Prognosen: 75 Milliarden vernetze Geräte" *SAP News Center*, 2020, [Online] https://news.sap.com/germany/2019/10/iot-chance-moeglichkeiten/. [Accessed: 20-Jul-2020].

[3] V. Steinhaus, "Milliarden vernetzte Gerte im Jahr 2022", *It-zoom.de*, 2020, [Online] https://www.it-zoom.de/mobile-busi -ness/e/50-milliarden-vernetzte-geraete-im-jahr-2022-19966/ [Accessed: 20-Jul-2020].

[4] HTML, *En.wikipedia.org.*, 2020, [Online] Available: https://en. wikipedia.org/wiki/HTML [Accessed: 20-Jul-2020].

[5] JSON, *Json. org.*, 2020, [Online] https://www.json.org/json-en. html [Accessed: 20-Jul-2020].

[6] XML, *En. wikipedia org.*, 2020, [Online] https://en.wikipedia. org/wiki/XML [Accessed: 20-Jul-2020].

[7] "Google Code Archive – Long-term storage for Google Code Project Hosting", *Code.google.com*, 2020, [Online] https://code. google.com/archive/p/htmlcompressor/ [Accessed: 20-Jul-2020].

[8] *Cs.uic.edu* 2020, [Online] https://www.cs.uic.edu/ boxu/domino/ rishe-ea-07-sb.xml-compression.eiswt07.published.paper.pdf [Accessed: 20-Jul-2020].

[9] "ECMAScript® 2020 Language Specification", *Ecma-international org.*, 2020, [Online] https://www.ecma-international.org /ecma-262/11.0/ [Accessed: 20- Jul- 2020].

[10] "Home – schema org.", *Iot.schema.org*, 2020, [Online] https://iot. schema.org. [Accessed: 20- Jul- 2020].

Received 6 August 2020

Ivan Sokol was born in 1963, received the Engineer degree in technical cybernetics from the Faculty of Electrical Engineering of the Slovak University of Technology, Bratislava in 1989. He has been working in practice since graduation. He worked for a short time in the area of medical electronics and automation. He currently works at Slovak Telekom a.s. in the position of senior security specialist with focus on the BIG DATA analysis.

Peter Hubinský was born in 1962, has received MSc and PhD degrees from Faculty of Electrical Engineering of the Slovak University of Technology in 1985 and 1992, respectively. After shorter work in praxis in industrial robotics he has returned and worked in the role of assistant, since 1999 Assoc. Professor and 2011 as a full Professor in area of robotics, mechatronics and dynamic system control. He cooperated for long period with foreign partners mainly in Germany. His current research interests are including resonance free control of mechatronic systems, mobile robot visual control and other related areas.