A preliminary study of loop-time delays in IoT platforms: the ThingSpeak case

Vítor Viegas[1,5], J. M. Dias Pereira[2,5], Pedro Girão[3,5], Octavian Postolache[4,5]

[1]*CINAV – Escola Naval, Base Naval de Lisboa, Alfeite, Almada, Portugal*
[2]*ESTSetúbal/IPS, Instituto Politécnico de Setúbal, Setúbal, Portugal*
[3]*Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal*
[4]*ISCTE-Instituto Universitário de Lisboa, Lisboa,*
[5]*Instituto de Telecomunicações, Lisboa, Portugal*
Email: vviegas2@gmail.com

***Abstract* – IoT platforms play an important role on modern measurement systems because they allow the ingestion and processing of huge amounts of data (big data). Usually, these platforms run as a service on the cloud and are accessed through open programming interfaces based on ubiquitous internet protocols (such as HTTP). Data analysis is done in batch, from time to time, or when a pre-configured event is triggered. Most of the platforms can also actuate on the physical world by issuing messages, thus closing the loop ingest-analyze-actuate.**

The continuous investment on IoT platforms has made them extremely reliable and performant, wondering if they can be used to control physical processes. The paper contributes to this discussion by evaluating the loop-time – defined as the delay between ingestion and actuation – of the ThingSpeak platform. The measuring methodology is explained, results are presented, and conclusions are extracted.

Keywords – IoT, ThingSpeak, loop-time, delay, time measurements

I. INTRODUCTION

Commercial IoT platforms are hosted on powerful data centers with high bandwidth, high storage capacity and high processing power. This allows them to ingest, store and analyze huge quantities of data in a robust and continuous way. The availability of big data enables the implementation of new control algorithms (such as predictive control based on artificial neural networks), as well as the improvement of existing features (such as automatic controller tuning, preventive maintenance or just-in-time asset management). For these reasons IoT platforms are very attractive to be used in the control of physical processes.

IoT platforms have being used by the industry at higher levels of automation, primarily for monitoring and supervision [1-7], with the goal of reducing or removing humans in the loop [8]. This is natural since higher business levels resist better to the unpredictability introduced by internet connections. There are some studies for control networks as well [9-10], but the constrains in terms of low-latency and real-time make harder the penetration of IoT platforms at lower automation layers.

Whether IoT platforms are used for monitoring, supervision or real-time control, it is important to know *how fast* and *how reliable* they are. For this purpose, we measured the time taken for a data packet to be ingested by a commercial IoT platform and returned back without any kind of processing. This so-called 'loop-time' is an indicator of *how fast* the platform is. We also compared the content of both packets, outgoing and incoming, to see if it remained the same and thus infer about *how reliable* the platform is. We chose the ThingSpeak platform because it has all the resources a typical IoT platform provides, is free (with some restrictions), and is very easy to use.

The paper is organized as follows: section II gives an overview of the ThingSpeak platform; section III explains the methodology used to measure the loop-time; section IV presents experimental results; and section V extracts conclusions.

II. THINGSPEAK

The ThingSpeak platform provides resources to store and process data in the cloud. The data is accessed using two open application programming interfaces (API): a REST API [11] that communicates over HTTP and follows the request-response model; and a MQTT API [12] that communicates over TCP/IP and follows the publish-subscribe model. Both APIs support data encryption mechanisms and provide authentication through unique read/write keys. The REST API is more popular because it is very easy to use and passes transparently through the network elements (e.g. routers). For these reasons, we used it in this work.

The ThingSpeak platform organizes information in data channels. Each channel includes eight fields that can hold any type of data, plus three fields for location, and one field for status. Each channel is also characterized by a unique ID, a name and a free description. It is not

possible to access the fields individually; all read/write operations are made at the channel level to optimize remote calls. All incoming data is time and date stamped and receives a sequential ID. Channels are private by default, but they can also be made public in which case no read key is required. Channels are provided at no charge for non-commercial, small projects that require less than three million messages/year (or ~8200 messages/day or ~5 messages/minute).

The ThingSpeak provides resources to control the dataflow inside the platform. These resources can be one of the following types:

- React: Executes an action when stored data meets a certain condition (e.g. when a given field of a given channel crosses a given threshold). The action can be as simple as the execution of a script or the issue of a remote message over HTTP.
- TimeControl: Orders the execution of an action once at a specific time, or periodically on a regular schedule, much like a software timer. The TimeControl supports the same actions as the React.
- ThingHTTP: Is a kind of notification over HTTP. It enables communication with remote entities such as devices, websites and web services.

The ThingSpeak platform relies on MATLAB scripts to process stored data. Scripts can be associated to a TimeControl to run one-time or periodically, or to a React to run whenever a given condition is met. Scripts can use the MATLAB toolboxes listed in [13], as long as the developer logs into to ThingSpeak using its MathWorks account and is licensed to use them. This opens the door to powerful data analytics, supported by robust, well-known software libraries. The results can be visualized on the web, directly from the ThingSpeak site, through ready-to-use, fancy charts. The visualization experience can also be enriched with custom widgets and MATLAB plots.

Fig. 1 shows the dataflow through the ThingSpeak platform. Data is ingested, stored and analyzed to extract meaningful information (if needed). The stored data can be visualized remotely through a web browser anytime, anywhere. Messages can be sent to third-party applications to signal a given event. The present work focusses on measuring the time it takes to upload data and receive a reply, assuming no processing is made in the interim.

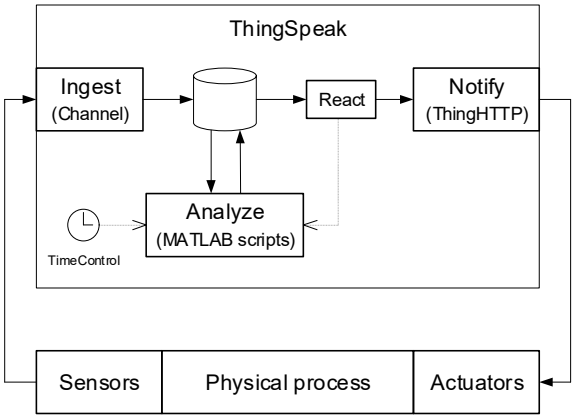

Fig. 1. Dataflow through the ThingSpeak platform.

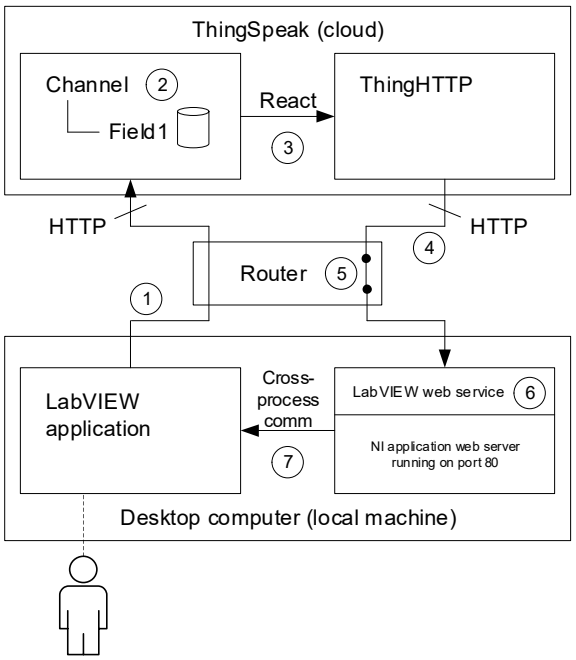

Fig. 2. Closed data path passing through the ThingSpeak platform.

III. METHODOLOGY

To measure the loop-time, we had to build a closed data path passing through the ThingSpeak platform, as shown in Fig. 2. The path includes several actors and stages that can be described as follows (the numbers in the list correspond to the numbers in the figure):

1. LabVIEW application: The LabVIEW application makes HTTP calls to the ThingSpeak platform, collects the replies and computes the time elapsed. Each call is a GET request intended to upload an unsigned 32-bit integer (U32) that is successively incremented. This integer is called 'source' as it uniquely identifies the request. The timestamp of the request, t_0, is registered to serve as reference for the loop-time. The application also provides a graphical interface to the user.
2. ThingSpeak channel (name = TestChannel; ID = 515584; access = private): The channel contains a single field (Field1) to store the source uploaded by the LabVIEW application.
3. React (condition type = numeric; test frequency = on data insertion; condition = TestChannel.Field1 > 0; run = each time the condition is met): A reaction is fired

each time the arriving source is positive, which is always because the source is a U32 number. The reaction instructs the ThingHTTP broker to make the request 78772 (see below).

4. ThingHTTP broker (request = 78772, URL = http://xxx.xxx.xxx.xxx//MyWebService/MyCallBack?Source; method = POST): The broker makes HTTP calls to remote endpoints. In the present case, the broker is configured to make a POST request to a web service running on the same machine as the LabVIEW application. The request sends back the source initially issued by the LabVIEW application.
5. Router: Outgoing and incoming HTTP messages pass through a local router. Outgoing messages are safe and so the router forwards them transparently. Incoming messages, however, are potentially dangerous and are blocked by default. To overcome this problem, we had to forward port 80 on the local router, so that POST requests coming from the ThingSpeak platform reach the LabVIEW web service.
6. LabVIEW web service: The LabVIEW web service is a stateless routine that receives a POST request, extracts the attached source, and registers the timestamp of the reply, t_1. The pair (source, t_1) is then sent back to the LabVIEW application by means of a UDP socket. The web service is hosted by the NI Application Web Server running on port 80 of the local machine.
7. UDP socket (URL = localhost; port = 61557): An external agent is needed to transfer data from the LabVIEW web service to the LabVIEW application because they are two separate processes. That agent is a simple UDP socket that listens for incoming data on port 61557.

The LabVIEW application issues a source and waits for the reply in the form of a pair (source, t_1). If the reply does not arrive within five minutes, or if the *incoming* source is different from the *outgoing* source, an error is registered. If the reply is successful, the loop-time is computed simply as the difference $t_1 - t_0$.

IV. EXPERIMENTAL RESULTS

The LabVIEW application, LabVIEW web service and DataSocket server were executed on a common desktop computer (Intel Core i7-4510U @ 2 GHz, 8 GB RAM, 256 GB SSD, Windows 10 Pro 64 bits). The access to the internet was made through a commercial provider (Vodafone Portugal) using a general-purpose router (Huawei's HG8247Q) with port 80 forwarded.

A. Measurements for one hour

The LabVIEW application was configured to issue sources every 20 s (approximately). Fig. 3 shows the loop-time for one hour of measurements (around 180 points). Fig. 4 shows the content of the data channel as visualized in the ThingSpeak site. Fig. 5 shows the statistical distribution of the loop-time. No errors were found during the test (incoming source always equal to outgoing source).

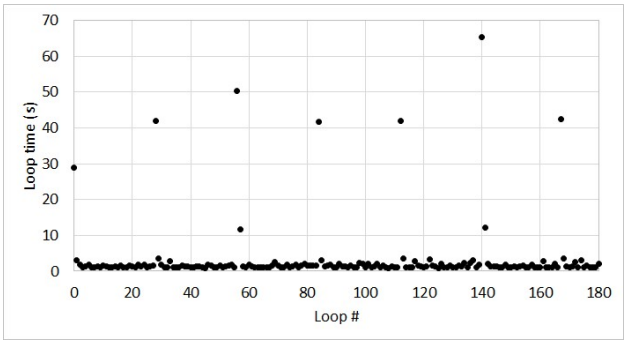

Fig. 3. Loop-time measurements for one hour. Amplitude parameters: maximum = 65.358 s; minimum = 0.927 s.

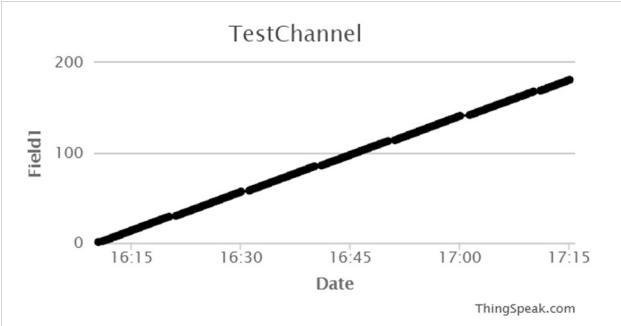

Fig. 4. Visualization of TestChannel.Field1 in the ThingSpeak site.

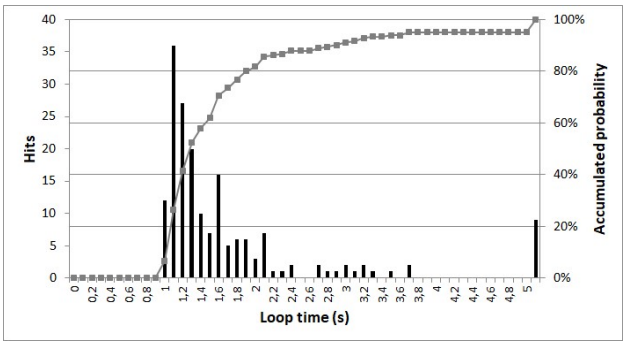

Fig. 5. Loop-time histogram (detail for the interval 0 to 5 seconds). Statistical parameters: mean = 3.256 s; standard deviation = 8.653 s; median = 1.279 s.

From the figures we see that the loop-time is less than 2 s with a probability of 82%. Nevertheless, every 10 min (around 30 points) we see that the loop-time increases very sharply reaching tens of seconds. This suggests that the ThingSpeak platform stores data in temporary buffers, which, from time to time or when they are full, have to be flushed and processed. The absence of errors reinforces the reliability of the platform.

B. Measurements for one day

The LabVIEW application was configured to issue sources every 6 minutes for 24 hours (around 240 points in total). Fig. 6 plots the results obtained. No errors were found during the test.

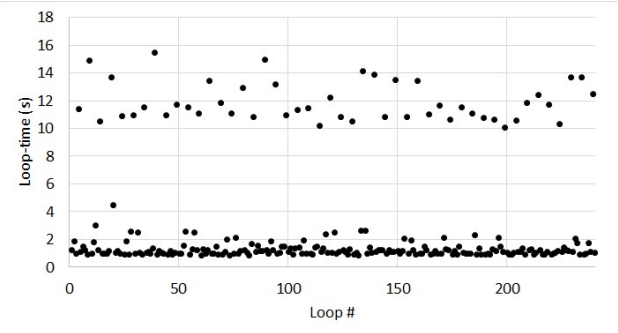

Fig. 6. Loop-time measurements for one day.

From the figure we see that the loop-time remained stable over 24 hours without significant fluctuations during the test. Every 5 points (around 30 min) the loop-time increased above 10 seconds, suggesting, again, the existence of buffering mechanisms. The absence of errors reinforces, again, the reliability of the platform.

V. CONCLUSIONS

In this paper we studied the responsiveness and reliability of the ThingSpeak platform. We measured the time needed for a data packet to loop back through the platform. We also verified if its content has been corrupted during the trip.

Tests were made covering periods of one hour and one day. The ThingSpeak platform revealed to be reliable since no data corruption was detected. The loop-time remained below 2 s during 82% of the time, with no significant fluctuations over the day. Nevertheless, peaks were detected periodically which may be caused by internal buffering mechanisms. This particular behavior deserves further research but has not a critical impact in a large number of monitoring applications supported in the cloud.

REFERENCES

[1] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo, T. Bangemann, "Migration of industrial process control systems into service oriented architecture", 38th Annual IEEE Conference on Industrial Electronics Society (IECON 2012), Oct. 25-28 2012, Montreal, Canada

[2] T. Hegazy, M. Hefeeda, "Industrial automation as a cloud service", IEEE Transactions on Parallel and Distributed Systems, Vol. 26, Issue 10, pp. 2750-2763, Oct. 1 2015

[3] R. Langmann, L. Meyer, "Automation services from the cloud", 11th IEEE International Conference on Remote Engineering and Virtual Instrumentation, 26-28 Feb. 2014, Porto, Portugal

[4] H. Sequeira, P. J. Carreira, T. Goldschmidt, P. Vorst, "Energy cloud: real-time cloud-native energy management system to monitor and analyze energy consumption in multiple industrial sites", 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014), 8-11 Dec. 2014, London, UK

[5] O. Givehchi, J. Jasperneite, "Industrial automation services as part of the cloud: first experiences", Jahreskolloquium Kommunikation in der Automation (KommA 2013), 13-14 Nov. 2013, Magdeburg, Germany

[6] O. Givehchi, H. Trsek, J. Jasperneite, "Cloud computing for industrial automation systems: a comprehensive overview", 18th IEEE Conference on Emerging Technologies & Factory Automation (ETFA),10-13 Sept. 2013, Cagliari, Italy

[7] A. Ito, T. Kohiyama, K. Sato, F. Tamura, "IoT-ready industrial controller with enhanced data processing functions", Hitachi Review, Vol. 67, No. 2, pp. 208-209, February 2018

[8] J. Pretlove, C. Skourup, "Human in the loop", ABB Review 1/2007

[9] L. Wang, A. Canedo, "Offloading industrial human-machine interaction tasks to mobile devices and the cloud", 19th IEEE Conference on Emerging Technology and Factory Automation (ETFA), 16-19 Sept. 2014, Barcelona, Spain

[10] O. Givehchi, J. Imtiaz, H. Trsek, J. Jasperneite, "Control-as-a-service from the cloud: a case study for using virtualized PLCs", 10th IEEE Workshop on Factory Communication Systems (WFCS 2014), 5-7 May 2014, Toulouse, France

[11] "REST API", https://www.mathworks.com/help/thingspeak/rest-api.html (accessed on April 10th 2019)

[12] "MQTT API", https://www.mathworks.com/help/thingspeak/mqtt-api.html (accessed on April 10th 2019)

[13] "Access MATLAB Add-On Toolboxes", https://www.mathworks.com/help/thingspeak/matlab-toolbox-access.html (accessed on April 10th 2019)