ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 6, No. 1, 342-348 (2021)

www.astesj.com

Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ

ISSN: 2415-6698

Study of latencies in ThingSpeak

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

[1]*CINAV – Escola Naval, Base Naval de Lisboa, Almada, 2810-001, Portugal*

[2]*Instituto de Telecomunicações, Lisboa, 1049-001, Portugal*

[3]*ESTSetúbal/IPS, Instituto Politécnico de Setúbal, Setúbal, 2914-508, Portugal*

[4]*Instituto Superior Técnico, Universidade de Lisboa, Lisboa, 1049-001, Portugal*

[5]*ISCTE – Instituto Universitário de Lisboa, Lisboa, 1649-026, Portugal*

ARTICLE INFO

Article history:
Received: 12 November, 2020
Accepted: 07 January, 2021
Online: 22 January, 2021

Keywords:
IoT
ThingSpeak
Latency
Delay
Measurement

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). Given the increasing use of these platforms, it is important to characterize their performance and robustness in real application scenarios. The paper analyzes the ThingSpeak platform by measuring the latencies associated to data packets sent to cloud and replied back, and by checking the consistency of the returned data. Several experiments were done considering different ways to access the platform: REST API, MQTT API, and MQTT broker alone. For each experiment, the methodology is explained, results are presented, and conclusions are extracted. The REST and MQTT APIs have similar performances, with roundtrip times between 1 s and 3 s. The MQTT broker alone is more agile, with roundtrip times below 250 ms. In all cases, the up and down links are far from being symmetric, with the uplink delay showing higher variance than the downlink delay. The obtained results can serve as a reference for other IoT platforms and provide guidelines for application development.

1. Introduction

Platforms for IoT (Internet of Things) have become key components in measurement and control systems because they are able to ingest, store and analyze huge quantities of data, on a 24/7 basis, at reasonable prices. They are hosted on the "cloud", which is a fancy name for data centers spread all over the world, equipped with high bandwidth, large storage capacity, and heavy processing power.

The term "cloud" is interesting because IoT platforms have indeed a broader view of the physical processes, as they were somewhere above in the sky. They have a broader view in terms of space because they gather data from different locations, and a broader view in terms of time because they store data persistently. This new level of awareness has flattened the traditional five-level automation pyramid [1] because field devices can now communicate directly with the cloud. Intermediate levels are being bypassed leading to an horizontal structure that is the basis of "smart factory" and "connected manufacturing" [2], two core concepts of "industry 4.0" [3].

Today, the cloud concentrates huge amounts of data making it the ideal place to run large-scale data analytics. Deep leaning, based on artificial neural networks, has benefited a lot from this scenario because it needs lots of data (big data) to perform well. As long as the data are good (big and diverse), deep learning is able to find good computational models for complex processes, even the hardest ones. With a good model in hands, new things can be done (such as preventive maintenance, and just-in-time asset management), and old things can be improved (such as robust control algorithms, and automatic controller tuning). IoT platforms play a key role in this movement because they are the "stage" where things are happening.

IoT platforms began to be used at the top of the automation pyramid because these levels do not need accurate timing. Typical applications include monitoring and supervision [4]-[10], with the

[*]Corresponding Author: Vítor Viegas, CINAV – Escola Naval, +351210902000, vviegas2@gmail.com

goal of reducing or removing humans in the loop [11]. There are also some applications for closed-loop control [12], [13], but the constrains in terms of low-latency and real-time make harder the penetration of IoT platforms at lower automation levels.

Whether IoT platforms are used for monitoring, supervision, or real-time control, it is important to know how fast and reliable they are. For that purpose, we took a well-known IoT platform – the ThingSpeak platform [14] – and measured the time it takes to upload a data packet and receive a reply. This so-called "roundtrip time" is an indicator of *how fast* the platform is. By checking the resemblance of both packets, outgoing and incoming, we can also have an idea on *how reliable* the platform is. We chose ThingSpeak because it is very easy to use, is open source, is free (with some restrictions), has an active community, and provides a comprehensive set of features, including persistent data storage, data analytics based on MATLAB, easy access through ubiquitous protocols, and security over SSL. For these reasons, the ThingSpeak is one of the makers' favorite platforms [15].

There are similar works trying to characterize experimentally the behavior of cloud servers and IoT platforms. For example, in [16], the author gives a tutorial on network latency measurements using PlanetLab as testbench. He measured the roundtrip times of mobile and non-mobile devices, connected through wireless (WiFi and 3G) and wired (Ethernet) interfaces, while pinging five different AWS servers spread around the world. In [17], the author and his team studied the performance of a cloud database by measuring the time needed to complete a writing on the database and getting back a reaction. They used a Siemens PLC to generate data, the IBM Cloud to store data and fire events, and an industrial computer to catch those events, all connected through a MQTT broker. In [18], the author measured the roundtrip time associated to a MQTT broker when it was accessed from two different continents (Brescia in Europe and São Paulo in South America). In [19], the author evaluated the efficiency and roundtrip time of three protocols commonly used in IoT, namely CoAP (constrained application protocol), WebSockets, and MQTT (message queue telemetry transport). In [20], the author made a quantitative performance analysis of the CoAP and MQTT protocols over various conditions of network capacity, packet loss probability, and link delay. In all these works, a substantial effort was put in characterizing experimentally the behavior of cloud services and the protocols used to access them.

The remaining of this paper is organized as follows: section 2 gives an overview of the ThingSpeak platform; sections 3 and 4 analyze the latencies of the ThingSpeak platform when it is accessed through two different application programming interfaces (API); section 5 focus on the ThingSpeak MQTT broker alone; section 6 discusses the obtained results; and section 7 extracts conclusions.

2. ThingSpeak

The ThingSpeak platform provides resources to store and process data in the cloud. The data are accessed through two well documented APIs: a REST API [21] that communicates over HTTP and follows the request-response model; and a MQTT API [22] that communicates over TCP/IP and follows the publish-subscribe model. Both APIs support authentication through unique read/write keys, but only the REST API supports data encryption through HTTPS. The REST API works well for one-to-one communications, while the MQTT API is best suited for one-to-many communications. The ThingSpeak MQTT broker only supports QoS = 0 (equivalent to "deliver at most once" or "fire-and-forget").

The ThingSpeak platform organizes information in data channels. Each channel includes eight fields that can hold any data type, 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 receive a sequential ID and a timestamp (with a 1 second resolution). 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 projects as long as they require no more than 8200 messages/day (~5 messages/minute).

The ThingSpeak provides the following resources to control the dataflow:

- React: Executes an action when stored data meet 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 remote call over HTTP, useful to communicate with remote entities such as devices, websites, and web services.

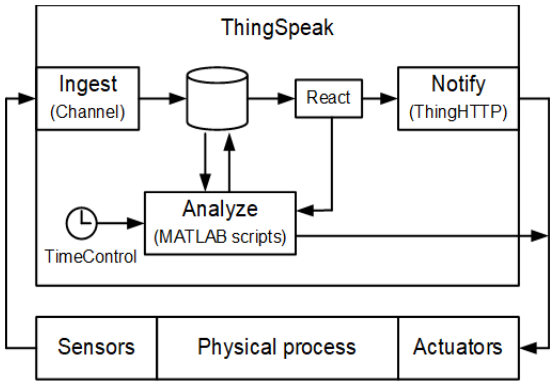


Figure 1: Dataflow inside the ThingSpeak platform

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 [23], as long as the user logs into ThingSpeak using its MathWorks account and is licensed to use them. This opens the door to powerful data analytics, supported by robust and well-known software libraries. The results can be visualized on the web, directly from the ThingSpeak site, through ready-to-use charts. The visualization experience can also be enriched with custom widgets and MATLAB plots.

Figure 1 shows the dataflow through the ThingSpeak platform. Data are ingested, stored in a database, and (optionally) analyzed by scripts that run periodically or when a given condition is met. Messages can be sent to third-party applications by a pre-configured ThingHTTP, or by a pre-programmed script using MATLAB functions. 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.

3. ThingSpeak accessed through the REST API

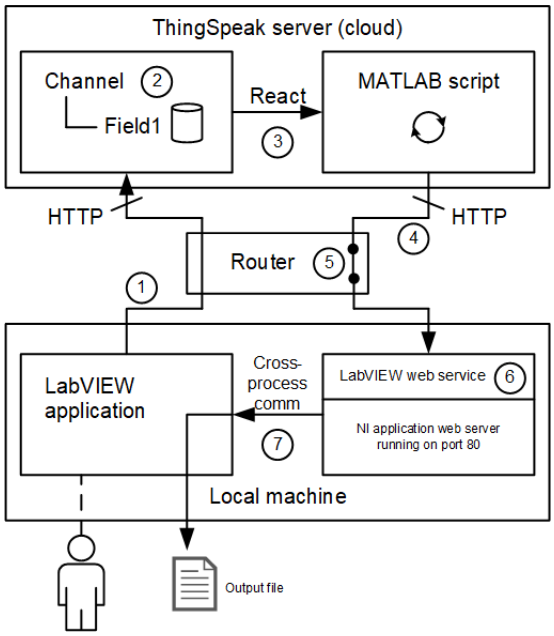
In this section we analyze the performance of the ThingSpeak platform when it is accessed through the REST API. We first explain the methodology used to measure latencies and then we present results.

3.1. Methodology

To measure REST API latencies, we built the closed data path shown in Figure 2, which includes the following stages (the numbers in the list correspond to the numbers in the drawing):

1. LabVIEW application: Is a custom application that runs on our local machine. It makes HTTP calls to the ThingSpeak server, collects the replies, and computes the time elapsed. Each call is a GET request that uploads an order number (n) that is incremented to identify the request. The timestamp of the request (t_{n0}) is registered to serve as reference for the roundtrip time.
2. ThingSpeak channel (name = TestChannel; ID = 515584; access = private): The channel contains a single field (Field1) to store the order number 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 a positive order number is received (which is always because n is an unsigned integer). The reaction instructs the MATLAB script to run (see below).
4. MATLAB script: Collects the timestamp at the ThingSpeak server and makes an HTTP call back to the local machine. As shown in Figure 3, the script is programmed to make a POST request to a web service running on the same machine as the LabVIEW application. The request sends back the order number and the timestamp at the ThingSpeak server (τ_n).
5. Router: Accesses from the local machine to an external server are inherently safe and the router forwards them transparently. However, connections in the opposite direction 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 server reach the LabVIEW web service. In other words, we had to expose the LabVIEW web service to the internet.
6. LabVIEW web service: Is a stateless routine that receives a POST request, extracts the attached order number (n) and timestamp (τ_n), and registers the timestamp of the reply (t_{n1}). The triplet (n, τ_n, t_{n1}) 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.

(n, τ_n, t_{n1}), adds the first timestamp (t_{n0}), and writes the quartet (n, t_{n0}, τ_n, t_{n1}) into the output file for further processing.




```
%get source
readChannelID=515584;
readApiKey='[redacted]';
n = thingSpeakRead(readChannelID,'ReadKey',readApiKey);
%get ThingSpeak's timestamp
tau = datetime('now','Format','HH:mm:ss.sss');
%call back
url='http://95.136.[redacted]:3582/MyWebService/MyCallBack';
%method = POST; media type = application/x-www-form-urlencoded
webwrite(url,'OrderNumber',n,'Tau',tau);
```

Figure 3: MATLAB script. The reading key of the channel and the IP address of the LabVIEW web service were erased for privacy

The LabVIEW application and the LabVIEW web service run both on our local machine (Intel i7-8550 CPU @ 1.80 GHz, RAM 16 GB, SSD 512 GB, NVIDIA GeForce MX150). The machine connects to an Ethernet port of a general-purpose router (model HS8247W from Huawei), which accesses the internet through a fiber optic link provided by Vodafone Portugal.

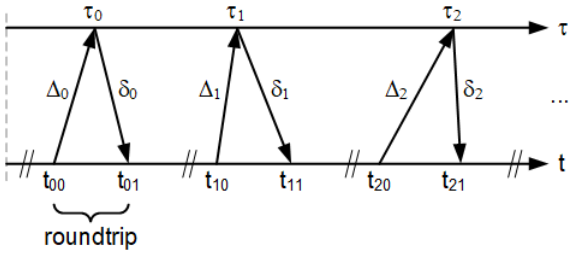
The LabVIEW application uploads order umbers (n) at multiples of 20 seconds to respect ThingSpeak free account limitations. On the nth upload, the quartet (n, t_{n0}, τ_n, t_{n1}) is saved on the nth line of the output file. Thus, it is possible to record the timelines shown in Figure 4, where the τ axis represents the timeline of the ThingSpeak server, and the t axis represents the timeline of the local machine. Of course, the two timelines are not

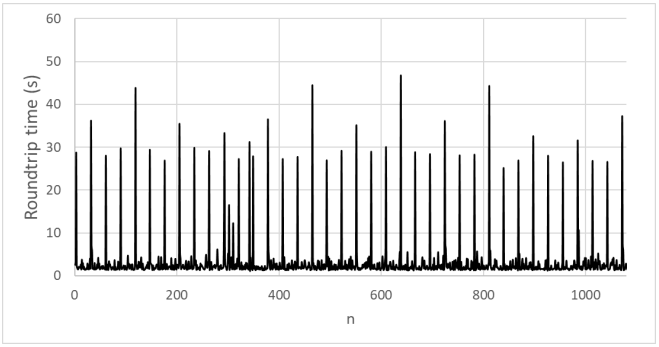
aligned because the clocks of the two systems are not synchronized. Yet, we can extract the following quantities from these timelines:

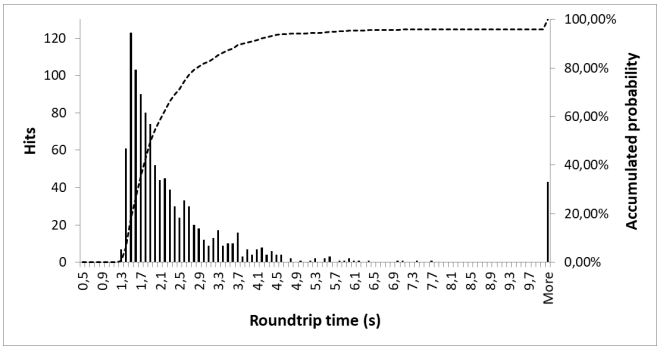
$$RT_n = t_{n1} - t_{n0} \quad (1)$$

$$d\Delta_n = \Delta_{n+1} - \Delta_n = (\tau_{n+1} - \tau_n) - (t_{(n+1)0} - t_{n0}) \quad (2)$$

$$d\delta_n = \delta_{n+1} - \delta_n = (t_{(n+1)1} - t_{n1}) - (\tau_{n+1} - \tau_n) \quad (3)$$







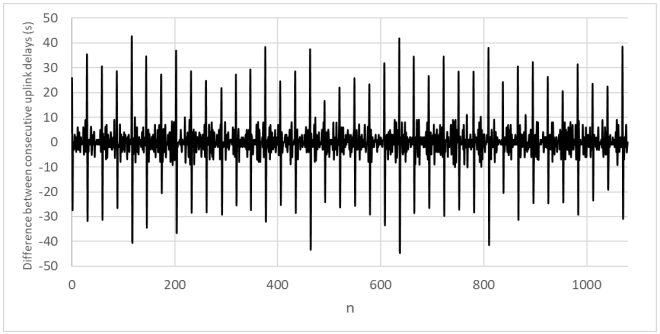
(b)

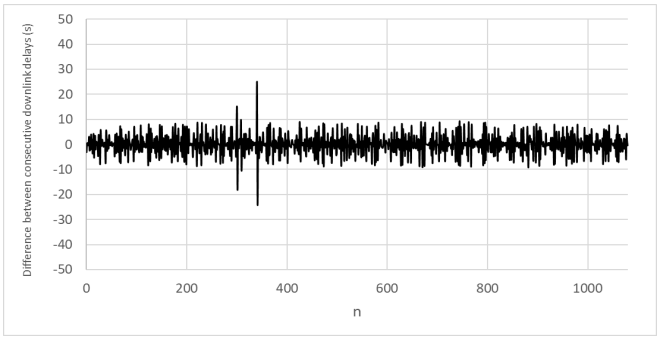
Figure 5: Roundtrip times of the REST API: a) plot; b) histogram

3.2. Results

We ran the closed loop illustrated in Figure 2 for 1080 times, from n=0 to n=1079, waiting approximately 20 seconds on each iteration. This number (1080) was a compromise between having statistically relevant results and limiting the test to a reasonable amount of time (almost six hours). Figure 5a shows the measured roundtrip times, and Figure 5b shows the corresponding histogram. The order number (n) was always replied correctly, which attests the robustness of the ThingSpeak platform.

Figure 5b we see that the roundtrip time has a heavy-tailed distribution, which is characteristic of multipath communication mediums as the Internet. The authors in [24] report similar results and suggest a lognormal distribution for the experimental data. In our case, we got mode = 1,66 s and a roundtrip time that is less than 2.9 s with a probability of 80%.





(b)

Figure 6: Difference between consecutive delays: a) uplink; b) downlink

Nevertheless, every 10 min (around 30 points) the roundtrip time increases very sharply up to tens of seconds, suggesting that the ThingSpeak platform stores data in temporary buffers, which, from time to time, are flushed and processed.

Figure 6 shows the difference between consecutive delays on the uplink and downlink directions. The differences are positive and negative because a higher delay on one iteration discounts on the next iteration. As expected, the uplink delay (Δ) is less stable than the downlink delay (δ) because the upload process is more complex than the reply (in terms of ThingSpeak internals).

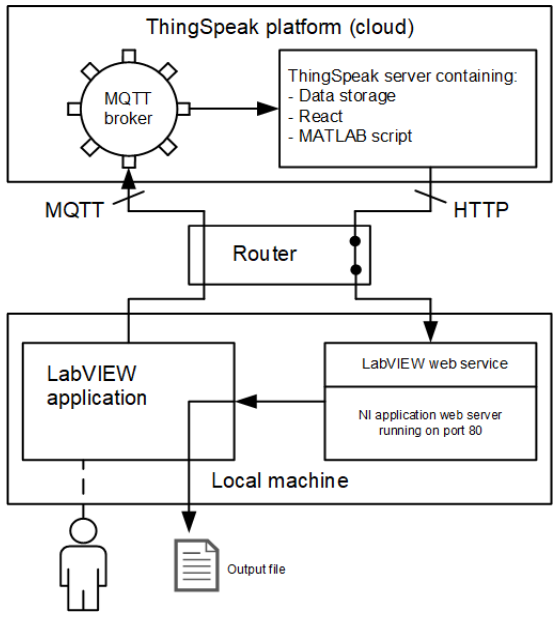
If we suppose that consecutive uplink delays are statistically independent, which makes sense because they correspond to different iterations, then the variance of the difference (dΔ or dδ) will be twice of the variance of the variable itself (Δ or δ). Therefore, if we compute the variances Var(dΔ) and Var(dδ) from the data of Figure 6, and divide the result by two, we get the variances of the delays: Var(Δ) = 37.52 s^2 and Var(δ) = 7.29 s^2. Applying the squared root, we get the standard deviations: σ_Δ = 6.13 s and σ_δ = 2.70 s.

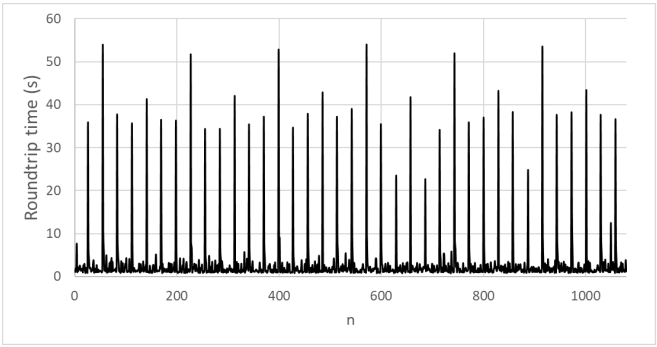
Finally, it is very difficult to infer about the mean value of the uplink and downlink delays because the clocks of the two systems

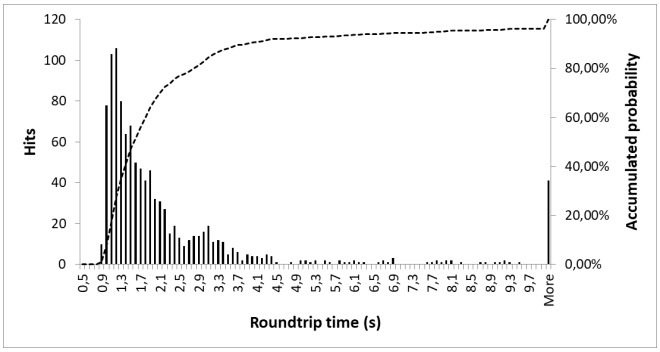
are not synchronized and, equally important, the two links are far from being symmetric.

4. ThingSpeak accessed through the MQTT API

built the closed data path shown in Figure 7, which is like that of Figure 2 with the difference that the order numbers are uploaded through the MQTT broker. The LabVIEW application was changed to publish order numbers (n) using a compatible MQTT driver [25]. The quartets (n, t_{n0}, τ_n, t_{n1}) were collected and saved into the output file as before.







(b)

Figure 8: Roundtrip times of the MQTT API: a) plot; b) histogram

We ran the closed loop 1080 times waiting approximately 20 seconds on each iteration. Figure 8a shows the measured roundtrip times, and Figure 8b shows the corresponding histogram. Again, the order number (n) was always replied correctly, with no mismatches or timeouts. From the graphs, we see that the roundtrip time follows a lognormal distribution with mode = 1,13 s and values below 2,8 s with a probability of 80%. We also see outliers coming every 10 min (as we saw in the REST API), suggesting that internal mechanisms of buffering and batch processing also apply to the MQTT API.

The random variables dΔ and dδ were also computed as before. The corresponding variances were extracted and divided by two, leading to Var (Δ) = 51.53 s^2 and Var (δ) = 5.99 s^2. Applying the squared root, we got the standard deviations of the uplink and downlink delays: σ_Δ = 7.18 s and σ_δ = 2.45 s.

5. MQTT broker alone

To analyze the performance of the MQTT broker alone we closed the loop without passing through the ThingSpeak server, as shown in Figure 9. The LabVIEW application was changed to publish order numbers (n) and subscribe the replies using the MQTT driver previously mentioned. In this case, only the triplets (n, t_{n0}, t_{n1}) were collected because the timeline of the ThingSpeak server (τ axis) is not available.

We ran the closed loop 1080 times waiting approximately 20 seconds on each iteration. Figure 10.a shows the measured roundtrip times, and Figure 10.b shows the corresponding histogram. As always, the order number (n) was always replied correctly, with no mismatches or timeouts. From the graphs, we see that the roundtrip time follows a lognormal distribution with mode = 0.189 s, and values below 0,250 s with a probability of 80%. We also see that the outliers observed in the previous tests have almost disappeared. This shows that the MQTT broker is more expeditious than the ThingSpeak server, probably because its buffering and processing needs are much less demanding.

6. Results and Discussions

Table 1 summarizes the latencies measured during all the experiments. From these results we can extract the following conclusions:

- The REST and MQTT APIs have similar performances with a slightly advantage for the MQTT API. In both cases, the roundtrip time was typically between 1 s and 3 s.
- The ThingSpeak server has internal mechanisms of buffering and batch processing that, periodically, introduce extraordinary delays. These mechanisms seem to be absent from the MQTT broker.
- The MQTT broker is very agile in distributing publications, with roundtrip times typically below 250 ms.
- The up and down links are far from being symmetric. The uplink is more complex since it has a higher variance.
- The tests lasted for several hours and were made in different days and in different times of the day, suggesting that the behavior of the ThingSpeak platform is time independent.

A final word about robustness: the order number (n) was never lost or corrupted, proving that the ThingSpeak is reliable, even for free accounts.

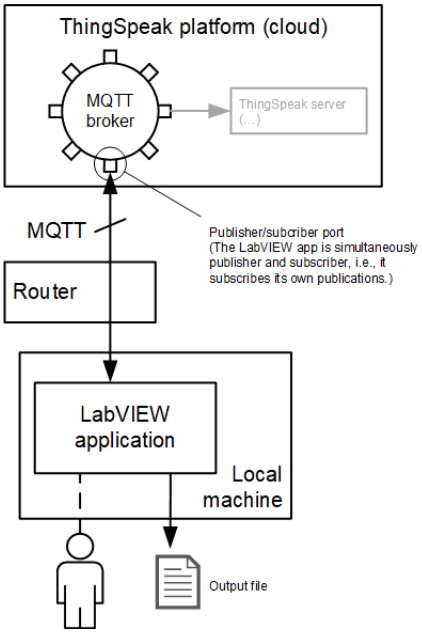
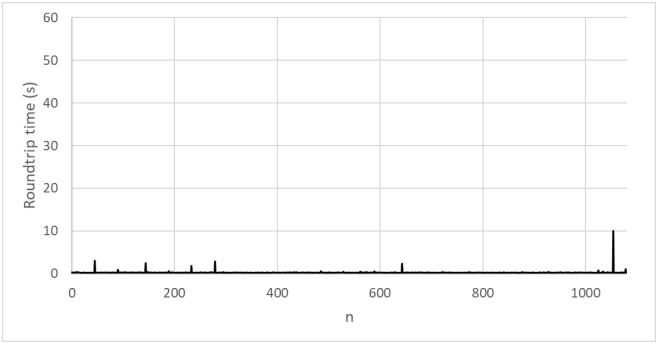
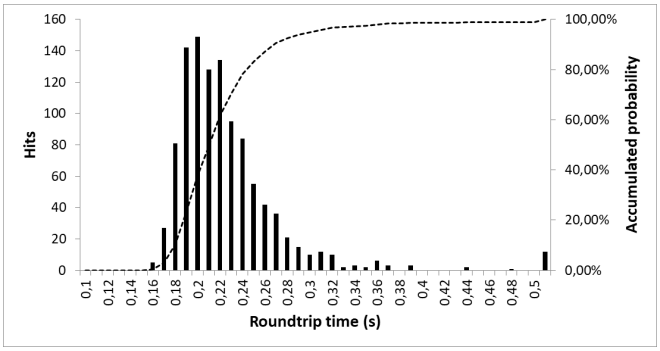


Figure 9: Data loop through the MQTT broker only



(a)



(b)

Figure 10: Roundtrip times of the MQTT broker alone (a) plot (b) histogram

Table 1: Experimental characterization of the ThingSpeak platform

		REST API	MQTT API	MQTT broker
Roundtrip time	**Min (s)**	1.25	0.86	0.158
	Mean (s)	3.28	3.27	0.243
	Mode (s)	1.66	1.13	0.189
	80th percentile (s)	2.90	2.80	0.250
Uplink delay	**Std. dev. (s)**	6.13	7.18	---
Downlink delay	**Std. dev (s)**	2.70	2.45	---

7. Conclusions

The paper reported the studies carried on the ThingSpeak platform to evaluate its responsiveness and reliability. We measured the time needed for a data packet to loop back through the platform, and we verified if its content has been corrupted during the trip. Tests were made for all access mediums (REST API, MQTT API and MQTT alone) covering periods of six hours.

We saw that the REST and MQTT APIs have similar performance with typical roundtrip times between 1s and 3s. We observed repetitive outliers that suggest that the ThingSpeak server has periodic mechanisms of buffering and batch processing. The MQTT broker alone did not show such outliers and performed significantly faster. In terms of reliability, no data was lost or corrupted.

We hope that the obtained results can serve as a reference for other IoT platforms and provide guidelines for application development.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] "ANSI/ISA-95.00.03-2013 Enterprise-Control System Integration - Part 3: Activity Models of Manufacturing Operations Management"

[2] R. Burke et al., "The smart factory - Responsive, adaptive, connected manufacturing," Deloitte Insights, https://www2.deloitte.com/us/en/insights/focus/industry-4-0/smart-factory-connected-manufacturing.html (accessed on 6 Oct. 2020).

[3] K. Schwab, "The fourth industrial revolution - What it means and how to respond," Foreign Affairs, 12 Dec. 2015, https://www.foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution (accessed on 6 Oct. 2020).

[4] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo and T. Bangemann, "Migration of industrial process control systems into service oriented architecture," IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, Montreal, QC, 2012, 5786-5792, doi: 10.1109/IECON.2012.6389039.

[5] T. Hegazy and M. Hefeeda, "Industrial automation as a cloud service," in IEEE Transactions on Parallel and Distributed Systems, **26**(10), 2750-2763, 1 Oct. 2015, doi: 10.1109/TPDS.2014.2359894.

[6] R. Langmann and L. Meyer, "Automation services from the cloud," 2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV), Porto, 2014, 256-261, doi: 10.1109/REV.2014.6784271.

[7] H. Sequeira, P. Carreira, T. Goldschmidt and P. Vorst, "Energy cloud: Real-time cloud-native energy management system to monitor and analyze energy consumption in multiple industrial sites," 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, 2014, 529-534, doi: 10.1109/UCC.2014.79.

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

[9] O. Givehchi, H. Trsek and J. Jasperneite, "Cloud computing for industrial automation systems - A comprehensive overview," 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), Cagliari, 2013, 1-4, doi: 10.1109/ETFA.2013.6648080.

[10] A. Ito, T. Kohiyama, K. Sato, F. Tamura, "IoT-ready industrial controller with enhanced data processing functions," in Hitachi Review, **67**(2), 208-209, Feb. 2018.

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

[12] L. Wang and A. Canedo, "Offloading industrial human-machine interaction

tasks to mobile devices and the cloud," Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), Barcelona, 2014, 1-4, doi: 10.1109/ETFA.2014.7005249.
[13] O. Givehchi, J. Imtiaz, H. Trsek and J. Jasperneite, "Control-as-a-service from the cloud: A case study for using virtualized PLCs," 2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014), Toulouse, 2014, 1-4, doi: 10.1109/WFCS.2014.6837587.
[14] "About ThingSpeak," https://thingspeak.com (accessed on 11 Oct. 2020).
[15] "Top IoT platforms for makers," https://ubidots.com/blog/top-iot-platforms/ (accessed on 30 Oct. 2020).
[16] Minseok Kwon, "A tutorial on network latency and its measurements," in Enabling Real-Time Mobile Cloud Computing through Emerging Technologies, IGI Global, 2015, 272-293, doi: 10.4018/978-1-4666-8662-5.ch009.
[17] Paolo Ferrari, Emiliano Sisinni, Alessandro Depari, Alessandra Flammini, Stefano Rinaldi, Paolo Bellagente, Marco Pasetti, "On the performance of cloud services and databases for industrial IoT scalable applications," in Electronics 2020, **9**(9), 1435, doi: 10.3390/electronics9091435.
[18] P. Ferrari, E. Sisinni, D. Brandão and M. Rocha, "Evaluation of communication latency in industrial IoT applications," 2017 IEEE International Workshop on Measurement and Networking (M&N), Naples, 2017, pp. 1-6, doi: 10.1109/IWMN.2017.8078359.
[19] S. Mijovic, E. Shehu and C. Buratti, "Comparing application layer protocols for the Internet of Things via experimentation," 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), Bologna, 2016, 1-5, doi: 10.1109/RTSI.2016.7740559.
[20] M. Collina, M. Bartolucci, A. Vanelli-Coralli and G. E. Corazza, "Internet of Things application layer protocol analysis over error and delay prone links," 2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), Livorno, 2014, 398-404, doi: 10.1109/ASMS-SPSC.2014.6934573.
[21] "REST API," https://www.mathworks.com/help/thingspeak/rest-api.html (accessed on 6 Oct. 2020).
[22] "MQTT API," https://www.mathworks.com/help/thingspeak/mqtt-api.html (accessed on 6 Oct. 2020).
[23] "Access MATLAB Add-On Toolboxes," https://www.mathworks.com/help/thingspeak/matlab-toolbox-access.html (accessed on 6 Oct. 2020).
[24] I. Antoniou, V.V. Ivanov, Valery V. Ivanov, P.V. Zrelovc, "On the log-normal distribution of network traffic," in Physica D: Nonlinear Phenomena, **167**(1-2), 72-85, July 2002. doi: 10.1016/S0167-2789(02)00431-1.
[25] "mqtt-LabVIEW," https://github.com/cowen71/mqtt-LabVIEW (accessed on 8 Oct. 2020).