Dynamic Cloud Service Management for Scalable Internet of Things Applications

Alex Mavromatis
High Performance Networks
University of Bristol
Bristol, United Kingdom
a.mavromatis@bristol.ac.uk

Sam Gunner
Intelligent Transport Systems
University of Bristol
Bristol, United Kingdom
sam.gunner@bristol.ac.uk

Theo Tryfonas
Engineering Systems & Design
University of Bristol
Bristol, United Kingdom
theo.tryfonas@bristol.ac.uk

Dimitra Simeonidou
High Performance Networks
University of Bristol
Bristol, United Kingdom
dimitra.simeonidou@bristol.ac.uk

Abstract— **The Internet of Things (IoT) is growing rapidly and official reports suggest that soon it will become one of the key technologies of the Internet infrastructure. Cloud services are already integrated with IoT, providing solutions for a large number of applications. Various Cloud IoT platforms are available for IoT users, both as open source and commercial solutions. One of the important features of a cloud IoT platform is the management of services and the deployment time required to set up a platform dedicated to a specific IoT use case. This paper presents a cloud IoT platform deployment mechanism that aims to improve the deployment time between an IoT network and a cloud IoT platform. The approach is based on a platform descriptor which describes the IoT application requirements. Therefore, software-based functionalities automate the provisioning of the platform to accommodate scalability in cloud IoT applications. The experimentation and evaluation of the proposed mechanism is conducted over a real data-center laboratory. We are comparing our mechanism with a Unix bash shell scripting deployment method and the results achieve over 50% improvement in both platform deployment time, as well as the time required to update the platform.**

Keywords— IoT, Cloud, Scalability

I. Introduction

The Internet of Things (IoT) is touted as being on track to revolutionize much of the way the world works, and real-time remote sensing is already allowing optimizations to be made across a broad range of sectors, from manufacturing to public transport, finance to healthcare. The remote device is only part of the technology picture however, as the amalgamation and processing of data at scale has to happen on hardware not constrained by power and size limitations. Physical servers are an option, but to deliver a scalable solution without the hassle of managing their own server farm system designers are increasingly turning to Cloud Computing.

Cloud Computing and IoT are two technologies that, when used together, are able to deliver end-to-end services and applications. Research and Development into the combination of these technologies is already gathering momentum, and implementation at scale is not trivial. The billions of connected devices expected by 2020 [1] mean that the ability to implement at scale is crucial, and this is as true for the researchers in the field as it is for industry.

This work has been supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 691735 (project REPLICATE).

Albeit state-of-art cloud platforms provide effective and reliable service, there is a need for greater automation in the way researchers specify and implement cloud platform instances, as this will aid the rapid deployment of experiments. Presently the provision and deployment of resources, both computing and storage, is most commonly done through a graphical user interface, and although this generally provides a user-friendly mechanism for launching single instances, it quickly becomes cumbersome as the number of deployments increase.

It is for this reason that we propose our Cloud IoT platform. The Cloud IoT Platform is a service providing solution for IoT deployments, aiming to facilitate the easy provisioning of cloud resource for IoT applications.

In this paper we present the cloud IoT platform architecture and show how it facilitates users in generating cloud-based resources tailored to the needs of their cloud based IoT applications. Our mechanism equips the user with a descriptor file that abstracts the platform elements and introduces a DevOps logic to IoT cloud service provisioning. With these descriptors a user can implement a dynamic resource allocation algorithm based on the application's needs, moving closer towards fully automated cloud IoT resource management. By automating this process our work aims to significantly reduce the deployment time of an end-to-end IoT application.

Directly after this introduction Chapter 2 gives some related work, followed by a detailed explanation of the Cloud IoT Platform architecture in Chapter 3. Chapter 4 provides some context for the platform, using a number of examples to describe how it would be used, Chapter 5 then evaluates the Cloud IoT Platform, and Chapter 6 contains our final conclusions.

II. Related Work

Cloud computing is well established in the industry where services are provided to users for a specific price. In the IoT era there is a lot of research on the integration with cloud and how the device and data management can be

A survey [3] on cloud IoT providers presents the selection of services and options available today in the market. This paper described the services that the cloud providers make available to users. The basic technologies and elements of IoT are similar across the platforms, however the implementation and costs are different. Furthermore, although all the platforms enable a wide range of applications to be developed, they do not permit on the fly reconfiguration of the platform resources. In [4] the authors present a methodology for selecting the most appropriate cloud platform for an application. Commercially available platforms are evaluated against a list of requirements taken from a number of use cases, and the results of this are analysed to aid cloud platform selection. However, the authors do not address questions on how possible future changes in requirement can be managed, or how resources on these platforms can be modified to accommodate new applications.

A general study in [5] presents the building blocks of a cloud IoT architecture. Furthermore, the authors analyse different applications that use cloud infrastructure for IoT and describe the advantages of the cloud implementation. This study does not focus on the need to have a generic cloud platform for new applications and IoT requirements. Further to having a generic approach on to the IoT platform architecture, increases in scale exacerbates the requirement to engineer efficient solutions, thereby optimizing the deployment time of an IoT application. The authors in [6] present the principles of the IoT cloud systems and how the whole lifecycle of an IoT application works. This study is interesting since the authors present an analysis of the end-to-end IoT deployment and highlight that the deployment and provisioning phase for IoT has to be diverse and on demand. Our proposed scheme facilitates the later by being based on continuous integration techniques where a deployment descriptor defines the cloud platform capabilities.

Improvement of IoT deployments are a common goal within R&D. The work in [7] presents a cloud IoT architecture for improving the deployment process and time. The authors implement the architecture and show improved results comparing the architecture with other IoT platforms. We identify that not only the deployment process needs to be optimized, but also the deployment control has to be accessible to the application user, therefore our system implements a REST Application Interface (API) able to provide updates on the IoT platform descriptor. This paper is focused on improving the cloud IoT services deployment and allowing the user to define the dynamic allocations of the cloud IoT platform based on the user's needs.

efficient and secure [2]. The challenge of managing billions of devices dynamically is open and R&D moves towards the direction of automation and DevOps to enhance scalability and efficient management.

III. Cloud IoT and Proposed Architecture

A. *Proposed Architecture & Platform Life-Cycle*

Figure 1 presents the proposed architecture, a bottom up design where the experimenter sets up an IoT network and then requires cloud resources to accommodate the data processing. The architecture comprises four layers where the first one is the IoT devices, with the second layer being the physical infrastructure such as computing and storage capacity within a data center. The third layer contains the middle-ware developed for this research. This middleware is comprised by two applications:

- **Physical Resource Manager (PRM):** The first software element developed for this paper is the PRM which is responsible for allocation of physical resources (such as storage, RAM and CPU capacity) for the IoT application. This is a nodeJS application running on the cloud server that allocates docker containers based on the descriptor provided by the user. A REST API is implemented and the application then waits for a POST request, the body of which is the descriptor JSON object.
- **IoT Services Manager (ISM):** The ISM is also a nodeJS application running at the cloud servers. ISM is the application responsible for pairing the platform service IDs to specific containers created by the PRM therefore providing isolation between the different users of the platform. Each platform user can have one or many unique service IDs. The ISM reads the descriptor and creates the user's Service ID, Service Path and authorization token, all of which are stored to a database. Finally, the ISM returns all the information to the user, including the user credentials and the platform instance details.

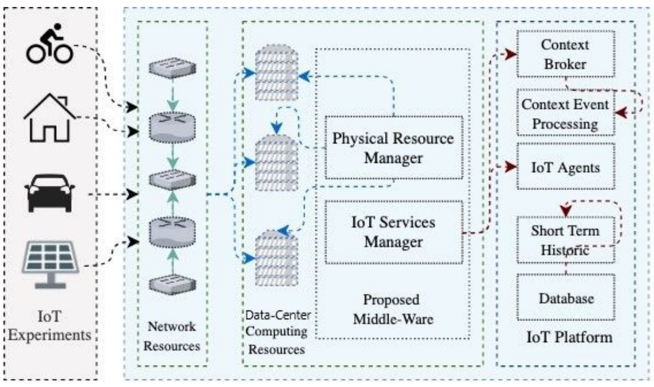
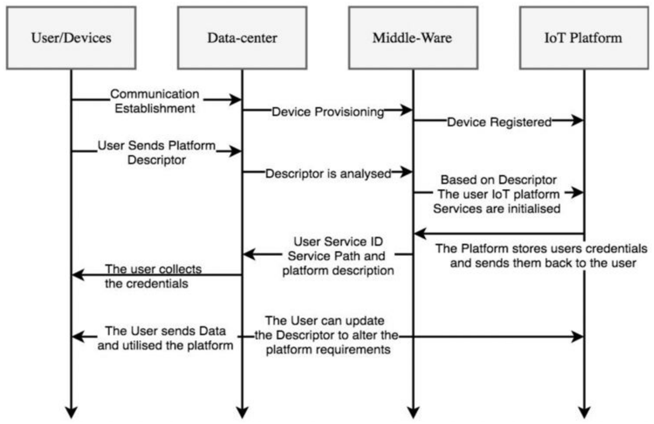


Fig. 1. Overall IoT Provisioning Architecture.

Figure 2 depicts the life cycle of the platform in the form of a flow diagram, showing all the transactions between the devices and the cloud services. The first step when establishing communications between a device and the IoT platform is the device provisioning [8]. This process is manual or dynamic, depending on the implementation. Once the device is authorized for communication data must be retrieved by the IoT platform and be made accessible to the user. At this stage the aforementioned ISM application is waiting for a platform descriptor, in which the user defines all the application requirements. The middleware software initializes the platform and creates instances for the end-user based on the descriptor details. Then it returns the platform instance information to the end-user (i.e. IP endpoint address, platform instance description etc.).



User/Devices
Data-center
Middle-Ware
IoT Platform
Communication Establishment
Device Provisioning
Device Registered
User Sends Platform Descriptor
Descriptor is analysed
Based on Descriptor The user IoT platform Services are initialised
User Service ID Service Path and platform description
The Platform stores users credentials and sends them back to the user
The user collects the credentials
The User sends Data and utilised the platform
The User can update the Descriptor to alter the platform requirements

Fig. 2. Flow Diagram of Proposed Scheme for Provisioning a New Device.

The platform descriptor is a JSON object created by the user and interpreted by the ISM application component of the Cloud IoT Platform. Once the devices are provisioned the user is presented with an interface which can be used to post the descriptor and initialize the platform equipped with the desired services and capacity. An example of the platform descriptor is presented in Figure 3. The descriptor requires all the fields specified in Figure 3, starting with the application name and a short description. The Service ID, Service Path and token are the user credentials within the platform. The ISM application reads the remainder of the descriptor and provisions the IoT platform accordingly, including the granting of access to specific IoT data protocols, allocating of a cap to the number of messages per day that can be sent, and launching the necessary platform applications. Once all the information about the user's application requirements are stored, the PRM executes any created container instances as detailed in the 'Resources' field of the descriptor. The 'Platform-Instances' element allows the user to specify the number of containers on which to run the platform. The descriptor also gives the user the option to enable load balancing where multiple instance are created.

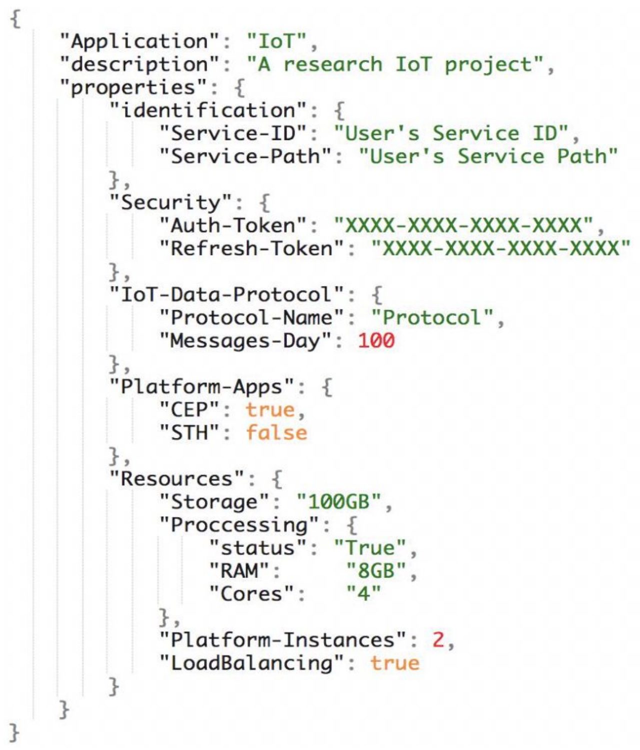

```
{
    "Application": "IoT",
    "description": "A research IoT project",
    "properties": {
        "identification": {
            "Service-ID": "User's Service ID",
            "Service-Path": "User's Service Path"
        },
        "Security": {
            "Auth-Token": "XXXX-XXXX-XXXX-XXXX",
            "Refresh-Token": "XXXX-XXXX-XXXX-XXXX"
        },
        "IoT-Data-Protocol": {
            "Protocol-Name": "Protocol",
            "Messages-Day": 100
        },
        "Platform-Apps": {
            "CEP": true,
            "STH": false
        },
        "Resources": {
            "Storage": "100GB",
            "Proccessing": {
                "status": "True",
                "RAM":     "8GB",
                "Cores":    "4"
            },
            "Platform-Instances": 2,
            "LoadBalancing": true
        }
    }
}
```

Fig. 3. An Example Descriptor Snippet Showing Fields Available to a System Designer: Our system uses this descriptor to provision the appropriate cloud computing resource.

B. Cloud IoT Platform Building Blocks

A Cloud IoT platform comprises different software elements running over an infrastructure that is accommodating data generated from IoT networks. Our implementation is based on the open source IoT platform FIWARE [9] the building blocks of which are presented in Figure 1. Our FIWARE IoT platform implementation contains five main building blocks:

- **Context Broker (CB):** The CB element is the main feature of the platform, accommodating multi-tenant users and the data from different IoT deployments. The operation of the CB is initialized by creating credentials for the user which are used for sending and retrieving the data being generated from the IoT Devices. The implementation of FIWARE requires a service id and a service path which are unique, and must used within any IoT data protocol. The CB implements a publish/subscribe philosophy able to accommodate a large number of users, assuming the resources permit.

- **IoT Agents:** This element is responsible for the convergence of different IoT data protocols. This includes protocols such as HTTP, MQTT and CoAP which are already enabled on the platform. This component allows extendibility, providing an API that can be used for developing new data handling agents.

- **NoSQL Database:** The third component, used to store the user's data, is a NoSQL MongoDB database integrated with the CB component. The CB is able to isolate the data using the service ID and therefore store the information using the service ID as a key.

The database is mainly used for historical data and analytics.

- **Complex Event Processing (CEP):** CEP is a powerful tool that allows the triggering of events and can be used to inform both the CB and applications themselves. These events are flexible and can be configured to enable a range of use cases. Furthermore, CEP can detect patterns that occur and alert the application concerned. This component enhances the platform by making it more dynamic and flexible. Although complex, and offering a range of additional tools, a level of abstraction is used and this simplifies significantly the task of creating effective events handlers. CEP is integrated with the CB, it can therefore monitor real time actions based on the CB subscription.
- **Short Term Historic (STH):** This module is an instance of the CB and is responsible for storing the historical data. This feature is very important for the application developer since it provides targeted data through a RESTful API which simplifies the use of the platform. For the purpose of this paper we have extended the STH component to consistently store data to the NoSQL database described above.

IV. Case Study Applications

The range of possible IoT applications is huge, but at the core of almost all is the collection, storage and processing of data, with the aim of aiding decision making through the condensing of (potentially massive) datasets down to a far smaller number of meaningful, salient pieces of information. In this paper we will briefly look at one such IoT application, and describe how the Cloud IoT Platform can aid both the initial deployment and subsequent enhancement of a project. The focus here is research, where specific goals (and far less specific methodologies) are known at the point of project conception. In this environment it is crucial that systems can be deployed quickly and easily, thereby lowering the barriers that might prevent a researcher from implementing her or his next idea.

A. Deployment of Real Time Civil Infrastructure Monitoring

One example of an IoT Application is 'Smart Infrastructure'. All developed nations are reliant on their civil infrastructure, and in many cases, this is aged and operating well outside its original design capacity (such as the metro networks of London or New York). Building new civil infrastructure is extremely expensive and can be very disruptive, so instead there is a desire to extend the life of existing infrastructure through proactive, preventative maintenance. It is hoped that by deploying a large number of IoT sensors onto a structure enough data can be gathered to allow accurate models to be made of it, and so predict failures before they happen. One extreme example of this is Hong Kong's Stone-cutter Bridge which has been instrumented with more than 1,700 sensors [10].

Bridge dynamics mean that in some situations a relatively low number of sensors can still provide useful information [11], so a smaller scale deployment might only consist of six structural sensors. Modern IoT wireless accelerometers can be very quickly deployed in key locations, with the use of low power wireless communications protocols such as 802.15.4 allowing extended duration deployments to run on a single set of batteries. A local wireless gateway can then relay vibration data in real time to a cloud server, possibly using a message broker system such as MQTT, where it could be stored in a real-time database such as InuxDB.

If samples are taken at 32Hz this will result in a total of 192 measurements a second, with the total number of messages likely to be between 200 and 250 per second due to system diagnostics. With these low load requirements, a VM of quite modest specification can be used, InuxDB's documentation [12] suggests that 4 cores and 4GB of RAM will be sufficient, and an extra 4GB of RAM is recommended for the MQTT message broker. HD space will depend on the duration of the deployment, but 500GB would sustain a deployment of many years.

With the requirements identified, our Cloud IoT Platform allows the necessary VM to be deployed extremely quickly. Values can be inserted into the descriptor, which is then uploaded to the platform to trigger the automatic provisioning of the necessary cloud resource. Such a deployment mode is ideal for circumstances where low-cost, temporary and/or opportunistic monitoring may be required, such as e.g. in [13].

B. Reconfiguration of Resources

Although initially set, as the research project progresses it is likely that system requirements will evolve. For example:

- The sample rate might be increased to give visibility of higher frequency components of the structural response.
- Sensors might be added to give information about vibrations along a different axis.
- Algorithms, developed using historical data, might be adapted to process real time data using a framework such as Spark.
- Non-technical project stakeholders, such as the infrastructure manager, might desire visibility of the data and so a user interface (e.g. Grafana) might be integrated into the system.
- Other data source, such as vehicle count, might be added to enrich the data.

Each of these changes has a required increase in computing resource associated with it. This might be more HD space and RAM to handle the increased number of messages, or more CPU cores to enable the real time processing and WebUI. In either case the Cloud IoT Platform can be used to quickly relocate the necessary resources with very little manual configuration required by the user. Instead, a modified service descriptor can be uploaded to the API, and the platform will automatically reconfigure the system to match the new requirements.

V. Experimental Setup & Evaluation

To evaluate our platform we have conducted a series of experiments designed to compare the deployment time and update time of a cloud IoT platform. The analysis is evaluating our platform descriptor based applications for cloud IoT platform deployment. As comparison we are using Unix bash shell scripting to automate the deployment process and evaluate both within the same experimentation environment. The results we present between the two approaches are average values with confidence interval of 95%.

A. Experimentation Set up

The experimentation set-up has been assembled within the High Performance Network Group at the University of Bristol data-center facilities. We have deployed the IoT platform and the ISM and PRM applications. We are emulating IoT devices for two different applications. During the experimentation the configuration of the platform is done through the descriptor, and this is benchmarked against traditional configuration done using bash shell scripts. The experiment ran for 24 hours and we compare the different approaches in terms of deployment time and platform update time. The experiments run twice, first implementing an emulated IoT networks of 100 devices, and then again with 1,000 devices. Connectivity is over WiFi and each experiment is repeated, first using MQTT as the IoT data protocol, and then using HTTP.

B. Evaluation & Results

In Figure 4(a) we present evaluation results of the IoT platform *deployment time* for 100 devices. This is the time that the system requires to configure all the user requested details concerning the platform. We are comparing the proposed descriptor-based solution with a bash shell script deployment mechanism. In both IoT data protocols our solution outperforms the shell script approach and the deployment time is reduced by over 50%.

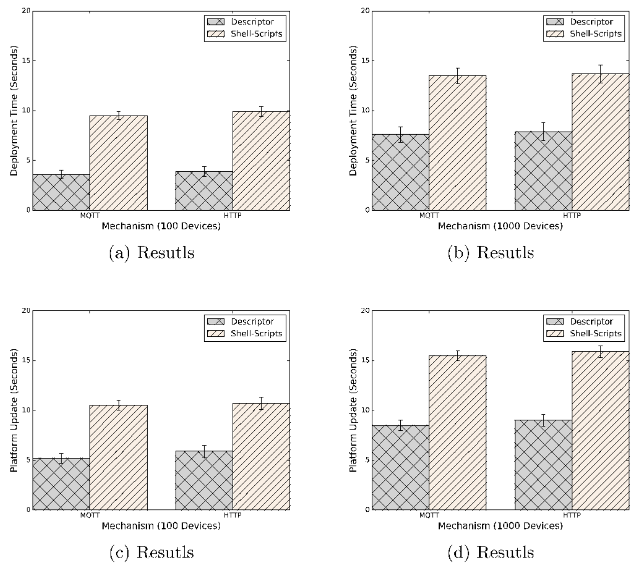


Fig. 4. Testing Results: deployment time of 100 & 1,000 nodes (a)-(b); configuration update performance for 100 & 1,000 nodes (c)-(d).

Figure 4(b) presents results of the deployment time for 1,000 IoT devices. The descriptor approach is still requiring 50% less time, as the larger size of the IoT network does not have any impact on the relative deployment performance of our proposed approach.

To further evaluate the system, we ran experiments where the user of the platform updates the platform's configuration. In Figure 4(c) the results of the configuration update time are presented. As shown, the descriptor mechanism is faster in updating the platform, therefore presents better performance for the 100 IoT devices scenario. Similarly Figure 4(d) presents the results for 1,000 IoT devices. The results show that our proposed architecture is not affected by IoT network size and still performs better than the generic bash shell script deployment set up.

VI. Conclusion

As IoT technologies evolve rapidly, multiple applications are developed both for research and industrial purposes. This paper focused on end-to-end IoT architecture, specifically addressing the problem of provisioning cloud resource for IoT applications. The cloud element of IoT applications is already a key enabler, as methods of storing and processing the large amount of generated data are required. A variety of cloud solutions are available today and most of them are able to facilitate the majority of the IoT requirements. We investigate the deployment time of an IoT platform and how devices are mapped to a data-center infrastructure. Our target is to present a mechanism able to reduce the deployment time and generally improve the control of the cloud IoT platform from the user perspective. An end-to-end architecture is presented including software components developed to strengthen our approach. The evaluation section of this paper is comparing the proposed mechanism with a Unix-based bash shell scripting technique for cloud IoT platform deployment. Our mechanism outperforms the shell scripting technique and provides improvements of 50% on the deployment and update time of a Cloud IoT platform based application. Our future research will aim to improve the quality of service of more complex applications by enhancing the networking within the data-center hosting the cloud IoT elements.

Acknowledgment

This work has been supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 691735 (project REPLICATE).

References

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything", Cisco Systems, 2011, Accessed March 2018.

[2] J. Zhou, Z. Cao, X. Dong and A. V. Vasilakos, "Security and Privacy for Cloud-Based IoT: Challenges", in IEEE Communications Magazine, vol. 55, no. 1, pp. 26-33, January 2017.

[3] T. Panzner and A. Kertesz, "A survey of IoT cloud providers", 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2016.

[4] P. Ganguly, "Selecting the right IoT cloud platform", 2016 International Conference on Internet of Things and Applications (IOTA), Pune, 2016.

[5] O. Mazhelis and P. Tyrvinen, “A framework for evaluating Internet-of-Things platforms: Application provider viewpoint”, 2014 IEEE World Forum on Internet of Things.

[6] H. L. Truong and S. Dustdar, “Principles for Engineering IoT Cloud Systems”, in IEEE Cloud Computing, vol. 2, no. 2, pp. 68-76, Mar.-Apr. 2015.

[7] V. C. Emeakaroha, N. Cafferkey, P. Healy and J. P. Morrison, “A Cloud-Based IoT Data Gathering and Processing Platform”, 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, 2015.

[8] Alex Mavromatis et al., “A Software Defined Device Provisioning Framework Facilitating Scalability in Internet of Things”, 2018 IEEE 5G World Forum (5GWF), Santa Clara, California, 446-451.

[9] FIWARE, https://www.fiware.org/, Accessed in September 2017.

[10] Eddy Dascotte, Jacques Strobbe, Ulf T. Tygesen, “Continuous stress monitoring of large structures”, 5th International Operational Modal Analysis Conference, Guimaraes, 2013.

[11] J.H.G. Macdonald “Pedestrian-induced vibrations of the Clifton Suspension Bridge, UK”, Proceedings of the Institution of Civil Engineers Bridge Engineering 161(2), 2008.

[12] InuxDB V1.5 Docs “Hardware sizing guidelines”, https://docs.influxdata.com/influxdb/v1.5/guides/hardware_sizing/, Accessed May 2018.

[13] S. Gunner, P.J. Vardanega, T. Tryfonas, J.H.G. Macdonald and R.E. Wilson, “Rapid deployment of a WSN on the Clifton Suspension Bridge, UK”, Smart Infrastructure and Construction, 170(3), 59-71, 2017.