A Private Gateway for Investigating IoT Data Management

Tamas Pflanzner and Attila Kertesz
Software Engineering Department, University of Szeged, 6720 Szeged, Dugonics ter 13, Hungary

Keywords: Internet of Things, Cloud Computing, Simulation, Gateway.

Abstract: By responding to the new trend represented by the appearance of the Internet of Things (IoT), several cloud providers have started to offer specific management services. In recent years, we have already seen that cloud computing has managed to serve IoT needs by making data generation, processing and visualization tasks transparent to the users. In IoT Cloud systems developers do not only have to buy and configure sensor devices, but they also have to develop so-called gateway applications to manage the data comings from these devices. In this paper we show how to develop such a private gateway, and present a comprehensive simulation environment, where IoT Cloud applications can be investigated without initial investments. Finally, we evaluate the proposed gateway with real sensor data.

1 INTRODUCTION

Internet of Things (IoT) systems follow a new trend, representing a dynamic global network infrastructure with self configuring capabilities (Sundmaeker et al., 2010), in which things can interact and communicate among themselves and with the environment through the Internet by exchanging sensor data, and react autonomously to events and influence them by triggering actions with or without direct human intervention. Such systems can be utilized in many application areas, thus they may have very different properties. According to recent reports in the IoT field (e.g. (Mahoney and LeHong, 2011)), there will be 30 billion devices always online and more than 200 billion devices discontinuously online by 2020. Such estimations call for smart solutions that provide means to interconnect and control these devices in an efficient way.

Cloud computing (Buyya et al., 2009) enables flexible resource provisions that have become hugely popular for many businesses to take advantage of responding quickly to customers demands. There is a growing number of cloud providers offering IoT-specific services, since cloud computing has the potential to serve IoT needs such as hiding data generation, processing and visualization tasks. With the help of these virtualized solutions, user data can be stored in a remote location and can be accessed from anywhere.

By realizing these capabilities, there are several, popular cloud platform providers available offering IoT specific services. Some of these IoT features are unique, but every IoT cloud provider has the basic capabilities to connect and store data from devices. Creating and examining such applications within IoT cloud environments represent a great challenge, since many things have to be managed at the same time, and a wide range of devices and data formats are available.

By addressing these needs, the main contributions of this paper are the followings: first, we characterize the problem area, then introduce how to develop real cloud gateway services to manage the simulated devices composing a complex semi-simulated environment. Finally, we evaluate the performance and scalability of the proposed solution.

The goal of our research with this proposal is to support the proliferation of integrated IoT, mobile and cloud technologies, and to enable experimenting with complex systems fulfilling the efficient management of user applications. By using our proposed tools, real things or devices can be substituted by mimicking their behavior, thus there is no need to perform initial investments by buying real sensors or devices to investigate IoT applications.

The remainder of this paper is presented as follows: Section 2 discusses emerging technological improvements clarifying the research scope and introduces an overview of related works. Section 3 presents our proposed IoT Cloud simulation environment, and also discusses the development possibilities of private cloud gateways for managing IoT de-

Pflanzner, T. and Kertesz, A.
A Private Gateway for Investigating IoT Data Management.
DOI: 10.5220/0006775505260532
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 526-532
ISBN: 978-989-758-295-0

vices, then it evaluates its scalability and device management features. Finally, the contributions are summarized in Section 5.

2 RELATED WORK

Cloud computing has become a widespread and reliable solution over the past decade. Overcoming interoperability issues of public cloud providers and various middleware implementations, the process of creating and managing cloud federations is clarified and applied (Kertesz, 2014). There are various reasons to optimize resource management in such federations: to serve more users simultaneously, to increase quality of service, to gain higher profit from resource renting, or to reduce energy consumption or CO2 emissions.

As depicted in Figure 1, in the past decade we experienced an evolution in Cloud Computing: the first clouds appeared in the form of a single virtualized datacenter, then broadened into a larger system of interconnected, multiple datacenters. As the next step cloud bursting techniques were develop to share resources of different clouds, then cloud federations were realized by interoperating formerly separate cloud systems. Once overall optimization issues in cloud federations addressing datacenter consolidation, operating costs and energy efficiency were developed, further research directions started to use clouds to support newly emerging domains, such as the Internet of Things. In the case of IoT systems, data management operations are better placed close to their origins, thus close to the users, which resulted in better exploiting the edge devices of the network. Finally, as the latest step of this evolution the group of such edge nodes formed the fog. Dastjerdi and Buyya defined fog computing as a distributed paradigm (Dastjerdi and Buyya, 2016), where cloud storage and computational services are performed at the network edge. This new paradigm enables the execution of data processing and analytics application in a distributed way, possibly utilizing both cloud and near-by resources. The main goal is to achieve low latency, but it also brings novel challenges in real-time analytics, stream processing, power consumption and security.

There are many simulators available to examine distributed and specifically cloud systems. Nevertheless, there are some more specific IoT simulators addressing similar issues compared to this study. Han et al. (Han et al., 2014.) have designed DPWSim, which is a simulation toolkit to support the development of service-oriented and event-driven IoT applications with secure web service capabilities. SimIoT (Sotiriadis et al., 2014.) was derived from the SimIC simulation framework. It introduces several techniques to simulate the communication between an IoT sensor and the cloud, but it is limited to compute activity models. The SimpleIoTSimulator (SimpleIoTSimulator, 2017) is an IoT sensor or device simulator that is able to create test environments made up of thousands of sensors and gateways on a computer. It supports many of the common IoT protocols (e.g. CoAP, MQTT, HTTP). Its drawback is that it needs a specific, RedHat Linux environment, while our approach is more heterogeneous, and focuses on IoT device simulation with mobile devices, which is easier to be applied. The Atomiton simulator (Atomiton, 2017) is probably the closest solutions to our concept. It manages virtual sensors, actuators and devices with unique behaviors. With this tool complex, dynamic systems can be created for specific applications. Unlike our open, mobile solution, it is a commercial, web-based environment with very limited documentation.

Kang et al. introduced the main types and features of IoT gateways in (Kang et al., 2017). This work is a detailed study in this regard, and presents the state-of-the-art and research directions in this field. In this paper we does not aim to propose a generic solution for all needs of an IoT system, but to provide a gateway solution that can be used together with MobIoTSim (Pflanzner et al., 2016) to enable a comprehensive simulation environment for investigating IoT clouds.

3 DEVELOPING A PRIVATE GATEWAY

3.1 A Comprehensive IoT Cloud Simulation Environment

The architecture of our vision for simulating IoT Cloud systems is depicted in Figure 2. Unlike traditional simulators, we try to stay as close to real world systems as possible. We take trace files saved from public applications, place them into an archive (I.), and use them in MobIoTSim (Pflanzner et al., 2016) (II.) to mimic real device behavior. We also develop gateway services (III.) to process and visualize sensor data and instantiate and operate them at private or public cloud providers.

The basic usage of the simulator is to: (i) connect the application to a cloud, where the data is to be sent, (ii) create and configure the devices to be simulated, and (iii) start the (data generation of the) required devices. These main steps represented by three main

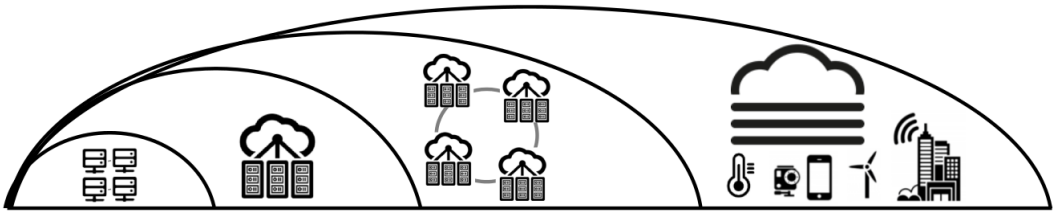

Figure 1: Evolution of cloud systems.

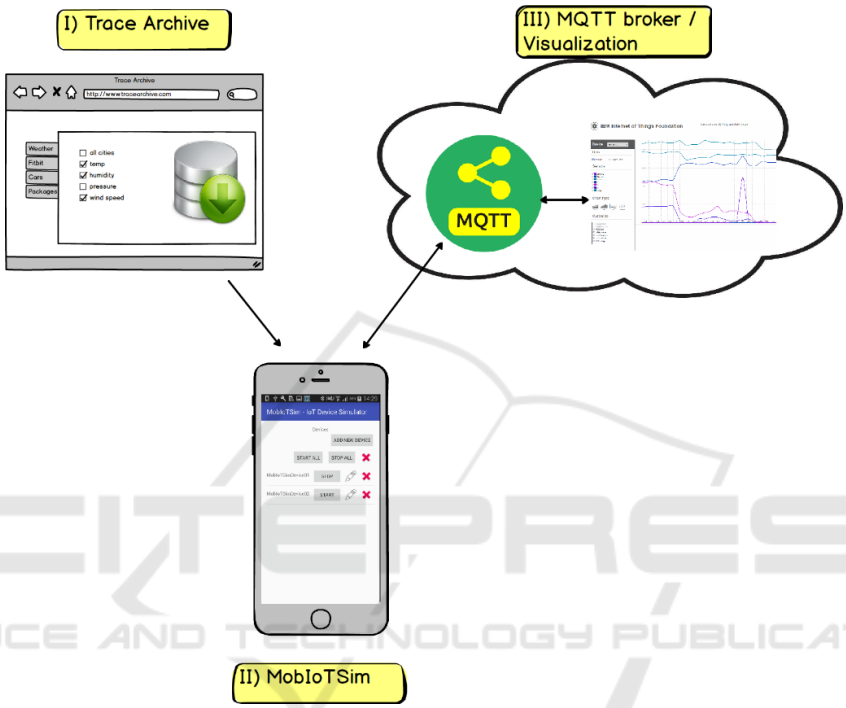

Figure 2: Simulating IoT Cloud systems.

parts of the application: the *Cloud settings*, the *Devices* and the *Device settings* screens.

We also developed an own, private gateway service (available in (Private Gateway, 2017)) for the Bluemix platform that is able to manage several devices simultaneously, and can send a notification to the MobIoTSim device simulator by responding to critical sensor values. This gateway service is basically an extended version of the IBM visualization application (IBM Watson, 2017). It has a web-based graphical interface to visualize sensor data coming from MobIoTSim. Messages (defined in JSON format) received from the simulated devices are managed by an MQTT server. It can also be used to send responses (or notifications) back to the simulated IoT devices in MobIoTSim.

It is easy to connect the simulator to such a private gateway. Since it has a predefined template called *Bluemix*, we only need to specify an organization identifier and the connection type (either TCP or TLS) to enable connection to the MQTT server. In the case of already available templates, the necessary URL is predefined. The application ID, the authentication key and token can be retrieved by registering to the gateway service, and these parameters can be used later to sign in to the data visualization site of the gateway. The simulated devices also need to be registered to the MQTT server of the gateway service – just like real devices –, by specifying their device and type identifiers and sensor data thresholds, which replies with their token identifiers (to be used for device setting). Once these settings are made, simulated devices can be defined and started in the same way as shown previously for the demo gateway. We can create advanced scenarios with this private gateway, such as managing more devices and responding to critical sensor data coming from the simulated devices.

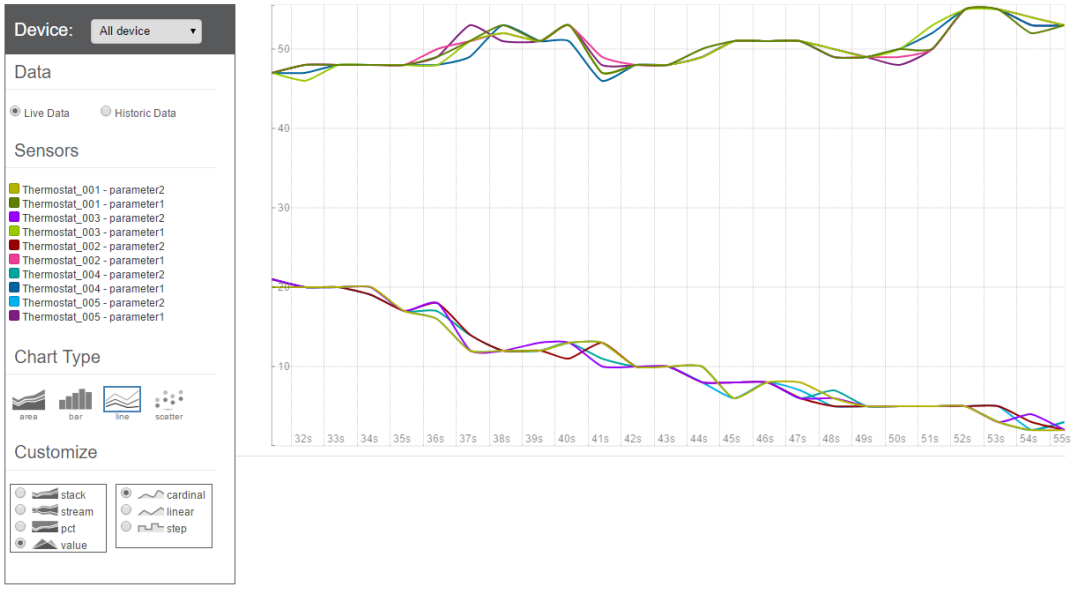

Figure 3: Visualizing the data in Bluemix sent by a group of MobIoTSim devices with the private gateway.

3.2 Evaluation

As a next step, we further revised our private gateway for Bluemix to handle up to hundreds of devices at the same time. This way we can see the data sent by all devices in a single real-time chart as shown in Figure 3. The node.js application managing the chart subscribes to the device topics with the MQTT protocol and waits for the messages. In this extended gateway, we use paging to overcome device management limitations (25 devices at a time). In order to enhance and better visualize many device data at the same time, we introduced device grouping for the chart generation.

Though the Bluemix platform provides some monitoring information for application services, custom Docker containers can be better monitored. For this purpose, we created the Docker version of our private gateway, which even became more portable this way.

We also examined and evaluated this gateway service. We deployed the private gateway in a Docker container as a micro application with the following parameters: OS - Ubuntu with Linux kernel 4.4.0, RAM - 256 MBs, CPU - Intel Xeon E5-2690 with 48 cores.

First, as an initial evaluation, we used the Thermostat template of MobIoTSim to create device groups of 100 and 450 simulated devices. We performed the simulations for 10 minutes and generated sensor messages in every second. We used the Grafana tool in Bluemix to monitor the resource consumption concerning CPU, memory and network usage. The results with median values can be seen in Figure 4. For

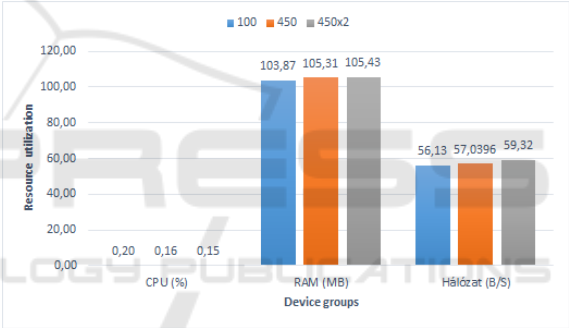

Figure 4: Initial evaluation with Thermostat devices.

the third case in the diagram we reset the message generation interval to 0.5 seconds, therefore we got 900 (denoted by 450x2) messages in every second. As we can see from the diagram we experienced only little load in all cases. The reason for this is that the gateway application only received the messages and forwarded them to a javascript-based chart generator run in a browser at the client side. So we performed no real data processing tasks, though in real world we may do that. As a result, we extended our private gateway by introducing stressing processes to support more realistic IoT operation. For CPU stressing we implemented a parameterizable Fibonacci number generation. We used the setting to count the 20th Fibonacci number upon each received message multiplied by a score representing the size of the message. (We suppose that larger messages need more processing computation.)

For the next experiment, we enabled stressing and created 10, 100 and 250 thermostat devices in groups.

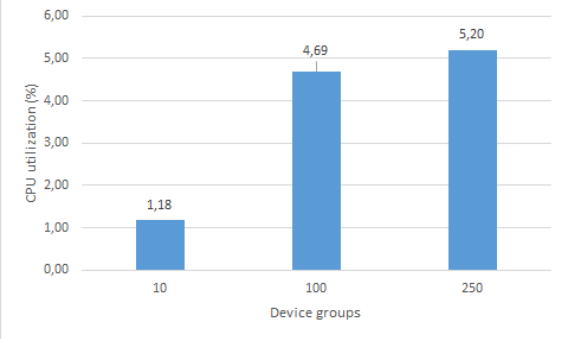

Figure 5: Evaluation results with CPU stressing.

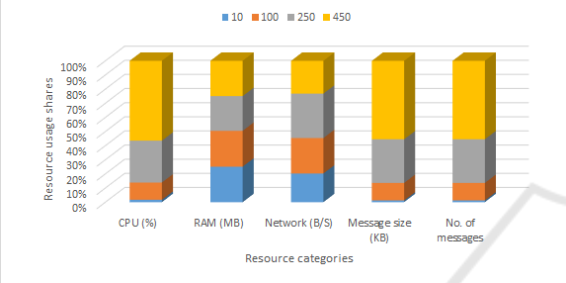

Figure 6: Comparison of different device groups.

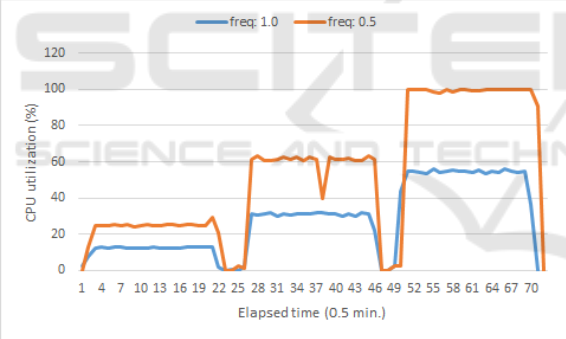

Figure 7: Comparison of different data generation intervals.

The averages of the measured results can be seen in Figure 5. Now we can see that the load of the CPU is increasing as we manage a higher number of devices at the same time. Therefore we extended this measurement and compared 10, 100, 250 and 450 devices in groups. Figure 6 shows the average results for the device groups and the appropriate resources. Next, we re-executed this scenario by reducing the data generation frequency to 0.5 seconds. The comparison of continuous measurements for 100, 250 and 450 devices of the 1 and the 0.5 seconds message generation can be seen in Figure 7.

Once we got some impression on working with a simple thermostat device, we chose a more complex one. As one of the earliest examples of sensor networks are from the field of meteorology and weather prediction, we chose to model a meteorological service and gathered real data from OpenWeatherMap (OpenWeatherMap, 2017). This service provides a global geospatial platform supporting the development and operation of data-driven products for agriculture, transportation and alike. It monitors and publishes actual weather conditions and forecast for more than 200 thousand cities using data from more than 40 thousand weather stations. Their database includes historical data, which is accessible through APIs.

As a final evaluation step, we used the Weather template of MobIoTSim to create device groups of 100 and 450 simulated devices. By using this template we can set up weather station parameters following the OpenWeatherMap format, or load previously saved OpenWeatherMap traces of certain cities. For this experiment, we randomly picked weather data of cities (one city for one simulated device) from earlier traces. Figure 8 shows the difference of the applied two templates in sizes, while Table 1 provide the detailed median measurement results. We can see that as the number of devices grows, the resource utilization also gets higher. Figure 9 highlights CPU utilization comparison of the two device types.

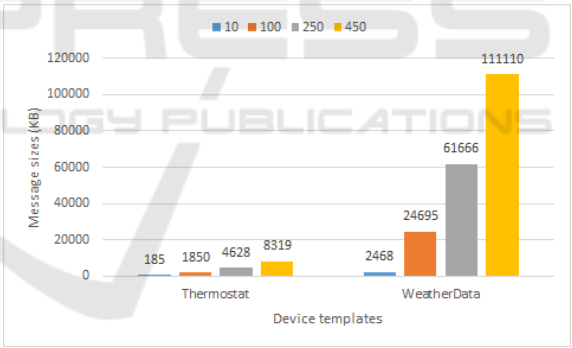

Figure 8: Comparison of message data sizes.

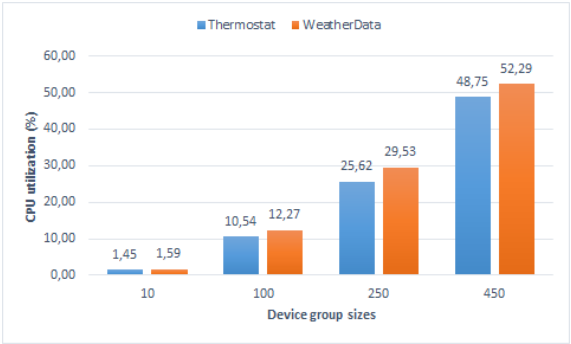

Figure 9: Comparison of CPU utilization of the two device templates.

Table 1: Results of measurements with the Weather template.

No. of devices	10	100	250	450
CPU util. (%)	1.59	12.27	29.53	52.29
Memory (MB)	110.07	110.22	110.35	110.05
Network (B/s)	890.66	881.34	855.16	853.60
Message size (KB)	2468	24695	61666 1	11110
No. of messages	6000	60046	149940	270165

Table 2: MQTT brokers.

	VerneMQ	Moquette	RabbitMQ	ActiveMQ	JoramMQ	Mosquitto
Price	free	free	free	free	free	free
Opensource	yes	yes	yes	yes	yes	yes
License	Apache 2.0	Apache 2.0	Mozilla Public	Apache 2.0	LGPL	Apache 2.0
Custer	yes	no	yes	yes	yes	no
Linux	yes	yes	yes	yes	yes	yes
MacOS	yes	yes	yes	yes	na	na
Windows	no	yes	yes	yes	yes	yes
Language	Erlang	Java	Erlang	Java	Java	C

4 FUTURE EXTENSIONS

Our future plan is to create a platform-independent gateway service, which can be later deployed to any cloud. Such a service should contain an MQTT broker to help the communication between the IoT environment and the cloud application. A graphical interface for a thing database is required to register IoT devices and generate API keys for cloud applications. The device registration makes the communication more secure.

We have already started to evaluate existing, open-source MQTT brokers, which should be the base of a platform-independent gateway service. Our criteria for the appropriate MQTT broker is to be open-source, scalable and extendable for dynamic device registration and management. In Table 2 we provide an overview of the most popular MQTT brokers we started to investigate. These brokers are the followings: VerneMQ (VerneMQ, 2017) (VerneMQ source, 2017), Moquette (Moquette, 2017) (Moquette source, 2017), RabbitMQ (RabbitMQ, 2017) (RabbitMQ source, 2017), ActiveMQ (ActiveMQ, 2017) (ActiveMQ source, 2017), JoramMQ (JoramMQ, 2017) (JoramMQ source, 2017), Mosquitto (Mosquitto, 2017) (Mosquitto source, 2017). Our future work will continue this evaluations, and select the most appropriate one to be incorporated to our gateway solution.

5 CONCLUSION

The IoT paradigm has appeared as the latest Internet revolution generating a huge amount of powerful devices for the online world. By responding to this trend, several cloud providers have started to offer specific management services for this new world. In this currently forming IoT ecosystem, users and system developers do not only have to buy and configure devices, but they also have to choose the right IoT cloud provider offering the combination of protocols and data structures fitting their applications.

To support users and researchers in this field, we have shown how to develop a specific gateway service to manage simulated devices composing a complex semi-simulated environment. We have also evaluated the performance and scalability of this IoT gateway service. Using these solutions, researchers and developers can investigate the behavior of IoT systems, and develop and evaluate IoT cloud applications more efficiently in a convenient, cost effective way.

Our future work will address the development of fully portable gateways (deployable at different cloud providers), and the creation of various device templates for other IoT application areas, such as smart cities.

ACKNOWLEDGEMENTS

The research leading to these results was supported by the UNKP-17-4 New National Excellence Program of the Ministry of Human Capacities of Hungary,

and by the Hungarian Government and the European Regional Development Fund under the grant number GINOP-2.3.2-15-2016-00037 (”Internet of Living Things”).

REFERENCES

H. Sundmaeker, P. Guillemin, P. Friess, S. Woelffle. 2010. Vision and challenges for realising the Internet of Things. CERP IoT, Cluster of European Research Projects on the Internet of Things, CN: KK-31-10-323-EN-C.

J. Mahoney and H. LeHong. 2011. The Internet of Things is Coming. Gartner report, https://www.gartner.com/doc/1799626/internet-things-coming.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Comp. Syst*, 25(6):599–616.

A. Kertesz. 2014. Characterizing cloud federation approaches. In: Cloud computing: challenges, limitations and R&D solutions. *Computer communications and networks*. Springer, Cham, pp. 277-296.

A. V. Dastjerdi, R. Buyya. 2016. Fog Computing: Helping the Internet of Things Realize Its Potential. *Computer*, vol. 49, pp. 112-.116, Aug. 2016. doi:10.1109/MC.2016.245

Zeng, X.; Garg, S.K.; Strazdins, P.; Jayaraman, P.P.; Georgakopoulos, D.; Ranjan, R. 2016. IOTSim: A simulator for analysing IoT applications. *Journal of Systems Arch.*.

S. N. Han, G. M. Lee, N. Crespi, N. V. Luong, K. Heo, M. Brut, P. Gatellier. 2014. DPWSim: A simulation toolkit for IoT applications using devices profile for web services. In proc. of IEEE World Forum on Internet of Things (WF-IoT), pp.544–547.

S. Sotiriadis, N. Bessis, E. Asimakopoulou, N. Mustafee. 2014. Towards simulating the Internet of Things. In proc. of IEEE Advanced Information Networking and Applications Workshops (WAINA), pp. 444–448.

SimpleSoft SimpleIoTSimulator, http://www.smplsft.com/SimpleIoTSimulator.html, June, 2017.

Atomiton IoT Simulator, http://atomiton.com/simulator.html, April, 2017.

H. Gupta, A. V. Dastjerdi, S. K. Ghosh, R. Buyya. 2016. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. arXiv:1606.02007, CLOUDS-TR-2016-2.

IBM Watson IoT Platform visualization application, https://github.com/ ibm-watson-iot/ rickshaw4iot, September, 2017.

B. Kang, D. Kim, H. Choo. 2017. Internet of Everything: A Large-Scale Autonomic IoT Gateway. *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 3, pp. 206–214.

Private Gateway Service for MobIoTSim, https://github.com/sed-szeged/MobIoTSimBluemixGateway, September, 2017.

OpenWeatherMap, http://www.openweathermap.org, June, 2017.

T. Pflanzner, A. Kertesz, B. Spinnewyn, S. Latre. 2016. MobIoTSim: Towards a Mobile IoT Device Simulator. In IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp. 21–27.

VerneMQ website, http://vernemq.com, November, 2017.

VerneMQ source code, https://github.com/erlio/vernemq, April, 2017.

Moquette website, http://andsel.github.io/moquette/. November in April, 2017.

Moquette source code, https://github.com/andsel/moquette, November, 2017.

RabbitMQ website, https://www.rabbitmq.com/, November, 2017.

RabbitMQ source code, https://network.pivotal.io/products/pivotal-rabbitmq, November, 2017.

ActiveMQ website, http://activemq.apache.org, November, 2017.

ActiveMQ source code, http://activemq.apache.org/source.html, November, 2017.

JoramMQ website, http://www.scalagent.com/en/jorammq-33/products/overview, November, 2017.

JoramMQ source code, http://joram.ow2.org, November, 2017.

Mosquitto website, https://mosquitto.org/, November, 2017.

Mosquitto source code, https://github.com/eclipse/mosquitto, November, 2017.