2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics

Load Balancing in the Fog of Things Platforms through Software-Defined Networking

Ernando Batista*†, Gustavo Figueiredo†, Maycon Peixoto*, Martin Serrano‡, and Cássio Prazeres*
* WISER @ DCC, Federal University of Bahia, Salvador, Bahia, Brazil
† GRADE @ DCC, Federal University of Bahia, Salvador, Bahia, Brazil
‡ UIoT @ Insight, National University of Ireland, Galway, Ireland
Email: {ernando.passos,maycon.leone,gustavobf,prazeres}@ufba.br, martin.serrano@insight-centre.org

***Abstract*—The centralization of Internet of Things (IoT) services (data and applications) running in the cloud has been a tendency since the introduction of the concept-design that the IoT enables the orchestration/management of a huge amount of Internet connected objects. However, the adoption of this conceptual design imposes deployment limitations as high latency and high bandwidth demands. This paper presents an approach to the need of executing load balancing for the Fog of Things Platforms via programmability for distributed Internet of Things environments through the use of Software-Defined Networks (SDN). In this approach each IoT Gateway is responsible for optimally managing a set of sensors through IoT managment Services by reducing the number of failures or overload network link(s) associated with an IoT Gateway. Failure on the IoT Gateway itself due to processing overhead may result in temporary or even permanent unavailability in IoT applications (Web, Mobile, and Desktop) that demand information provided by those devices. We address these problems using Fog of Things load balancing approach and here we present our findings, deployed and implemented progress and experiments results.**

***Index Terms*—Internet of Things, SDN, Fog Computing, Load Balancing.**

I. Introduction

The Internet of Things (IoT) allows the coordination and orchestration of a huge amount of Internet connected objects [1][2]. Since these objects should be largely distributed and location aware, IoT applications have to deal with high degree of uncertainty, due to, for instance, frequent infrastructure changes or interference by adverse weather conditions [3]. Therefore, the network infrastructure to support the IoT have to be dynamic, scalable, transparent in terms of hardware, inter-operable, and with low latency [4].

In order to fulfill part of such requirements, Cloud Computing plays an important role for IoT [5]. However, this centralization may impose limitations as high latency and high bandwidth demand. In turn, such limitations can be overcome by combination of Cloud Computing with another paradigm: Fog Computing. According Bonomi et al. [4][6], Fog Computing is a paradigm that provides compute, storage, and networking services, between devices and traditional clouds, at the edge of network. In this way, Fog Computing try to reduce, or even to avoid, the communication and processing overload between edge devices and cloud data centers.

As consequence, Prazeres and Serrano [7] proposed Fog of Things (FoT), which is a paradigm for design and implementation [8] of Fog Computing platforms for IoT. In FoT, as in Fog Computing, part of data processing capacity and service delivery operations are performed locally in "small servers" (IoT gateways) and data processing that demands for heavy computational resources can occur in the cloud.

A FoT based platform can present critical scenarios, such as situation in which: several sensors are managed by the same gateway – causing processing overhead; or several sensors are managed by neighborhood gateways – causing communication overhead (network congestion). In these scenarios, some problems may occur: increase in the time required to exchange information among devices; loss of samples (temperature, luminosity, etc.) due to communication and/or processing overload; and, consequently, inconsistencies of information provided to IoT applications (Web, Mobile, and Desktop).

In order to fulfill part of such implication related to: a) location awareness; b) the centralization of services in the cloud; c) limitations as high latency and high bandwidth demand; and, d) processing and communication overhead; in this paper, we explore the possibility for using the combination of Fog of Things with a solution based on network programmability for overcoming such limitations. This solution aims in realize load balancing among gateways in a FoT platform via SDN (Software-Defined Networking) paradigm. As a result, we performed tests with and without our load balancing proposal and the tests indicated improvement in the performance of the gateways (reduction in the response time) and, consequently, reduction of lost samples is reported.

The remainder of this paper is organized as follows. Section II describes some state-of-the-art works related with our proposal. Section III presents the Fog of Things paradigm. Then, in Section IV, we present our proposal for load balancing among gateways in a FoT platform. Design of experiments and result analysis are presented in Sections V and VI. Finally, we present final remarks and future works in Section VII.

II. Related Works

The related work discussed in this section present solutions specifically to load balancing strategy in IoT/Fog Gateways. So far, we have studied some approaches [9][10][11] presenting

978-1-5386-7975-3/18/$31.00 ©2018 IEEE
DOI 10.1109/Cybermatics_2018.2018.00297

solutions to improve the performance in IoT/Fog scenarios using the SDN paradigm, however the cited works do not use the SDN paradigm aiming to improve the performance of an IoT application but to optimise the data collection/distribution.

Beraldi, Mtibaa and Alnuweiri [12] in the CooLoad (Cooperative Load Balancing), performs load balancing based on buffers available on Fog servers. In this approach, each Fog server (as a Gateway) has one buffer, so according to the overhead on a given buffer, the tasks are re-allocated to other servers with available buffer. CooLoad was modeled as a "Quasi Birth and Death processes" (QBD-processes) in continuous time Markov chains. The architecture for Fog Computing proposed by Kapsalis et al. [13] was implemented in the CloudSim environment. The main objective is to perform management and execution of IoT tasks through some layers (Hub Layer, Device Layer and Cloud Layer). For each new task, information about the IoT task to be performed are required, as a requirement of time, complexity and memory. To perform the resource allocation, the approach uses a scoring based function according to Fog Broker computational utilization criteria. In the experiments, the authors described some performance improvement in comparison to the traditional scenario (Cloud Computing). However, some characteristics of the network infrastructure are not taken into account as end-to-end path states, link information, and number of hops among hosts.

Ningning et al. [14] proposes a task-oriented load balancing mechanism using the graph partitioning strategy. Fog nodes are made up of virtual machines or virtual machine clusters. Fog nodes with high computational power can execute multiple virtual machines and physical nodes with low computational power are associated with other nodes in order to form a virtual machine. To perform the load balancing, information about the resources of the virtual machines (CPU, memory and others) and network resources (delay) is used as parameters. In the performed experiments, they used real devices with different computational power. The metric chosen to evaluate the load balancing was the average of node migrations that need to be made to balance the scenario. This approach is focused on migration, association and partitioning of nodes (virtual machines), differently from most IoT environments and the FoT paradigm used in this paper.

III. Fog of Things (FoT)

According to Prazeres and Serrano [7], Fog of Things paradigm goes further than the Fog Computing in some directions, such as: i) by using all the processing capacity of the network edge through performing data processing and service delivery on devices, gateways (very small servers) and small local servers; ii) by defining profiles for gateways and servers at the edge of the network, in order to define and implement IoT Services to be delivered in a distributed way; iii) by distributing these IoT Services also at the network edge through a message-service oriented middleware [8]. Therefore, as shown in Figure 1 the following components compose a FoT based platform [7]: FoT-Devices (sensors and actuators, for instance), FoT-Gateways, FoT-Servers, applications and others.

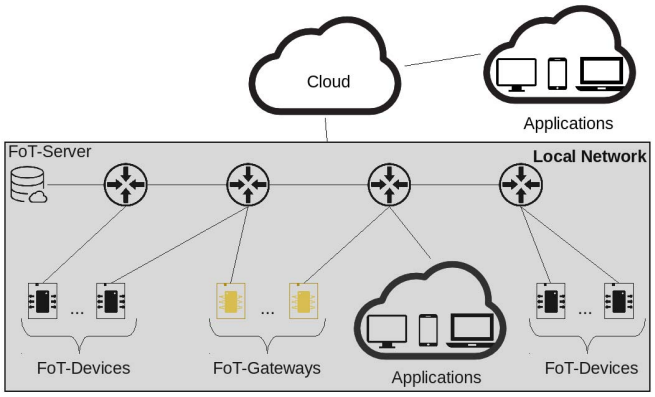


Fig. 1. Overview of a FoT based platform.

FoT-Device (see Figure 1) is the basic component in the FoT paradigm. The concept of FoT-Device goes further than the general concept of Device proposed by Bassi et al. [15], because FoT-Devices must perform the most basic functionality of a FoT-based platform, in terms of local and parallel data processing: transformation of raw data into structured content.

FoT-Gateway (see Figure 1) is the basic node managed by a FoT-based platform and can be called simply as a gateway, since its most basic and main objective is to map from communication layer protocols (Ethernet, WiFi, ZigBee, Bluetooth, etc.) to HTTP (RESTful Web Services). Therefore, FoT-Gateways must offer services to access/manage FoT-Devices (Thing as a Service paradigm - TaaS) and other IoT services.

FoT-Server (see Figure 1) is a basic and optional node manager for a FoT-based platform. FoT-Servers should provide specifics capabilities or information that are not supported by FoT-Devices and FoT-Gateways, for example, long-term historical data from FoT-Devices.

Application (see Figure 1) is any kind of application that uses the HTTP-based REST API (i.e. RESTful Web Services) provided by FoT-Gateways. Thus, applications can be Web, mobile (Android, iPhone or Windows Phone) or even traditional desktop applications.

IV. FoT-Gateways Load Balancing

In this section, we present an IoT Scenario (Section IV-A) where loss of samples (temperature, for instance), caused by a processing or a communication overload, can occur; and we also present our load balancing approach in order to deal with the lost samples.

A. Problem Statement: an IoT Scenario

As described in Section III, FoT-Gateways act as managers for the FoT-Devices and the basic services for a FoT based platform are those related with the management performed by the FoT-Gateways. Thus, several FoT-Devices can be managed by the same FoT-Gateway, which may cause overload in two ways: the amount of FoT-Devices is higher than the processing

capacity of a FoT-Gateway; the amount of exchanged messages between FoT-Devices and FoT-Gateways is higher than the communication capacity of the network links.

As an example in order to verify the impact of such two overload types, we create a virtual aggregation of FoT-Devices (in this case a sensor), which we nominate "virtual sensor". The main objective of a virtual sensor is to provide a means for abstraction in data access for several sensors. Figure 2 shows an example of utilization for virtual sensor, in which an application (Web, Mobile, or Desktop) requires information (samples) about the temperature in an environment (a room, for instance) with four FoT-Devices (temperature sensors). The samples provided by the FoT-Devices are processed and stored in FoT-Gateways and only the samples average value generated by the virtual sensor are sent to the application.

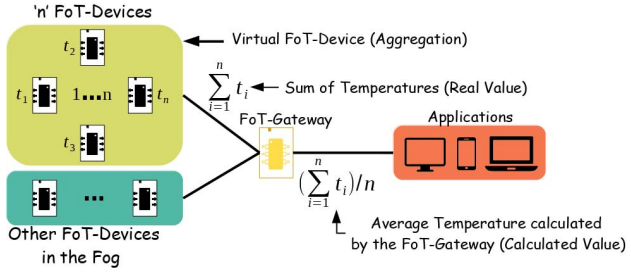
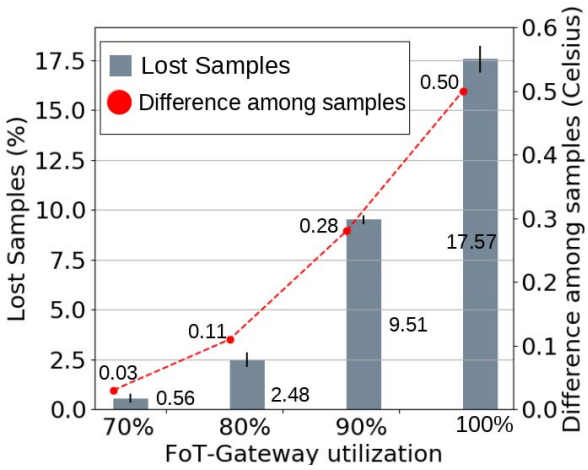


Fig. 2. IoT scenario: Virtual aggregation of FoT-Devices.

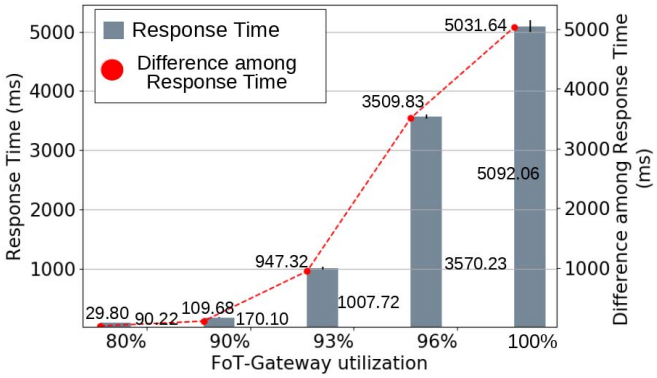
Using the abstraction provided by the virtual sensor, the application considers all FoT-Devices of a group as a single device. In critical scenarios, as intermittent network instability or gateway overload, if one or more samples are not processed/stored by the gateway an inconsistency can occur. As can be viewed in Figure 2, this inconsistency can be calculated by the difference between the average of: the real value of the samples generated by the FoT-Devices; and the value calculated by the FoT-Gateway for an application.

Figure 3(a) presents the results of a simulation with a FoT-Gateway working from 70% to 100% of its processing capacity, because, in our experiments, there was no lost samples below 70%. In this simulation, the FoT-Gateway performance is affected as processing overload occurs, which directly affects the amount of lost samples (percentage) and the increasing in the average sample inconsistency. In this case, a lost sample is the temperature value collected by a FoT-Device and not processed by the FoT-Gateway due to a processing overload. In addition, the red line in Figure 3(a) shows the average inconsistencies behavior (real temperature x application temperature) according to the FoT-Gateways processing.

Figure 3(b) presents the results of a simulation about network links utilization between FoT-Gateways and FoT-Devices. The link network utilization was configured to vary from 80% to 100%, because, in our experiments, response time variation was irrelevant below 80%. In this simulation, the response time affects the order in which samples reach



(a) Lost samples caused by processing overload in FoT-Gateways.



(b) Increase of response time caused by communication overload.

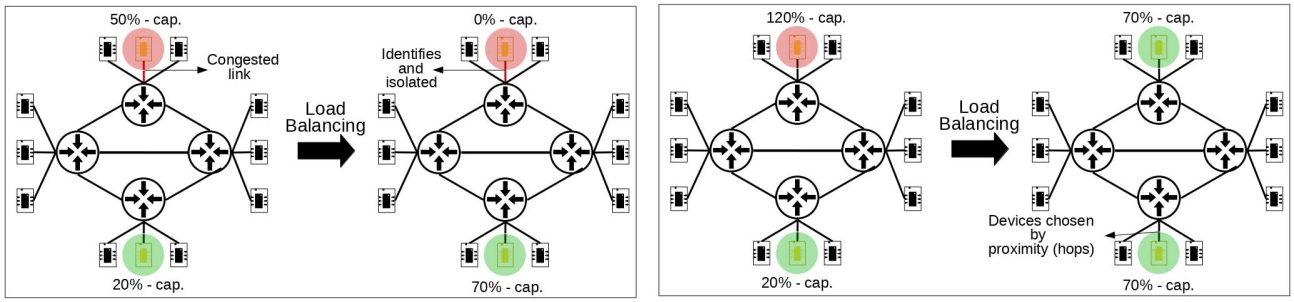
Fig. 3. Analysis for Lost Samples and Response Time.

the FoT-Gateway, which directly affects the average sample inconsistency. In this case, samples will not be lost, but it may reach the FoT-Gateway after the average to be calculated due a communication overload. In addition, the red line in Figure 3(b) shows the difference between the means in ideal scenarios (below 50% link utilization) and the scenarios of (80%, 90%, 93%, 96%, and 100%).

Therefore, to avoid such scenarios shown in Figure 3(a) and Figure 3(b), an IoT environment should provide aspects of dynamism and self-management that identify occurrences of processing and/or communication overload. Thus, in Section IV-B, we present our approach for load balancing among FoT-Gateways in an IoT platform based on the FoT paradigm.

B. Load Balancing

The problem described in Section IV-A comes up due to two main reasons: the FoT-Gateways loose performance as the number of FoT-Devices that it manages increases and the samples that are over delayed as communications links connecting the FoT-Devices to the FoT-Gateway become congested. Both problems are a consequence of the dynamic nature of the IoT environment where demands may vary over time. Therefore,



(a) Identification and isolation of FoT-Gateway with congested link.

(b) Change from unbalanced scenario to balanced scenario.

Fig. 4. Load Balancing Examples.

to avoid this problem, it is imperative to consider specificities of the IoT environment while promoting an adequate usage of both the FoT-Gateway and the communication links.

The Load Balancing Algorithm (Algorithm 1 and Algorithm 2), proposed in this section takes into consideration specific characteristics of the FoT scenario like the load on FoT-Gateways as well as the network usage. It relies on the programmability and flow deviation capability enabled by the SDN paradigm to balance resources demanded from the FoT-Gateways while keeps communication links operating under acceptable levels of load. Traditional load balancing algorithms do not consider specific characteristics found in the scenario used with FoT-Gateways and FoT-Devices (FoT paradigm). Relevant information in the FoT context would not be considered, in addition the implementation of own algorithm allows the insertion of new parameters in the balancing.

The materialization of this fact can be observed in experiments varying the number FoT-Devices managed by each FoT-Gateway to represent the FoT-Usage (**Change to Utilization**) (Figure 3(a)). The figure clearly shows that high utilization of FoT-Gateways rises the distributing the load among several FoT-Gateways of lost samples, which in turn increases the difference between the computed temperature and the correct temperature measurement. The objective of the algorithm is to promote the best performance of FoT-Gateways in the management of FoT-Devices. For that, the algorithm considers two complimentary information: (i) the state of the network regarding the utilization and delay of the links, and the number of hops between FoT-Gateways and FoT-Devices (see Figure 4(a)); (ii) the processing capacity of FoT-Gateways and the amount of managed devices (see Figure 4(b)). The Algorithm 1 shows how this information is taken into consideration during the load balancing.

The Algorithm 1 runs periodically in the controller after the creation of the network topology (hybrid topology - mesh and linear), by default the topologies are started with unbalanced scenarios - but maintaining proportionality according to the number of devices. The Algorithm 1 verifies the information of the FoT-Gateways and network (through OpenFlow/SDN), taking into consideration the load of both, network and FoT-Gateways (see Figure 5). Thus, it is possible to handle with

```
1  Function FoT Load Balancing ()
       Data: g_over: List of overloaded FoT-Gateways in
             descending order of processing;
             g_new: List of FoT-Gateways that accept new
             FoT-Devices in ascending order of processing;
             net_inf: List with FoT-Gateway link information;
2      while True do
3          g_over,g_new=controller.update_processing_list();
4          net_inf= controller.get_network_events();
5          foreach g_new as g do
6              if net_inf.get(g).use<=0.9*net_inf.get(g).cap then
7                  g_new.first=g[index];
8                  break;
9              end
10         end
11         if g_over!=null && g_new!=null then
12             change flow (g_over.first, g_new.first);
13         end
14         else if g_new!=null then
15             foreach net_inf as net do
16                 if net[index].use>0.9*net[index].cap then
17                     change flow (net[index], g_new.first);
18                 end
19             end
20         end
21     end
```

Algorithm 1: Load Balancing Among FoT-Gateways.

extreme situations in which the network is degraded (high utilization and high delay) and the FoT-Gateway is under low utilization. Based on the IoT scenario (Figure 2) and the results (Figure 3(b)), it is considered as inadequate in the Algorithm 1 links with utilization above 90% of its capacity (line 6/16).

In line 3, the Algorithm 1 receives two lists: the first one has the FoT-Gateways that are overloaded and the second list contains the FoT-Gateways with available capacity to manage additional FoT-Devices. This information is collected in every FoT-Gateway and stored in the controller. If both lists are non empty, the change of managers is performed (line 11). If, on the other hand, there are no overloaded FoT-Gateways the algorithm checks the information about the network to balance the load among FoT-Gateways (lines 14 to 21).

The function **Change Flow**, described in Algorithm 2, is called by the Algorithm 1 to remove the flow generated by a FoT-Device that was managed by an overloaded FoT-Gateway

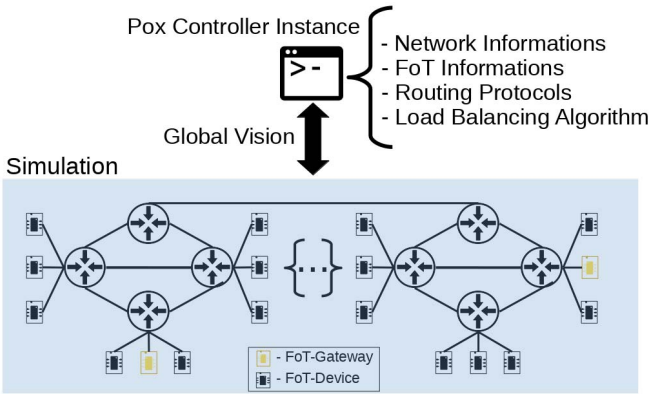


Fig. 5. Experiment Overview.

```
1  Function Change Flow (g_over, g_new)
       Data: floyd_warshall: Run Floyd-Warshall Algorithm in
             controller;
             l_hops: List of FoT-Devices in ascending order
             (hops);
             l_sensors: List of FoT-Devices managed by g_over;
2      l_sensors=get_data_sensors(g_over);
3      foreach l_sensor do
4          hops=floyd_warshall.get_hops(g_over,l_sensors[index]);
           if hops>0 then
5              l_hops.add(l_sensors[index],hops);
6          end
7      end
8      l_hops.sorted('ascending order');
9      controller.install(g_new,l_hops.first);
10     update_services_and_json_files(g_new,l_hops.first);
```

Algorithm 2: Change Flow Function.

and to pass it on to a FoT-Gateway with lower utilization. In addition, Algorithm 2 uses the hop count as the criterion to identify the FoT-Devices that will be reallocated, which means that FoT-Devices are reallocated to the closest FoT-Gateway (lines 3 and 8). Both algorithms use the POX controller and the communication between switches and controller is performed using the OpenFlow protocol version 1.0. To get and verify the network state information, we resort to some events specific of the POX controller like 'PortStatsReceived" and "FlowStatsReceived". Thus, it is possible to get information about switches (ports/utilization), links (capacity/delay/utilization) and FoT-Gateways/FoT-Devices attached to the switches.

V. Design of Experiments

The environment was implemented in the Mininet[1] emulation environment with some modifications to represent the FoT-Gateways and FoT-Devices components. The FoT-Devices and FoT-Gateways carries out data representation and information exchange through Web standards and protocols. Both elements were represented by Mininet's native component (hosts) using Web Services and information exchange in JavaScript Object Notation (JSON). These services enable hosts behavior as predicted in the FoT paradigm (see Figure 6). Thus, in the experimentation part in this article, hosts can behave as FoT-Devices, providing information about environment, for example, temperature and luminosity. In addition, hosts can behave as FoT-Gateways by performing management of FoT-Devices, i.e. by requesting new FoT-Devices information.

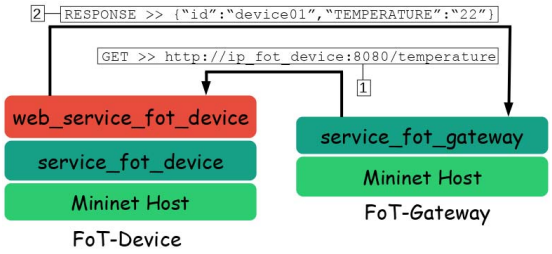


Fig. 6. Communication between FoT-Device and FoT-Gateway.

There are two different types of workload in the environment. The first one is the traffic due to information exchange between FoT-Devices and FoT-Gateways, in which case the information occurs in random time ranging from 6 to 12 requests per minute. The second one represents the congestion in the links, realized by iPerf [2]. An information exchange between FoT-Gateways and FoT-Devices are as follows: (i) the FoT-Gateway asks for a sample (i.e., temperature, luminosity) to FoT-Device; (ii) the FoT-Device replies with the sample value; (iii) if the FoT-Gateway is not overloaded, it shall processes and stores the received sample.

TABLE I
Factors and Levels Used in the Experimental Design.

Factors	Levels	Response variables	
Number of FoT-Gateways	2	Lost samples	Response time
	4		
Number of FoT-Devices	31		
	66		
Load Balancing Algorithm	Yes		
	No		

In order to conduct the full factorial design plan [16], the main parameters from our experimental work are shown in the Table I. **(I - Factors)**: Number of FoT-Gateways, Number of FoT-Devices, and Load Balancing Algorithm. These factors are chosen because several previous experiments show the meaningful influences performed by them in the environment. **(II - Levels)**: 2 or 4 FoT-Gateways; 31 or 66 FoT-Devices. These levels are chosen to show a light scenario and other overloaded scenario. In order to show the differences regarding the algorithms, it was used an approach with load balancing and the other without load balancing. In turn, **(III - Response Variables)** are defined by the **Response Time** between FoT-Gateway and FoT-Device, and **Lost Samples** as the percentage of samples discarded by FoT-Gateways, allowing to visualize the commitment of the system with the time and the accuracy.

[1]http://mininet.org/

[2]https://iperf.fr/

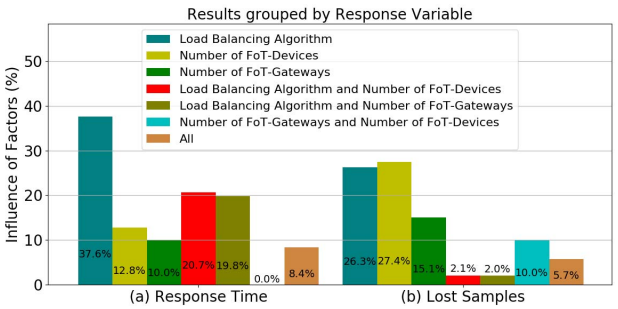


Fig. 7. Influence of factors on the response time and lost samples.

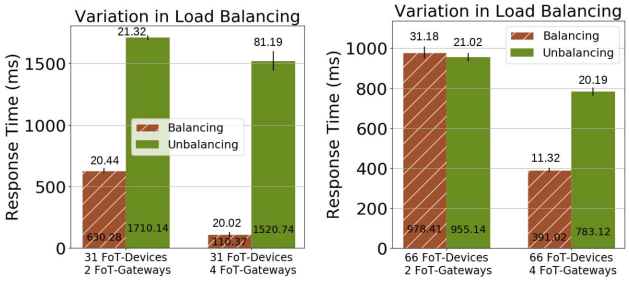
In the experiments model, the average and standard deviation are calculated for each combination (factors and levels), which will serve as the basis for calculating the sum of squares, resulting in the influence of each factor on the response variables. Figure 7(a) shows the influence of each factor and its combinations on the overall response time of the experiment. It is considered as significant influence the ability of a given factor to modify the values of the response.

The following setup was used to conduct the experiments: Operating System Ubuntu 12.04 LTS, Intel Core i7-4770 - 3.40GHZ processor/ 8GB of RAM. To generate the results from planning model used in this work, 8 simulations -each with 10 replications and each replication lasting 720 seconds- were performed for each scenario, that is, for each combination of factors and levels. All values in the replication stage were used, since the results obtained in each scenario did not present significant changes. According to Figure 7(a) the most significant factor is the Load Balancing Algorithm with 37.6% of influence on response time. The algorithm is the factor that most influences the response time because load balancing is performed, thus reducing the overall waiting time. Figure 7(b) shows the influence of each factor and its combinations on the overall percentage of samples discarded from the experiment (Lost Samples). In this scenario, while the load balancing Algorithm has 26.3% of influence, the most influence on this response variable are the Number of FoT-Devices with 27.4%. This behavior occurs because there is the increase of workload, turning the system overloaded.

The results shown the factors used in the design of experiments had different influences on the response variables. Furthermore, the proposed algorithms had a considerable influence on both scenarios (response time and lost samples). In order to increase analytical information of our work, Section VI presents a detailed analysis of how the load balancing algorithm impacts the response variables of the system.

VI. Analysis of Results

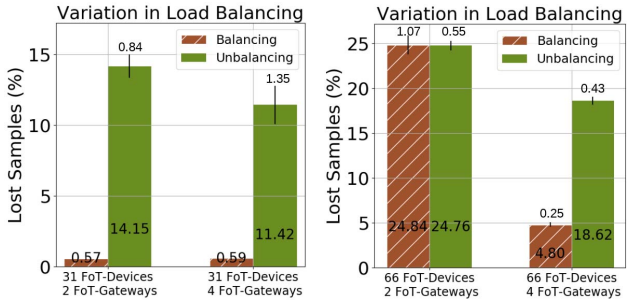
This section presents an analysis about how our load balancing approach influenced the response variables defined in Section V. In order to do that, we present the data generated from our experiments for each combination (a scenario) of factors and levels defined in Table I. Furthermore, we compare the results with and without our load balancing approach in the same scenario (factors with the same levels).



(a) Scenario with 31 FoT-Devices. (b) Scenario with 66 FoT-Devices.

Fig. 8. Analysis for Response Time.

The execution of the Algorithms 1 and 2 promoted reduction in the global response time in three of four scenarios shown in Figure 8(a) and Figure 8(b). Our load balancing approach had no impact in the last scenario in Figure 8(b) (66 FoT-Devices with 2 FoT-Gateways) and, consequently, there was no reduction in global response time, because it is a saturated scenario (too many devices and too few gateways).

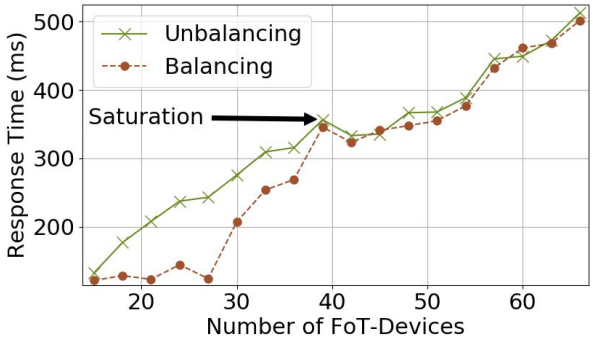


(a) Scenario with 31 FoT-Devices. (b) Scenario with 66 FoT-Devices.

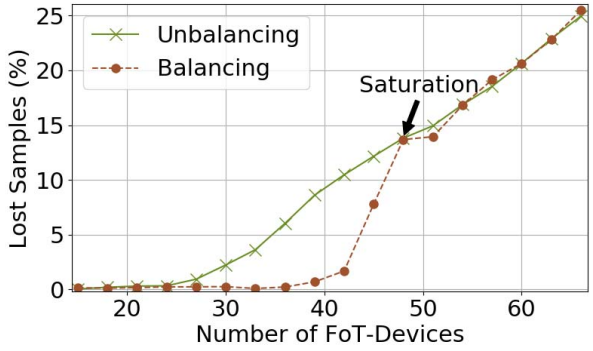
Fig. 9. Analysis for Lost Samples.

The execution of the Algorithms 1 and 2 also promoted reduction in the amount of lost samples in three of four scenarios shown in Figure 9(a) and Figure 9(b). Our load balancing approach had no impact in the last scenario in Figure 9(b) (66 FoT-Devices with 2 FoT-Gateways) and, consequently, there was no reduction in the amount of lost samples, because it is the same scenario aforementioned and shown in Figure 8(b).

In addition, Figure 10(a) and Figure 10(b) show a scenario with three FoT-Gateways with information about the behavior of response time and lost samples according to the variation in the amount of FoT-Devices. These results differs from those presented in Figure 5 and Figure 6, because Figure 10(a) and Figure 10(b) show results with less granularity. As a result, it is possible to view detailed information about the response variables (lost samples and response time) according to the variation in the amount of FoT-Devices. Moreover, it is possible to identify the moment at which the saturation occurs. After saturation, the values with and without our



(a) Impact of the Algorithms 1 and 2 on the response time.



(b) Impact of the Algorithms 1 and 2 on the lost samples.

Fig. 10. Analysis for Response Time and Lost Samples.

load balancing approach show similar results, indicating the inefficiency of the algorithm for saturated scenarios.

VII. Final Remarks

We presented a load balancing approach for FoT-Gateways (FoT paradigm) and network links. The main objective in this approach is to improve the performance in IoT scenarios based on Fog Computing. In this way, we evaluated two metrics: lost samples and response time. The results demonstrated that for the most of the scenarios evaluated, the utilization of our approach resulted in better performance when compared the same scenarios without our load balancing approach.

As a future work, we are developing a solution for IoT scenarios (saturated scenarios) in which our load balancing approach is not efficient. The proposal for this future work is based on virtualization, i.e., if all FoT-Gateways are saturated, we plan to virtualize new FoT-Gateways at the FoT-Server level. Then, we can use the same load balancing approach proposed in this paper to be performed in real and virtual FoT-Gateways. As a result, we believe that this solution can be evaluated with the same metrics used in this paper: lost samples and response time. In addition, we plan to increase the number of devices in the experiments and compare our solution with traditional load-balancing algorithms.

Acknowledgments

Authors would like to thank: CAPES, CNPq, and FAPESB organizations for supporting the Graduate Program in Computer Science at the Federal University of Bahia; CNPq (205266/2014-2) for the support to Cassio Prazeres; partnership MCTIC-UFBA - (Ministry of Science, Technology, Innovation and Communication of Brazil) for the financial support through the PROPCI/PROPG/PROPESQ/UFBA-004/2016; and FAPESB (BOL3271/2015) for the support to Ernando Batista and likewise to Science Foundation Ireland (SFI) for the support to Martin Serrano through the grant (SFI/12/RC/228).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] H. B. Pandya and T. A. Champaneria, "Internet of things: Survey and case studies," in *2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO)*, Jan 2015, pp. 1–6.

[3] M. Abdelshkour, "IoT, from cloud to fog computing," http://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing, March 2015, [Online; accessed 30-October-2017].

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Workshop on Mobile Cloud Computing*. ACM, 2012, pp. 13–16.

[5] M. serrano, "A self-organising archotecture for cloud by means of infrastrucutre performance and event data," in *Proceedings of the IEEE cloudcom confernece 2013*. New York, NY, USA: IEEE, 2013, pp. 481–486. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6753835

[6] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, ser. Studies in Computational Intelligence, N. Bessis and C. Dobre, Eds. Springer International Publishing, 2014, vol. 546, pp. 169–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-05029-4_7

[7] C. Prazeres and M. Serrano, "SOFT-IoT: Self-organizing fog of things," in *2016 30th International Conference on Advanced Information Networking and Applications Workshops*, March 2016, pp. 803–808.

[8] C. Prazeres, J. Barbosa, L. Andrade, and M. Serrano, "Design and implementation of a message-service oriented middleware for fog of things platforms," in *Proceedings of the Symposium on Applied Computing*. ACM, 2017, pp. 1814–1819.

[9] P. Bull, R. Austin, and M. Sharma, "Pre-emptive flow installation for internet of things devices within software defined networks," in *3rd International Conference on Future Internet of Things and Cloud*, 2015, pp. 124–130.

[10] J. M. Llopis, J. Pieczerak, and T. Janaszka, "Minimizing latency of critical traffic through sdn," in *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, Aug 2016, pp. 1–6.

[11] P. Du, P. Putra, S. Yamamoto, and A. Nakao, "A context-aware iot architecture through software-defined data plane," in *2016 IEEE Region 10 Symposium (TENSYMP)*, May 2016, pp. 315–320.

[12] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *2017 Second International Conference on Fog and Mobile Edge Computing*, 2017, pp. 94–100.

[13] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis, "A cooperative fog appfog computing dynamic load balancing mechanism based on graph repartitioningroach for effective workload balancing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 36–45, March 2017.

[14] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, no. 3, pp. 156–164, March 2016.

[15] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. van Kranenburg, S. Lange, and S. Meissner, Eds., *Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*, 1st ed. Springer-Verlag Berlin Heidelberg, 2013.

[16] R. Jain, *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley, 1991.