2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

An Algorithm to Optimise the Load Distribution of Fog Environments

Euclides C. Pinto Neto, Gustavo Callou, Fernando Aires
Department of Statistics and Informatics
Federal Rural University of Pernambuco
Recife, Pernambuco
Email: {euclides.carlosn, gustavo.callou, fernandoaires}@ufrpe.br

***Abstract*—Internet of things, a trend of the following years, makes it possible to develop new applications and services as well as creates a huge amount of data to be processed. In order to support this new paradigm, an extension of cloud computing, named Fog Computing, has been developed. Fog computing improves the cloud security, availability and performance by providing a distributed and powerful communication environment with short delay. Therefore, this new paradigm complements the cloud computing. However, the fog faces several issues such as quality of service (QoS) and multi-tenancy optimisation and load balancing. This paper proposes the algorithm called Multi-tenant Load Distribution Algorithm for Fog Environments (MtLDF) to optimise the load balancing in Fogs environments considering specific multi-tenancy requirements (delay and priority). Finally, we present case studies to show the applicability of the proposed algorithm in comparison to a Delay-Driven Load Distribution (DDLD) strategy.**

***Keywords*—Fog Computing, Cloud Computing, Load Balacing, Multi-tenancy.**

I. Introduction

Nowadays, the Internet traffic is very large and is expected to grow 50 times by 2020 [1]. These data are created and used by many industries, such as Banking, Sales and Manufacturing. These industries demand an even smarter and faster way to process this data because it may lead to a competitive advantage for them. Therefore, many vendors offer solutions related to data management in order to increase productivity and profit, which may involve Cloud Computing and Internet of Things (IoT).

This data growth is a result of a process in which many devices get connected to Internet and start sharing information with each other, i.e., Internet of Things (IoT) [2]. IoT is creating a new global connected sensor world, in which massive number of devices are connected to the Internet generating a huge amount of data [3]. However, IoT has many challenges such as interoperability, security, data centers technology, standards, and server technologies [4].

In order to support the IoT, a powerful distributed system is required. An increasing number of system developers use cloud technologies to provide IoT services [5]. The advance of parallel computing, distributed computing and grid computing enabled the constructions of a new computing model called cloud computing, which aims to share data, processing, and services transparently among many users [6]. This computing model can support IoT in many ways, such as processing, storage, scalability and data analysis. However, the cloud may be a slow option for real-time applications because it may be too far from the users. A faster option is to bring the cloud to the user by creating simpler clouds near to the devices. This small cloud is called Fog, which takes the Cloud Computing to the edge of the network with low latency and location awareness [7].

The Fog Computing takes the Cloud Computing to the edge of the network. It has low latency and location awareness, wide-spread geographical distribution, mobility, very large number of nodes, predominant role of wireless access, strong presence of streaming and real time applications, and heterogeneity [7]. However, there are many issues related to Fog computing such as quality of service (QoS), security and networking [8], which involves load balancing.

Load balancing aims to share the coming load between the available nodes. Note that this concept is widely used. Examples of its applications can be seen in heterogeneous cellular networks [9], data centres [10] and smart grids [11]. In other words, the load balancing aims to minimise the resource consumption, which reduces energy consumption and carbon emission rate [12]. It has application on security as well, treating DDoS attacks [13], for example. There are not many works on load balancing applied to fog computing. Therefore, this paper proposes an algorithm for performing load balancing on fog computing that considers two specific metrics: delay and priority. Furthermore, load balancing can reduce energy consumption by evenly distributing the load, minimising the resource consumption [14].

This paper is organised as follow. Section II discusses related works and intersections among them. Section III presents our multi-tenant load distribution algorithm and discusses its details. Section IV shows the three case studies to present the algorithm applicability through experiments. Finally, Section V discusses the results obtained in our experiments and Section VI presents the conclusion of this paper as well as future works.

II. Related Works

As Cloud Computing is a popular and important research topic, many works are focused on the issues it faces. On the other hand, few of them focus on Fog Computing, which is a new concept and not as popular as Cloud Computing.

978-1-5386-1645-1/17/$31.00 ©2017 IEEE

Nidhi et al. [14] discuss many load balancing techniques in cloud computing and compare them considering several parameters such as performance, scalability and associated overhead. None of them uses Energy consumption (EC) and Carbon Emission (CE) as metrics or consider the multi-tenancy characteristics. Our strategy considers resource utilisation, response time and multi-tenancy.

Sidra et al. [15] present a survey on load balancing algorithms in cloud computing over the period of 2004-2015. Many strategies are presented, such as Static load balancing, which is made at compile time, and Honey Bee, which increases throughput and minimise response time. The advantages and limitations of each strategy is discussed as well. However, it does not show a study of these algorithms in a mult-tenant environment, which may present specific requirements.

John et al. [16] present an algorithm for predicting future workload for multiple services. The multi-tenant nature of cloud computing systems is considered as well as a simple load balancing algorithm, called Round Robin, which distribute the load to each node sequentially. Therefore, this work shows that the multi-tenant prediction model can predict future network loads, and it can be used to develop an energy-aware cloud architecture. However, it does not considers other parameters in the distribution or multi-tenants requirements as resource utilisation.

A Power Load Distribution Algorithm is proposed in [17] to optimise the energy flow in power systems. More specifically, the algorithm optimises the flow distribution of the energy flow model (EFM). EFM is responsible for estimating energy consumption and cost issues of power infrastructures without crossing the restrictions of the power capacity that each device can support. IoT was not the focus of this work and the case studies were conducted on data centre and cloud computing systems.

The authors in [18] propose an extension of IoT architecture focused on multi-tenancy. It is intend to be used in IoT decentralised applications based on objects and adds a multi-tenancy layer to set users priorities. Objects have credentials, which contains information related to priorities and permission. So that users can even share their objects with other users. However, this work focus on the architecture and issues as power saving, resources utilisation and load balancing are not treated.

Chien et al. [19] present a load balancing strategy for cloud environments based on estimating the end of service time, which reduce response and processing times. Thus, as a load balancing tool, it improves the system in some points such as optimises resource utilisation and minimises response time. However, this paper does not consider the multi-tenant applications or IoT architectures.

Wang et al. [20] present a dynamic load balancing method of cloud centre based on Software-defined Networking (SDN). OpenFlow protocol, used in SDNs, offers task scheduling flexibility, real-time monitoring of the service node flow and load condition. In this approach, four modules are considered and all of them are put in the SDN controller: traffic detection module, which deals with dynamic traffic monitoring and statistics; load calculation module, which estimates the load distribution; dynamic load scheduling module, which aims to achieve high performance load balance process; flow management module, which supports load balance strategy. However, this work aims in general purposes cloud environments. Fog computing and IoT are not taken into account, as well as multi-tenancy load balancing.

III. Multi-tenant Load Distribution Algorithm

This section presents the proposed multi-tenant load distribution algorithm, which was implemented in Java language, and is an strategy applied to the Fog Computing for sharing the load among nodes considering specific tenants requirements. This work adopts the tenant maximum acceptable delay (TMAD), which is an important metric in terms of QoS [21], and tenant priority (TP) requirements, which is used to sort the tenants according to their importance for the system and is based on plan each tenant pays for, i.e., special plans may lead to a higher priority operation. However, this can be extended to consider different metrics. Furthermore, the considered workload in our modelling comes from read-heavy and write heavy applications [22], which demands a heavy processing and low-latency.

Figure 1 shows our proposed architecture. In this figure, the *IoT end users* represent the tenant devices which are able to communicate with the Fogs to get the workload done. In this architecture, a Fog is able to communicate with other Fog and/or a higher layer named *Fog Management Layer* (FML). This management layer is responsible for sending the load to the proper nodes and, thus, managing the Fogs. In order to accomplish this, a table for managing the nodes utilisation and the communication delay from a Fog to another is adopted.

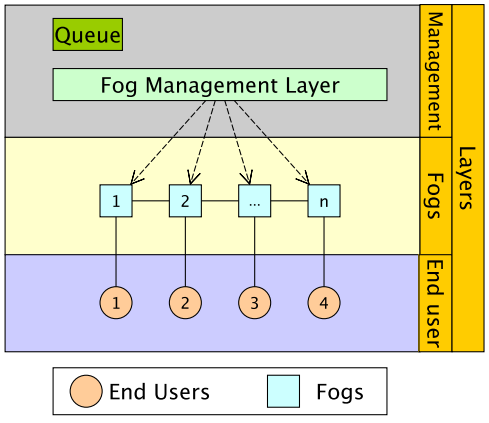

Fig. 1. Architecture of our strategy.

When end users send a load to any Fog, it consults the FML that defines which node will be the responsible for this operation. This decision is made based on the utilisation level. In addition, this node selection also takes into account the delay, which must be less than the delay requirement (TMAD) of the current tenant. Assuming a fact in which all

the only available nodes have a delay higher than the TMAD of the current tenant, a node is selected anyway. The best performance and availability are achieved when all nodes are available, in which more flexibility on the architecture can be obtained according to the defined requirement (e.g., delay).

Finally, considering the case when no one node is available, all new loads are sent to a queue located in the FML. Each element inside the queue is composed of three attributes: load, tenant and Fog origin. The tenant identification is the only way to get the specific tenant requirements. The Fog origin used for computing the delay related to the current end user device position. In this scenario, when any node becomes available, the FML takes the load with highest tenant priority (TP) and send it to any available node. The first criteria in this process is the TP, the second criteria is the arrival order. The workflow presented in Figure 2 illustrates how the algorithm works.

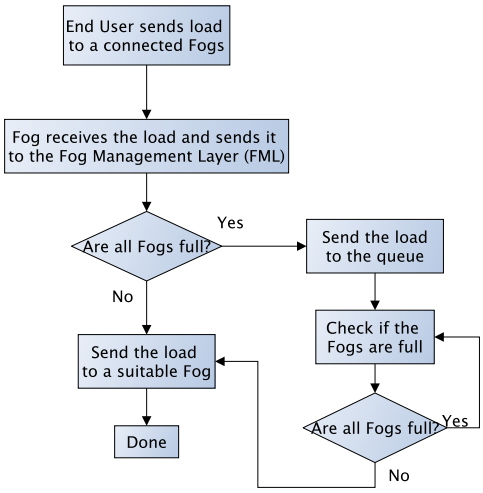

Fig. 2. Workflow that explains the operation of our algorithm.

Figure 2 shows the workflow model of our system. First, end users send the load to a reachable Fog. Then, the Fog receives the load and consults the FML in order to discover which fog will receive the load. After that, assuming there are Fogs available to receive that load, the load is sent to a suitable Fog, considering the load balancing. Otherwise, considering the Fogs are full, the load is sent to the queue. The next step checks if a Fog (or a set of Fogs) became available to receive the load. Finally, the load is processed by the Fogs. Additionally, we should stress that the proposed strategy focuses only on Fog environments, especially redundant Fogs. The requirements can be adapted and it can be applied to groups of Fogs, that share data, or a single Fog. However, there are cases in which this communication among Fogs may not be feasible, for instance, when the fogs are too far way.

IV. Case Studies

This section presents three case studies in order to show the applicability of the proposed strategy. The main goal of the first study is show the applicability of our strategy in a small scenario considering three different tenants. After that, we present a case study in the same scenario considering five different tenants. Finally, the last case study assumes a larger scenario with two different tenants.

In all case studies we consider the Fogs are equals in terms of topology, so that we consider the load to be represented in terms of the whole system capacity, i.e., how much of all the Fogs are needed to process the workload. In our experiments we use the concept of states. A state represents the current system condition after the occurrence of a set of events. The state representation considers events in a period of time, i.e., we assume the occurrence of a set of events which are represented through tables. Therefore, we group the load sent for each tenant in tables. The nodes considered in the case studies have the same processing power. Finally, in our experiments we compare our algorithm (MtLDF) to an approach based on delay value, that we call Delay Driven Load Distribution (DDLD). This approach does not considers the resource utilisation, its choices are made considering only the delay due to the fact of delay is a important metric in terms of QoS [21]. In order to quantify the load distribution efficiency we use the following equation:

$$f(n) = 1 - \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |load(Fog(i)) - load(Fog(j))|}{(n-1) * (\sum_{i=1}^{n} load(Fog(i)))} \quad (1)$$

This equation represents the load distribution among Fogs or even nodes. Its result varies from 0, which indicates the load is completely unbalanced, to 1, which indicates the load is completely balanced. The *n* represents the number of Fogs (or nodes) in the problem. The function *load(Fog(n))* represents the load the Fog *n* holds. The numerator of this equation is the sum of the differences of the load of all Fogs, comparing one by one. Note that it is a positive value. The denominator of this equation is the sum of the load of all Fogs multiplied by the number of Fogs minus 1, i.e, this division aims to normalise the value. Note that the numerator is the difference among all Fogs and the sum operator goes up to (*n*-1). Thus, in order to normalise this value we divide it by the total load times the number of iterations in the sum operator, i.e., (*n*-1). Finally, the value 1 is subtracted by the result of this division. In other words, the greater the difference of load among Fogs, the higher the result of the division and the lower the result of the load distribution function.

A. Case Study I

This case study aims to show the applicability of our proposed strategy considering the topology presented in Figure 3. In this example, we have four Fogs connected to each other. The connections and its respective delay, 25ms, are also illustrated. Each Fog has four available nodes that users can send load. All Fogs communicate with the Fog Management Layer (FML) and its respective queue.

TABLE I. STATE DESCRIPTION OF CASE STUDY I

Tenant	TMAD	TP	Fog	Load Added (%)
TenantA	10ms	3	1	25
			2	3
			3	2
			4	5
TenantB	150ms	5	1	10
			2	3
			3	2
			4	5
TenantC	150ms	6	1	10
			2	5
			3	3
			4	2

Furthermore, we consider three different tenants (Tenant A to C in Table I) among the large number of devices. Table I presents the state description adopted for the case study I. In this table, the column TMAD represents the values related to the tenant maximum acceptable delay, TP is the tenant priority, and load added corresponds to the capacity used for each fog.

Assuming the presented topology, the tenant A sends a load that demands 35% of the whole system to be processed whereas the tenants B and C send loads that demands 20% of the processing power of the whole system to be done. Note that, in this case study, the full capacity of the system is not used. In our equation the DDLD approach reached 0.833. Considering that the perfect scenario in terms of load balancing the result of this equation is 1, the DDLD performed well. However the MtLDF reached 0.978, which is much closer to 1, i.e., MtLDF presented a more efficient load balancing in comparison to DDLD. Figure 4 presents the load in each Fog using DDLD (green bars) and MtLDF (grey bars). Note that the results shown through the grey bars are more constant than the others depicted by the green ones.

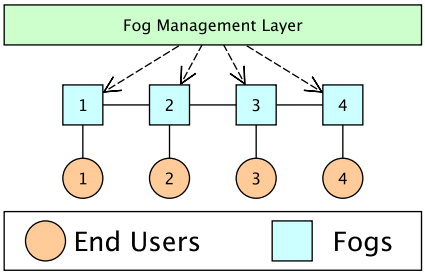

Fig. 3. Topology adopted in Case Study I

B. Case Study II

This case study aims to present the applicability of our strategy in a small topology considering five different tenants. Four Fogs are not connected to each other are adopted. Thus, the load balancing is applied only inside the Fogs and they cannot share load to each other. Each Fog has four available nodes that users can send load to. All the Fogs communicate with the Fog Management Layer and its respective queue. Furthermore, we consider five different tenants among the large number of devices (Tenant A to E as shown in Table II). In this case, the tenants A, B and C sends, respectively,

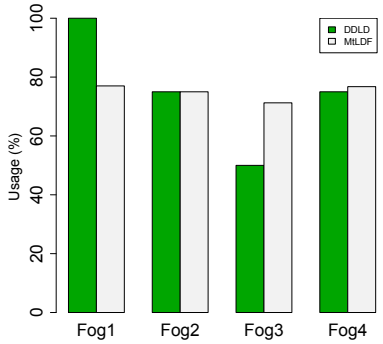

Fig. 4. Case Study I: Load sent to the system Fogs using DDLD and MtLDF

TABLE II. STATE DESCRIPTION OF CASE STUDY II

Tenant	TMAD	TP	Fog	Load Added (%)
TenantA	100ms	4	1	6
			2	2
			3	1
			4	1
TenantB	80ms	7	1	3
			2	0
			3	1
			4	1
TenantC	150ms	6	1	12
			2	2
			3	4
			4	2
TenantD	200ms	5	1	10
			2	5
			3	5
			4	5
TenantE	150ms	10	1	12
			2	3
			3	5
			4	5

a load that demands 10%, 5% and 20% of the whole system capacity. In our equation, DDLD and MtLDF reached the same value, 0.674, i.e., as the Fogs are not connected to each other, they are not able to share load and the load distribution is made internally. The proposed approach focuses on load distribution among Fogs. Note that the distribution efficiency for DDLD and MtLDF are identical.

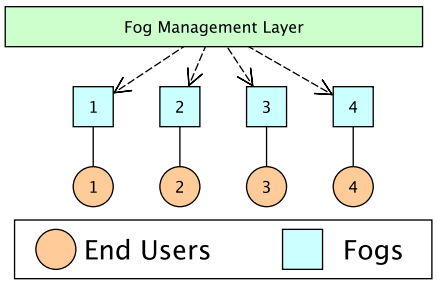

Fig. 5. Topology adopted in Case Study II

In our experiment, the assumed strategy considers the four Fogs as distinct group of Fogs. Due to the fact that they cannot communicate among them, four different algorithm

executions were needed. The load sent to the system nodes in our experiments are shown in Figure 6. The green bars, which sometime has value equals to 0, illustrate the load in each node using DDLD, whereas the grey bars illustrate the load in each node using our approach, MtLDF. Each couple of bars, grey and green, represents the load in one node. Note that the load sent to Fogs using DDLD and MtLDF are equal because Fogs are not connected to each other and cannot share loads. However, inside the Fogs, MtLDF balances the load and try to keep the nodes utilisation around the same. It can be considered an internal load balancer, i.e., the nodes of other Fogs are not taken into account.

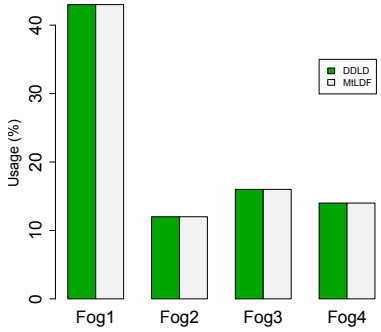

Fig. 6. Load sent to the system nodes using DDLD (green) and MtLDF (grey) in Case Study II

C. Case Study III

The main goal of this case study is to analyse a more complex topology. Figure 7 illustrates the under analysis scenario which is composed of ten Fogs connected to each other. In this topology, the delay of the connections of Fogs is 25ms. Each Fog has twenty available nodes that the users can send load to. All the Fogs communicate with the Fog Management Layer and its respective queue. Furthermore, we consider two different tenants among the large number of devices. Table III shows the adopted values for both tenants. In this case, the tenant A demands sends a load that demands 34% of the whole system and the tenant B, 32%. Assuming the equation 1, DDLD and MtLDF reached, respectively, 0.888 and 0.992 of distribution efficiency, which shows that MtLDF is more effective than DDLD in terms of load distribution in this scenario.

Figure 8 shows the load sent to the Fog system. The green bars illustrate the load in each Fog using DDLD, whereas the grey bars illustrate the load in each Fog using our approach, MtLDF. Note that the graph depicts that the MtLDF is able to share the load among all the fogs available in a much more effective way than DDLD does. Therefore, assuming our approach, all the fogs reached similar usage level.

V. Discussion

MtLDF can provide a better resource utilisation of a Fog system. Even in those cases in which the Fogs are not connected, the load is balanced among the internal

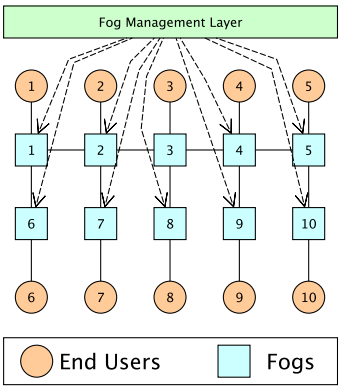

Fig. 7. Topology adopted in Case Study III

TABLE III. State description of Case Study III

Tenant	TMAD	TP	Fog	Load Added (%)
TenantA	20ms	10	1	5
			2	1
			3	1
			4	6
			5	3
			6	2
			7	3
			8	6
			9	5
			10	2
TenantB	120ms	5	1	4
			2	3
			3	3
			4	4
			5	1
			6	3
			7	5
			8	2
			9	2
			10	5

nodes. When DDLD is employed, this distribution is not so effective. The presented case studies are developed on different topologies. Each of those has its own characteristics such as connectivity and size. Note that the queue only receives loads when the Fogs are full. So that, when any node becomes available, the load with highest TP contained in the queue is sent. The TMAD must be carefully thought, because it can impact on the system performance. In some cases, it can concentrate the load in few parts of the system that respect its requirements.

In all case studies, the MtLDF performed well and reached high levels in our distribution equation. Table IV presents the distribution value of load among Fogs for both approaches considering the three case studies. Note that, in the case studies I and III the MtLDF presented a higher distribution value, with 0.978 and 0.992 respectively, in comparison with the DDLD, which reached 0.833 and 0.888. Considering the case study II the MtLDF reached the same distribution efficiency than the DDLD (0.674). A possible explanation for this results is due to the scenario under analysis, in which the Fogs are not connected to each other, and, thus, they cannot share load among them. It shows that MtLDF performed as good as or better than DDLD in all case studies.

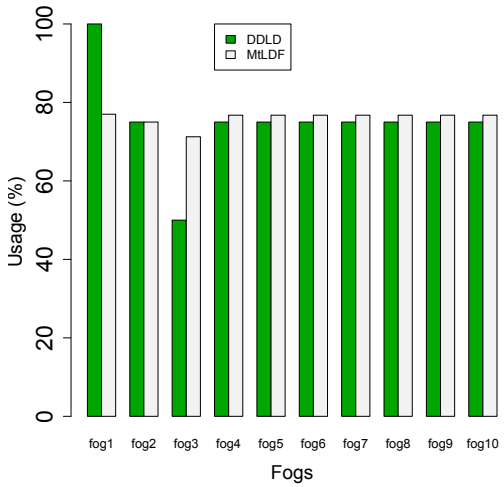

Fig. 8. Load sent to the system Fogs using DDLD (green) and MtLDF (grey) in Case Study III

TABLE IV. EQUATION RESULTS FOR ALL CASE STUDIES USING DDLD AND MTLDF

Case Study	DDLD	MtLDF
I	0.833	0.978
II	0.674	0.674
III	0.888	0.992

VI. CONCLUSION

This paper proposed a multi-tenant load distribution algorithm in Fog environments called MtLDF. The experiments showed that MtLDF can distribute the load in a more effective way than Delay-Driven Load Distribution (DDLD) does. The resource utilisation was also increased according to those values found in our distribution equation. In some cases, the distribution value was higher than 97%. As future work the authors intend to improve the algorithm and add new functions and metrics. In addition, we can try to improve the load balancing across the Fog-Cloud layers, and to consider some features such as disk I/O operations.

ACKNOWLEDGMENT

The authors would like to thank FACEPE and CNPq for the financial support of this project.

REFERENCES

[1] Yong-Hee Jeon, *"Impact of Big Data: Networking Considerations and Case Study"*, International Journal of Computer Science & Network Security; Dec2012, Vol. 12 Issue 12, p30, 2012.

[2] M. Dixit, J. Kumar and R. Kumar, *"Internet of things and its challenges"*, Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on, Noida, 2015, pp. 810-814, 2015.

[3] J. Gao, L. Lei and S. Yu, *"Big Data Sensing and Service: A Tutorial"*, Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on, Redwood City, CA, 2015, pp. 79-88, 2015.

[4] M. Dixit, J. Kumar and R. Kumar, *"Internet of things and its challenges"*, Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on, Noida, 2015, pp. 810-814, 2015.

[5] W. Zeng, M. Koutny and P. Watson, *"Opacity in Internet of Things with Cloud Computing (Short Paper)"*, 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA), Rome, 2015, pp. 201-207, 2015.

[6] Xu Wang, Beizhan Wang and Jing Huang, *"Cloud computing and its key techniques"*, Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on, Shanghai, 2011, pp. 404-410, 2011.

[7] Bonomi, F., Milito, R., Zhu, J., Addepalli, S., *"Fog computing and its role in the internet of things"*, Proceedings of 1st Workshop on Mobile Cloud Computing (MCC 2012), pp. 1316. ACM, 2012.

[8] S. Yi, C. Li, Q. Li, *"A Survey of Fog Computing: Concepts, Applications and Issues"*, in Proceedings of the 2015 Workshop on Mobile Big Data, pp. 3742, 2015.

[9] I. Sohn; S. Lee, *"Distributed Load Balancing via Message-Passing for Heterogeneous Cellular Networks"*, in IEEE Transactions on Vehicular Technology , vol.PP, no.99, pp.1-1, 2016.

[10] L. Yu; L. Chen; Z. Cai; H. Shen; Y. Liang; Y. Pan, *"Stochastic Load Balancing for Virtual Resource Management in Datacenters"*, in IEEE Transactions on Cloud Computing , vol.PP, no.99, pp.1-1, 2016.

[11] K. Kouzelis, I. Diaz De Cerio Mendaza, B. Bak-Jensen, J. R. Pillai and B. P. Bhattarai, *"Allocation of power meters for online load distribution estimation in smart grids"*, Smart Grid Technologies - Asia (ISGT ASIA), 2015 IEEE Innovative, Bangkok, 2015, pp. 1-6, 2015.

[12] Kansal, Nidhi Jain; Chana, Inderveer, *"Cloud Load Balancing Techniques : A Step Towards Green Computing"*, International Journal of Computer Science Issues (IJCSI);Jan2012, Vol. 9 Issue 1, p238, 2012.

[13] S. Zinno, G. Di Stasi, S. Avallone and G. Ventre, *"A Load Balancing Algorithm against DDoS attacks in beyond 3G wireless networks"*, Euro Med Telco Conference (EMTC), 2014, Naples, 2014, pp. 1-6.

[14] Nidhi Jain Kansal1, Inderveer Chana, *"Cloud Load Balancing Techniques : A Step Towards Green Computing"*, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[15] S. Aslam and M. A. Shah, *"Load balancing algorithms in cloud computing: A survey of modern techniques"*, 2015 National Software Engineering Conference (NSEC), Rawalpindi, 2015, pp. 30-35.

[16] J. J. Prevost, K. Nagothu, B. Kelley and M. Jamshidi, *"Load prediction algorithm for multi-tenant virtual machine environments"*, World Automation Congress (WAC), 2012, Puerto Vallarta, Mexico, 2012, pp. 1-6.

[17] Ferreira, Joao, Gustavo Callou, and Paulo Maciel. *"A power load distribution algorithm to optimize data center electrical flow."* Energies 6.7 (2013): 3422-3443.

[18] S. Cherrier, Z. Movahedi and Y. M. Ghamri-Doudane, *"Multi-tenancy in decentralised IoT"*, Cloud Networking (CloudNet), Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on, Milan, 2015, pp. 256-261.

[19] N. K. Chien, N. H. Son and H. Dac Loc, *"Load balancing algorithm based on estimating finish time of services in cloud computing"*, 2016 18th International Conference on Advanced Communication Technology (ICACT), Pyeongchang Kwangwoon Do, South Korea, 2016, pp. 228-233.

[20] W. Yong, T. Xiaoling, H. Qian and K. Yuwen, *"A dynamic load balancing method of cloud-center based on SDN"*, in China Communications, vol. 13, no. 2, pp. 130-137, Feb. 2016.

[21] B. Bhuyan, H. Sarma, N. Sarma, A. Kar and R. Mall, *"Quality of Service (QoS) Provisions in Wireless Sensor Networks and Related Challenges"*, Wireless Sensor Network, Vol. 2 No. 11, 2010, pp. 861-868. doi: 10.4236/wsn.2010.211104.

[22] Z. Ren, W. Shi and J. Wan, *"Towards realistic benchmarking for cloud file systems: Early experiences,"* Workload Characterization (IISWC), 2014 IEEE International Symposium on, Raleigh, NC, 2014, pp. 88-98. doi: 10.1109/IISWC.2014.6983048