Enabling High Performance Fog Computing Through Fog-2-Fog Coordination Model

Mohammed Al-khafajiy[1], Thar Baker[1], Atif Waraich[1], Omar Alfandi[2], Aseel Hussien[1]
[1]Faculty of Engineering and Technology, Liverpool John Moores University, Liverpool, UK
Email:M.D.Alkhafajiy@2016.ljmu.ac.uk; T.Baker@ljmu.ac.uk, A.I.Waraich@ljmu.ac.uk; A.Hussien@ljmu.ac.uk
[2]College of Technological Innovations, Zayed University, Abu Dhabi, UAE. Email:Omar.alfandi@zu.ac.ae

***Abstract*—Fog computing is a promising network paradigm in the IoT area as it has a great potential to reduce processing time for time-sensitive IoT applications. However, fog can get congested very easily due to fog resources limitations in term of capacity and computational power. In this paper, we tackle the issue of fog congestion through a request offloading algorithm. The result shows that the performance of fogs nodes can be increased be sharing fog's overload over several fog nodes. The proposed offloading algorithm could have the potential to achieve a sustainable network paradigm and highlights the significant benefits of fog offloading for the future networking paradigm.**

***Index Terms*—Internet of Things, Fog Computing, Resource management, High performance computing, Fog-to-Fog**

I. Introduction

Within the emerging of Internet of Things (IoT), billions of devices will be connected to the IoT network in the near future, therefore, a considerable amount of data will be generated or gathered every second [1]. The current network paradigm, which relies on centralised data-centres (a.k.a. Cloud computing), becomes impractical solution for IoT data due to the long distance between the data source and designated data-cente [2], [3]. In other words, the amount of time taken by data to travel to a data-centre makes the importance of the data vanish. Therefore, the network topology have been evolved to permit data processing at the edge of the network, introducing what so-called `Fog computing` [7], [8]. Fog computing is a network paradigm that provides computation, storage and networking services between IoT things (e.g, sensors and wearable as in [4]–[6]) and traditional cloud computing data-centres [3]. Fog computing is typically located at the edge of network, so it provides low latency, location awareness and improves Quality of Services (QoS) [9] for data streaming and real-time applications/services. It is worth noting that the fog layer in the IoT architecture consists of heterogeneous devices clustered together and forming a fog domains. Each fog device has its own coverage range where the desired fog services are provided and due to node heterogeneity, the type and quality of the provided services (e.g., processing speed and storage capacity) are vary from one fog node to another [3].

Fog can provide elastic resources to large scale processing systems, thus it can work independently (i.e., fog$\Longleftrightarrow$fog) and/or federated with cloud (i.e., fog$\Longleftrightarrow$cloud) [1], [10], [11]. We refer to fog$\Longleftrightarrow$fog as `fog-to-fog coordination`, while fog$\Longleftrightarrow$cloud as `fog-to-cloud collaboration`. Simply put, horizontal nodes cooperation is a *coordination* process, while the vertical nodes cooperation is a *collaboration* process as per Figure 1. In fog-to-cloud collaborations, the fog and cloud will aid each other to serve end-user; hence they can improve the ability to handle big-data acquisition, aggregation, reducing data transportation as well as balancing the computation power used for data processing. While in fog-to-fog coordination, the fogs works together in order to achieve one task for the end-user. Moreover, could also be a relevant decision-making involved within when to use the cloud instead of the fog and vice versa [11]. Delay sensitive applications may gain priority over others to make use of the fog resources while applications that are require more computation power with no time sensitivity maybe directed to the cloud [3], [10]. The decision-making can also be influenced by the demand and location of these resources. For instance, in a large-scale environment monitoring system, local and regional data can be aggregated and mined at fog nodes that can provide timely feedback especially for emergency case such as toxic pollution alerts, while computational intensive tasks can be scheduled for the cloud.

Eventhough fog can be very beneficial for real-time system, it may lose its importance when it gets congested [11], [12], [15]. The fog congestion can be identified/noticed when the number of arrived service's requests are higher than the number of served/processed service's requests at a given time on specific fog due to its limited capability to cope with volume of arrived service's requests. Hence, when fog is congested, the latency could raise for newly arrived services, hence, this mean the importance of fog will be vanished [11], [13]. This will rises the demand for a proper resource management that can handle fog's congestion. In this paper we propose a fog load balancing algorithm which aims at minimising the latency of service's requests through a load balancing model that addresses fog resource managements problem when it become congested.

The reset of this paper is organized as follows: Section II discusses some related work to fog computing and task offloading. Section III describes the proposed fog computing workload balancing and the offloading algorithm. Simulation results and evaluations are presented Section IV. Finally, Section V concludes the paper.

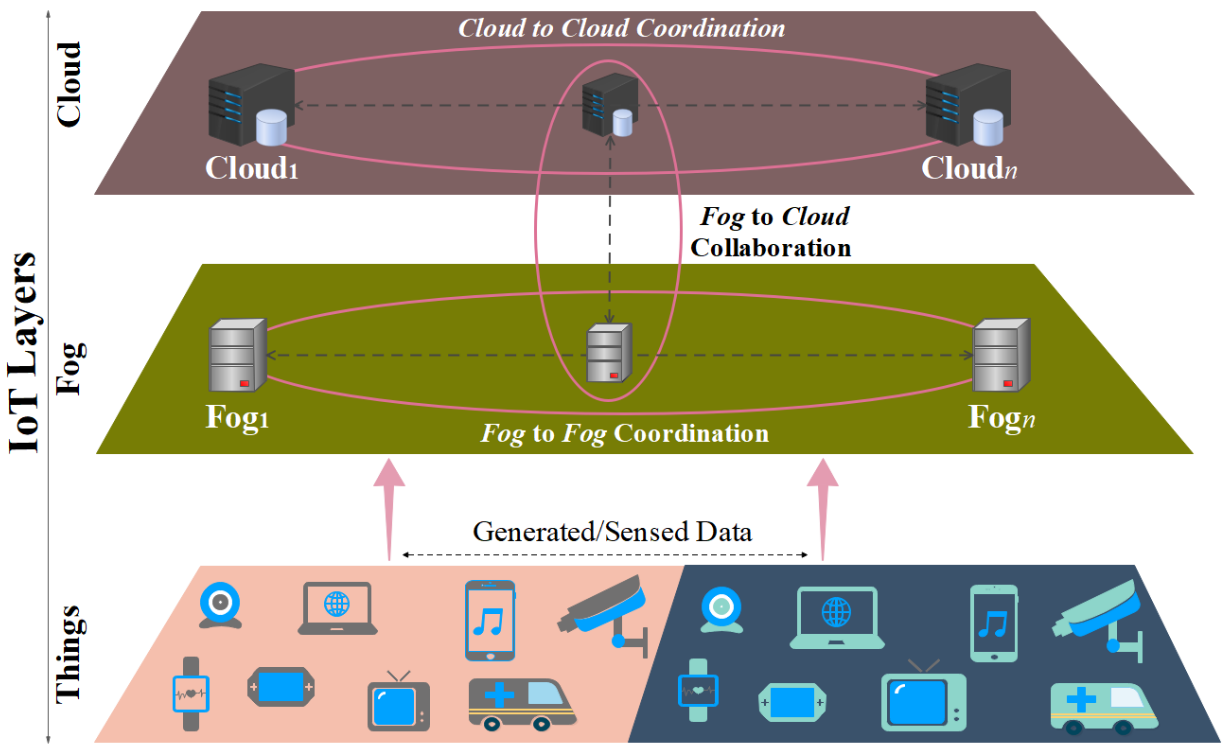

Fig. 1. IoT Layers and horizontal *versus* vertical nodes cooperation

II. Related Work

Recent researches founds that the employment of fog nodes at the edge of the network can improve QoS for IoT applications [14]. However, fog computing is still in its conceptual stage. This section discusses some existing related studies in the area of fog computing that have attempted to use fog for minimizing delay and looked on fog workload balancing. The related research in this section found in a comprehensive studies on fog computing in [14], [16], [18].

Yousefpour et al [11] proposes a delay minimizing collaboration and offloading policy for fog devices that aims to reduce the service delay for IoT applications. They also develop an analytical model to evaluate the policy and show how the proposed framework helps to reduce IoT service delay. In [17], the authors proposed a locality-aware workload sharing scheme for mobile edge computing environments. In the proposed scheme, each node is aware of its neighbouring nodes and their current workloads and utilizes such information for workload sharing in case of high workload. Another distributed latency-aware task processing and offloading model is proposed in the study [19]. In this model, gateways, which act as fog nodes, dynamically exchange their processing and storage capability information. Based on this information, fog nodes probabilistically forward their task to their neighbouring fog nodes or the cloud, when there is a limit in local processing or storage [14]. Lee et al [20] study the problem of fog network formation and task distribution in a cloud-fog architecture. Their main contribution in task allocation for fog nodes by accounting for the dynamic formation of a fog network. Since the locations fog node neighbours is an uncertainty, the authors use an online approach for quickly obtaining information of the fog network and minimizing computational latency accordingly.

Drolia et al [21] argue that offloading computation-intensive tasks, such as those of image recognition, to fog nodes is not always a right decision. In addition, the fog nodes themselves may become the bottleneck of processing delay, if many tasks are offloaded to them. Hence, they propose to use the available resources on edge devices and propose a collaborative scheme between edge devices and fog nodes to pre-fetch parts of the trained models of image recognition onto the device. Comparably, Chen et al [22] introduce the notion of edge computing coalition, which is a collaborative edge-based resource pool of small cell base stations with cloud computing capabilities to serve computation requests. The collaborative edge computing scheme accommodates more computation workloads by offloading workload from overloaded nodes to lightly loaded nodes. Li et al [23], proposes a systematic optimisation framework with the key idea that relaxing Quality of Results (QoR) in applications where a perfect result is not always necessary. Relaxing QoR alleviates the required computation workload and enables a significant reduction of response time and energy consumption. In the proposed framework, the authors consider a mobile edge environment where the computing tasks can be divided, offloaded, and processed in parallel by distributed edge nodes. Beraldi et al [24] proposes and formulate a cooperative offloading policy between two edge data centres for load balancing. The model is based on a simple rule: if a service request arrives at one data center when its buffer is full, the request is offloaded to the other cooperating data center and served by that data center. In addition, the study in [25] analyses an offloading policy between multiple fog data centres installed at the edge of the network in a ring topology. The study also quantitatively models and estimates the gain achieved via cooperation between neighbouring fog data centres in a ring topology.

III. Proposed Fog Workload Balancing

The fog computing network architecture is either application specific or application agnostic, similar to other large-scale distributed systems (e.g., cloud computing). However, to the best of our knowledge, there is no standard fog network architecture for fog computing [2], [18]. In this paper, we adopt a general fog computing architecture in line with [10], [11], [26], [27], given it the mostly renowned fog architecture. The main architecture's layer of the adopted network architecture is composed of $things$, $fogs$, and $cloud$ layers as per Figure 1.

A. Fog Workload distribution

Considering a scenario when a fog node is free and can accept requests based on its current load, it will processes the received service request. However, when the fog node is busy processing other services, it may need to offload the request to other available fog node. There are two approaches to model the interaction among fog nodes in the fog layer [2]; i) Is centralised model, which relies on a central authority node that controls the offload interaction between nodes. ii) Is distributed approach, in which relies on a universal protocol that allows direct interaction among nodes. In the distributed model, there is no need for a centralised node to share the state of fogs, instead, each fog node runs a protocol to distribute their state information to the neighbouring fog nodes and send request out when it become congested. The distributed model is more suitable than centralised model for scenarios where things nodes are not static [2], [11]. Thus this supports the mobility and the flexibility in data acquisition. Therefore, we have adopted this model of interactions in our fog layer.

B. Total Service Request Delay

The service request can be define as a set of tasks, each of which have to be processed in order to provide the desired service for end-user. The service request can be processed in the thing, fog, and/or cloud layers. Processing a service request can be accomplished in different scenarios over different layers in the IoT network, for instance, a request can be processed in the fog layer by multiple fog nodes or between fog and cloud. The total service request delay S_d is calculated using Equation 1.

$$\begin{aligned} S_d = \rho_i^I * S_{proc}^I \\ + \rho_i^F * [D_{tran}^F + D_{prop}^F + D_{com}^F] \\ + \rho_i^C * [D_{tran}^c + D_{prop}^c + D_{com}^c] \end{aligned} \tag{1}$$

Where ρ_i^I is the probability that the $thing_i$ processes the request locally at the IoT layer, ρ_i^F is the probability of sending the request to the fog layer, and ρ_i^C is the probability that the IoT node sends the request directly to the cloud, having ρ_i^I + ρ_i^F + ρ_i^C = 1. Thus, S_{proc}^I is the average processing delay of the $thing_i$ when it processes its own request. D_{prop}^F is propagation delay, D_{tran}^F is sum of all transmission delays. Similarly, D_{prop}^C is propagation delay for cloud server, D_{tran}^C is sum of all transmission delays to cloud server. The D_{com} is the total computational delay for a service request over the fog or the cloud layer, based on the transmission delay (D_{tran}), propagation delay (D_{prop}) and processing delay (D_{proc}) and its computed using Equation 2.

$$D_{com} = D_{tran} + D_{prop} + D_{proc} \tag{2}$$

In order to correctly calculate the delay, we should be clear about where the service will be formed and what parameters are involved in the process. Therefore, in this research our focus on discussing the service processing delay over $fogs$ layer only based on delay parameters of service's transmission, propagation and processing delays.

C. Problem Formulation and Constraints

In the proposed fog system, we aim at improving the QoS, therefore it is crucial to guarantee the minimal service delay to end-users during service processing at the fog layer. The three sources of delay (i.e., transmission, propagation and processing delays) are considered in our proposed model. The total delay for a service in the fog layer is computed by adding the time of uploading service's packets to fog layer ($\tau_{\uparrow}$) from a $thing$, adding to it, the time where a service wasted waiting the queue (τ_{que}) until it get processed by the fog node and this added to the total time as time delay for processing the service (τ_{proc}), and finally the response time to the $thing$ ($\tau_{\downarrow}$). For simplification, we assume that($\tau_{\uparrow}$=$\tau_{\downarrow}$), having ([$\tau_{\uparrow}$=$\tau_{\downarrow}$]=2$\tau_{\downarrow\uparrow}$) as per Equation 3, because logically the returned packets contents normally is similar or smaller than the sent packets.

$$\tau_s = \tau_{\uparrow} + \tau_{que}^s + \tau_{proc} + \tau_{\downarrow}$$

$$\tau_s = \tau_{que}^s + \tau_{pro} + 2\tau_{\downarrow\uparrow} \tag{3}$$

The problem we are focusing on in this paper is how to have an optimal workload on fog nodes with achieving minimal delay for IoT services. Thus, achieving reasonable load includes executing/processing the desired services within the threshold limit of fog capability. In addition, low latency for IoT services include delivering the services before service request's deadline (s_d) with the desired QoS. Therefore, the research problem can be defined as follow:

$$
\begin{aligned}
P: &\quad max[\tau_s] \leqslant s_d, \forall s \in S && (4)\\
\text{s.t.} &\quad f_c^{min} \leqslant f_w \leqslant f_c^{max} && (5)\\
&\quad \sum \lambda_s \leqslant \sum \mu_f && (6)\\
&\quad \lambda_s \xrightarrow{min[D_p]} f_i && (7)\\
&\quad \tau_s \leqslant s_d, \forall s \in S && (8)
\end{aligned}
$$

The constraints of this research is intensively on reduce service latency, therefore, our constraints written with focus on achieving minimal service delay. In constraint (5), we indicates that (f_w) is strictly bounded by upper limit (f_c^{max}) and lower limit (f_c^{min}) of fog capabilities based on CPU frequency (unit $hertz$). Constraint (6) imposes that the total traffic arrival rate (λ_s) to a fog should not exceed the the service rate (μ_f) of this fog. Constraint (7) impose the first destination for the IoT services traffic generated at the things layer will be to a fog node with minimal cost of propagation delay within the fog layer (ideally, lowest propagation delay is for the nearest fog node). Constraint (8) is strictly bound the service time τ_s by service deadline s_d.

D. Task Offloading

The task offloading model is proposed to balance the workload on fog by distribute service's requests from the congested fog node to another avalibale fogs within the fog layer. In order to balance services traffic in fog layer, we impose that fogs in any giving location are reachable for each other, having a mesh fog networing. The fog network modeled as a mesh network to enable fogs communication with each other directly as well as sharing task processing and data transfers among each other. Thus, this assumption is line with [13], [28].

Since we have the constraint ($\lambda_s \xrightarrow{min[D_p]} f_i$) that services are directed to nearest fog node, its valid that services arrival rate can significantly vary from one fog to another [13] depends on fog node location. For instance, imagine the scenario where a fog node placed in the city-centre and another on the edge of the city-centre, obviously, the node that is placed in the city-centre will have a higher workload compared to the one outside the city-centre. Therefore, due to fog's geo-location, the load will vary. In such similar scenario, offloading the services request from a loaded fog (i.e., congested) to an idle fog node could be essential to mitigate the workload and keep the service latency to the minimal, thus improving the QoS.

The decision factors where a node is congested and offloading is required is relay significantly on fog workload (f_w), which associate with the service traffic arrival rate (λ_s) and total processing rate (i.e., service rate μ) that can be achieved based on fog CPU frequency. In addition, service processing time τ_s, which ideally should not exceed service deadline (s_d). Therefore, to make the decision of offloading by fog node is when $\tau_s > s_d$, and can written as follow, having (Off_s) for offloading service decision:

$$Off_s = \begin{cases} 1, & \text{if } \tau_s > s_d \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Thus:

$$\tau_s > s_d, \forall s \in S$$

$$\tau_{que}^s + \tau_{pro} + \tau_{\downarrow} > s_d$$

The Off_s value is set to either 0 or 1, where the value 0 refers to no offloading required. While, value 1 refers to offloading is required as it indicates that the new arrived service will suffer form delay and won't be able to meet the service s_d, hence, service offloading is required.

The best neighbouring fog nodes, to share the overload with a congested fog, are those that provide a service with a minimal delay. Algorithm 1 finds the best node to handle the overload from a congested node, and than offload the overload from the congested node.

Algorithm 1: TASK OFFLOADING

```
   Input: FogNode $(F_n)$; FogLoad $(F_l)$; OverLoad $(O_l)$.
   Parameters : FogCapacity $(F_c)$; Propagation $(D_p)$.
   Initialisation: $F_n = \phi$; $F_c = \phi$; $F_l = \phi$; $O_l = \phi$.
   Result: Share the Overload with best available node
1  $F_L = list\{\phi\}$          ▷ $F_L$ initiate fog list
2  $F_L = list[F_n] \longleftarrow getFogNodes(out : (F_n, F_c))$
3  $F_L = sort(F_L$, by $F_c$ DESC$)$
4  for $each\ F_n \in F_L$ do
5      if $Fn \longleftarrow (F_l \geq Fc_{max})$ then
6          $F_L = pop(Fn)$       ▷ remove busy node
7      else
8          $\tau_s = \sum_{i=1}^{n}[\tau_{que}^i + \tau_{pro}^i + \tau_{\downarrow}]$
9          if $(\tau_s < s_d)$ then
10             $list.add(F_n, \tau_s)$
11             continue
12         else
13             $F_L = pop(Fn)$
14             $F_L = update(F_L)$
15 if $F_L \neq \phi$ then
16     $F_n = min[F_L(\tau_s, D_p)])$
17     $F_l^n = F_l + O_l$
18 else
19     $searchAgain(F_l = \phi)$
20     goto:1
21 return $F_n$
```

The first part of the algorithm perform the process of finding the best available fog node(s) to handle the overload of the congested fog node. Lines 2-3 of the algorithm initiate the list of active fogs in the domain alongside with the node's capacity and current load (i.e., queue size). The list of available nodes will be refined by removing the nodes that are already busy with other services (i.e., $\lambda_i = \mu_i$) as per

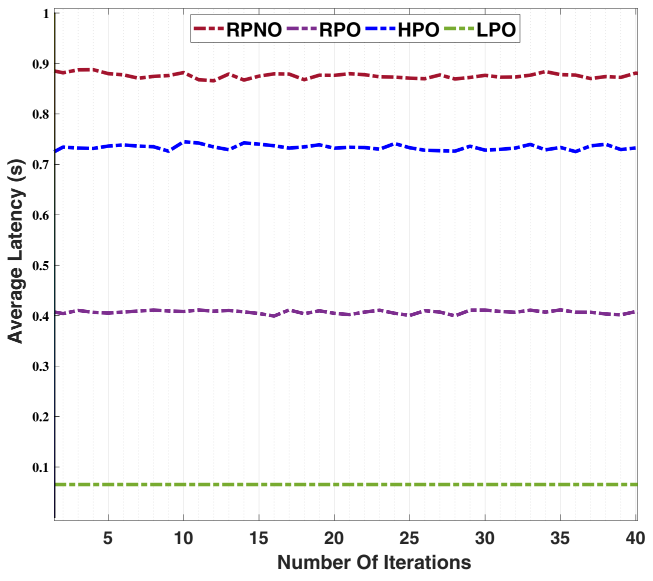

Fig. 2. Average latency based on packet's size

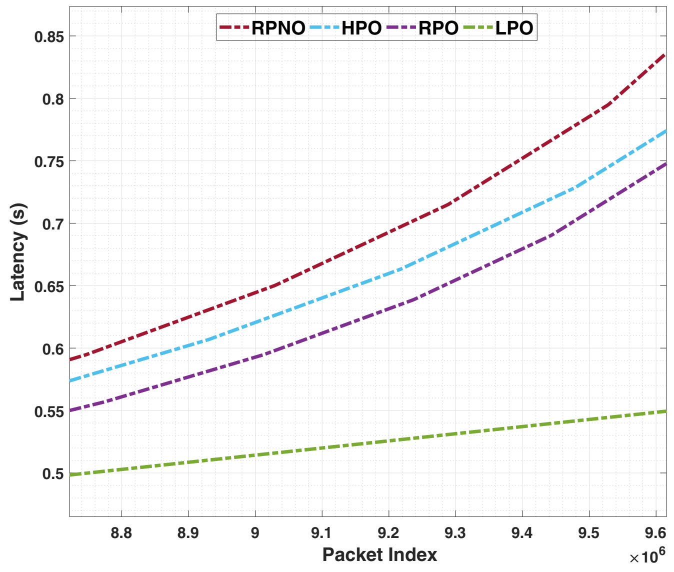

Fig. 3. Latency per packet to show the variations upto packet index 10^6

lines 5-7. In the remaining part of the algorithm, lines 8-13 compute the time required for a service to run on each of the available nodes. If the time is within the limit allowed for the service (i.e, before S_d), the algorithm will be keep the node in the list and log the expected service processing time against the fog node as per line 10. Moreover, if the τ_s on F_n is less than S_d, than F_n will be automatically removed from the fog list as per line 13. The last part of the algorithm shows the process of receiving the list of best available nodes and if the list is not empty, that means there is at least one fog able to take the overload from the congested fog. However, if there is more than one fog node in the list, the algorithm will direct the overload to a fog node that can provide minimal τ_s and has the lowest propagation delay D_p as per lines 14-17. The outcome of this algorithm will be a fog node address and estimated service time τ_s to assist the congested fog node.

IV. Performance Evaluation

In this section we evaluate the proposed the offloading model which aims at enabling optimal fog workload with minimal latency for the IoT services, through a simulation. The system model has been simulated by implementing an environment in MATLAB that mimic real-world interactions between IoT nodes and fog nodes. The simulation settings are presented in Table I.

A. Simulator setup

The simulator was developed using Matlab 2019a, with the aim of simulating a real-world interactions network between thing and fog nodes focuses on achieving minimum delay for IoT services requests sent from things through fog-to-fog coordination model. The simulation runs on a Lenovo ideaPad with Intel Core $i5$ processor and $8GB$ of RAM. The basic components of our simulator are fog node with the proposed offloading algorithm and two type of things; one that can provide heavy-packets (e.g., camera) and another that provide low-packets (e.g., ambient sensor).

TABLE I
Simulation Settings

Parameter	Value
Operating system	Win 10
Simulation environment	Matlab 2019a
Number of fog nodes	12
Fog node CPU	$[0.5 - 1.5]$GHz
Network topology	Mesh
Number of service's requests	10^6
Package Size	$[0.2 - 80]$KB

Fog node: is presented as a class specifies hardware characteristics of fog devices and their connections to other fog devices and things. The main attributes of this class are the processing time and queuing time alongside to the offloading algorithm to handle the overload.

Things node: the instances of the thing class are entities that act as IoT ambient sensor. The class contains attributes representing the characteristics of a sensor, like the size of the packet (i.e., heavy and low packets), the number of packets per second. By setting appropriate values of these attributes, different devices can be simulated (e.g., simulate data from a light sensors or from a cameras).

B. Performance Result

The performance metric that we used to evaluate our algorithm is based on the average time taken to process a service. This metric mainly reflects the efficiency of service's computation, the lowest the service time, the better. Figure 2 illustrates the performance of our proposed offloading algorithm based on the average response time for all received services according to packets size; having heavy packets offloading (HPO), low packets offloading (LPO) and random packets offloading (RPO). The RPO is a mix between heavy and low packet's sizes. The RPNO refers to random packet's size processed without offloading. In Figure 2 the fog nodes set with different capabilities; hence, nodes are vary in their service rate μ to mimic real-world scenarios where fog nodes are not equals in capacities. It is clear that using the offloading algorithm would influence the average processing time. Thus, Over 40 iterations, the arrearage latencies for all LPO, RPO and HPO are lower than the arrearage latency for RPNO through all iterations. Figure 3 shows the impact of the increasing the number of packets on the latency. In this simulation, the packets are varied from 1 to 10^6 packet per second. The packets acquisition rate is incremental parameter from 1% to 100%, thus, this rate is fixed at any given timestamps for constancy. It is obvious that increasing the number of arrived packets will increase the overall latency. However, the proposed offloading algorithm still achieving lower latency than processing the packets without offloading from a congested fog node.

V. Conclusion and Future Work

The main advantage of fog computing is that the hardware resources are placed "closer" to IoT Things. This allows services relaying on IoT inputs to be carried out with minimal latency and benefit real-time applications by provide a fast and reliable service response time. Although, fog is beneficial, it could lose its importance when it get congested due to its resources limitations in term of capacity and computational power. In this paper, we tackle the issue of fog congestion through request offloading algorithm. The result shows that the performance of fogs nodes can be increased be sharing fog's overload over several fog nodes. In our future work, we are planing to extend the simulation using a data processing techniques (e.g, MapReduce or Spark). Also, study security issues that associate with malicious fogs and/or attacks that aims at reducing fog performances.

References

[1] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," in IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 416-464, First quarter 2018.

[2] Al-khafajiy M, Baker T, Al-Libawy H, Maamar Z, Aloqaily M, Jararweh Y. Improving fog computing performance via fog-2-fog collaboration. Future Generation Computer Systems. 100:266-80. 2019.

[3] M. Al-khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers and O. Alfandi, "Fog Computing Framework for Internet of Things Applications," 2018 11th International Conference on Developments in eSystems Engineering (DeSE), Cambridge, United Kingdom, pp. 71-77. 2018.

[4] Al-khafajiy M, Baker T, Chalmers C, Asim M, Kolivand H, Fahim M, Waraich A. Remote health monitoring of elderly through wearable sensors. Multimedia Tools and Applications. 24:1-26.Jan 2019.

[5] Maamar Z, Baker T, Faci N, Al-Khafajiy M, Ugljanin E, Atif Y, Sellami M. Weaving cognition into the internet-of-things: Application to water leaks. Cognitive Systems Research. 56:233-45. 1 Aug 2019.

[6] Maamar Z, Baker T, Faci N, Ugljanin E, Atif Y, Al-Khafajiy M, Sellami M. Cognitive computing meets the internet of things. In13th International Conference on Software Technologies, ICSOFT, Porto, Portugal, July 26-28, 2018; pp. 741-746. SciTePress. 2018.

[7] Cisco Systems, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," Www.Cisco.Com, p. 6, 2016.

[8] D. Evans, "The Internet of Things - How the Next Evolution of the Internet is Changing Everything," CISCO white Pap., no. April, pp. 1–11, 2011.

[9] M. Al-khafajiy, T. Baker, A. Waraich, D. Al-Jumeily and A. Hussain, "IoT-Fog Optimal Workload via Fog Offloading," 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, 2018, pp. 359-364.

[10] Fan Q, Ansari N. Towards Workload Balancing in Fog Computing Empowered IoT. IEEE Transactions on Network Science and Engineering. 2018 Jul 4.

[11] A. Yousefpour, G. Ishigaki, R. Gour and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 998-1010, April 2018.

[12] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z.Patrikakis, "A Cooperative Fog Approach for Effective Workload Balancing," IEEE Cloud Comput., vol. 4, no. 2, pp. 36–45, Mar. 2017.

[13] Wang X, Ning Z, Wang L. Offloading in Internet of vehicles: A fog-enabled real-time traffic management system. IEEE Transactions on Industrial Informatics. 2018 Mar 16.

[14] Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F, Niakanlahiji A, Kong J, Jue JP. All one needs to know about fog computing and related edge computing paradigms: a complete survey. Journal of Systems Architecture. 2019 Feb 12.

[15] Al-Khafajiy M, Webster L, Baker T, Waraich A. Towards fog driven IoT healthcare: challenges and framework of fog computing in healthcare. InProceedings of the 2nd International Conference on Future Networks and Distributed Systems; p. 9; ACM. 26 Jun 2018.

[16] Mouradian C, Naboulsi D, Yangui S, Glitho RH, Morrow MJ, Polakos PA. A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE Communications Surveys & Tutorials. 2017 Nov 9;20(1):416-64.

[17] A. Jonathan , A. Chandra , J. Weissman , Locality-aware load sharing in mobile cloud computing, in: Proceedings of the10th International Conference on Utility and Cloud Computing, in: UCC '17, ACM, 2017, pp. 141–150.

[18] Naha RK, Garg S, Georgakopoulos D, Jayaraman PP, Gao L, Xiang Y, Ranjan R. Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions. IEEE Access. 2018 Aug 22.

[19] K. Desikan , M. Srinivasan , C. Murthy , A novel distributed latency-aware data processing in fog computing-enabled iot networks, in: Proceedings of the ACM Workshop on Distributed Information Processing in Wireless Networks, ACM, 2017.

[20] G. Lee , W. Saad , M. Bennis , An online secretary framework for fog network forma- tion with minimal latency, in: 2017 IEEE International Conference on Communi- cations (ICC), IEEE, 2017.

[21] U. Drolia , K. Guo , P. Narasimhan , Precog: prefetching for image recognition appli- cations at the edge, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, ACM, 2017.

[22] L. Chen , J. Xu , Socially trusted collaborative edge computing in ultra dense net- works, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, ACM, 2017

[23] Y. Li, Y. Chen, T. Lan and G. Venkataramani, "MobiQoR: Pushing the Envelope of Mobile Edge Computing Via Quality-of-Result Optimization," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017.

[24] R. Beraldi , A. Mtibaa , H. Alnuweiri , Cooperative load balancing scheme for edge computing resources, in: 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, 2017.

[25] C. Fricker , F. Guillemin , P. Robert , G. Thompson , Analysis of an offloading scheme for data centers in the framework of fog computing, ACM Trans. Model. Perform. Eval. Comput. Syst. (TOMPECS), 2016.

[26] Deng R, Lu R, Lai C, Luan TH, Liang H. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet of Things Journal. 2016;

[27] Kim WS, Chung SH. User-Participatory Fog Computing Architecture and Its Management Schemes for Improving Feasibility. IEEE Access. 2018 Mar 19.

[28] Salahuddin MA, Al-Fuqaha A, Guizani M. Reinforcement learning for resource provisioning in the vehicular cloud. IEEE Wireless Communications. 2016 Aug;23(4):128-35.