Keep it Flat (KiF): Resource Management in Integrated Cloud-Fog Networks

Neam M. Farroukh
Computer Science Department
American University of Beirut
Beirut, Lebanon
Email: nmf14@mail.aub.edu

Mohamed Nassar
Computer Science Department
American University of Beirut
Beirut, Lebanon
Email: mn115@aub.edu.lb

Shady Elbassuoni
Computer Science Department
American University of Beirut
Beirut, Lebanon
Email: se58@aub.edu.lb

Haidar Safa
Computer Science Department
American University of Beirut
Beirut, Lebanon
Email: hs33@aub.edu.lb

***Abstract*—Fog computing extends the cloud services and brings them to the edge of the network. By taking advantage of edge devices that have sufficient resources (i.e., storage, compute, and bandwidth), the cloud becomes closer to the edge. The fog has been proved as a promising solution for avoiding unbearable latency and network capacity saturation with the proliferation of Internet of Things (IoT) end-devices. Lately, researchers have investigated the impact of cloud-fog cooperation on the performance of the network in terms of latency, network capacity and security. While the cloud can handle heavy-weight delay-tolerant tasks, the fog becomes in charge of all light-weight delay-sensitive tasks. In such integrated networks, resource management becomes a key challenge that must be addressed effectively. In this paper, we design and study two different resource management strategies at the fog layer: a flat one versus a clustered one. Both strategies are formalized as optimization problems and constrained by minimum resource allocation requirements, as well as Quality of Service (QoS) and privacy requirements. The comparison of the two strategies shows the superiority of the flat approach in terms of overall performance and fog delay, while the clustered approach results in lower number of overall tasks being rejected.**

***Keywords*—fog computing; resource management; security; latency; optimization.**

I. Introduction

Computing in general is an on-demand utility model where users opt to benefit from services without worrying about where these services are hosted or how they will be delivered. Cloud computing is a well established technology defined as a tool that provides ready-to-consume resources like CPU, I/O, and memory based on the users' demands. Lately, the number of smart end-devices, renowned as the Internet of Things (IoT), has been proliferating at a tremendous scale. Consequently, such a number of devices is going to produce trillion gigabytes of data [1]. Imagine this huge amount of data being sent to a centralized and far located cloud. This will cause network saturation and severe degradation in users' experience. Thus, cloud computing will lose its luster for latency-sensitive applications that require resources just in the vicinity.

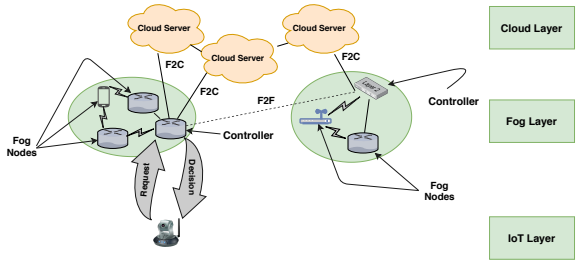


Figure 1. Network Architecture.

To address this issue and to meet the delay and mobility requirements of various IoT applications, it is necessary to have an intermediate control layer that resides between the end-devices and the far located cloud. This shifting from the core to the edge of the network is termed as edge computing. Subsequently, Cisco proposed Fog Computing (FC) in 2012 [2]. According to Cisco, the fog is just another cloud layer that is closer to the IoT devices. The fog extends the assets of the original cloud as storage, computing and networking services to the edge of the network by taking advantage of devices, e.g., access points, routers, that are rich in resources and located near end-devices. The aim of the fog is to provide lower latency and better user experience.

Resource management in fog computing is still considered as a key challenge due to the limited computational resources of edge devices and their heterogeneity. It is essential to address this challenge in a way that optimizes the fog resources while satisfying the QoS and privacy requirements of the IoT applications and their computational tasks. Therefore, the research question that we address in this paper is the

Copyright (c) IARIA, 2021. ISBN: 978-1-61208-878-5

following: *Given a backend cloud with "infinite" resources and considerable latency, a pool of fog nodes with different characteristics and arriving IoT tasks with different requirements, what is the best distributed and hierarchical strategy to support scheduling and assignment decisions?*

The rest of the paper is organized as follows: In Section II, we present background information. In Section III, we review related work. Section IV discusses the system model. The scheduling strategies are detailed in Section V. Simulation and experimental results are tailored in Section VI. Finally, Section VII concludes the paper and sheds the light on future work.

II. Background

Cloud computing replaced traditional hosting by enabling customers to rent compute resources like applications, storage and virtual machines through Internet. Similar to any other utility like water and electricity, users do not need to worry about managing and maintaining the utility infrastructure [3]. Cloud computing follows the Pay-as-You-Go usage model, which facilitates the scaling and customization of computing resources. The edge is any computing and network resources that reside between the end-users and the cloud data centers. The aim behind fog computing is to perform all the processing and computing at the proximity of data resources and thus to minimize the latency. An IoT device is any device that is able to transmit and receive data, and has an attached sensor or actuator. IoT devices are becoming part of every aspect of our lives since they give more control on routine work and personal tasks. For that, IoT applications have been deployed in various areas such as smart homes, smart cities, transportation, and healthcare. Examples of IoT applications are door locks, smart heating, coagulation testing in medicine and smart traffic signals.

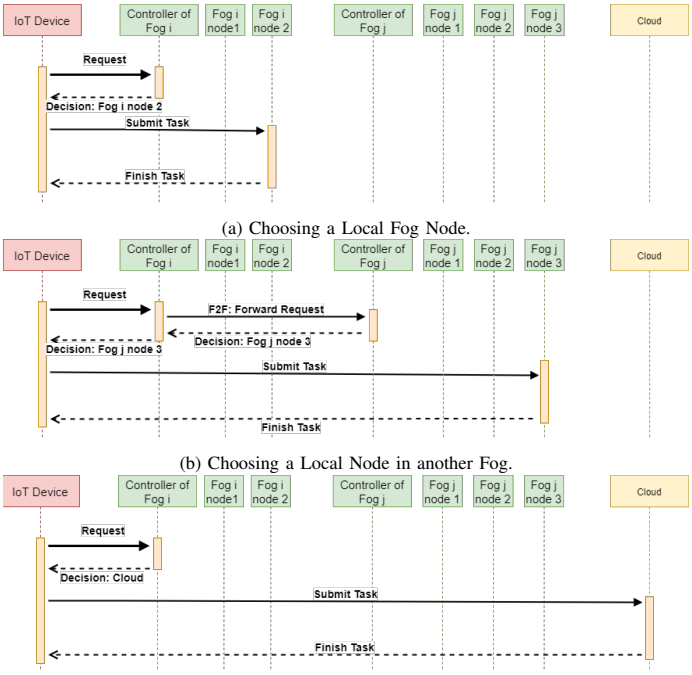
III. Related Work

A task scheduling algorithm in the fog layer based on priority levels is proposed in [4]. The fog nodes in the fog layer can communicate with each other for efficient resource allocation and load balancing. The tasks are first processed in the fog layer based on their priority levels. Only when all the micro datacenters in the fog layer are saturated that tasks are propagated to the cloud layer. A more real-time oriented resource management approach is proposed in [5]. Factors, such as fluctuating relinquish probability of the customer, service type, service price, and variance of the relinquish probability are taken into account. In the proposed architecture, the fog node is capable of predicting the consumption of resources for a particular customer. The Distributed Earliest deadline First (DEF) algorithm was proposed in the context of symbiotic fog computing [6]. The presented model accounts for dynamic resources that arrive into the system for an interval of time and lend a fraction of their computing capacities against financial incentives. The assignment design is seen as a recommendation system. Given a task's requirements, multiple nodes can be recommended based on task similarity, node similarity and node previous performance on a similar task. A Fog Resource Selection algorithm (FResS) is proposed in [7]. The proposed model collects and maintains a repository of performance data in the form of execution logs, and uses the data to train a neural network model. When a new task arrives, the neural network model predicts the amount of required resources and uses it for task placement and estimating the execution time. The load balancing problem under the constraint of achieving the minimum latency in Fog networks is also addressed in [8]. A reinforcement Q-learning based decision-making process is proposed to find the optimal offloading decision with the assumption of unknown reward and transition functions. The proposed process allows fog nodes to offload an optimal number of tasks to the cloud. The lack of approaches for the leasing and releasing of resources in fog computing is also highlighted in [9]. A conceptual framework and an optimization problem for fog resource provisioning are presented. The optimization problem has the goal to provide delay-sensitive utilization of available fog nodes. The resource provisioning plan is generated by an orchestration node. We follow a similar approach in this paper. Our work is different from previous related work for two reasons: (1) we focus on the orchestration topology and differentiate between a flat one versus a clustered one, (2) we add abstract variables representing security and privacy to our model.

IV. System Model

Our model (Figure 1) is derived from [10]: the cloud manages the fog and handles the heavy-weight delay-tolerant IoT tasks, while the fog is responsible for handling the light-weight delay-sensitive IoT tasks. Each fog cluster is assigned a number of IoT devices that are directly connected to it. In the flat scenario, an IoT device can reach to any of the other fog clusters for requesting a task. In the clustered scenario, the task issued from an IoT device is directed to the controller of the cluster it is connected to. Fog controllers communicate to migrate a task from a cluster to another. Inter-cluster communication should be harnessed to achieve load balancing and efficient task distribution. Moreover, fog controllers communicate with the cloud to avoid fog saturation and offload tasks that are deemed heavy weight or delay-tolerant.

Each fog node f_j is represented as a profile vector $\{CPU_j, Memory_j, PM_j\}$ corresponding to its available CPU type and available Instructions Per Second (IPS), memory, and privacy, at time t, respectively. We omit t to simplify the notation. The privacy measure is a trust value representing the security and privacy strength of a fog node. Many approaches can be used to build such a trust model, [11] in particular. Each task T_i arriving at the fog layer is represented by a profile vector $\{CPU_i, Memory_i, SL_i, IC_i, Bytes_i, MAD_i, SC_i\}$. The entries in the vector correspond to the minimum required CPU type and available IPS, the minimum required memory, the minimum security level, the task instruction count, the task data size in bytes, the maximum allowed delay, and the scheduling class (priority level), respectively. The scheduling class is one if



(a) Choosing a Local Fog Node.

(b) Choosing a Local Node in another Fog.

(c) Offloading Task to Cloud.

Figure 2. Three Possible Scenarios for an IoT task.

the task is delay tolerant and zero otherwise. The maximum allowed delay attribute determines the latency requirement of the task. Thus, a fog node can be assigned a task if the total time it takes to process the task along with the total transmission, routing and propagation delays do not exceed the maximum allowed delay attribute of the task. We represent the link between any two network devices, source s and destination d, by the profile vector $\{BW_{s,d}, PD_{s,d}\}$ corresponding to the upload bandwidth and propagation delay, respectively.

Each fog controller is logically connected to a set of IoT devices. The IoT device is homed with one or more fog controllers. Upon receiving a request from an IoT device, the controller's job is to take a decision on the best way to handle this task. The controller replies to the IoT source with the decision and additional information that helps proceeding with the task. As shown in Figure 2, three scenarios are possible:

(a) the controller predicts that the task can be accomplished by a local node and responds with the IP address of this node,
(b) the controller has a busy cluster, estimates that some extra delay is tolerable and forwards the task to a neighbor cluster's controller. Note that this can be done in an iterative way by forwarding the task and relaying the response to the client, or in a recursive way by directly responding with the IP address of the next fog controller.
(c) the controller decides that the task is delay-tolerant and not computationally affordable at the fog layer at this moment, the task can be offloaded to the cloud. The controller replies with IP address of the cloud service.

In case where the constraints cannot be fulfilled by any of these three scenarios, the task is rejected and the IoT device has to try at another time, or try another controller in the case of multi-homing.

V. The Task Allocation Strategies

Our formulation uses Integer Linear Programming (ILP) since our decision variables are discrete (0 or 1). ILP problems are NP-complete, however, efficient solvers can be used to deal with our formulations. For instance, we have used the *pywraplp* linear solver module from OR-Tools [12].

A. Flat-Based Fog Node Selection

In this variation, the controller has the full knowledge of the fog layer. It can build this knowledge by receiving periodic updates from the other controllers. The updates contain timely information about the available fog nodes and their capabilities. The controller has to keep measurements of the Round Trip Times (RTTs) for all the nodes. We define X_i^j as the decision variable for our optimization problem. $X_i^j = 1$ means that Task T_i is assigned to fog node f_j and 0 otherwise. In case where the output of the optimization is all-zeros decision variables, none of the reachable fog nodes is suitable for executing the task. If the scheduling class of the task is one, the decision will be to offload the task to the cloud, otherwise the task will be rejected. The Spare Time (ST_i^j) is the difference between the maximum allowed delay and the predicted total delay for node f_j (TD_i^j) as in (1) and (2).

$$ST_i^j = MAD_i - TD_i^j \tag{1}$$

$$ST_i^j \geq 0 \tag{2}$$

We denote P_i^j the difference between the minimum required privacy/security level for task T_i and the privacy measure PM_j of the fog node f_j as in (3) and (4).

$$P_i^j = SL_i - PM_j \tag{3}$$

$$P_i^j \geq 0 \tag{4}$$

Assuming that a fog controller receives a batch of tasks $i = 1, 2, .., N$ and has a set of reachable nodes $j = 1, 2, ..., M$, the controller has to solve for a sequence of objective functions. Each function finds an integer vector assignment X_i that maximizes the weighted average of the spare time and the privacy difference as controlled by a variable α. The sequence of objective functions is depicted in (5) and is subject to constraints (6), (7), (8), (9), and (10).

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{M} (\alpha ST_i^j + (1-\alpha) P_i^j) * X_i^j \\ & \forall i \in \{1, \ldots, N\} \end{aligned} \tag{5}$$

Each task can be assigned to exactly one fog node. This constraint is formalized in (6). To guarantee (2) and (4), we add the constraints in (7) and (8).

$$\begin{array}{lll} \sum_{j=1}^{M} X_i^j \leq 1 & & \forall i \\ X_i^j \in \mathbb{Z} & \forall j & \forall i \\ X_i^j \geq 0 & \forall j & \forall i \end{array} \tag{6}$$

$$\sum_{j=1}^{M}(ST_i^j * X_i^j) \geq 0 \quad \forall i \tag{7}$$

$$\sum_{j=1}^{M}(P_i^j * X_i^j) \geq 0 \quad \forall i \tag{8}$$

Moreover, we need to meet the resources requirements of CPU and memory of the task by comparing it to the dynamic remaining (available) CPU and memory at the node as in (9) and (10).

$$\sum_{j=1}^{M}(\text{CPU}_j^R - \text{CPU}_i) * X_i^j \geq 0 \quad \forall i \tag{9}$$

$$\sum_{j=1}^{M}(\text{Mem}_j^R - \text{Mem}_i) * X_i^j \geq 0 \quad \forall i \tag{10}$$

B. Clustered-Based Fog Node Selection

A second variation is to go along the clustered topology. We allow the controller to select the best cluster instead of directly searching for the best node. For this purpose, we represent each fog cluster *k* by a profile vector: $\{AvgCPU_k, AvgMem_k, AvgPM_k\}$. This vector specifies the cluster's average available CPU, average available memory, and average privacy measure, respectively. These values are computed based on the profile vectors of the fog nodes belonging to each cluster. Each fog controller periodically receives timely cluster profiles from the other controllers. The profile can also be retrieved in a pull manner. This method reduces the size of the optimization problem as we are currently looking for the best cluster rather than the best fog node. We define Y_i^k as the decision variable for the cluster C_k and task T_i. $Y_i^k = 1$ means that cluster C_k is selected for task T_i, and the request will be forwarded to its fog controller. If C_k happens to be the controller's cluster, a local fog node is selected. If the output of the optimization is the all-zeros vector, the decision is solely based on the task scheduling class. If it is delay tolerant, we offload it to the cloud, otherwise, we reject it.

The Spare Time (ST_i^k) is the difference between the maximum allowed delay and the predicted total delay for a node f_j in cluster C_k in average, as in (11) and (12).

$$ST_i^k = MAD_i - TD_i^k \tag{11}$$

$$ST_i^k \geq 0 \tag{12}$$

We denote P_i^k the difference between the minimum required privacy/security level for task T_i and the average privacy measure of the cluster C_k, as in (13) and (14).

$$P_i^k = SL_i - AvgPM_k \tag{13}$$

$$P_i^k \geq 0 \tag{14}$$

The sequence of objective functions is defined in (15) and is subject to constraints (16), (17), (18), (19), and (20).

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{M}(\alpha ST_i^k + (1-\alpha)P_i^k) * X_i^k \\ & \forall i \in \{1, \ldots, N\} \end{aligned} \tag{15}$$

α is a hyper-parameter used to control the weights of privacy versus QoS. Each task can be assigned to exactly one cluster node as in (16). To guarantee (12) and (14), we add the constraints in (17) and (18), respectively.

$$\begin{aligned} & \textstyle\sum_{k=1}^{K} Y_i^k \leq 1 && \forall i \\ & Y_i^k \in \mathbb{Z} && \forall k \quad \forall i \\ & Y_i^k \geq 0 && \forall k \quad \forall i \end{aligned} \tag{16}$$

$$\sum_{k=1}^{K}(ST_i^k * Y_i^k) \geq 0 \quad \forall i \tag{17}$$

$$\sum_{k=1}^{K}(P_i^k * Y_i^k) \geq 0 \quad \forall i \tag{18}$$

The selected cluster also needs to meet the resources requirements of CPU and memory of the task by comparing them to the dynamic available average CPU and average memory at the cluster, as in (19) and (20).

$$\sum_{k=1}^{K}(\text{AvgCPU}_k^R - \text{CPU}_i) * Y_i^k \geq 0 \quad \forall i \tag{19}$$

$$\sum_{k=1}^{K}(\text{AvgMem}_k^R - \text{Mem}_i) * Y_i^k \geq 0 \quad \forall i \tag{20}$$

C. Delay Calculation

The Total Delay (TD) is an important optimisation factor since it represents the QoS contribution to the decision taken by the controller. The total (or end-to-end delay) for a given task is the time difference between the moment when the task has been issued and the moment marking the end of the task execution. The controller has to estimate the round trip time between the IoT device and each of the fog nodes in the selection pool. We describe the delay calculation for the flat versus clustered scenarios:

Flat. The delay for task T_i is the sum of the delay at the controller C_j and the delay at the selected fog node f_j:

$$TD_i^j = d_{C_j} + d_{f_j} \tag{21}$$

Clustered. The delay for task T_i is the sum of the delay at the controller C_j, the delay at the chosen cluster's controller C_k and the delay at the selected fog node $f_{k'}$:

$$TD_i^j = d_{C_j} + d_{C_k} + d_{f_{k'}} \tag{22}$$

Other delay calculations are also possible. For instance, a task can be forwarded from a cluster to another more than one time. The task can be offloaded to the cloud, or rejected. We do not consider the delay calculation for these cases in our work. Note that we overloaded the notation of a cluster C_k to denote the controller at this cluster. We only consider the iterative scenarios.

The delay at a controller C_k or at fog node *j* is calculated as the sum of four terms:

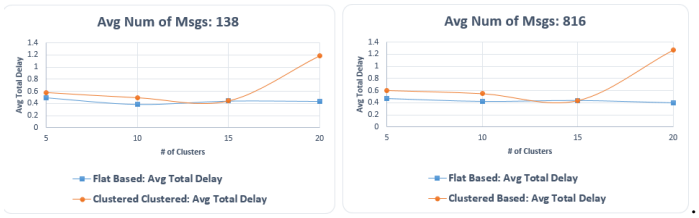
$$d = d_{\text{transmission}} + d_{\text{propagation}} + d_{\text{processing}} + d_{\text{queuing}} \tag{23}$$

Note that some of these terms are also composed as a sum of multiple delays of the same nature. For example, $d_{\text{transmission}}$ involves the round trip transmission. $d_{\text{propagation}}$ involves the round trip propagation as well. The profile vectors $\{BW_{s,d}, PD_{s,d}\}$ $\forall s$ $\forall d$ are used in these calculations.

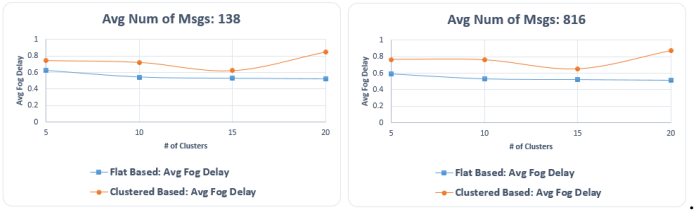
VI. Simulation and Experimental Results

To implement and analyze our proposed variations, we used the Yet Another Fog Simulator (YAFS) [13]. Preliminary simulations showed that a value of $\alpha = 0.9$ achieves a good balance in between privacy and QoS. Four types of topology were created, each containing 5, 10, 15, and 20 clusters, respectively. Each fog cluster consists of a number of fog nodes, having a single controller and a number of IoT devices directly connected to it. Each cluster in every type of topology has a small number of fog nodes, since as mentioned in [10], small scale fogs would result in better performance metrics. In the simulation, the number of fog nodes belonging to a cluster ranges between 4 and 8. Each fog cluster has a range between 3 to 5 IoT devices directly connected to it. To set the characteristics of the fog nodes, we used values from real servers. As for the privacy measure attribute, we sampled values from a uniform distribution. We assumed that each cluster has a range of privacy measures. The privacy measure for any fog node belonging to the cluster is within this specified range. For example, assume having fog cluster k with privacy measure range between 0.3 and 0.5. Each value in the range represents a trust value. The trust value can be obtained in reality based on a security and privacy assessment tool as described in [11]. The CPU and IPS parameters of the cloud are set higher than any value being set to the fog nodes. This is due to the fact that the cloud has higher processing capabilities than any fog node. As for the connections, the bandwidth between an IoT device and a fog node could be either 54 Mbits/s as in wireless 802.11g networks or 100 Mbits/s as in fast Ethernet. The bandwidth between the controllers, which act as routers for IoT devices, and the cloud is set to 10 Gbits/s. While the bandwidth between fog controllers is set to 100 Mbits/s. We adopted these values from the topology created in [14]. We created five sets each containing 100 different types of tasks. For every set of tasks and each type of topology (which differs by the number of clusters available), four experiments were performed. In each experiment, different simulation times were set to increase the number of tasks being generated.

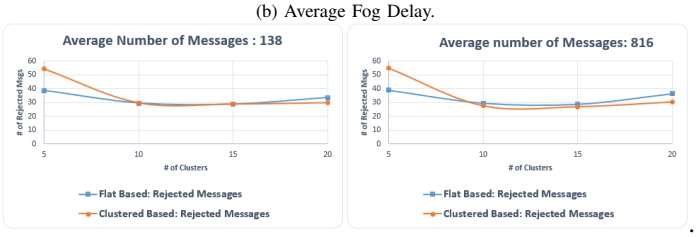
The results were evaluated based on the average fog delay which is the delay of tasks being executed in the fog layer, the average total delay, the number of tasks being rejected, and the number of tasks being offloaded to the cloud. We present the results for when a low and a high number of tasks are generated based on simulator time (138 tasks and 816 tasks on average). Our goal is to evaluate the behavior of both variations when the number of tasks being generated in the network increases. The results for both approaches are shown in Figure 3. The results of the flat based approach show that the number of tasks being rejected and offloaded to the cloud and the average fog and total delay decreases as the number of fog clusters in the network increases from 5 to 20. This decrease is due to the higher probability of task to node assignment with the increase in the number of fog nodes that the generated IoT tasks can be assigned to in the



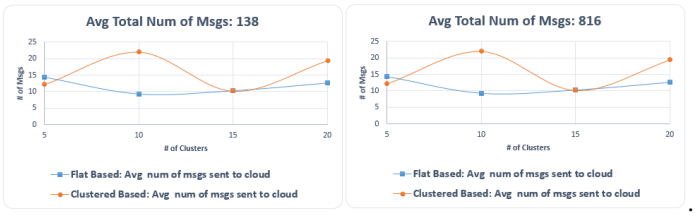
(a) Average Total Delay.



(b) Average Fog Delay.



(c) Rejected Tasks.



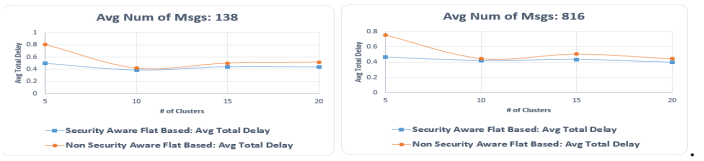
(d) Offloaded Tasks to Cloud.

Figure 3. Flat Based vs. Clustered Based.

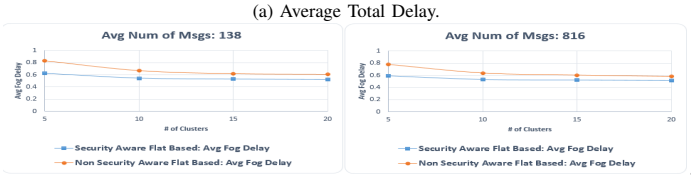
network. As for the results of the clustered based approach, it shows that the average fog delay and number of tasks being rejected and offloaded to the cloud decreases as the number of clusters available in the network increases. The average total delay starts to decrease as the number of clusters increases to reach 15 then increases as this number reaches 20. This can be justified by the increase in the number of clusters that the controller has to examine when selecting a suitable fog node for assignment. Comparing results of both variations in Figure 3 shows that the flat based variation is more applicable when having a large scale fog topology consisting of 20 or more clusters. This can be justified by the fact that this variation gives lower values for the fog and total delays as desired and decreases the number of tasks being propagated to higher layers (cloud). On the other hand, when having an average scale topology of around 15 clusters, both variations behave the same and thus both are applicable.

Figure 4 and Figure 5 show the impact of including and excluding the privacy and security factors from the formalized optimization problem. The figures show that a security aware variation leads to a higher probability of task rejection and task propagation to higher layers. For repeatability, we provide the

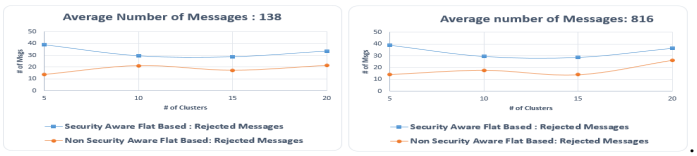
Copyright (c) IARIA, 2021. ISBN: 978-1-61208-878-5



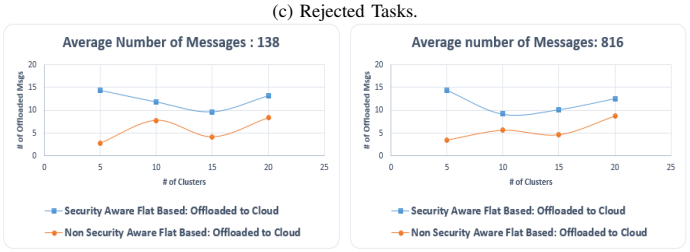
(a) Average Total Delay.



(b) Average Fog Delay.

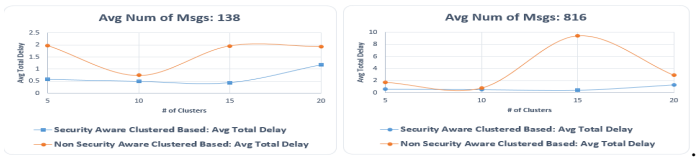


(c) Rejected Tasks.

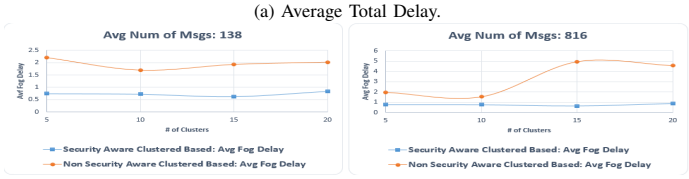


(d) Offloaded Tasks to Cloud.

Figure 4. Flat Based Fog Selection: Security Aware vs. Non Security Aware.

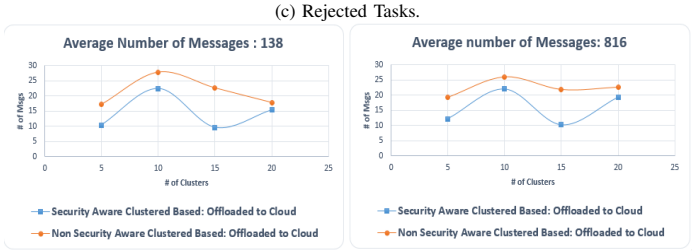


(a) Average Total Delay.



(b) Average Fog Delay.

(c) Rejected Tasks.



(d) Offloaded Tasks to Cloud.

Figure 5. Clustered Based Fog Selection: Security Aware vs. Non Security Aware.

source code of all experiments at [15].

VII. Conclusion and Future Work

In this paper, we proposed, designed and studied two run-time resource allocation strategies, flat-based fog node selection and clustered-based fog node selection.

The results showed that the flat-based strategy permits better performance, especially when the number of clusters in the fog increases. We attribute this result to the potential number of layers of indirection required for demanding IoT tasks. The flat approach promotes a "take it or leave it" behaviour.

In future work, we intend to provide more insights on solving the optimisation problems and the scalability of the solving method for large-scale settings. We want to report measurements on using different solvers, their solving time and accuracy. We will also consider the comparison between solving the optimization problems in batch mode and in direct (one-by-one) mode. We will also consider refining our model to include inter-cluster communication and the exchange of updates. We would like to assess our approaches based on real-world data, and explore whether using reinforcement learning can lead to better recommendations for task allocation in the fog.

Acknowledgements

This work was partially supported by a grant from the university research board of the American University of Beirut (URB-AUB-2020/2021).

References

[1] B. Varghese, N. Wang, D. S. Nikolopoulos, and R. Buyya, "Feasibility of fog computing," *arXiv preprint arXiv:1701.05451*, 2017.

[2] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*. ACM, 2015, pp. 37–42.

[3] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*. IEEE, 2012, pp. 877–880.

[4] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized task scheduling in fog computing," in *Proceedings of the ACMSE 2018 Conference*, 2018, pp. 1–8.

[5] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*. IEEE, 2015, pp. 105–110.

[6] V. Kochar and A. Sarkar, "Real time resource allocation on a dynamic two level symbiotic fog architecture," in *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2016, pp. 49–55.

[7] N. Mostafa, I. Al Ridhawi, and M. Aloqaily, "Fog resource selection using historical executions," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2018, pp. 272–276.

[8] J.-y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–7.

[9] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for iot services in the fog," in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*. IEEE, 2016, pp. 32–39.

Copyright (c) IARIA, 2021. ISBN: 978-1-61208-878-5

[10] L. Peng, A. R. Dhaini, and P.-H. Ho, "Toward integrated cloud–fog networks for efficient iot provisioning: Key challenges and solutions," *Future Generation Computer Systems*, vol. 88, pp. 606–613, 2018.
[11] R. Shaikh and M. Sasikumar, "Trust model for measuring security strength of cloud computing service," *Procedia Computer Science*, vol. 45, pp. 380–389, 2015.
[12] L. Perron and V. Furnon, "Or-tools," Google, https://developers.google.com/optimization/ [accessed July 2021].
[13] I. Lera, C. Guerrero, and C. Juiz, "Yafs: A simulator for iot scenarios in fog computing," *arXiv preprint arXiv:1902.01091*, 2019.
[14] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *2017 IEEE international conference on edge computing (EDGE)*. IEEE, 2017, pp. 17–24.
[15] N. Farroukh, "KiF Github Repository," https://github.com/NeamFarroukh/Keep-it-Flat-KiF-Resource-Management-in-Integrated-Cloud-Fog-Networks.

Copyright (c) IARIA, 2021. ISBN: 978-1-61208-878-5