A Distributed Resource Allocation Algorithm for Task Offloading in Fog-enabled IoT Systems

Hoa Tran-Dang, Dong-Seong Kim

***Abstract*—In the IoT-based systems, the integration of fog computing allows the fog nodes to offload and process tasks requested from IoT-enabled devices in a distributed manner to reduce the response delay. However, achieving such a benefit is still challenging in the heterogeneous fog systems in which long task queues of powerful fogs can contribute to an average long delay of task execution. To handle the conflict of resource request for task processing this paper proposes a distributed fog resource allocation algorithms, namely MaxRU (Maximum Resource Allocation). Through the simulation analysis, MaxRU show potential advantages in reducing the average delay in the heterogeneous fog environment compared with the existing solutions.**

***Index Terms*—Fog-enabled IoT Systems, Fog Computing, Task Offloading, Resource Allocation, Workflow.**

I. Introduction

The Internet of Things (IoT) paradigm has been widely adopted in practical applications such as smart cities [1], smart grids [2] since it enables the interconnection and interoperability of IoT-enabled physical and virtual entities to create smart services and informed decision makings for monitoring, control, and management purposes [3]. Currently, the mutual benefits gained from the combination of fog and cloud enable the resulting IoT-fog-cloud systems to provide uninterrupted IoT services with various QoS requirements for the end users along the things-to-cloud continuum [4]. However, employing the fog computing raises another concern regarding decisions whether the tasks should be processed in the fog or in the cloud. There are many factors impacting on the offloading decision policies such as offloading criteria, application scenarios [5]. Basically, in the most of existing offloading techniques the tasks are probably offloaded to the best surrogate nodes (i.e., offloadees), which have the most ample resources (e.g., large storage capacity, high speed processing) and reliable communication network conditions in terms of delay, bandwidth between them and their neighbors, the IoT devices, and even the cloud servers. However, such the fog offloading solutions face significant challenges regarding the workload distribution among the complicated heterogeneous fog devices characterized by different computation resources and capabilities. The challenge is further amplified by increasing the rates of service requests, which probably make the task queues of resource-rich fog nodes longer. As a result, the requirements of latency-sensitive applications can be violated because of excessive waiting time of long queue. Furthermore, reliance of the remote cloud servers to fulfill the tasks may not help in improving the situation due to high communication delay or networking related disturbance.

These issues exposed from the above use cases do urge the need to develop an adaptive offloading mechanism that is based on the fog resource awareness to make the best context-aware offloading decisions. The work in [6] proposed an offloading policy for fog nodes to minimize the averaged delay for providing the IoT services to the IoT nodes in the IoT-fog-cloud application scenarios. The policy to offload the tasks in the fog layer or forward them to the cloud is decided based on an intensive analytical delay model, which takes into account the IoT-to-cloud and fog-to-cloud communication delay. In particular, the fog-to-fog communication is exploited to reduce the service delay through an efficient task sharing mechanism, which accounts for not only the queue length status of fog nodes but also the types of IoT service requests (i.e., light and heavy tasks). In conclusion, the service provisioning tasks are dynamically assigned to be processed in the fog landscape or in the cloud by an optimized manner. The various numerical results associated in the proposed framework show the role of fog layer in reducing the service delay as expected. Closely related to the work [6] is FOGPLAN framework as introduced in [7] for QoS-aware dynamic service provisioning. In this scheme, to ensure the QoS requirement of delay-sensitive IoT applications the fog service controllers continuously monitor their task execution performance and the service request rate to dynamically decide whether to deploy a new request into or to release a queued request from the queues. This key mechanism enables the fog node to remove the queued tasks, which no longer violate the prescribed delay requirement. As a result, the percentage of services satisfying the QoS constraints is increased significantly while reducing the overall cost. However, the resources of fogs are not made full use of in these works since a majority of heavy tasks is likely processed by the cloud servers.

In this paper, we examine the performance of fog layer to process all the requesting tasks. In addition, we proposed a MaxRU (Maximum Resource Allocation) algorithm to handle the unbalanced load among the heterogeneous fog environment as well as improving the service provisioning delay.

II. System Model and Problem Formulation

A. System Model

The work considers an IoT-Fog-Cloud system as illustrated in Fig. 1, which basically comprises of three layers, namely

Hoa Tran-Dang and Dong-Seong Kim are with department of IT Convergence Engineering, Kumoh National Institute of Technology, Korea, e-mail: {hoa.tran-dang, dskim}@kumoh.ac.kr. Corresponding Author: Dong-Seong Kim

Corresponding Author: Dong-Seong Kim, e-mail: dskim@kumoh.ac.kr.

IoT, Fog, and Cloud layer to provide IoT services.

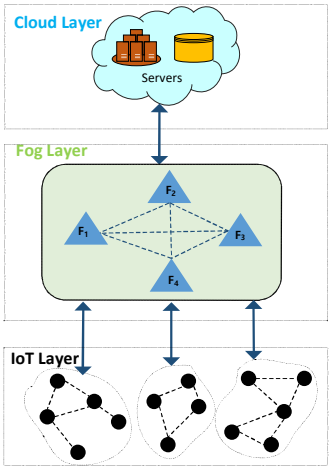

Fig. 1: A typical three-tier architecture of IoT-Fog-Cloud system for providing specific kinds of IoT services.

For readability, important terminologies used in this paper are defined and clarified as following definitions:

Definition 1: An IoT service A requested by IoT nodes is modeled as a tuple $A = (a, R_a, f(a))$, where a is input data size needed to be processed, R_a represents required resources for processing a, and $f(a)$ is the output data size of processing.

Definition 2: The service provisioning delay D_A is defined as the time interval elapsed from when a fog receives the service request A (i.e., the input data a) until the requesting IoT node receives the response (i.e., the output of input data processing $f(a)$.

In our IoT-Fog-Cloud system, the IoT nodes primarily submit the service requests to their closest connected fogs.

Definition 3: A fog node is called *primary host* of a service request A if it receives the service request A primarily.

Definition 4: A fog node is called *host* (or *service host*) of a service request A if it eventually takes charge of processing a, and then sending the computation result $f(a)$ to the IoT node.

B. Workflow Model of IoT Service Provisioning

Provisioning an IoT service to a requesting IoT device is modeled as a workflow, which includes a set $\mathcal{T}_A$ of m tasks, $\mathcal{T}_A = \{t_1, ..., t_m\}$. Basically, there are two types of tasks in the set: data communication and data processing tasks. The first type refers to tasks for sending and receiving the data through communication channels, while the second type is for executing the data by dedicated algorithms and software. Fig. 2 illustrates such an overall workflow, in which each task is sequentially executed, with output data as the input of its subsequent task. To describe the parametric context of each task, we define a_i and $f(a_i)$ as the input and output data size of the task t_i, respectively. In addition, $\gamma_{t_i} = f(a_i)/a_i$ represents the input-output data ratio of task t_i, which is dependent on the types of tasks and the associated task execution methods [8]. Notably, $\gamma_{t_i} = 1$ for the data communication tasks.

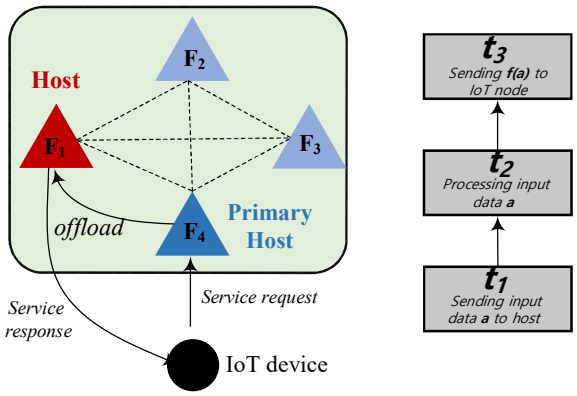

Fig. 2: A general workflow for service provisioning processes includes possibly three tasks, which are executed sequentially.

From the above stated perspective, each primary host is based on the available resources and workload state of its neighborhood to select the efficient service host. Each fog maintains it own neighbor resource table containing the updated information about the available resources. These tables are updated and shared periodically among the neighboring nodes to support the primary host to make offloading decisions. Table I shows an example of neighbor resource table stored by the fog node F_1, which records the resource states of neighbors with respect to residual memory (M_r), clock frequency, round-trip time (RTT), and waiting time in queue (W).

Node ID	Fog specification & Resource Status			
	M_r (MB)	Frequency (GHz)	RTT (ms)	W (ms)
F_2	200	10	2.5	350.2
F_3	100	5	3.1	500
F_4	400	2.5	4.8	239.1

TABLE I: Resource table of neighbors of fog node F_1

C. Problem Formulation

Fig. 3 illustrates the concern problem in IoT-Fog-Cloud systems, in which the primary host F_2 cannot process the input data of service request A due to lack of resources. Meanwhile, offloading the task to the fog neighbors F_1 and F_3

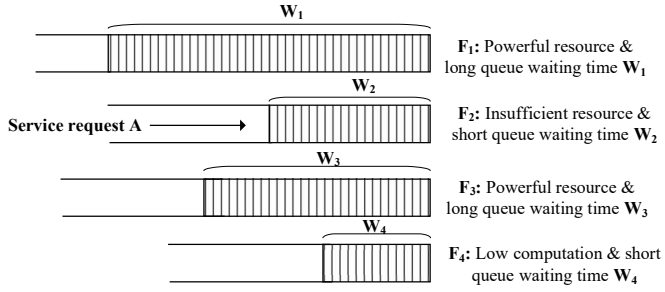

Fig. 3: The heterogeneity and unbalanced workload of fog environment expose issues in offloading tasks.

may lead to extensive delay since there are high workloads in queues of these fog nodes. In addition, F_4 can offload the task but resulting long delay due to low computational capability. Such issue urges a need to design an algorithm for efficiently allocating which fogs process which tasks in order to achieve the minimized average delay.

III. Mathematical Formulation of Task Execution Minimization

Generally, the problem is turned into resource and communication scheduling, which simultaneously maps appropriate fog resources to process corresponding tasks and then establish an order to send the processing results to the IoT node. Based on the workflow, we aims at finding a schedule to execute the workflow on fog and cloud computing resources so that the makespan of schedule (i.e., service provisioning delay) is minimized. We define a schedule for each workflow of provisioning the service A as a tuple $S_A = (\mathcal{T}_A, \mathcal{C}_A, M_A, D_A)$, which include a set $\mathcal{T}_A$ of tasks in the workflow, a set $\mathcal{C}_A$ of fog resources, a task-to-resource mapping M_A, and the total execution time of schedule (i.e., total service provisioning delay) D_A.

In each schedule, $\mathcal{C}_A = \{F_1, F_2, ..., F_n\}$ includes the neighbor fogs of primary host, which need to be assigned for executing the tasks. M_A represents a task-resource mapping and is comprised of tuples of the form $m_{t_i}^{r_j} = (t_i, r_j, ST_{t_i}, ET_{t_i})$, one for each task in $\mathcal{T}_A$ of workflow. A mapping tuple $m_{t_i}^{r_j}$ implies that task t_i is scheduled to be processed by resource r_j and is expected to start executing a time ST_{t_i} and complete by time ET_{t_i}, where $t_i \in \mathcal{T}_A$, and $r_j \in \mathcal{C}_A$.

The objective of the optimization is to find an optimal workflow that minimizes the service provisioning delay, that is modeled as follow:

$$\begin{aligned} \textbf{P:} \ \min \quad & D_A \\ \text{s. t.} \quad & Constraints. \end{aligned} \tag{1}$$

1) Objective Function: The objective function is to minimize the service provisioning delay $D(A)$, which is calculated as follow:

$$D_A = ST_m + \sum_{j \in \mathcal{C}_A^+} \alpha_{mj} T_{mj}^{proc} - T_{cur}, \tag{2}$$

where ST_m is the start execution time of final task t_m of workflow, and α_{mj}=1 if the task t_m is processed by the resource r_j ($r_j \in \mathcal{C}_A^+$), T_{cur} is current time captured by the primary host as it receives the request A, and T_{mj}^{proc} is time to process the task t_m. Since t_m is the final task of workflow, T_{mj}^{proc} is time to transfer the result $f(a)$ from the host r_j (i.e., a fog or cloud) to the requesting IoT node. We assume that there are l hops to connect the host r_j to the IoT nodes, thus, T_{mj}^{proc} is derived by:

$$T_{mj}^{proc} = \sum_{l} \frac{f(a)}{R_l} + T_{mj}^{prop}, \tag{3}$$

where $f(a)$ is the output data size in bits, and R_l is the data rate of l'th link on the path to transmit $f(a)$ from r_j to the IoT node, and T_{mj}^{prop} is propagation time from a node m to a node j, which can calculated from RTT (see Section VI).

2) Constraints:

- To execute a task, a resource must have an available sufficient resource to process the input data of task. Since all requesting services may have different requirements on kind of resource features (e.g., available storage, CPU processing capability, and/or their combinations) to process the input data, we use the operator $\succeq$ to indicate the sufficiency of resource. Thus, the constraints are modeled as follow:

$$\sum_{j \in \mathcal{C}_A} \alpha_{ij} \mathbf{R}_j^{avai} \succeq \mathbf{R}_{t_i}^{req}, \forall t_i \in \mathcal{T}_A, \tag{4}$$

  where $\mathbf{R}_j^{avai}$ is the current available state of resource r_j, and $\mathbf{R}_{t_i}^{req}$ is required resource to execute the task t_i.
- Each task is processed only by a single resource, thus that implies the following constraint.

$$\sum_{j \in \mathcal{C}_A} \alpha_{ij} = 1, \forall t_i \in \mathcal{T}_A. \tag{5}$$

- In respecting to the queued tasks, all start times of tasks must be greater than the current waiting times of queues of resources, which process the tasks:

$$ST_{t_i} - T_{cur} \geq \sum_{j \in \mathcal{C}_A} \alpha_{ij} W_j, \forall t_i \in \mathcal{T}_A. \tag{6}$$

- Regarding the order of tasks, as a task t_i is finished completely before a task t_j is executed, their start times must satisfy the following condition:

$$\beta_{ij}(ST_i + \sum_{k \in \mathcal{C}_A} \alpha_{ik} T_{ik}^{proc}) \leq ST_j, \forall \{t_i, t_j\} \in \mathcal{T}_A. \tag{7}$$

  where β_{ij} is defined as:

$$\beta_{ij} = \begin{cases} 1, & \text{if } \mathcal{O}_{t_j} > \mathcal{O}_{t_i}, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

  Notably, the processing orders of tasks t_i (i.e., $\mathcal{O}_{t_i}$) are dependent on the their data dependence and can be obtained based on the constructed workflow. Furthermore, the processing time T_{ik}^{proc} for the communication tasks can be adopted from Equation (3). Meanwhile, for the data processing tasks the time required for a resource r_i to process a_k-bit data of task t_k is calculated as follow:

$$T_{ik}^{proc} = \frac{a_k \gamma_{ik}}{f_i}, \tag{9}$$

  where f_i is the computation capability of r_i, i.e., the CPU frequency (in CPU cycles per second), and γ_{ik} is the processing density of r_i to process the data of task t_k (in CPU cycles per a data bit).
- As a pair of tasks t_i, t_j involves sending and receiving a set of data, they are executed in parallel by the transmitter and the receiver, thus:

$$\sigma_{ij}(ST_i - ST_j) = 0, \forall \{t_i, t_j\} \in \mathcal{T}_A, \tag{10}$$

  where σ_{ij} is defined and retrieved from the built workflow as follow:

$$\sigma_{ij} = \begin{cases} 1, & \text{if } \mathcal{O}_{t_j} = \mathcal{O}_{t_i}, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

- Multi-task single resource constraint: Unlike the general resource provisioning scheduling algorithms for scientific workflows on the cloud with multi-core servers as assumed in [9], the resource allocation for offloading mechanisms in the heterogeneous fog environment faces additional concerns regarding constraints on concurrent usage of resource and time. In this work, we suppose that no multiple tasks are executed in parallel by a single resource. In other hand, as a task t_i and a task t_j are assigned to be executed by a resource r_k, their start times must respect to following condition:

$$(\alpha_{ik}ST_i - \alpha_{jk}ET_j)(\alpha_{jk}ST_j - \alpha_{jk}ET_i) \leq 0, \quad \forall\{t_i, t_j\} \in \mathcal{T}_A \ \& \ r_k \in \mathcal{C}_A. \tag{12}$$

A. Solution Deployment Analysis

In this work, we employed particle-swarm optimization (PSO) approach to resolve the problem **P** [10]. The set of solutions include an optimal one to offer the minimal service delay and other feasible sub-optimal ones. Practically, the optimal solution may not be deployed due to unavailability of expected task-resource mapping, which in turn is basically caused by unavailable resource of involved fogs, or unreliable communication dropping the resource allocation requests. To cope with this situation, we utilize an additional feasible solution (i.e., sub-optimal solution) in the set as a backup solution. Suppose that a fog obtains a set $\mathcal{S}_A = \{S_A^*, S_A^1, ..., S_A^p\}$ of feasible solutions for offloading a task A, which includes the optimal solution S_A^* and a number p of sub-optimal solutions. We denote D_A^k ($k = \overline{1...p}$) as the expected service delay as the corresponding solution S_A^k is implemented. The feasible solutions are selected such that their corresponding delay are less different, or $|D_A^* - D_A^k|/D_A^* \leq \gamma_{th}\%$, where γ_{th} is acceptable threshold of difference between delays obtained by any feasible solution and the optimal solution. To facilitate the selection of solution, the feasible solutions in the list are sorted in a descending order according to the expected delay.

IV. Maximal Resource Utilization based Allocation in Fog Environment

A fog node can receive multiple resource allocation requests simultaneously for executing different tasks, some of which may expose an overlap of expected resource usage. As illustrated in Fig. 4, five task execution requests t_1, t_2, t_3, t_4, and t_5 cannot be served altogether by a fog node F_3 due to their mutual conflicts of some requesting resources (e.g., resource overlap between t_3 and t_5). Accordingly, the request t_2 totally is rejected since the resource is already allocated for processing t_1. Meanwhile, the three remain tasks t_3, t_4, and t_5 must compete for usage of resource.

Due to lack of global information, the fog nodes make decisions on which task requests are accepted to process in a distributed manner. In a simple way, the fog nodes can employ the first-come first-serve (FCFS) mechanism to handle the requests. However, it probably leads to degrade the overall system performance in terms of delay in the cases that the requests for implementing the optimal solutions are denied. In addition, deploying the backup solutions or the cloud-based solutions in the worst cases can attribute to an increased delay averagely.

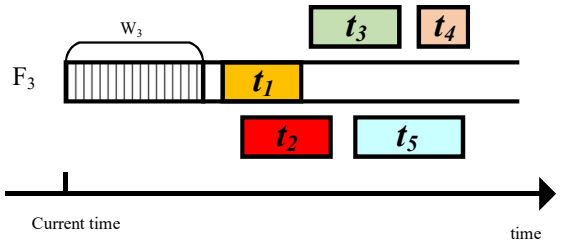

Fig. 4: Multiple requests for executing tasks expose possible conflicts of resource usage in fogs.

In MaxRU scheme, some of received requests are accepted to be served by a certain fog so as the resource utilization of the fog is maximized. We denote T_i^r as a feasible allocation solution, which is a set of p_i possible maximum requesting tasks ($0 \leq p_i \leq p$) such that their expected resources are non-overlapping, $T_i^r = \{t_1^{r'}, t_2^{r'}, ..., t_{p_i}^{r'}\}$. Notably, $t_j^{r'}$ is selected from the set T^r, $\forall j = 1...p_i$, and the tasks in T_i^r is sorted in an ascending order of start time (ST). As the allocation solution T_i^r is deployed in a certain fog, its resource utilization RU_i is defined and calculated by:

$$RU_i = \frac{\sum_j^{p_i}(ET_j - ST_j)}{ET_{p_i} - ST_1}, \tag{13}$$

where ST_j and ET_j are start time and end time of execution of task $t_j^{r'}$, which are obtained from the optimization problem solutions.

Suppose that there exists a maximum $\overline{p}$ allocation solutions (i.e., $T_1^r, ..., T_{\overline{p}}^r$), thus the maximum resource utilization can be derived by $MaxRU = max\{RU_k\}, \forall k \in \{1, \overline{p}\}$. Consequently, in MaxRU mechanism the allocation solution which offers the maximal resource utilization is selected to be deployed.

Algorithm 1 summarizes key procedures to implement MaxRU mechanism.

V. Simulation and Performance Evaluation

A. Simulation Environment Setup

1) Characteristics of IoT Services: In this work, the input data of all the computing service requests is characterized by sizes, which are uniformly distributed as U[1-5] MB. We vary the service request rate of IoT nodes according to the list $\mu = \{0.01, 0.02, 0.03, 0.04, 0.05\}$ (requests per second) to investigate the performance of systems in the cases of increasing workload. The input-output ratio of data execution tasks is set to be 0.1 for all the types of data.

2) System Configuration: The fog layer is composed of 10 heterogeneous fog nodes, which are characterized different configurations. The heterogeneity of fog nodes are featured by computation capability (i.e., CPU processing density) γ and CPU frequency f, which are assumed to be uniformly distributed in U[200, 2000] (cycles/bit) and U[1, 15GHz] (U[10^9, 15×10^9](cycles/s)), respectively [11]. The RAM and

Algorithm 1: MaxRU: Maximal Resource Utilization based Allocation

Input: $T^r = \{t_1^r, t_2^r, ..., t_p^r\}$, $T^a = \{t_1^a, ..., t_q^a\}$
// Sets of p requesting tasks and q allocated tasks
Output: $Res = \{t_1^r.res, t_2^r.res, ..., t_p^r.res\}$ // Response to requests, $t_1^r.res = 1$: accepted; $t_1^r.res = 0$: rejected

begin
  // Rejecting tasks which violate the allocated tasks
  for $t_i^r \in T^r$ **do**
    if $RC\_OVERLAP(t_i^r, T^a) = 0$ **then**
      $t_i^r.res \leftarrow 0$ // Rejecting t_i^r
      $REMOVE(t_i^r, T^r)$ // Update T^r
  // Accepting tasks that their requesting resources do not overlap with that of the other tasks
  for $t_i^r \in T^r$ **do**
    if $RC\_OVERLAP(t_i^r, T^r \backslash \{t_i^r\}) = 1$ **then**
      $t_i^r.res \leftarrow 1$ // Accepting t_i^r
      $REMOVE(t_i^r, T^r)$ // Update T^r
  // Find $\overline{p}$ feasible allocation solutions
  $\mathcal{T}_{sol} = \{T_1^r, ..., T_{\overline{p}}^r\} = ALLOC\_SOL(T^r)$
  // Sort tasks in an ascending order of ST
  for $T_i^r \in \mathcal{T}_{sol}$ **do**
    $T_i^r \leftarrow SORT_{ST}(T_i^r)$; // $t_i^r.ST \leq t_{i+1}^r.ST$ $\forall t_i^r \in T_s^r$
    Calculate RU_i as Equation 13
  if $RU_k = MaxRU$ **then**
    T_k^r is selected to be deployed
    for $t_j^r \in T_k^r$ **do**
      $t_j^r.res \leftarrow 1$

storage capacity are two another factors characterizing the heterogeneity of fog environment, which are randomly distributed in {1GB, 512 MB, 256 MB} and U[1 GB,6 GB], respectively [4].

3) Communication Delay and Channel Capacity: The propagation delay can be estimated by having the round-trip time (RTT), which itself can be expressed as RTT(ms)=0.03× distance (km)+5 [6], [12]. Thus, this work assumes that the propagation delay between the IoT nodes and the fog nodes, among fog nodes, and between fog nodes and the cloud servers are uniformly distributed between U[1, 2] (ms), U[0.5, 1.2] (ms), and U[15, 35] (ms) [6], respectively. Furthermore, the data rate between the IoT layer and fog layer is 54 Mb/s according to IEEE 802.11 a/g (i.e., WiFi standard) [4].

B. Comparative Approaches

To evaluate the performance of TPRA algorithm in terms of average service provisioning delay, we use AFP and TPRA algorithms [13] for the comparative study. In AFP mode the tasks always are processed by the best neighbors or cloud if the request time overcomes the fog offloading limit ($e_{\mathcal{M}}$). Meanwhile, TPRA [13] is based on the task priority with expected delay consideration. We examine the performance of algorithms as the service request μ and λ are varied.

C. Evaluation and Analysis

This section presents and analyzes the performance of systems with respect to the service provisioning delay as different offloading strategies are employed in different simulation scenarios.

Fig. 5 depicts the average service delay achieved by four comparative algorithms TPRA, MaxRU, and AFP as the service request rate μ and heterogeneity level (λ) of fog environment are varied.

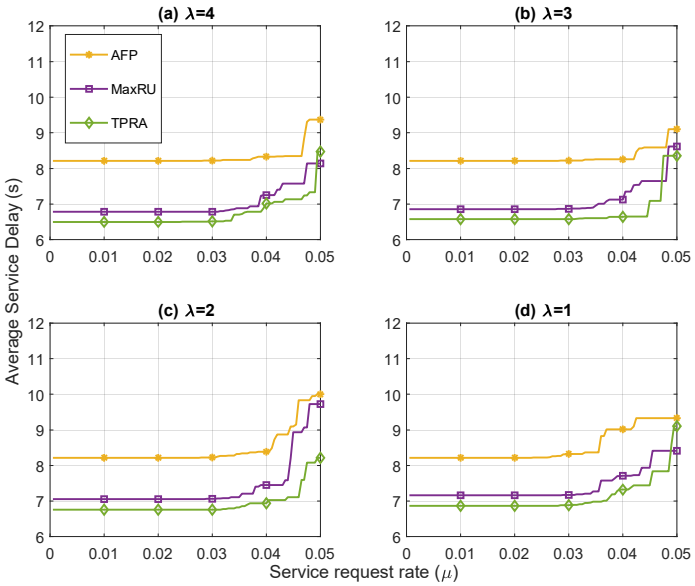

Fig. 5: The average service delay offered by the comparative schemes as a function of λ and μ

As expected, the cloud-assisted solution AFP totally incurs the longest delay, which is resulted in from the dominance of delays of communication tasks since the data is transmitted over long distances from the fog layer to the remote cloud server. In parallel, the other fog-based offloading solutions (i.e., TPRA, MaxRU) achieve a lower delay regardless the variations of μ and λ. Such the result completely confirms the essential role of fog computing platforms in improving the system performance in terms of service delay as recognized in the existing literature. In particular, since the rate of service requests directly impacts on the queuing status of fog nodes, the performance gain obtained by the fog-supported computing systems is get larger in cases of lower μ.

In addition, as μ is higher more workloads are computed by the cloud in the AFP scheme since the neighbor fog nodes probably have long queues of waiting tasks. Furthermore, the scarcity of powerful computing fogs leads to a significant increase in the amount of tasks forwarded to the cloud server in the AFP policy. Therefore, as displayed in Figs. 5 the performance gap between TPRA, MaxRU and AFP is higher

when μ increases. Whereas, the proposed offloading mechanism TPRA enables more services hosted and processed by the fog landscape through associated efficient load balancing among the fogs. As a result, the average service delay offered by our proposed solutions is lower than that of AFP regardless of μ.

Fig. 6 supports such the claim through the percentage of workload (P_{wl}^{F}), which is processed by the fog layer. As shown in this figure, as the rate of request is low the task division mechanism is not significantly beneficial. However, MaxRU based offloading solutions enable the fogs to serve up to 75% of IoT service requests at the highest request rate ($\mu = 0.05$), which implies approximately only 25% of requests needed to be processed by the cloud. Meanwhile, the comparative scheme (AFP) can handle a smaller percentage of requests (roughly 55%) in the fog layer and thus up to 45% requests are forwarded to the cloud.

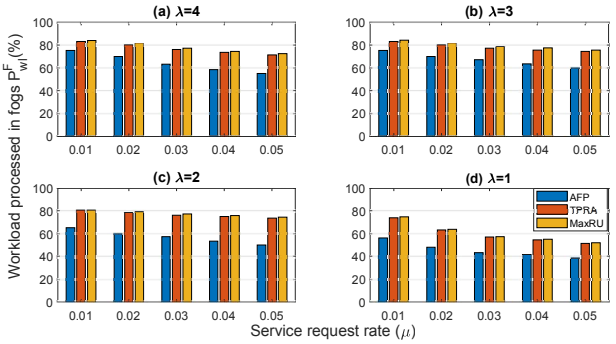

Fig. 6: Percentage of workload processed in the fog landscape as a function of λ and μ.

VI. Conclusions

Offloading the tasks to the best neighbor fogs may contribute to the excessive delay due to the long queuing tasks of powerful nodes. In addition, the workload distribution is unbalanced among the heterogeneous fog environment, which results in the underutilized resources. This paper introduced TPRA, a resource allocation algorithms for assigning the computing tasks efficiently to the fog resource. Each IoT service provisioning is modeled as a workflow, which specifies the schedule of processing involved task. In addition, TPRA handles the conflict of requesting resource based on the priority of tasks, that is achieved by the workflow optimization. The primary simulation results show the advantage of proposed algorithms in reducing delay and balancing the workload in the heterogeneous fog environment.

Acknowledgment

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2020-2020-0-01612) and supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation), Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2018R1A6A1A03024003), and Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2020R1I1A1A01073019).

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[2] Y. Saleem, N. Crespi, M. H. Rehmani, and R. Copeland, "Internet of things-aided smart grid: Technologies, architectures, applications, prototypes, and future research directions," *IEEE Access*, vol. 7, pp. 62 962–63 003, 2019.

[3] H. Tran-Dang and D. Kim, "An information framework for internet of things services in physical internet," *IEEE Access*, vol. 6, pp. 43 967–43 977, 2018.

[4] H. Tran-Dang and D.-S. Kim, "FRATO: Fog resource based adaptive task offloading for delay-minimizing IoT service provisioning," vol. 32, no. 10, pp. 2491–2508.

[5] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, Oct. 2018.

[6] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, April 2018.

[7] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "Fogplan: A lightweight qos-aware dynamic fog service provisioning framework," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080–5096, June 2019.

[8] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "Dats: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3423–3436, April 2019.

[9] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, April 2014.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[11] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. Zhou, "Femto: Fair and energy-minimized task offloading for fog-enabled iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4388–4400, June 2019.

[12] A. Qureshi, "Power-demand routing in massive geo-distributed systems," Ph.D. dissertation, USA, 2010.

[13] H. Tran-Dang and D.-S. Kim, "Task priority-based resource allocation algorithm for task offloading in fog-enabled IoT systems," in *2021 International Conference on Information Networking (ICOIN)*. IEEE.