A Many-to-One Matching based Task Offloading (MATO) Scheme for Fog computing-enabled IoT Systems

Hoa Tran-Dang
Dept. of IT convergence engineering
Kumoh National Institute of Technology
Gumi-si, South of Korea
{hoa.tran-dang}@kumoh.ac.kr

Dong-Seong Kim
Dept. of IT convergence engineering
Kumoh National Institute of Technology
Gumi-si, South of Korea
dskim@kumoh.ac.kr

2022 International Conference on Advanced Technologies for Communications (ATC) | 978-1-6654-5188-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ATC55345.2022.9942992

***Abstract*—Fog computing networks have been widely integrated in IoT-based systems to improve the quality of services (QoS) such as low response service delay through efficient offloading algorithms. However, designing an efficient offloading solution is still facing many challenges including the complicated heterogeneity of fog computing devices and complex computation tasks. In addition, the need for a scalable and distributed algorithm with low computational complexity can be unachievable by global optimization approaches with centralized information management in the dense fog networks. In these regards, this paper proposes a distributed computation offloading framework (MATO) for offloading the splittable tasks using matching theory. Through the extensive simulation analysis, the proposed approaches show potential advantages in reducing the average delay significantly in the systems compared to some related works.**

***Index Terms*—Fog computing, parallel computation, task offloading, task partition, distributed algorithm, stable matching.**

I. Introduction

Practically, the Internet of Things (IoT) has become an integral element for realizing smart practical systems such as smart cities [1], smart factories [2], smart logistics, and supply chain [3], [4]. The fundamental aspect of IoT-based systems is to connect all devices through the Internet protocol to exchange high volume data and process them to create smart services and applications. Owing to limited computation resources, network, storage, and energy, IoT devices are inadequate for executing all computational tasks, especially tasks with huge volumes and complex data structures. Cloud computing is an essential solution to this problem because it provides powerful resources to fulfill tasks efficiently [5], [6]. However, cloud computing-based solutions do not always meet the expected quality of service (QoS) and quality of experience (QoE) requirements for some classes of IoT applications, especially latency-sensitive ones because of the long physical distance between the IoT devices and the remote cloud servers, scarce spectrum resources, and intermittent network connectivity.

This context has led to an integration of fog computing in middle between the IoT and cloud layer to process and offload most tasks on behalf of the cloud servers, thereby allowing for the QoS and QoE requirements to be met [7], [8]. Besides providing the cloud like services to EUs, the fog computing potentially improves the performance of fog-based systems such as reduction of service delay [9], and energy saving [10].

However, to realize these benefits of fog computing paradigm, there requires efficient resource allocation strategies to perform task offloading operations [11]. Indeed, there are many factors that challenge the design and development of effective offloading strategies such as the heterogeneity of computing devices, various types of computational tasks with different requirements [12] such as many applications related to artificial intelligence (AI), machine learning (ML), and augmented reality (AR). There are a large number of centralized optimization techniques and algorithms proposed in the literature to provide optimal solutions to the aforementioned resource allocation problems [13]. However, these solutions require a centralized control to gather the global system information, thus incurring a significant overhead and computation complexity of algorithms. This complexity is further amplified by the rapidly increase of density and heterogeneity of FCNs [14] when dealing with combination integer programming problems [15].

The aforementioned limitations of optimization have lead to a second class of game theory based offloading solutions that can avoid the cost-intensive centralized resource management as well as substantially reduce the complexity of algorithms such as POMT [16] and POST [17]. Despite their potentials, these approaches pose several limitations. First, classical game theoretical algorithms such as best response require some information regarding actions of other players [18]. Correspondingly, many assumptions are introduced in the game theory based algorithms to simplify the system models that, in some case, are impractical. Second, most game-theoretic solutions, for example, Nash equilibrium, investigate one-sided stability notions in which equilibrium deviations are evaluated unilaterally per player. In addition, the stability must be concerned by both sides of players (i.e., resource providers and resource requesters) in the context of fog computing

environment.

Ultimately, managing resource allocation effectively in such a complex environment of IFC systems leads to a fundamental shift from the traditional centralized mechanism toward distributed approaches. Recently, matching theory has emerged as a promising technique for solving resource allocation problems in many applications such as wireless communication networks [19] because it can alleviate the shortcomings of game theory and optimization-based approaches. Many matching-based algorithms also are proposed in the fog environment to reduce the latency such as METO [20], and DATS [21]. Alternatively, while the optimization and game theory based solutions are efficient in a some limited scenarios, the matching-based approaches have potential advantages owing to the distributed and low computational complexity algorithm. However, to reap the benefits of matching theory for task offloading and resource allocation in the fog computing environment, there is a need of advanced frameworks to handle their intrinsic properties such as heterogeneity of fog computing devices and the complex structure of computation tasks. In these regards, this paper proposes a distributed computation offloading algorithm to minimize the overall task execution delay in the fog-enabled IoT systems.

II. Many-to-One (M2O) Matching Model

Although the most of matching models in the literature deal with two sets of agents, there are other matching models involving only one set (i.e., stable roommate problem [22]), and three sets of agents [23]. In the scope of paper, we consider a M2O matching model of two distinct sets of agents, in which each agent of one side can be matched with multiple agents of the other side but the reverse is not valid. We denote these two sets as $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ and $\mathcal{Y} = \{y_1, y_2, ..., y_k\}$. Fig. 1 shows an example of M2O matching.

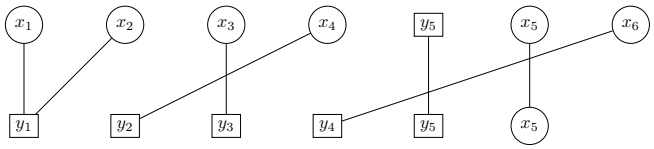


Fig. 1: An example of a M2O matching established between two sets $\mathcal{X}$ and $\mathcal{Y}$ in which $\mathcal{M}(y_1) = \{x_1, x_2\}$, $\mathcal{M}(y_2) = \{x_4\}$, $\mathcal{M}(y_3) = \{x_3\}$, $\mathcal{M}(y_4) = \{x_6\}$, x_5 and y_5 are unmatched agents (i.e., $\mathcal{M}(y_5) = \{y_5\}$, $\mathcal{M}(x_5) = \{x_5\}$).

Generally, a M2O matching can be defined as follow:

Definition 2.1: The outcome of M2O matching model is a matching $\mathcal{M}$: $\mathcal{X} \cup \mathcal{Y} \mapsto \mathcal{X} \cup \mathcal{Y}$ such that the three following constraints are satisfied:

- $|\mathcal{M}(x)| = 1$ for every agent $x \in \mathcal{X}$ and $\mathcal{M}(x) = x$ if x is unmatched,
- $|\mathcal{M}(y)| = q_y$ for every agent $y \in \mathcal{Y}$; if the number of agents in $\mathcal{M}(y)$ is k and $k < q_y$, then $\mathcal{M}(y)$ will have $q_y - k$ copies of y,
- $\mathcal{M}(y) = x$ if and only if x is an element of $\mathcal{M}(y)$.

Recall that each agent y has a positive quota q_y to represent the maximum number of agents in the set $\mathcal{X}$ it can be matched with. With this definition, $\mathcal{M}(x) = y$ means that agent x is matched with agent y, and $\mathcal{M}(y) = \{x_1, x_2, y, y\}$ indicates that the agent y with the quota $q_y = 4$ has been matched with two agents x_1 and x_2 and has two unfilled matching positions.

The objective of matching games is to achieve stability instead of optimality, at which there is no incentive incurred to devise the current matched pairs of agents. A stable matching is defied in the following definition 2.2.

Definition 2.2: A matching $\mathcal{M}$ is pairwise stable if there is no blocking pair (x, y).

Alternatively, a matching is to stably match agents in two sets based on their individual, objectives, and exchanged information. Each agent builds a ranking of other agents in the other side individually using a preference relation, and then creating its own preference list (PL). Basically, a preference can calculated based on an objective utility function that quantifies the QoS achieved by a certain matching between two agents of two sides. Defining $\mathcal{P}(x)$ and $\mathcal{P}(y)$ as preference lists of agent $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, respectively. Basically, $\mathcal{P}(x) = \{y_1, y_2, x, y_4, y_5, ...\}$ illustrates that agent x prefers y_1 to y_2 (denoted as $y_1 \succ_x y_2$), and prefers keeping the position unfilled over other agents like y_4 and y_5. A blocking pair is defined as follow:

Definition 2.3: (x, y) is a blocking pair for a matching $\mathcal{M}$ if three following conditions are satisfied:

- $\mathcal{M}(x) \neq y$,
- $y \succ_x \mathcal{M}(x)$,
- $x \succ_y \mathcal{M}(y)$.

Based on the created PLs, agents can implement distributed algorithms called deferred acceptance algorithm (DAA) to derive the matching outcome. Deferred acceptance (DA) algorithm known as the Gale-Shapley algorithm [24] is the classical algorithm to find the stable matching in marriage problem, which can be used to find the matching in M2O matching model. The DA algorithm is an iterative procedure, in which each players in one set make proposals to the other set, whose players, in turn, decide to accept or reject these proposals, respecting their quota and PLs. Basically, an acceptance decision is made when the proposing agent has a higher rank order of PL than the current matched agent. With this procedure, the algorithm admits many distributed implementations, which do not require the players to known each other's preferences. In addition, the DA algorithm ensures to achieve the stable matching outcome in a polynomial time.

III. System Model

A. Fog Computing Network

This paper considers a FCN with flat architecture (i.e., no centralized fog controller) as illustrated in Fig. 2, which includes a set $\mathcal{F}$ of N FNs ($\mathcal{F} = \{F_1, ..., F_N\}$) and a cloud data center C providing M virtual machines (VMs). FNs can connect together through wired and wireless links for sharing resources. A cloud server can be reached by any FN for

providing the cloud services such as computation, storage, and caching. The FNs operate in a distributed manner to process and offload tasks. At an arbitrary scheduling time slot, a FN can be in one of three modes: (i) a task node (TN) that has tasks in its queue, (ii) busy node (BN) that is busy processing tasks, and (iii) helper node (HN) that has available resource to offloading tasks. The state of nodes can change over scheduling times. For example, a BN at a time slot τ will become a HN in the next time slot $(\tau + 1)$ if there is no task resided in its queue. Notably, the cloud always serves as a HN because it has a set of virtual machines.

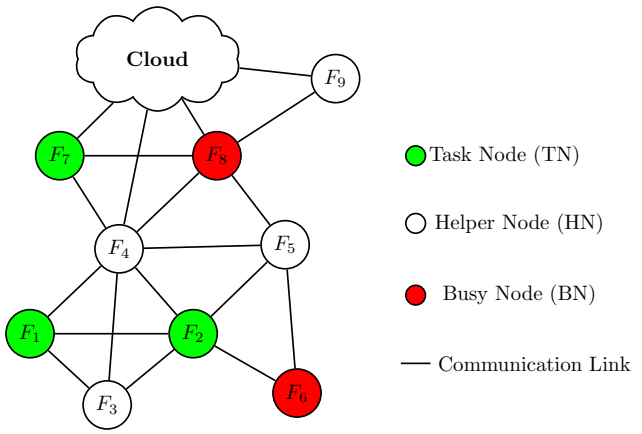


Fig. 2: An illustrative model of FCN includes three TNs (F_1, F_2, F_7), two BNs (F_6, F_8), and four HNs (F_3, F_4, F_5, F_9), which can connect to the cloud to request the cloud services.

To design an efficient offloading policy, TNs are based on a table of neighbor resources that contains the updated information about the available resources of neighbor HNs. We define $\mathcal{H}_i$ as the set of HNs neighboring to a TN F_i, thus $\mathcal{H}_i = \{F_p, F_k, ..., C\}$. We assume that the fog devices uses a common protocol to communicate and update their resource status periodically [25]. The computing capability of a FN is represented through CPU frequency (f (cycles/s)), and CPU processing density (γ (cycles/bit)).

B. Computation Offloading Model

Computation tasks from the end user devices such as IoT sensors, and smart phones arriving at the queues of FNs follows an exponential distribution with average rate λ tasks per second (tasks/s). Fig. 3 shows the overview of task offloading model in the FCN.

At a certain time slot τ, the FCN has a set $\mathcal{T} = \{t_1, t_2, ..., t_K\}$ including K computational tasks required to be processed. A single task t_i currently resided in the queue of TN F_i can be processed locally by F_i or fully offloaded by a HN F_k ($F_k \in \mathcal{H}_i$). In addition, a task t_i with a size a_i can be divided into $m = |\mathcal{H}_i| + 1$ subtasks, which are denoted as t_{ik} $(k = 0, ..., N)$. These substasks then can be processed in parallel by multiple HNs. As illustrated in

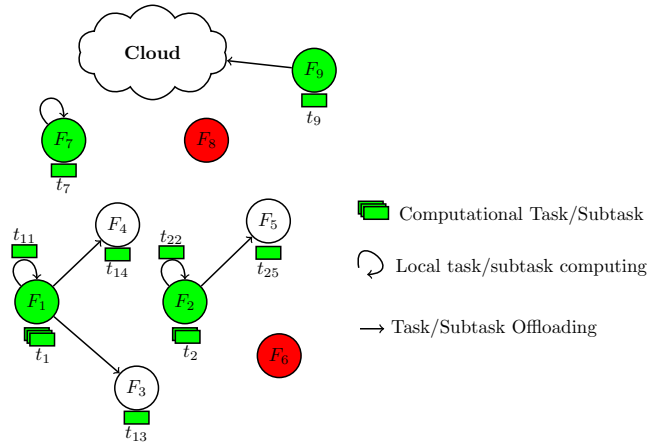


Fig. 3: The offloading model includes full and partial offloading.

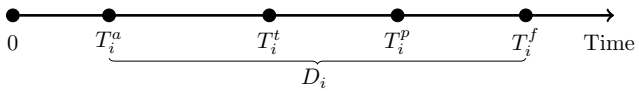
Fig. 3, two substasks t_{14} and t_{13} divided from t_1 can be computed simultaneously by F_4 and F_3 respectively while t_{10} is computed locally. We define a_{ik} as the data size of subtask t_{ik}, thus we have:

$$\sum_{k=0}^{N} a_{ik} = a_i, \forall t_i \in \mathcal{T}. \quad (1)$$

In addition, when $a_{ik} = 0$, the HN F_k is not assigned to offload t_i.

IV. Problem Formulation

The overall delay for executing a task t_i (D_i) is an interval from the arrival time of task (T_i^a) at the queue of F_i until finish processing time (T_i^f) at other fog node (F_i or some HNs). As illustrated in Fig. 4, $D_i = T_i^f - T_i^a$.



T_i^a : Arrival time of task t_i at a TN F_i
T_i^t : Transmission time of task t_i to an arbitray HN F_k
T_i^p : Processing time of task t_i at F_k
T_i^f : Finish time of task t_i at F_k

Fig. 4: Timeline to calculate the delay to execute an arbitrary task T_i.

When the partial offloading policies are applied for executing t_i, we define D_{ik} as the delay to process the subtask t_{ik}. Suppose that there is no data dependency between subtasks of t_i, therefore the subtasks can be computed in parallel by different HNs. In addition, no aggregation of subtask processing results is required, thus $D_i = \max\{D_{ik}\}$, $k = 1, ..., N$. Notably, $D_{ik} = T_{ik}^f - T_i^a$.

We define o_{jk}^i to represent the scheduling order between subtasks t_{ij} and t_{ik} for wireless transmission. Particularly, if

t_{ij} is scheduled on wireless link before t_{ik}, we have $o^i_{jk} = 1$, otherwise, $o^i_{kj} = 0$. Since t_{ij} is transmitted either before or after t_{ik}, we have

$$o^i_{ij} + o^i_{ik} = 1, \forall i, j = 0, ..., N \ \& \ i \neq j. \tag{2}$$

We define T^t_{ij} as the start time for transmitting t_{ij}. There are no overlap between their wireless transmission times of two subtasks t_{ij} and t_{ik}, therefore their start transmission times must satisfy:

$$T^t_{ij} - (T^t_{ik} + \frac{a_{ik}}{r_{ik}}) \geq -Lo^i_{jk}, \forall j, k \in \mathcal{H}_i, \tag{3}$$

where L is a large possible constant and r_{ik} is the data rate on the channel between F_i and F_k.

Additionally, the start processing time T^p_{ij} of t_{ij} must not be smaller than the arrival time at a HN F_j, therefore:

$$T^p_{ij} \geq T^t_{ij} + \frac{a_{ij}}{r_{ij}}, \forall j \in \mathcal{H}_i. \tag{4}$$

Finally, the end time T^f_{ij} to finish t_{ij} is calculated as follow:

$$T^f_{ij} = T^p_{ij} + \frac{a_{ij}\gamma_j}{f_j}, \forall j \in \mathcal{H}_i. \tag{5}$$

Recall that $D_{ij} = T^f_{ij} - T^a_i$, therefore $D_i = \max_j\{D_{ij} | j \in \mathcal{H}_i \cup \{i\}\} = \max_j\{T^f_{ij}\} - T^a_{ij}$. We denote $\mathbf{P}(t_i|\mathcal{H}_i)$ as a problem to minimize the execution delay of t_i given the available computation resource set $\mathcal{H}_i$. The optimization problem $\mathbf{P}$ is modeled as follow to minimize D_i:

$$\mathbf{P}(t_i|\mathcal{H}_i): \min_{\boldsymbol{\alpha}_i, \boldsymbol{T}^t_i, \boldsymbol{T}^p_i, \boldsymbol{o}_i} D_i \quad \text{s. t.} \quad (1), (2), (3), (4). \tag{6}$$

where $\boldsymbol{\alpha}_i$=$[a_{ii}, a_{ij}, a_{ik}, ...]$, $\boldsymbol{T}^t_i = [T^t_{ii}, T^t_{ij}, T^t_{ik}, ...]$, $\boldsymbol{T}^p_i = [T^p_{ii}, T^p_{ij}, T^p_{ik}, ...]$, and $\boldsymbol{O}_i = [o^i_{ij}, o^i_{jk}, ...]$, $\forall j, k \in \mathcal{H}_i$.

As each fog node selfishly aims to minimize its own task execution delay, the minimization of overall delay of all tasks is infeasible because of coupling resource problem. In addition, achieving the global optimization (i.e., minimizing the sum of total delay of all tasks) face a critical challenge regarding the computation complexity, especially in dense fog networks. In the next section, we propose a distributed algorithm based on matching theory for allowing the fog nodes to perform task offloading in a distributed manner with low computational complexity.

V. Proposed MATO Scheme

A. Overview

Based on the system model, we model the computational offloading problem in the FCNs as a M2O matching game between the set of tasks $\mathcal{T}$ and the set of HNs $\mathcal{H}$. Unlike the M2O matching models studied in the related work, this paper considers a M2O matching with group model in which a task t_i prefers a group of HNs, which can collaboratively offload the task t_i with the minimized execution delay. Define $\overline{\mathcal{H}}_i$ as a subset of $\mathcal{H}_i$ and $\overline{\mathcal{H}}^*_i$ is the most preferred subset of HNs to derive the minimal execution delay achieved by solution of problem $\mathbf{P}(t_i|\mathcal{H}_i)$. Notably, as $\overline{\mathcal{H}}^*_i$=$\{F_i\}$, t_i is processed locally by F_i. Basically, for each task t_i, there is a optimal subset to offer the task offloading solution with minimal delay, thus inherently incur the coupling resource problem as illustrated in Fig. 5.

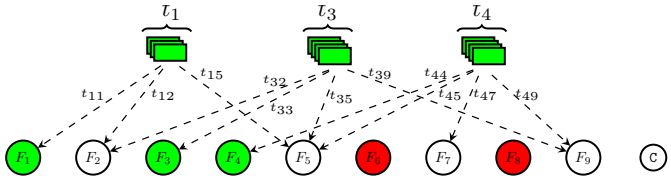


Fig. 5: An example of optimal resource allocation solutions for offloading three tasks T_1, T_3, and T_4: $\overline{\mathcal{H}}^*_1 = \{F_1, F_2, F_5\}$, $\overline{\mathcal{H}}^*_3 = \{F_2, F_3, F_5, F_9\}$, $\overline{\mathcal{H}}^*_4 = \{F_4, F_5, F_7, F_9\}$ obtained from their own optimization problems. Because a HN can only process a task/subtask, the resource contention occurs at F_2, F_5, and F_9.

The MATO scheme includes two algorithms, which are developed to create the PLs of agents (i.e. tasks and HNs), and finally produce a stable matching based on the DA algorithm.

B. PL Construction

1) PLs of tasks: Faced with a set $\mathcal{H}_i$ of HNs, each task t_i can rank all subsets of $\mathcal{H}_i$ based on the minimal delay they can offer to offload t_i. However, when $|\mathcal{H}_i|$ is large in dense fog networks, this method is inefficient because it incurs high overhead of computation. To overcome this issue, we first determine quotas for task agents as follow.

Definition 5.1: The quota q_i for a task t_i is the number of HN optimized by the problem $\mathbf{P}(t_i|\mathcal{H}_i)$. Alternatively, $q_i = |\overline{\mathcal{H}}^*_i|$, and $q_i \leq |\mathcal{H}_i|$.

Consequently, the tasks may have different quotas for matching with agents of HNs, that is contrary to the model in [21] considering the same quota for all tasks.

Definition 5.2: A HN F_k is acceptable to a task t_i if and only if $D^*_i(t_i|F_k) < D^*_i(t_i|F_i)$.

This means that delay achieved by full offloading $D^*_i(t_i|F_k)$ is lower than delay achieved by local computing $D^*_i(t_i|F_i)$.

Finally, we have an acceptable list of HN agents denoted as $\mathcal{L}_i$ that t_i can match with, and $\mathcal{L}_i = \mathcal{H}^*_i \cup \{F_k, F_j, ...\}$, where F_k, F_j are HNs selected according to Definition 5.2 and F_k, $F_j \in \mathcal{H}_i$. Each t_i can construct its own PLs from the subsets of $\mathcal{L}_i$. Basically, $\mathcal{P}(t_i)$ = $\{\overline{\mathcal{H}}^*_i, \overline{\mathcal{H}}^x_i, \overline{\mathcal{H}}^y_i, ..., F_i\}$, where $\overline{\mathcal{H}}^x_i$, $\overline{\mathcal{H}}^y_i$ are subsets of $\mathcal{L}_i$ and $D^*_i(t_i|\overline{\mathcal{H}}^*_i) < D^*_i(t_i|\overline{\mathcal{H}}^x_i) < D^*_i(t_i|\overline{\mathcal{H}}^x_i) < ... < D^*_i(t_i|F_i)$, and $D^*_i(t_i|\overline{\mathcal{H}}^x_i)$ is obtained from solution of $\mathbf{P}(t_i|\overline{\mathcal{H}}^x_i)$

2) PLs of HNs: We adopt the concept of processing efficiency (PE) as presented in [21] to evaluate the capability of HNs in offloading. Recall that PE of a HN F_k with respect to a TN F_i denoted as $\rho(k, i)$ is calculated as follow:

$$\rho(k, i) = \frac{1}{r_{ik}} + \frac{\gamma_k}{f_k}. \tag{7}$$

This parameter specifies the efficiency of HN F_k to offload a bit data sent from a TN F_i. Based on PE, each HN can construct its PL over individual tasks. The PL of a HN F_j is in this form, $\mathcal{P}(F_k) = \{t_m, t_n, t_p\}$. Basically, for a pair of tasks t_m and t_n, $t_m \succ_{F_k} t_n$ if $\rho(k,m) < \rho(k,n)$.

C. Matching Algorithm

Based on the PLs of tasks and HNs, MATO employs the DA algorithm to find the stable matching as shown in Algorithm 1. We implement the task-optimal stable matching algorithm, in which the task proposes and the role of HNs is to reject or accept the proposals based on their PLs.

Algorithm 1: M2O matching algorithm

Input: $\mathcal{P}(t_i)$, $\mathcal{P}(F_k)$, $\forall\ t_i \in \mathcal{T}$ & $F_k \in \mathcal{H}$

// PLs of all tasks and HNs

Output: $\mathcal{M}$

begin

  Step 1: Each task t_i proposes to its most prefered subset of HNs (i.e., $\overline{\mathcal{H}_i^*}$)

  For every $F_k \in \overline{\mathcal{H}_i^*}$:

  if $t_i == \mathcal{P}_k[0]$ // $\mathcal{P}_k[0]$ is the most prefered agent of F_k in the PL of F_k

  then

    F_k accepts the proprosal

  else

    F_k rejects the proprosal

    t_i removes $\overline{\mathcal{H}_i^*}$

    Update $\mathcal{P}(t_i)$

  while *There is a rejection from HNs* **do**

    Each task t_i continues to proposes to its most prefered subset of HNs in the updated PLs including all of HNs accepted the previous proposals

VI. Simulation and Performance Evaluation

A. Simulation Environment Setup

Table I summarizes the important parameters and values for the simulation scenario, where U[x,y] indicates the uniform distribution on interval [x, y]. The scenario includes 100 FNs deployed uniformly in a region of area 200 m $\times$ 200 m. The coverage of each FN is 40 m. The cloud provides 50 VMs for offloading tasks.

All the simulation results are averaged over 100 simulation rounds and each round lasts 1000 seconds according to the clock of CPU. We compare the proposed MATO approach with DATS, and POST.

B. Evaluation and Analysis

Fig. 6 depicts the average delay achieved by POST, DATS, and MATO when β is varied.

When there is a large percentage of HNs in the network (i.e., β is small (0.1, 0.2)), MATO, DATS, and POST have a similar performance. With this scenarios, there are abundant resources to offer the optimal solutions for offloading tasks. When β increases, the performance gap between MATO and DATS, POST is larger. When $\beta = 0.9$, there presents a high workload in the network and a small percentage of available resources (HNs). Therefore, it is likely to have more tasks computed locally by TNs. That leads to a considerable increase of delay for the three algorithms. However, MATO still outperforms DATS, and POST because it can allocate the resources better through the stable matching. Concretely, DATS and POST assign the best HNs to tasks in a greedy manner. Meanwhile, MATO uses the principals of stability in the matching theory to allocate the resource fairly.

To further examine the potential of MATO, we evaluate the delay reduction ratio (DRR) defined as the total delay reduction compared to the total delay of local computation. Fig. 7 shows the simulation result regarding the DRR achieved by the three algorithms. All schemes employed task division and associated task parallel processing, hence reducing the delay considerable compared with the local computing. In particular, MATO is able to reduce delay substantially compared to DATS, and POST owing to the optimal scheluding mechanism.

TABLE I: Parameters for simulation

Parameters	Values
Number of FNs, N	100
Number of VMs provided by cloud, M	50
Task size, a_i	{5, 10, 15}(MB)
Active ratio of TNs, β	{0.1-0.9}
Processing density of FNs, γ	U[500,1000] (cycles/bit)
CPU frequency of FNs, f	{1.0, 1.5, 2.0, 2.5} (GHz)
Processing density of each VM, γ_c	100 (cycles/bit)
CPU frequency of each VM, f_c	10 (GHz)
Transmitting power of FNs, P_t	U[100,350] mw
Transmitting power of GW, P_t	20w
Noise power spectral density, N_0	-174 dBm/Hz
Bandwidth, B	U[15-90] KHz

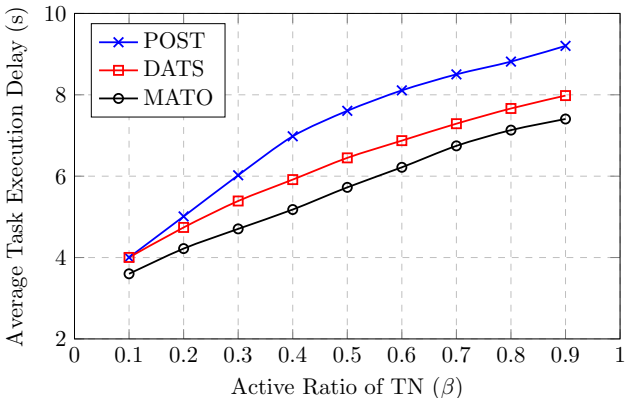


Fig. 6: The average task execution delay offered by the comparative offloading schemes.

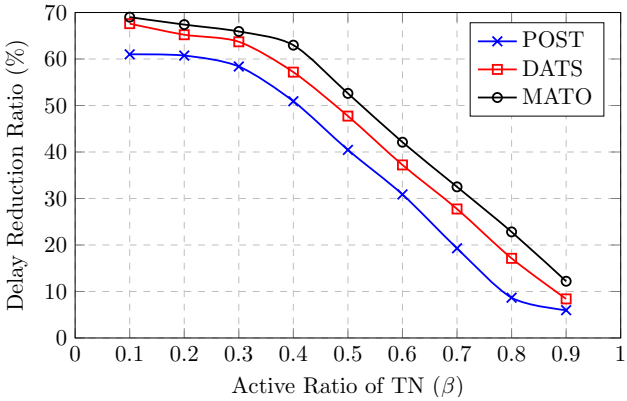


Fig. 7: The delay reduction ratio offered by MATO, DATS, and POST.

VII. Conclusions and Future Works

This paper presented MATO, a distributed computation offloading framework used for the heterogeneous FCNs to reduce the delay of task execution using matching theory. The intensive simulation results show that MATO is an effective offloading solution to achieve the reduced execution delay for fog computing environment, in which the fog computing devices are heterogeneous complicated, and different kinds of tasks. By applying task division technique and the associated parallel computation, MATO allows a task to be processed by multiple HNs to significantly reduce the overall task execution delay. The optimal scheduling of subtask transmission and subtask processing is also integrated to contribute to the delay reduction objective. The matching theory based principals are used to remove the rational and selfishness of individual TNs and simultaneously offer the fair resource allocation.

Acknowledgments

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] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4590–4602, Oct 2018.

[3] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D.-S. Kim, "The internet of things for logistics: Perspectives, application review, and challenges," pp. 1–29.

[4] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D. Kim, "Toward the internet of things for physical internet: Perspectives and challenges," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4711–4736, 2020.

[5] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009, pp. 44–51.

[6] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, Mar. 2016.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC 2012*. ACM Press, 2012.

[8] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, Jan. 2018.

[9] 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.

[10] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: a green computing paradigm to support iot applications," *IET Networks*, vol. 5, no. 2, pp. 23–29, 2016.

[11] 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.

[12] K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali, and D. A. Ibrahim, "A review of fog computing and machine learning: Concepts, applications, challenges, and open issues," vol. 7, pp. 153 123–153 140.

[13] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2018.

[14] G. Lee, W. Saad, and M. Bennis, "An online optimization framework for distributed fog network formation with minimal latency," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2244–2258, 2019.

[15] K. Guo, M. Sheng, T. Q. S. Quek, and Z. Qiu, "Task offloading and scheduling in fog ran: A parallel communication and computation perspective," *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 215–218, 2020.

[16] Y. Yang, Z. Liu, X. Yang, K. Wang, X. Hong, and X. Ge, "Pomt: Paired offloading of multiple tasks in heterogeneous fog networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8658–8669, 2019.

[17] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "Post: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3170–3183, 2020.

[18] S. Durand and B. Gaujal, "Complexity and Optimality of the Best Response Algorithm in Random Potential Games," in *Symposium on Algorithmic Game Theory (SAGT) 2016*, pp. 40–51. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01404643

[19] E. A. Jorswieck, "Stable matchings for resource allocation in wireless networks," in *2011 17th International Conference on Digital Signal Processing (DSP)*, pp. 1–8.

[20] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Meto: Matching-theory-based efficient task offloading in iot-fog interconnection networks," vol. 8, no. 16, pp. 12 705–12 715.

[21] 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.

[22] D. Gusfield, "The structure of the stable roommate problem: efficient representation and enumeration of all stable assignments," *SIAM Journal on Computing*, vol. 17, no. 4, pp. 742–769.

[23] K. Eriksson, J. Sjöstrand, and P. Strimling, "Three-dimensional stable matching with cyclic preferences," *Mathematical Social Sciences*, vol. 52, no. 1, pp. 77–87.

[24] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," vol. 69, no. 1, pp. 9–15.

[25] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via fog-2-fog collaboration," *Future Generation Computer Systems*, vol. 100, pp. 266–280, Nov. 2019.