JYX JYVÄSKYLÄN YLIOPISTO UNIVERSITY OF JYVÄSKYLÄ

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Samanta, Amit; Chang, Zheng; Han, Zhu

Title: Latency-Oblivious Distributed Task Scheduling for Mobile Edge Computing

Year: 2018

Version: Accepted version (Final draft)

Copyright: © 2018 IEEE.

Rights: In Copyright

Rights url: http://rightsstatements.org/page/InC/1.0/?language=en

Please cite the original version:

Samanta, A., Chang, Z., & Han, Z. (2018). Latency-Oblivious Distributed Task Scheduling for Mobile Edge Computing. In GLOBECOM 2018 : Proceedings of the 2018 IEEE Global Communications Conference. IEEE. IEEE Global Communications Conference. https://doi.org/10.1109/GLOCOM.2018.8647673

Latency-Oblivious Distributed Task Scheduling for Mobile Edge Computing

Amit Samanta*, Zheng Chang†, and Zhu Han‡§
*Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India
†Faculty of Information Technology, University of Jyvaskyla, FI-40014 Jyvaskyla, Finland
‡University of Houston, Houston, TX §Kyung Hee University, Seoul, South Korea
E-mails: amit.samanta049@gmail.com, zheng.chang@jyu.fi, zhan2@uh.edu

***Abstract*—Mobile Edge Computing (MEC) is emerging as one of the effective platforms for offloading the resource- and latency-constrained computational services of modern mobile applications. For latency- and resource-constrained mobile devices, the important issues include: 1) minimize end-to-end service latency; 2) minimize service completion time; 3) high quality-of-service (QoS) requirement to offload the complex computational services. To address the above issues, a latency-oblivious distributed task scheduling scheme is designed in this work to maximize the QoS performance and goodput for the MEC services. Unlike most of the existing works, we consider the latency-oblivious property of different services in order to achieve the optimized goodput and service latency. Furthermore, we design an optimal decision engine for efficiently offloading the computational services. Simulation results are presented to demonstrate the effectiveness of the proposed offloading scheme over other existing state-of-the-art solutions, in terms of service latency, goodput, service completion time and fairness.**

I. Introduction

Due to advances in new technologies, extensive mobile applications is expending beyond our daily life enormously in the era of Internet-of-Things (IoT) [1]. Hence, the computation of mobile applications has become more intense and complicated. The recent report from Chaffey's suggest that the users spend at least 15 % of their time on playing mobile games and another 20 % for refreshment, which demands extreme computational power and storage [2], [3]. However, the mobile devices have very limited capacity in terms of processing power, storage and battery life. To provide solutions to these shortcomings, mobile computational offloading techniques have been proposed to execute intensive computational tasks to more effective and qualified computing devices.

In recent years, Mobile Edge Computing (MEC) [4]–[7] have gained a lot of popularity among researchers, due to it's efficient computational offloading capability at the network edge. More importantly, it supports a productive and incentive computation offloading mechanism, while adjusting different network- and user-level optimization. Primarily, MEC platforms are deployed at the network edge in order to optimize the execution requirements of computational service offloading. They are expected to favour the offloading mechanism, while strongly decreasing the network latency between the edge users and servers. The core element of any mobile computation offloading framework is the optimal decision engine, since it determines when a task needs to be offloaded to an external (MEC) server. Offloading a task to an external server may incur expensive overhead, hence the offloading decision shall be based on predictions of the latency and total time required to offload their services, as well as the computation time among other possible metrics. It is not easy to predict the service latency of an application method ahead of its execution. An offloading framework addresses these challenges to make efficient offloading decisions and also to provide application developers and/or users a way to integrate their application into the offloading framework. Most offloading frameworks proposed so far predict the available bandwidth or execution time [8]. They have completely ignored the variation in latency requirements of mobile edge services, which is termed as *latency-oblivious*. Another common assumption is that the inputs of the computational services do not vary much, which can lead to imprecise estimation of latency requirements. However, modern mobile application requires different processing powers and latency capabilities to process their tasks at local operator clouds, private edge clouds (also called MEC hosts) and remote clouds. Therefore, it is necessary to accurately estimate the latency-requirement of mobile services. Hence, it is important to propose a latency-oblivious distributed task scheduling scheme for MEC.

This paper presents a distributed task scheduling framework, while taking into consideration of optimal latency requirements of edge services. It is equipped with the novel algorithms aimed to estimate the optimal latency-requirements of edge applications, as well as we aim to minimize the service completion time. Our main contributions are summarized as:

1. We propose a latency-oblivious distributed task scheduling scheme for mobile edge devices to minimize the service latency, while maintaining high throughput of the network. We also present a cost-effective service offloading framework for MEC architecture to provide optimal resources and profit to edge devices for efficient computational offloading.
2. We propose an optimal latency estimation engine, which accurately estimates the service latency of different mobile applications to support effective offloading decision for the MEC platform. We also consider the optimal latency-constraint for efficient task scheduling.
3. Simulation results demonstrate that our algorithm can

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

effectively schedule the available tasks from edge devices to servers. The results also show that the proposed scheme provides higher throughput while minimizing service latency. It also yields the best performance, in terms of service utility and service completion time, under different traffic rate, compared with other solutions.

The rest of the paper is organized as follows. Section II describes the related work. In Section III, we present the system model for MEC. Section V describes a latency-oblivious adaptive task scheduling scheme for MEC, in particular, our service offloading framework is designed to provide maximum revenue. Section VI conducts extensive simulations to validate our proposed scheme, and Section VII concludes the paper.

II. Related Work

The problem of task scheduling for MEC platform in the presence of different network dynamics is a challenging job. Over the years, only a few researchers have addressed some of the important issues related to this problem. Liu *et al.* proposed latency and reliability-aware task scheduling scheme for MEC [9]. They did not consider the latency-oblivious property for different tasks. Chang *et al.* proposed energy-efficient optimization framework for fog computing system [10]. Ren *et al.* proposed a partical offloading mechanism to minimize the delay for MEC platform [11]. Chen *et al.* proposed a multi-task offloading scheme for multi-user in MEC [12]. Fan *et al.* proposed application-aware workload allocation for IoT-enabled edge platform [13]. Genez *et al.* proposed latency-aware policy consolidation scheme for edge computing systems [14]. Neto *et al.* proposed user-level online offloading framework for MEC systems [1]. Ranadheera *et al.* proposed an optimal framework for computation offloading and activation of edge servers using minority game [15]. Lyu *et al.* proposed energy-efficient admission algorithm for delay-sensitive tasks in MEC [16]. Yang *et al.* proposed MEC-empowered Energy efficient task offloading scheme in 5G [17]. Liu *et al.* proposed multiobjective optimization framework for computation offloading in fog computing [18].

In summary, most for the existing studies [1], [9]–[19] mainly focus on the energy-efficient offloading and resource allocation in MEC platform. They did not consider any network dynamic in terms of service latency and data traffic. They assumed that the latency requirement of different tasks are available to them a prior, but in real life, it is very tough to get to know the actual and practical latency requirements of them. As discussed previously, network dynamics inherently changes the latency requirements of edge devices inherently, which reduces the network throughput and also increases the service completion time. This motivates us to study latency-oblivious distributed task scheduling scheme to minimize the total end-to-end delay, while maintaining high throughput, under different network dynamics.

III. System Model

Without loss of generality, we assume there are n mobile devices denoted by, $\mathcal{H} = \{H_1, H_2, \cdots, H_n\}$, coexisting in an

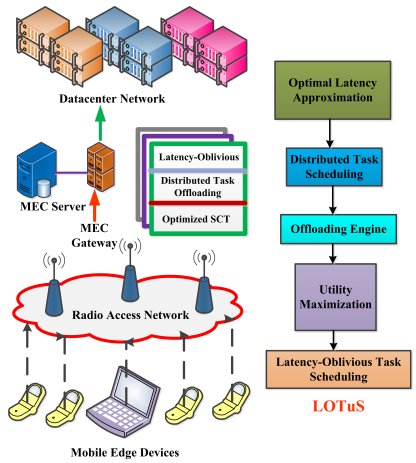


Figure 1: Latency-Oblivious Task Scheduling in MEC

area offload the computational services, as shown in Fig. 1. Each of the edge devices is comprised of different kind of services denoted by $\mathbb{S} = \{S_1, S_2, \cdots, S_K\}$ and they belong to different real-time mobile applications (i.e., self-driving car, video analytic, augmented-reality, object detection system etc.). The mobile devices offload their computational services to edge servers denoted by, $\mathcal{Y} = \{Y_1, Y_2, \cdots, Y_m\}$. The owners of the edge servers is considered to be associated with different edge computing platforms severed by edge service providers denoted by $\mathcal{X} = \{X_1, X_2, \cdots, X_o\}$. The traffic flow T_{flow} of computational services arrives at the edge servers by following Poisson distribution [20]–[23]. Further, the offloading process of computational services is scheduled according to their priorities, which is determined by service competition time and QoS. The edge devices require fair amount of resources in order to offload the services optimally. We assume that the mobile device H_i has maximum $\mathbb{R}_i^{max}$ and minimum $\mathbb{R}_i^{min}$ resource requirements, which is known to the edge platform.

Fundamentally, the edge devices have very limited energy to offload their computational services, therefore the service offloading mechanism is very important to minimize the energy-consumption rate of mobile devices. Also, they have a very stringent latency-requirements, hence it is necessary to estimate it correctly. Here, we consider the initial energy of an edge device H_i is $\mathcal{E}_{ini}^i$. Along with energy consumption, it is necessary to minimize the service offloading cost in order to maximize the fairness of mobile devices. Here, the total service offloading price for an edge device H_i is denoted by $\mathbb{C}_{off}^t$. Therefore, we propose a latency-oblivious distributed task scheduling scheme to minimize the service latency and completion time. Further, we propose latency-oblivious distributed task scheduling algorithm for MEC.

IV. Optimal Latency Approximation

As previously discussed that the mobile devices have a very stringent latency requirement to offload the computational services. However, it is very tough to know the actual latency

requirements of mobile devices practically. Also, the latency requirements of mobile devices change dynamically, as it is oblivious to the edge platform. Further, due to heavy network load and congestion, the service offloading latency increases abruptly in the network, which inherently increases the service completion time of tasks. First, we need to estimate the total service latency encountered by mobile devices, while offloading the computational services. Later, we propose a distributed task scheduling scheme, while taking into consideration of unique priorities of different tasks.

• ***Execution Latency***: The execution latency is deepened on the average waiting time and total network delay of incoming and outgoing flows. Mathematically,

$$\mathcal{D}_i^{exe}(t) = \mathbb{T}_i(F_i) + \mathbb{T}_n et(i) \tag{1}$$

where $\mathbb{T}_i(F_i)$ and $\mathbb{T}_n et(i)$ denote the average waiting time and total network delay of incoming and outgoing flows F_i.

• ***Queuing Latency***: The service queuing delay in the network for available tasks is denoted as:

$$\mathcal{D}_i^{que}(t) = \mathcal{G}\mathbb{Q}_i(t) \tag{2}$$

where $\mathbb{Q}_i(t)$ denotes the average queuing latency and $\mathcal{G}$ denotes the number of existing tasks in the queue.

• ***Offloading Latency***: The offloading latency $\mathcal{D}_i^{off}(t)$ is directly proportional to the total waiting time to offload the computational services on edge devices, which is:

$$\mathcal{D}_i^{off}(t) = \frac{\mathbb{J}_i^t}{q_i^t} \tag{3}$$

where $\mathbb{J}_i^t$ and q_i^t denote the total service length and time to execute the service S_k at time t, respectively.

Hence, the total estimated service latency $\mathbb{D}_{to}^t$ for edge device ED_i is the addition of both service execution latency $\mathbb{D}_{EL}^t$ and service offloading latency $\mathbb{D}_{off}^t$, defined as:

$$\mathcal{D}_i^{tot}(t) = \mathbb{T}_i(F_i) + \mathbb{T}_n et(i) + \mathcal{G}\mathbb{Q}_i(t) + \frac{\mathbb{J}_i^t}{q_i^t} \tag{4}$$

V. Distributed Task scheduling

After estimating the latency requirements of mobile devices, we now model a distributed task scheduling scheme for edge devices to maximize the goodput and QoS of the network. Suppose, we assume that the computational services require $\mathcal{T}$ slots to offload them efficiently to edge servers. Here, we consider a time frame with different time slots. We describe the length of time slot and index of time-slot by t and $t \in \mathcal{T} = \{1, 2, \cdots, \}$, respectively. In a time-slot, if more than one edge device chooses a particular channel for offloading their services, then we have used the carrier sense multiple access mechanism to overcome the possible collisions in the network. To minimize the service completion time, it is necessary to estimate the total cost for computational service offloading, which is discussed below:

Estimation of Cost Factors: The edge computing platform estimates different cost factors in order to offload the services efficiently. The cost factors are discussed as follows:

- *Edge Storage Cost*: The storage cost of an edge service in time slot t is equal to the storage cost of all its replicas in edge servers $\mathcal{Y}$ in time slot t [24]. Thus, we have

$$\mathcal{C}_i^{sto} = c_i^{sto}(t)V(t) \tag{5}$$

where $c_i^{sto}(t)$ denotes the storage cost of edge servers $\mathcal{Y}$ per unit size per unit time and $V(t)$ denotes the size of the edge service in time slot t.

- *Replication Cost*: The propagation cost for updating replicas of the edge services, which is defined as:

$$\mathcal{C}_i^{rep} = c(Y, Y_p) = \sum_{i \in n} \big([w^s(t) \times z_l(Y)] + w^s(t)[z_l(Y') + V(t) \times \mathbb{L}(Y_p)]\big) \tag{6}$$

where $c(Y, Y_p)$ denotes the transfer cost between Y and Y_p, Y' denotes the edge server excluding Y and Y_p that hosts a replica, $w^s(t) \times z_l(Y)$ denotes the initial replication cost, $w^s(t)[z_l(Y') + V(t) \times \mathbb{L}(Y_p)]$ denotes the replication cost after transferring to another server, $w^s(t)$ denotes the total number of service replications, $\mathbb{L}(Y_p)$ out-network price of edge server Y_p per unit size.

- *Management Cost*: The server management cost, $\mathcal{C}_i^{man}$, is dependent on the mapping price $\mathcal{C}_i^{map}$ and initial sever development price $\mathcal{C}_i^{sd}$ [25]. It is defined as:

$$\mathcal{C}_i^{man} = w^s(t)[\mathcal{C}_i^{map} + \mathcal{C}_i^{sd}] \tag{7}$$

- *Migration Cost*: The total cost for virtual machine (VM) creation, management and migration is defined as:

$$\mathcal{C}_i^{vm} = \mathcal{C}_i^{vm_c re} + \mathcal{C}_i^{vm_m ang} \tag{8}$$

where $\mathcal{C}_i^{vm_c re}$ and $\mathcal{C}_i^{vm_m ang}$ denote the unit VM creation price and management price, respectively.

The total cost $\mathcal{C}_i^{tot}$ charge by edge platform is defined as, $\mathcal{C}_i^{tot} = \mathcal{C}_i^{sto} + \mathcal{C}_i^{rep} + \mathcal{C}_i^{man} + \mathcal{C}_i^{vm}$.

A. Offloading Decision Engine

In order to schedule the tasks, we need to design an optimal offloading decision engine. First, we design the net utility function for each mobile devices based on the estimated cost factors and other decision metrics (i.e., energy consumption rate, QoS, service completion time etc.). Further, we design a latency-oblivious distributed task scheduling algorithm to minimize the total latency.

Definition 1. *The traffic flow of available task is depended on the size of the task and total time required to complete it. Mathematically,*

$$F_i^{in}(t) = S_{siz}\mathcal{T}_S \sum_{i \in n} F_i^S(t) \tag{9}$$

where $\sum_{i \in n} F_i^S(t)$ denote the total number of aggregated tasks coming from mobile devices, S_{siz} and t_i denote the size of the task and total time required to complete it, respectively.

Definition 2. *The service capacity is depended on the base computing capacity and total resources allocated to available*

tasks [26]. It is mathematically defined as:

$$\mathbb{C}_i = \mathbb{C}_i^{ba} + \lambda \sum_{i \in n} \eta_{i,re} \tag{10}$$

where $\mathbb{C}_i^{ba}$ denotes the base computing capacity, λ denote the total number resource blocks and $\eta_{i,re}$ denotes the total resources allocated to tasks. Mathematically, we have, $\eta_{i,re} = \frac{\zeta_i}{\mathbb{R}_i}$. Here, ζ_i and $\mathbb{R}_i$ denote the data rate and received data rate per resource blocks, respectively.

Definition 3. *The energy consumption rate of mobile device is deepened on the energy of device H_i when it is idle and energy of device H_i when it is full loaded. Mathematically,*

$$\mathcal{E}_i = \mathcal{E}_{ini}^i + \mathcal{E}_{load}^i \frac{\mathbb{C}_i^{ba}}{\mathbb{C}_i^{max}}$$

where $\mathcal{E}_{ini}^i$ denotes the energy of device H_i when it is idle, $\mathcal{E}_{load}^i$ denote the energy of device H_i when it is full loaded and $\mathbb{C}_i^{max}$ denote the maximum base computing capacity.

Definition 4. *The average number of requests $\mathcal{N}_{req}$ is depended on the total number of aggregated requests and probability of coming $\mathcal{B}$ service requests. Mathematically,*

$$\mathcal{N}_{req} = \pi_{\mathcal{B},i}^S \frac{\sum_{i=0}^K \gamma_i^S}{\mathcal{B}} \tag{11}$$

where $\sum_{i=0}^K \gamma_i^S$ denotes the aggregated service requests and $\pi_{\mathcal{B},i}^S$ denotes the probability of coming $\mathcal{B}$ service requests.

Definition 5. *The service completion time is directly proportional to the mean service rate per server and number of servers allocated to serve the delay-sensitive workload. Mathematically,*

$$SCT_i = \frac{1}{\mathbb{W}_i \Theta_i(t)/b_i(t)} \tag{12}$$

where $b_i(t)$ denotes the number of servers allocated to serve the delay-sensitive workload, $\Theta_i(t)$ denotes the mean service rate per server and $\mathbb{W}_i$ denotes the arrival rate at time t.

Definition 6. *The QoS-level is defined as the ratio of total number of computational services offloaded successfully to edge servers and the total service latency of edge devices. It is defined as:*

$$\mathbb{Q}_i = \frac{\mathcal{X}_i}{\sum_{i \in n} \sum_{t \in \mathcal{T}} \mathcal{D}_{tot}^t}, \tag{13}$$

where $\mathcal{X}_i$ denotes data size of a computational service for mobile device H_i, and $\mathcal{D}_{tot}^t$ denotes the total estimated service latency for mobile device H_i.

Definition 7. *The service queue is developed to store the different task request from mobile devices. Hence, we model a service quque [27], which is defined as:*

$$Q_i(t+1) = [Q_i(t) - a_i(t) f_i + \mathbb{H}_i(t)]^+ \tag{14}$$

where f_i denotes the total resource provided by a single server during one time slot, $\mathbb{H}_i(t)$ denotes the resource demand of a task at time t, $a_i(t)$ denotes the number of servers allocated to serve the delay-tolerant task workload at time t and $Q_i(t)$ denotes the amount of the unfinished task workload at the beginning of time slot t.

B. Utility Maximization Framework

Using the Definitions 1 – 7, we formulate net utility $\mathcal{U}_i$ for computational service offloading from edge devices to servers, which is expressed as:

$$\mathcal{U}_i = \Upsilon_1 \frac{\mathbb{C}_i \mathbb{Q}_i F_i^{in}(t)}{\mathcal{N}_{req}} - \Upsilon_2 \left[\frac{\mathcal{E}_i}{\mathcal{E}_{max}} + \frac{\mathcal{C}_i^{tot}}{\mathcal{C}_i^{max}} + \frac{1}{\mathbb{W}_i \Theta_i(t)/b_i(t)} \right], \tag{15}$$

where Υ_1 and Υ_2 denotes the scaling factors for distributed task scheduling. $\mathcal{C}_{max}$ is the maximum cost set by the edge platform. Having computed the net utility for each mobile device, the mobile device with the maximum net utility value emerges as the winner and get to schedule its services first than the others. Thus, without the loss of generality, we formulate the optimization problem as:

$$\textbf{(P1)}: \underset{t>0}{\text{maximize}} \sum_{i \in n} \mathcal{U}_i, \tag{16}$$

$$\mathcal{E}_i \geq \mathcal{E}_{max}, i \in n, \tag{17}$$

$$\mathbb{C}_i \geq \mathbb{C}_{th}, i \in n, \tag{18}$$

$$\text{Subject to} \quad SCT_i \geq SCT_{th}, i \in n, \tag{19}$$

$$\mathcal{Q}_i \geq \mathcal{Q}_{th}, i \in n, \tag{20}$$

$$\mathcal{C}_i^{tot} \geq \mathcal{C}_{max}, i \in n. \tag{21}$$

(16) presents the primary optimization function for distributed task scheduling. (17) describes that the energy consumption rate, $\mathcal{E}_i$, is to be greater than the threshold energy consumption rate, $\mathcal{E}_{max}$. The service capacity of a task, $\mathbb{C}_i$, is to be greater than the threshold service capacity, $\mathbb{C}_{th}$, as shown in (18). (19) represents that the service completion time of a task, SCT_i, is to be grater than the threshold service completion time, SCT_{th}. The QoS-level, $\mathcal{Q}_i$, is to be greater than the threshold QoS-level, $\mathcal{Q}_{th}$, as shown in (20). (21) denotes that the total estimated cost for mobile device $\mathcal{C}_i^{tot}$, is greater than the threshold cost, $\mathcal{C}_{max}$. Solving the optimization problem using the Lagrangian multiplier, we get,

$$\begin{aligned} \Delta_{\mathcal{U}} = \sum_{i=1}^n \frac{\omega_i}{\mathcal{U}_{th}} \Gamma_i \Big(\mathcal{E}_i, \mathbb{C}_i, SCT_i, \mathcal{Q}_i, \mathcal{C}_i^{tot} \Big) \\ - \Xi_1 \Big(\sum_{i=1}^n \mathcal{E}_i - \mathcal{E}_{max} \Big) - \Xi_2 \Big(\sum_{i=1}^n \mathbb{C}_i - \mathbb{C}_{th} \Big) \\ - \Xi_3 \Big(\sum_{i=1}^n SCT_i - SCT_{th} \Big) - \Xi_4 \Big(\sum_{i=1}^n \mathcal{C}_i^{tot} - \mathcal{C}_{max} \Big). \end{aligned}$$

where Ξ_1, Ξ_2, Ξ_3 and Ξ_4 denote the different constraints for Lagrangian Multipliers and ω_i denotes priority levels of different services in edge devices.

Theorem 1. *The considered Latency-Oblivious Distributed Task Scheduling scheme for MEC (LOTuS) is NP-hard.*

Proof. We prove the NP-hardness of this problem via a polynomial-time reduction from an uncapacitated facility location (UFL) problem [28], which is designed to be NP-hard problem. The relaxation can be done by considering the distance and cost functions in a UFL instance as the latency estimation and task scheduling functions in our problem – LOTuS, respectively, and setting the other things in LOTuS to zero. This concludes our proof. □

Here, (16) can be represented as a convex optimization problem which can be solved using interior point algorithm. Hence, our main objective is to maximize the value of $\mathcal{U}_i$ using the Lagrange multiplier. We have used gradient descent method to solve the problem. We also design a heuristic algorithm to find a local optima at each stage with the aim of finding a global optima.

Algorithm 1 Algorithm for Heuristic Task Scheduling

Inputs:
- Set of mobile devices ($\mathcal{H}$), set of services $\mathcal{S}$ and total time $\mathcal{T}$.

Output: Optimized service completion time $S\bar{C}T_i$ and waiting time T_{wa}.

Set $T_{wa} = 0$.
Set $\mathcal{Z} = n$ and $A = m$.
for each mobile device H_i **do**
  if $\mathcal{T} < T_{wa}$ **then**
    First, calculate the traffic flow of available task $F_i^{in}(t)$.
    Estimate service capacity $\mathbb{C}_i$.
    Calculate QoS level $\mathcal{Q}_i^{con}(t)$.
    Estimate energy consumption rate $\mathcal{E}_i$.
    Calculate average number of requests $\mathcal{N}_{req}$.
    Design utility function $\mathcal{U}_i$.
    if $\mathcal{U}_i \geq \mathcal{U}_{th}$ **then**
      Updated set of mobile devices $\bar{\mathcal{Z}} = \mathcal{Z} \cap H_i$.
      Optimized service completion time $S\bar{C}T_i$.
      Update waiting time $T_{wa} = T_{wa}$.
    else
      Updated set of mobile devices $\mathcal{Z} = \mathcal{Z}$.
      Non-optimal service completion time ($\widehat{SCT_i}$).
      Update waiting time $T_{wa} = T_{wa} + 1$.
    end if
  end if
end for
Return $S\bar{C}T_i$ and T_{wa}.

We discuss the algorithm for the heuristic task scheduling scheme. As shown in Algorithm 1, first, we need to provide three inputs – set of mobile devices ($\mathcal{H}$), set of services $\mathcal{S}$ and total time $\mathcal{T}$. As the designed problem is NP-hard, hence we design this heuristic packet scheduling scheme to optimize the service completion time and QoS in the network. Initially, we set the waiting time T_{wa} to 0. Thereafter, for each mobile device H_i, we conduct the scheduling algorithm. When the total time less than the waiting time, i.e., $\mathcal{T} < T_{wa}$, then we calculate the traffic flow $F_i^{in}(t)$. Afterward, we estimate packet transmission rate $\mathcal{F}_i(\mathcal{Z}, t)$ and calculate the QoS level $\mathcal{Q}_i^{con}(t)$. Also, we estimate the total service capacity $\mathbb{C}_i$ and calculate the energy consumption rate $\mathcal{E}_i$. Using the estimated and calculated variables, we design a utility function $\mathcal{U}_i$ for heuristic scheduling. If the utility function $\mathcal{U}_i$ greater than the threshold utility function $\mathcal{U}_{th}$, then we update the set of mobile devices $\bar{\mathcal{Z}} = \mathcal{Z} \cap H_i$. Also, we update the waiting time T_{wa} as well. Along with, we also get the optimized service completion time $S\bar{C}T_i$ using Lagrangian Multiplier. The process is stopped, when the waiting time crosses a predefined maximum waiting time T_{wa}^{max}. To optimize the service completion time for different available tasks using (16), we use the Lagrangian optimization technique to get the optimal value.

Theorem 2. *The worst-case computational complexity for heuristic task scheduling algorithm is $O(\mathcal{M}n^2)$, where n is the number of mobile devices.*

Proof. At first, each mobile device tries to offload their computational services to edge servers in order to get the optimal service completion time. Therefore, to get the optimal value, the worst case computation complexity of the task scheduling algorithm is $O(\mathbb{G}n^2)$. Before scheduling the tasks, the mobile devices try to minimize the total network latency in the absence of multiple mobile devices. To minimize the total latency, we proposed a latency-oblivious approximation scheme for each mobile devices. Hence, the worst case complexity of service latency approximation algorithm is $O(\mathbb{J}n)$. Thus, combining both the algorithms, we have,

$$T(n) = \mathbb{L}_1\{\mathbb{G}T(n^2) + \mathbb{J}T(n)\} + \mathbb{L}_2 T(1). \tag{22}$$

By combining the worst-case complexity for both the algorithms, we obtain, $O(\mathcal{M}n^2)$, where $\mathcal{M} = \mathbb{G} + \mathbb{J}$. Hence, we observe that the total computational complexity of LOTuS in the worst case, is $O(\mathcal{M}n^2)$ with n as the number of mobile devices, which completes the proof of Theorem 2. □

VI. Performance Evaluation

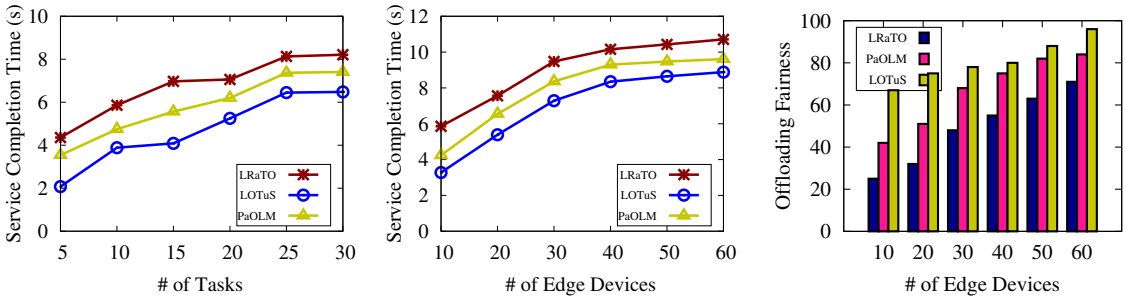
We present simulation results of the proposed scheme LOTuS compared to existing schemes. The simulation parameters used in the experiments are shown in Table I.

Table I: Experimental Parameters

Parameter	Value
Bandwidth	20 MHz
Total number of CPU cycles of computation task	1,000 Megacycles
Service deadline	[4000, 6000] ms
Computation resource demand	[10, 20] MHz
Transmission power of edge device	100 mWatts
Computation capability of edge device	0.7 GHz
Computation capability of the MEC server	100 GHz
Data traffic arrival modeled as Poisson process	[0, 10] unit/sec
Expected size of data traffic	100 Mbits
Computation service arrival (mean size = 1 Mbit)	[0, 10],

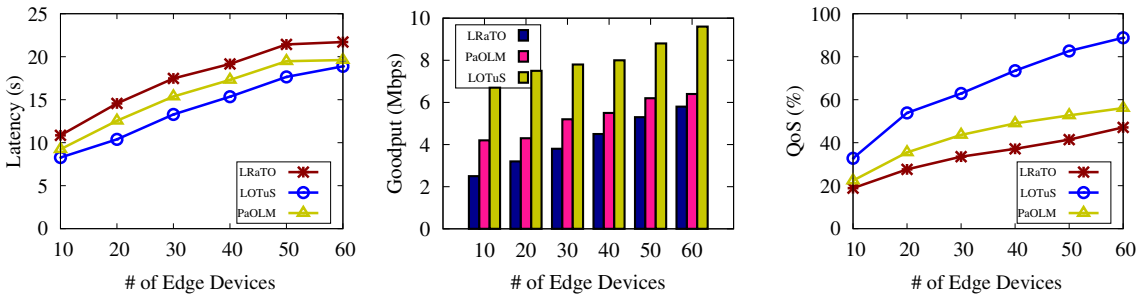
A. Experimental Setup

Parameter Settings: we have listed the experiential setup in Table I. We consider 200 edge devices which are distributed over an area of 1000m x 1000m and one MEC server located to one base station. The MEC server located in the base station, whose computation capability is 100 GHz and the computation capability of edge device is 0.7 GHz. Each base station has 50 orthogonal wireless channels for the computational service offloading from edge devices to edge servers. Here, the cellular backhaul delay coefficient is considered to be 0.0001 sec/KB [29], [30]. The total time duration to offload the computation services of mobile edge devices are randomly distributed between 5 and 10 ms. The corresponding computation file size of each computational



(a) Service Completion Time (b) Service Completion Time (c) Fairness

Figure 2: Analysis of service completion time and fairness



(a) Latency (b) Goodput (c) QoS

Figure 3: Analysis of latency, goodput and QoS

service varies within the range 300 and 800 KB. The delay requirements of edge devices are considered to be within 0.5-1 s. We used the D-ITG traffic generator [31] to model IoT traffic flows from real traces presented in [32].

Workload: We implemented our scheme in 10 servers, each machine configured with Intel core-i5 processor and 1.7 GHz CPU. For this work, we consider two types of traffic workloads – latency-sensitive (i.e., edge services) and latency-tolerant (i.e., cloud services) workloads. Here, the higher priority is given to delay-sensitive traffic than delay-tolerant traffic workloads. Thus, we ran edge services at a higher priority than the normal and background services, respectively.

Benchmarks: To evaluate the performance, we use two benchmarks - LRaTO [9] and PaOLM [11]. LRaTO [8] proposed a latency and reality-aware task offloading for MEC. They also implemented a resource allocation scheme, which tries to assign the optimal resources to edge devices and also minimizes the service latency of the network. However, they do not consider any latency-oblivious property of edge devices, which is unknown to the edge platform. PaOLM [11] proposed a partial offloading scheme for MEC in order to minimize the total latency of the network. This work also minimizes the energy consumption rate of the edge devices.

B. Results and Discussion

Fig. 2(a) shows the service completion time in the network for a varying number of tasks requested from different mobile devices. As the number of tasks increases in the edge platform, the service completion time also increases. To overcome this problem, we proposed a latency-oblivious distributed task scheduling scheme, which efficiently estimates the service latency requirements of different tasks and accordingly the tasks are scheduled among edge servers to minimize the service completion time. From the figure, we observe that the service completion time using proposed approach – *LOTuS* increases with the variation in a number of tasks. However, it performs better in compared to existing schemes – *PaOLM* and *LRaTO*, where we observe our scheme perform better in terms of service completion time by 12% and 17%, respectively. Fig. 2(b) presents the service completion time for varying number of mobile devices in the network. From the figure, we observe that the cumulative service completion time incurs for executing all the tasks available to edge platform. To optimize the service completion time, we proposed distributed task scheduling scheme, using that we are able to minimize the task execution time, which eventually minimizes the service completion time. It performs better than the existing approaches – *PaOLM* and *LRaTO*. Hence, our proposed approach – *LOTuS* outperforms the existing approaches by 8% and 10%, respectively. Fig. 2(c) shows the fairness among different tasks for varying number of mobile devices. From the figure, we oversee that the fairness for the proposed scheme – *LOTuS* increases with the increase in number of mobile devices. As the proposed task scheduling scheme provides fair resources to different tasks using the optimal offloading