Quantifying the Influence of Intermittent Connectivity on Mobile Edge Computing

Miao Hu, *Member, IEEE*, Di Wu, *Senior Member, IEEE*, Weigang Wu, Julian Cheng, *Senior Member, IEEE*, and Min Chen, *Senior Member, IEEE*

Abstract—Mobile edge computing (MEC) is a key technology that enables the deployment of applications (or services) at the proximity of mobile users. However, the performance of mobile edge computing is sensitive to the quality and availability of underlying connection links. It is still unclear to what extent intermittent connectivity affects the performance of mobile edge computing. In this paper, we make the first attempt to quantify the influence of intermittent connectivity on mobile edge computing from a theoretical perspective. Specifically, we propose an analytical framework based on discrete-time Markov chain and derive a closed-form expression of the task processing time under different network conditions. Our model can be further extended to account for the case with group task arrivals. We also conduct extensive simulations to examine the accuracy of our proposed analytical models with both synthetic and real-world user mobility traces. The results show that our model can well capture the influence of intermittent connectivity on MEC. Our model sheds important insights into the impact of intermittent connectivity on task processing in MEC, which we believe should be taken into account when designing future MEC systems.

Index Terms—Intermittent connectivity, mobile edge computing, task processing time

1 Introduction

Mobile edge computing (MEC) is a new form of network architecture that brings cloud computing capabilities to the edge of a cellular network [1], [2], [3]. In MEC, local cellular base stations are equipped with abundant computing and storage resources, and allow offloading of tasks from end devices. With the support of MEC, in addition to performing computations on remote cloud servers, a fraction of computation tasks can be handled locally at nearby base stations. Thus, MEC can significantly reduce the burden on resource-constrained end devices (e.g., smart phones, pads) in terms of computing power, storage and battery life, and can also achieve much lower latency compared with the traditional cloud computing paradigm. Note that MEC can be adopted to support both latency-sensitive applications (e.g., augmented reality [4], [5], video streaming [6], [7], online gaming [8], [9]) and delay-tolerant applications (e.g., connected vehicles [10], [11], Internet of things [12], [13], tactical network [14], [15]). In this paper, we focus on the delay-tolerant MEC service scenario, in which the remote cloud access is unavailable (e.g., data processing in the battlefield) and only edge servers in the proximity can provide services to users.

In an MEC system, applications on end devices can be split into a set of small tasks, among which some are processed at the edge cloud servers (or base stations) to reduce processing latency. However, the performance of MEC is sensitive to the quality and availability of underlying connection links. As we know, the phenomenon of intermittent connectivity prevails in a mobile network. Due to signal interference or user mobility, the communication link between an end device and an edge cloud server may not be always available. The quality and availability of MEC service exhibit a time-varying nature. Moreover, when the network is congested by traffic, the service capability of an MEC system is also significantly deteriorated. Thus, it is essential to fully understand how the change of underlying link conditions affects the performance of MEC and to what extent the impact is.

To answer the above questions, one approach is to conduct extensive experiments under various mobility patterns and investigate the performance of MEC from hand-on experiences. The major drawback of such an approach is that it is too time-consuming. Another approach is to develop an analytical model of an MEC system, which takes the impact of intermittent connectivity into account. The analytical model also allows us to examine extreme scenarios that cannot be covered by small-scale experiments. However, the development of such an analytical model for MEC is challenging. The reasons are multi-fold: *First*, the communication link becomes intermittent with user movement. It is challenging to accurately model the effects of intermittent connectivity in the MEC environment. *Second*, different task generation patterns will result in various levels of network traffic load, which in turn affect the performance of MEC. We need to consider the effect of network traffic load in the model. *Third*,

- *M. Hu, D. Wu, and W. Wu are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, Guangdong 510275, China, and also with the Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, Guangdong 510006, China. E-mail: {humiao5, wudi27, wuweig}@mail.sysu.edu.cn.*
- *J. Cheng is with the School of Engineering, The University of British Columbia, Kelowna, BC V1V 1V7, Canada. E-mail: julian.cheng@ubc.ca.*
- *M. Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: minchen2012@hust.edu.cn.*

Manuscript received 27 Dec. 2018; revised 18 May 2019; accepted 28 June 2019. Date of publication 10 July 2019; date of current version 8 Mar. 2022. (Corresponding author: Di Wu.)
Recommended for acceptance by C. Wu.
Digital Object Identifier no. 10.1109/TCC.2019.2926702

due to the limited capacity of edge servers, an offloaded task may not be processed immediately upon arrival. It needs to wait until the completion of previously queued tasks. The delay in the waiting queue should also be considered when developing the model of MEC.

In this paper, we attempt to understand the influence of intermittent connectivity on the performance of MEC from a theoretical perspective. Towards this purpose, we develop an analytical model of an MEC system, which captures the effects of intermittent connectivity on processing time of offloaded tasks. The complete task processing procedure can be divided into multiple stages, including task generation, task offloading, task queueing, task execution, and task result retrieval. Our model allows us to derive a closed-form expression for the expected task processing time. By utilizing a discrete-time Markov chain, we can derive the distribution of task queueing time at the edge cloud, based on which we can obtain the variance (or worst-case) of task waiting time. In addition, our model can be generalized to the case with group task arrivals. Our proposed model provides a tractable analytical framework for an MEC system, which can help design efficient task offloading algorithms.

The contributions of this paper can be summarized as follows:

- To the authors' best knowledge, this is the first work that develops an analytical model to understand the influence of intermittent connectivity on the performance of MEC. We model the whole procedure of task processing in MEC, which includes task generation, task offloading, task queueing, task execution and task result retrieval.
- We derive a closed-form expression for task processing time in an MEC system under various conditions with intermittent connectivity and random task generation. Specifically, our model takes intermittent connectivity and various levels of network load into account. We also obtain the distribution of task queueing time in the MEC system.
- We verify the accuracy of our model by performing extensive simulations with both synthetic and real-world user mobility traces. The results indicate that the proposed model can well capture the influence of intermittent connectivity on the performance of MEC. Based on our results, we also discuss potential improvements of existing task offloading algorithms.

The remainder of this paper is organized as follows. Section 2 presents the related work. The system model and definitions are introduced in Section 3. Section 4 builds analytical models for task processing time on MEC systems. Experimental results for the analytical models are presented in Section 5, where we also discuss the potential applications of our proposed analytical models. Section 6 concludes this paper and suggests several future research directions.

2 Related Work

In recent years, mobile edge computing has received significant attention from both academia and industry. Previous works mostly focused on how to efficiently perform task offloading in MEC systems (e.g., [16], [17], [18], [19], [20], [21]). These studies assumed that edge cloud servers are always available to the users, which does not always hold due to dynamic user movement and limited edge processing capacity.

The intermittent connectivity could cause a serious performance degradation on the quality and availability of edge service in both task offloading and result retrieval [22], [23], [24]. To reveal the influence of intermittent connectivity on MEC performance, Li et al. [25] assumed that mobility information of users can be known *a prior*, which is not reasonable in real MEC systems. As a practical solution, Shi et al. [26] modeled the intermittent connectivity as the received signal strength fluctuation to overcome the network uncertainty caused by variable connectivity. However, the measured signal strength cannot directly reveal the potential MEC performance with intermittent connectivity. Zhang et al. [27] defined the edge cloud service availability as a constant ratio of the task processing time and the average user contact time within the target edge cloud. But the average contact duration also cannot effectively describe the intermittent connectivity properties. By field measurements, it is observed that the random link connection can be statistically modeled [28], [29], [30]. Nevertheless, how these fitted distribution results affect the MEC performance (e.g., task offloading time and result retrieving time) is still unclear.

The limited edge processing capacity also affects the availability of an MEC system, especially on task queueing at the edge cloud servers. Wang et al. [31], [32] exploited the contact patterns regulated by the device mobility patterns; however, task queueing at the edge cloud server was not taken into consideration. There also exist several research works on evaluating MEC performance in terms of service access probability, task success rate, task execution speed and so on. Li et al. [33] showed that the edge cloud access probability can be evaluated by the link connection time between mobile users and the edge cloud. Again, the balance between the random task generation and limited edge processing capacity is not highlighted in the modeling.

In addition, it is also essential to consider task execution time in the modeling of MEC performance. Champati et al. [34] stated that the processing time required by a task is generally unknown prior to the processing action and later derived a closed-form expression for the task processing time. Their derivation is based on the assumption that an MEC system is perfectly connected, without considering existence of intermittent connectivity. Recent studies [35], [36], [37] proposed learning-driven methods to predict the task execution time given some practical assumptions. But these studies did not explore the influence of task execution property on MEC performance.

Different from the above works, our study aims to quantify the influence of conditions (i.e., intermittent connectivity and random task generation) on MEC performance by modeling the complete task processing stages (e.g., task generation, task offloading, task queueing, task execution, and task result retrieval).

3 System Model

3.1 Overview of an MEC System

We first describe the model of a typical MEC system, which contains *mobile users, edge cloud servers* and *communication*

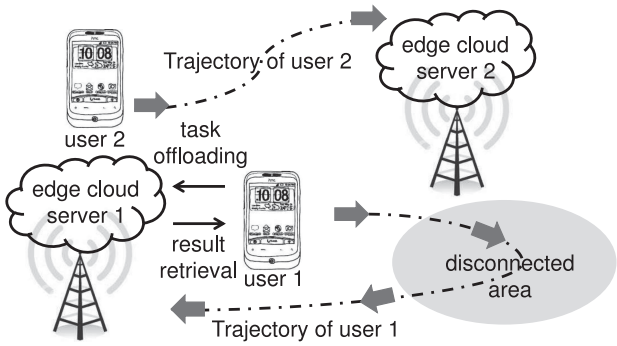


Fig. 1. A typical MEC system with pure edge cloud servers.

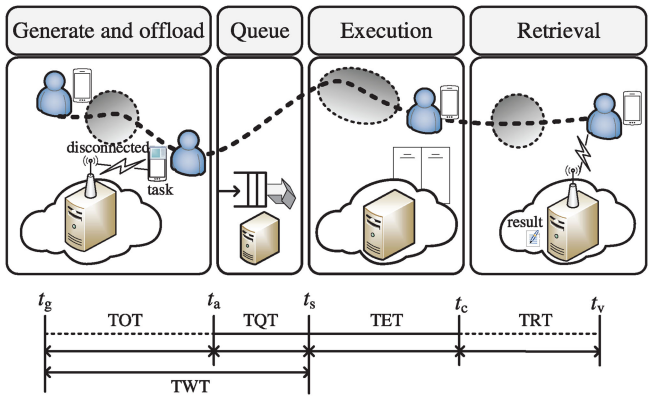


Fig. 2. Timeline of task processing in MEC.

links. The whole MEC system is illustrated in Fig. 1. In such an MEC system, edge cloud servers are installed on the base stations to provide computing services to resource-constrained end devices (e.g., smart phones, pads). Note that the MEC system can be composed of either pure edge cloud servers or hybrid edge/cloud servers [38]. In our model, we consider an MEC system with pure edge cloud servers, which is especially useful in mission-critical scenarios (e.g., battle-field, wild rescue). Anyway, our model can be easily extended to the scenario with hybrid edge/cloud servers.

In our model, the task processing procedure can be briefly described as below. After a task is generated by a user, it waits for the establishment of a link connection to an MEC server. Once the communication link is initialized, the task is off-loaded to the edge cloud server. If no other tasks are currently being processed at the same edge cloud server, it will directly be executed; otherwise, it enters into the queueing state until the completion of previous tasks. After the task result has been generated, the edge cloud server waits the establishment of a link connection to the task initiator and sends it back.

The user movement can cause intermittent connectivity, where the user might lose the wireless connection to the edge cloud servers (e.g., the disconnected area in Fig. 1). Due to the problem of intermittent connection, a task may need to wait for a while until the task initiator moves into the transmission range of a base station. In other words, a link disconnection can delay task offloading and result retrieval. Besides, the edge cloud servers have limited resources to serve all the users in the system. For example, user 2's task generation can affect the offloading process and the task execution process for tasks generated by user 1 at the edge cloud servers.

In this paper, we consider a simplified scenario in which each edge cloud server handles offloaded tasks independently. Specifically, when a user offloads a task to one edge cloud server, the user should also retrieve the results from the same edge cloud server. In case that the results cannot be retrieved within a timeout period τ (e.g., due to connectivity problem), the user has to re-send the task request to another edge cloud server. For the initial responsible edge cloud server, it will also drop the results after τ.

Next, we explore the influence of the intermittent connectivity on a general MEC system.

3.2 Timeline of Task Processing in MEC

Fig. 2 illustrates the timeline of task processing procedures, which can be divided into multiple stages, including *task generation*, *task offloading*, *task queueing*, *task execution*, and *task result retrieval*. To ease our presentation, we formally define a set of terminologies to be used in our analysis.

- *Task offloading time (TOT)*, which is defined as the interval between the time point of task generation and the time point of accessing the MEC system.
- *Task queueing time (TQT)*, which is defined as the queueing time for a task before its execution at the edge cloud servers after its access of the MEC system.
- *Task execution time (TET)*, which is the time taken by the execution of a task at the edge cloud server.
- *Task retrieval time (TRT)*, which is the time needed to retrieve all the execution results of a task.

As shown in the example (see Fig. 2), $t_{\mathrm{g}}^{(i)}$ denotes the generation time for task i, and $t_{\mathrm{a}}^{(i)}$ denotes the time to access the edge cloud server. Let $T_{\mathrm{o}}^{(i)}$ denote the TOT for task i, and we have $T_{\mathrm{o}}^{(i)} = t_{\mathrm{a}}^{(i)} - t_{\mathrm{g}}^{(i)}$, where the connection establishment time is much lower than T_{o} and can be ignored.

TOT is mainly determined by the intermittent connectivity. To describe user mobility patterns, we define the *link connection period* as the continuous connection interval between a user and an edge cloud server. Similarly, we define the *link disconnection period* as the continuous link breakage time interval. We model the link connection and disconnection periods as exponential random variables with parameters η and θ, respectively. Such an assumption has been widely used in previous studies [28], [29], [30]. We also conduct studies on real-world traces in Section 5 to verify that the assumption is reasonable.

After a task has been offloaded to an edge cloud server, it might not be processed immediately, which is determined by the task scheduling scheme. Suppose that the First In, First Out (FIFO) scheduling algorithm is used, which simply queues tasks in the order that they arrive in the queue. That is, a task may need to wait until the completion of execution for tasks that have been offloaded to the edge cloud server previously. Let $t_{\mathrm{s}}^{(i)}$ denote the time to start the execution of task i at the edge cloud server. Let $T_{\mathrm{q}}^{(i)}$ denote TQT for task i, and we have $T_{\mathrm{q}}^{(i)} = t_{\mathrm{s}}^{(i)} - t_{\mathrm{a}}^{(i)}$.

TQT is mainly affected by the network load (i.e., the task generation patterns). The task generation interval can be fitted with a specific distribution, and the Poisson distribution is assumed to follow the assumptions made in previous

TABLE 1
Defined Symbols

Symbol	Definition
λ	the computation task arrival rate
η	the connected period rate
θ	the disconnected period rate
μ	the task service rate at the edge cloud
$t_{\mathrm{g}}^{(i)}$	the generation time for the ith task
$t_{\mathrm{a}}^{(i)}$	the time for task i accessing the edge cloud
$t_{\mathrm{s}}^{(i)}$	the time executing task i at the edge cloud
$t_{\mathrm{c}}^{(i)}$	the time completing task i at the edge cloud
$t_{\mathrm{v}}^{(i)}$	the time retrieving task i from the edge cloud
$T_{\mathrm{o}}^{(i)}$	the offloading time for task i
$T_{\mathrm{q}}^{(i)}$	the queueing time at edge for task i
$T_{\mathrm{e}}^{(i)}$	the execution time at edge for task i
τ_{e}	the average TET at the edge cloud server
$T_{\mathrm{r}}^{(i)}$	the task result retrieval delay for task i
p_C	prob. that user is connected with edge cloud
p_D	prob. that user is disconnected with edge cloud
L	the number of tasks queued at the edge cloud
p_i	the probability of state i, $i \in \{0, 1\}$
Z_1, Z_2	the residual disconnection time
T	the consecutive tasks' generation interval
T'	the every other tasks' generation interval
N_τ	the number of divided queueing delay states
ϵ	the defined queueing delay state interval
S_i	the ith queueing delay state
$\mathbf{P}$	the transition prob. matrix for TQT state
π_i^m	the probability that $T_q^{(m)} \in S_i$
τ	the maximum queueing delay threshold
p_{s}	the edge serviceability
$\hat{p}_{\mathrm{s}}$	the estimated edge serviceability
G	the number of generated tasks in one time

studies [39], [40]. The modeling methodology can be further extended to other task generation patterns. Let λ denote the task arrival rate, and the average task arrival interval is $1/\lambda$, i.e., $\mathrm{E}\,[t_{\mathrm{g}}^{(i)} - t_{\mathrm{g}}^{(i-1)}] = 1/\lambda$.

We define the *task waiting time (TWT)* as the waiting time for a task to be executed at the edge cloud server after its generation. TWT equals to the sum of TOT and TQT. Let $T_{\mathrm{w}}^{(i)}$ denote the TWT for task i, then we have $T_{\mathrm{w}}^{(i)} = T_{\mathrm{o}}^{(i)} + T_{\mathrm{q}}^{(i)}$.

Once the task is allocated with resources, it will be executed immediately. Let $t_{\mathrm{c}}^{(i)}$ denote the time that task i is completed at the edge cloud server. Let $T_{\mathrm{e}}^{(i)}$ denote TET for task i, and we have $T_{\mathrm{e}}^{(i)} = t_{\mathrm{c}}^{(i)} - t_{\mathrm{s}}^{(i)}$.

TET is dominant and non-negligible in the edge computing scenario, and different task execution patterns will affect its value. For example, in the scenario where a user accumulates a certain amount of tasks before offloading them to the edge cloud, T_{e} is identically distributed. In the practical scenarios, T_{e} can be modeled with more general distributions. In the following analysis, let μ denote the task execution rate at the edge cloud, and let τ_{e} denote the average task execution time. Then we have $\tau_{\mathrm{e}} = 1/\mu$.

After the completion of task processing at the edge cloud, the result needs to be retrieved back to the user that generates the task. Let $t_{\mathrm{v}}^{(i)}$ denote the time that task i is retrieved back to the user, and $T_{\mathrm{r}}^{(i)}$ denote TRT for task i. Then we have $T_{\mathrm{r}}^{(i)} = t_{\mathrm{v}}^{(i)} - t_{\mathrm{c}}^{(i)}$. Similar to TOT, TRT is more relevant to the intermittent connection. In other words, this value is mainly determined by the user mobility pattern.

Overall, the symbols used in this paper are summarized in Table 1.

4 Analysis of MEC Performance With Intermittent Connectivity

In this section, we conduct theoretical analysis on the performance of MEC with intermittent connectivity.

4.1 Modeling Intermittent Connectivity in MEC

For the tractability of the following analysis, we assume that the characteristics of link connections for different tasks are independent with each other. Namely, the offloading delay T_{o} is identically distributed for each generated task. Therefore, we have the cumulative distribution function (CDF) of $T_{\mathrm{o}}^{(i)}$ as

$$\begin{aligned} &\Pr\{T_{\mathrm{o}}^{(i)} \le 0\} = \Pr\{T_o = t_0\} = 1 - p_0 \\ &\Pr\{T_{\mathrm{o}}^{(i)} \le t\} = 1 - p_0 + p_0 \Pr\{Z_1 \le t - t_0\}, \text{ for } t > t_0. \end{aligned} \tag{1}$$

where t_0 is the initialization time incurred for offloading link construction, p_0 denotes the probability that the link is disconnected at any given time, and Z_1 represents the time the communication link continuously being at the disconnection state. According to the characteristics of exponential distribution, Z_1 also follows an exponential distribution with parameter θ, i.e., $\Pr\{Z_1 \le t\} = 1 - e^{-\theta t}, t \ge 0$. Then we have

$$\Pr\{T_{\mathrm{o}}^{(i)} \le t\} = 1 - p_0 e^{-\theta t}, \quad t \ge 0. \tag{2}$$

Proposition 1. *Let $p_{ij}(t)$ denote the probability of the link connection that belongs to state j at time t given the initial state i at time 0, where $i, j \in \{0, 1\}$. Then we have*

$$p_{10}(t) = p_{00}(t) = \eta/(\eta + \theta), \tag{3}$$

and

$$p_{11}(t) = p_{01}(t) = \theta/(\eta + \theta). \tag{4}$$

Proof. Let $N_{ij}(t)$ denote the number of visits to state j in $(t_{\mathrm{g}}, t_{\mathrm{g}} + t)$ given that the system enters state i at time t_{g}, then $N_{ij}(t)$ does not depend on the start time. Let $\psi_{ij}(t)$ denote the expectation of $N_{ij}(t)$, and we have [41]

$$\psi_{11}(t) = \int_0^t \psi_{01}(t - x) dC(x) \tag{5a}$$

$$\psi_{01}(t) = \int_0^t [1 + \psi_{11}(t - x)] dI(x), \tag{5b}$$

where $C(\cdot)$ denotes the CDF of the connected period, and $I(\cdot)$ denotes the CDF of the disconnected period.

Equations (5a) and (5b) can be used to determine ψ_{11} and ψ_{01}. One way to accomplish this is to take the Laplace-Stieltjes transform of (5) [41], [42]. For $\psi_{ij}(t)$, let

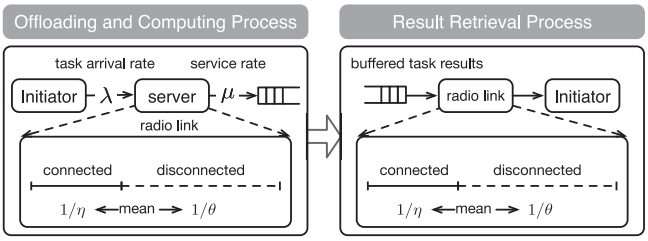


Fig. 3. An illustration of the queueing modeling concept for the task processing time derivation.

$\Psi_{ij}(s)$ denote the Laplace-Stieltjes transform, and we have

$$\Psi_{11}(s) = \Psi_{01}(s)C(s), \tag{6a}$$

$$\Psi_{01}(s) = I(s) + \Psi_{11}(s)I(s). \tag{6b}$$

Combining (6a) and (6b), we have

$$\Psi_{01}(s) = \frac{I(s)}{1 - C(s)I(s)},$$

and

$$\Psi_{11}(s) = \frac{C(s)I(s)}{1 - C(s)I(s)}.$$

Similarly, we can solve $\Psi_{10}(s)$ and $\Psi_{00}(s)$ in terms of $C(s)$ and $I(s)$. Note that

$$N_{10}(t) - N_{11}(t) = \begin{cases} 1, & \text{if } S(t) = 1 \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, we have

$$\begin{aligned} p_{10}(t) &= \mathrm{E}[N_{10}(t) - N_{11}(t)] = \psi_{10}(t) - \psi_{11}(t), \\ p_{11}(t) &= 1 - p_{10}(t). \end{aligned} \tag{7}$$

Similarly, we have $p_{01}(t) = \psi_{01}(t) - \psi_{00}(t)$ and $p_{00}(t) = 1 - p_{01}(t)$.

In general, the Laplace-Stieltjes transforms are difficult to invert. However, it is possible to describe the behavior under large t using asymptotic formulas [42]. In particular, we have

$$\begin{aligned} \psi_{10}(t) &= \frac{t}{1/\eta + 1/\theta} - \frac{1/\eta}{1/\eta + 1/\theta} + \frac{l^{(2)}}{2(1/\eta + 1/\theta)^2} + o(1), \\ \psi_{11}(t) &= \frac{t}{1/\eta + 1/\theta} - 1 + \frac{l^{(2)}}{2(1/\eta + 1/\theta)^2} + o(1), \end{aligned}$$

where $l^{(2)}$ denotes the second moment of H, and we have $H(t) = \int_{\mathrm{g}}^{t} I(t-x)dC(x)$. Then we can obtain

$$p_{10}(t) = p_{00}(t) = \eta/(\eta + \theta),$$

and

$$p_{11}(t) = p_{01}(t) = \theta/(\eta + \theta).$$

This completes the proof of Proposition 1. □

Corollary 1. *To guarantee that the MEC system is stable, i.e., its long run averages exist, we require that*

$$\lambda \mathrm{E}(T_{\mathrm{e}}) < \frac{1}{1 + \eta/\theta}. \tag{8}$$

Proof. Let p_C and p_D denote the fraction of time that the edge cloud is connected and disconnected with the task initiator, respectively. Then we have

$$p_C = \frac{1/\eta}{1/\eta + 1/\theta} = \frac{1}{1 + \eta/\theta}, \tag{9}$$

and

$$p_D = \frac{1}{1 + \theta/\eta}. \tag{10}$$

We require that $\lambda/\mu \leq p_C$ to guarantee that the queueing system is stable. That is, we require that

$$\lambda \mathrm{E}(T_{\mathrm{e}}) < \frac{1}{1 + \eta/\theta}.$$

This completes the proof of Corollary 1. □

A model of an unstable system must use protocols to accommodate the effects of the instability. In the following analysis, we determine the task processing performance in a stable MEC system.

4.2 Analayis of Task Processing in MEC

Recall that $T_{\mathrm{w}} = T_{\mathrm{o}} + T_{\mathrm{q}}$, where T_{o} is mainly affected by the user mobility and T_{q} is influenced by conditions, e.g., the intermittent connectivity and the task generation pattern. We consider a general MEC task processing model as shown in Fig. 3. Then we have the following proposition.

Proposition 2. *Suppose that each task in the queue at the edge cloud server has a mean execution time $\mathrm{E}(T_{\mathrm{e}})$, then we have*

$$\mathrm{E}(T_{\mathrm{w}}) = \frac{\mathrm{E}(T_{\mathrm{e}}) + p_D/\theta}{1 - \lambda \mathrm{E}(T_{\mathrm{e}})}. \tag{11}$$

Proof. Assume that an arriving task finds $\mathrm{E}(L)$ tasks on average in the queue and each of them has a mean execution time $\mathrm{E}(T_{\mathrm{e}})$. With probability p_D, the edge cloud server is already disconnected on arrival, in which case the task has an extra mean delay of $1/\theta$. Then we have [43]

$$\mathrm{E}(T_{\mathrm{w}}) = \mathrm{E}(T_{\mathrm{o}}) + \mathrm{E}(T_{\mathrm{q}}) = \underbrace{p_D \cdot \frac{1}{\theta}}_{\text{Part I}} + \underbrace{(\mathrm{E}(L) + 1)\mathrm{E}(T_{\mathrm{e}})}_{\text{Part II}}, \tag{12}$$

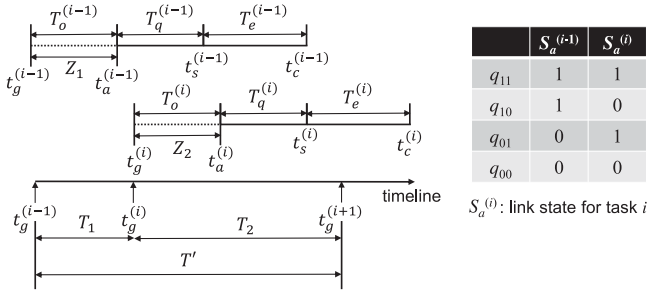
where Part I represents the disconnection time on arrival at the edge cloud, and Part II represents the execution time for the offloaded task at the edge cloud. Then, with the Little's law (i.e., $\mathrm{E}(L) = \lambda \mathrm{E}(T_{\mathrm{w}})$), we have

$$\mathrm{E}(T_{\mathrm{w}}) = \frac{\mathrm{E}(T_{\mathrm{e}}) + p_D/\theta}{1 - \lambda \mathrm{E}(T_{\mathrm{e}})}.$$

This completes the proof of Proposition 2. □

From Proposition 2, we have the following corollary.

Corollary 2. *The expected queueing time at the edge cloud server can be obtained as*



	$S_a^{(i-1)}$	$S_a^{(i)}$
q_{11}	1	1
q_{10}	1	0
q_{01}	0	1
q_{00}	0	0

Fig. 4. Time illustration for two consecutively arrived tasks.

$$\mathrm{E}(T_{\mathrm{q}}) = \frac{\mathrm{E}(T_{\mathrm{e}}) + p_D/\theta}{1 - \lambda \mathrm{E}(T_{\mathrm{e}})} - p_D/\theta. \tag{13}$$

In the previous analysis, the expected task waiting time $\mathrm{E}(T_{\mathrm{w}})$ and the expected task queueing time $\mathrm{E}(T_{\mathrm{q}})$ are given as closed-form expressions. However, in most practical task offloading scenarios, the distribution can provide more insights on the MEC performance, which will be analyzed in the following.

4.3 Analysis on Distribution of Task Queueing Time

We propose a discrete-time Markov chain (DTMC) model to derive the distribution of the task queueing time at the edge cloud server as follows. The modeling contains two steps: first we discuss the dependence between continuously generated tasks, and then we build a DTMC model to model the changing of TQT with time.

4.3.1 Dependence on Continuously Generated Tasks

Tasks are numbered sequentially as $\{0, 1, \ldots, n, n+1, \ldots\}$ according to the order of their generation time. Overall, the delay for task i to be executed at the edge cloud server is the sum of the offloading delay $T_{\mathrm{o}}^{(i)}$ and the queueing delay $T_{\mathrm{q}}^{(i)}$, where the queueing delay is defined as

$$T_{\mathrm{q}}^{(i)} = \max\{0, t_{\mathrm{c}}^{(i-1)} - [t_{\mathrm{g}}^{(i)} + T_{\mathrm{o}}^{(i)}]\}, \tag{14}$$

where $t_{\mathrm{c}}^{(i-1)} = t_{\mathrm{g}}^{(i-1)} + T_{\mathrm{o}}^{(i-1)} + T_{\mathrm{q}}^{(i-1)} + \tau_{\mathrm{e}}$. When a task accesses a idle edge cloud server, it does not need to queue and $T_{\mathrm{q}}^{(i)} = 0$; otherwise, when the previously arrived task at the edge cloud server has not been completed, the newly arrived task need to queue at the edge cloud server and $T_{\mathrm{q}}^{(i)} = t_{\mathrm{c}}^{(i-1)} - [t_{\mathrm{g}}^{(i)} + T_{\mathrm{o}}^{(i)}]$.

Let T denote the generation time interval between two consecutive tasks, and (14) can be rewritten as

$$T_{\mathrm{q}}^{(i)} = \max\{0, T_{\mathrm{q}}^{(i-1)} + \tau_{\mathrm{e}} - T + T_{\mathrm{o}}^{(i-1)} - T_{\mathrm{o}}^{(i)}\}, \tag{15}$$

where $T = t_{\mathrm{g}}^{(i)} - t_{\mathrm{g}}^{(i-1)}$. That is, the queueing at the edge cloud server has cumulative effect.

Given the queueing delay $T_{\mathrm{q}}^{(i-1)}$ for task $(i-1)$, we have the CDF of the queueing delay for task i at the edge cloud server as

$$\Pr\{T_{\mathrm{q}}^{(i)} \le t\} = \Pr\{T_{\mathrm{q}}^{(i-1)} + \tau_{\mathrm{e}} - T - T_{\mathrm{o}}^{(i)} + T_{\mathrm{o}}^{(i-1)} \le t\}, \tag{16}$$

where $t \ge 0$.

To study the factors affecting the queueing delay, we illustrate timelines of two consecutive tasks $(i-1)$ and i as shown in Fig. 4. In Fig. 4a, we can see that $t_{\mathrm{s}}^{(i)} = t_{\mathrm{c}}^{(i-1)}$. That is, task i's computation start time depends on the previous task $(i-1)$'s completion. Fig. 4b shows the state transition probability q_{ij} for the generation time interval between two consecutive tasks, where the current task's generation time lies in state i while the following one transfers to state j. That is, $\{S_a^{(i-1)} = i, S_a^{(i)} = j\}$, $i, j \in \{0, 1\}$, where $S_a^{(i)}$ represents the communication link state for task i. Let q_{ij} denote the probability of the corresponding communication link state combination. Recall that the connection and disconnection durations are both exponentially distributed and the two states are independent, we have $q_{ij} = p_i p_j$, $i, j \in \{0, 1\}$. Let $T_{\Delta}^{(i)}$ denote the queueing delay difference between two consecutive tasks i and $(i-1)$, and we have

$$T_{\Delta}^{(i)} = T_{\mathrm{q}}^{(i)} - T_{\mathrm{q}}^{(i-1)}. \tag{17}$$

Based on $T_{\mathrm{q}}^{(i)}$ defined in (15), we have the CDF of $T_{\Delta}^{(i)}$ as

$$\begin{aligned}
&F_{T_{\Delta}^{(i)}}(t) = \Pr\{T_{\Delta}^{(i)} \le t\} \\
&= \Pr\{T_{\mathrm{q}}^{(i)} - T_{\mathrm{q}}^{(i-1)} \le t\} \\
&= \Pr\{T + T_{\mathrm{o}}^{(i)} - T_{\mathrm{o}}^{(i-1)} \ge \tau_{\mathrm{e}} - t\} \\
&= q_{11} \Pr\{T \ge \tau_{\mathrm{e}} - t\} + q_{00} \Pr\{T + Z_1 - Z_2 \ge \tau_{\mathrm{e}} - t\} \\
&\quad + q_{10} \Pr\{T + Z_1 \ge \tau_{\mathrm{e}} - t\} + q_{01} \Pr\{T - Z_2 \ge \tau_{\mathrm{e}} - t\} \\
&= q_{11} \underbrace{[1 - F_T(\tau_{\mathrm{e}} - t)]}_{\text{Part I}} + q_{10} \underbrace{[1 - F_{D_1}(\tau_{\mathrm{e}} - t)]}_{\text{Part II}} \\
&\quad + q_{01} \underbrace{[1 - F_{D_2}(\tau_{\mathrm{e}} - t)]}_{\text{Part III}} + q_{00} \underbrace{[1 - F_{D_3}(\tau_{\mathrm{e}} - t)]}_{\text{Part IV}},
\end{aligned} \tag{18}$$

where $D_1 = T + Z_1$, $D_2 = T - Z_2$, and $D_3 = T + Z_1 - Z_2$. The distributions of these variables are derived in the following.

For Part I, we have the probability density function (pdf) of T as $f_T(\tau_{\mathrm{e}}) = \lambda \exp(-\lambda \tau_{\mathrm{e}})$, where $T = t_{\mathrm{g}}^{(i+1)} - t_{\mathrm{g}}^{(i)}$.

For Part II, we have $D_1 = T + Z_1$, and its pdf as

$$f_{D_1}(t) = \begin{cases} \theta^2 t e^{\theta t}, & \text{if } \theta = \lambda \\ \theta\lambda(e^{-\lambda t} - e^{-\theta t})/(\theta - \lambda), & \text{otherwise.} \end{cases}$$

For Part III, D_2 follows a distribution of the difference of two exponentially distributed variables ($D_2 = T - Z_2$). Based on the independence assumption, the joint pdf of T and Z_2 can be represented as

$$f_{T,Z_2}(t_1, t_2) = \lambda\theta \exp(-\lambda t_1) \exp(-\theta t_2).$$

Then we have the pdf of D_2 as

$$f_{D_2}(z) = \frac{\lambda\theta}{\lambda + \theta} \begin{cases} e^{\theta t}, & \text{if } t \le 0 \\ e^{-\lambda t}, & \text{if } t > 0. \end{cases}$$

The derivation of $F_{D_2}(t)$ is divided into two cases. When $t \le 0$, we have

$$F_{D_2}(t) = \Pr\{D_1 \le t\} = \tfrac{\lambda}{\lambda+\theta} e^{\theta t},\ t \le 0.$$

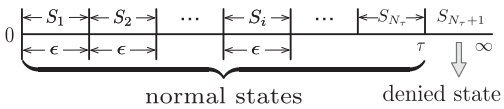


Fig. 5. The queueing delay is represented by N_τ states with equal interval ϵ and one state with infinite length.

Otherwise, when $t > 0$,

$$F_{D_2}(t) = 1 - \int_t^{\infty} \int_0^{t_1 - t} \lambda\theta e^{-\lambda t_1} e^{-\theta t_2} dt_2 dt_1 = 1 - \frac{\theta}{\lambda + \theta} e^{-\lambda t},\ t > 0.$$

For Part IV, we have $D_3 = T + Z_1 - Z_2 = D_1 - Z_2$. Similar to the derivation in part III, we have the joint pdf of T and Z_2 as $f_{D_1,Z_2}(t_1, t_2) = f_{D_1}(t_1) f_{Z_2}(t_2)$. We have

$$f_{D_3}(t) = \begin{cases} \frac{\theta^2\lambda e^{-\lambda t}}{\theta^2 - \lambda^2} - \frac{\theta\lambda e^{-\theta t}}{2(\theta - \lambda)}, & \text{if } t \geq 0 \\ \frac{\theta^2\lambda e^{\theta t}}{\theta^2 - \lambda^2} - \frac{\theta\lambda e^{\theta t}}{2(\theta - \lambda)}, & \text{otherwise.} \end{cases}$$

4.3.2 CDF and pdf of Task Queueing Time

Following (18), we can obtain the dependence on $T_{\mathrm{q}}^{(i)}$ between continuously generated tasks. Based on this modeling, we propose a DTMC based analytical framework to derive the distribution of the task queueing time at the edge cloud server.

As illustrated in Fig. 5, we divide the possible queueing delay into $N_\tau + 1$ states. The first N_τ states S_i ($i = 1, 2, \ldots, N_\tau$) with the same length ϵ ($\epsilon = \tau / N_\tau$) belong to interval $[(i-1)\epsilon, i\epsilon]$. State $S_{N_\tau+1}$ represents that the queueing delay value is greater than τ. The edge service is denied when the queueing delay falls into the state $S_{N_\tau+1}$, which is named as *denied state* and the other states are *normal states*. The transitions between states are illustrated in Fig. 6. Let π_i^m denote the probability that the queueing delay lies in state S_i after m tasks arrival. Initially, π_i^0 represents the probability of initial queueing delay state and we have $\pi_i^0 = \Pr\{T_{\mathrm{q}}^{(0)} \in S_i\}$. Then we have $\pi_i^{m+1} = \pi_i^m \mathbf{P}$, where $\mathbf{P}$ denotes the state transition probability matrix. Let π_i^* denote the convergent probability that the queueing delay stays in the state S_i, then we can achieve the results of CDF and pdf of task queueing time in the following proposition.

Proposition 3. *The CDF of task queueing time is obtained as*

$$\Pr\{T_{\mathrm{q}}^{(i)} \leq \tau\} = 1 - \pi^*_{N_\tau+1}, \tag{19}$$

and the pdf of the task queueing time can be obtained as

$$\Pr\{T_{\mathrm{q}}^{(i)} = \tau\} = \pi^*_{N_{\tau-1}+1} - \pi^*_{N_\tau+1}. \tag{20}$$

Proof. Recall that π_i^* denotes the convergent probability that the queueing delay stays in the state S_i, which satisfies that

$$\pi_i^* = \pi_i^* \mathbf{P}. \tag{21}$$

Based on (21), we have $\pi^*_{N_\tau+1}$ denote the probability that the task queueing time is greater than τ. In complementary, we have the CDF of task queueing time as

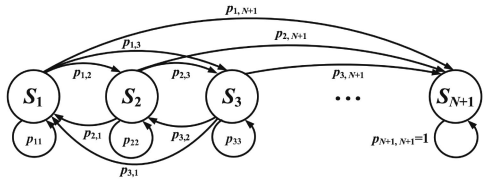


Fig. 6. An illustration of finite state transition.

$$\Pr\{T_{\mathrm{q}}^{(i)} \leq \tau\} = 1 - \pi^*_{N_\tau+1},$$

and the pdf of the task queueing time as

$$\begin{aligned} \Pr\{T_{\mathrm{q}}^{(i)} = \tau\} &= \Pr\{T_{\mathrm{q}}^{(i)} \leq \tau\} - \Pr\{T_{\mathrm{q}}^{(i)} \leq \tau - 1\} \\ &= (1 - \pi^*_{N_\tau+1}) - (1 - \pi^*_{N_{\tau-1}+1}) \\ &= \pi^*_{N_{\tau-1}+1} - \pi^*_{N_\tau+1}. \end{aligned}$$

This completes the derivation of Proposition 3. □

With the distribution in Proposition 3, we can further derive the variance of (or worst-case) task waiting time, e.g., we can estimate the variance of task queueing time as

$$Var(T_{\mathrm{q}}) = 2\int_0^{\infty} \tau(1 - F(\tau))d\tau - \left(\int_0^{\infty} (1 - F(\tau))d\tau\right)^2, \tag{22}$$

where $F(\tau) = \Pr\{T_{\mathrm{q}} \leq \tau\}$. The CDF and pdf of task queueing time need the results of the convergent probability π_i^*, which depends the state transition probability matrix $\mathbf{P}$ as in (21). In the following, we derive the state transition probability for Proposition 3.

4.3.3 State Transition Probability

In the state transition probability matrix $\mathbf{P}$, each element p_{ij} represents the probability that the queueing delay changes from the state S_i to the state S_j between two consecutive task arrivals. When $i, j \leq N_\tau$, we have

$$\begin{aligned} p_{ij} &= \Pr\{T_{\mathrm{q}}^{(i)} \in S_j | T_{\mathrm{q}}^{(i-1)} \in S_i\} \\ &= \Pr\{(j-1)\epsilon \leq T_{\mathrm{q}}^{(i)} < j\epsilon | (i-1)\epsilon \leq T_{\mathrm{q}}^{(i-1)} < i\epsilon\} \\ &= \int_{(j-1)\epsilon}^{j\epsilon} \left[\int_{(i-1)\epsilon}^{i\epsilon} f_{T_\Delta^{(i)}}(y - x)dx\right] dy. \end{aligned} \tag{23}$$

From the definition of the queueing delay difference in (17), we can conclude that the transition probability p_{ij} is not affected by the current state S_i. Instead, p_{ij} is determined by the interval between the states S_i and S_j, that is, we have $p_{ij} = p_{mn}$ when $j - i = n - m$ holds. Therefore, we can estimate p_{ij} as

$$\begin{aligned} p_{ij} &= \int_{(j-i)\epsilon}^{(j-i+1)\epsilon} f_{T_\Delta^{(i)}}(t)dt \\ &= F_{T_\Delta^{(i)}}[(j - i + 1)\epsilon] - F_{T_\Delta^{(i)}}[(j - i)\epsilon]. \end{aligned} \tag{24}$$

Two special cases should be highlighted in deriving p_{ij} when $i \neq N_\tau + 1$, which include:

- When $i \leq N_\tau$ and $j = N_\tau + 1$, we have

$$\begin{aligned} p_{ij} &= \Pr\{T_{\mathrm{q}}^{(i)} \in S_j | T_{\mathrm{q}}^{(i-1)} \in S_i\} \\ &= \Pr\{T_{\mathrm{q}}^{(i)} \geq N_\tau \epsilon | (i-1)\epsilon \leq T_{\mathrm{q}}^{(i-1)} < i\epsilon\} \\ &= 1 - F_{T_\Delta^{(i)}}(\tau). \end{aligned} \tag{25}$$

- When $i \leq N_\tau$ and $j = 1$, the derivation of p_{ij} can be split into two sub-cases:
  - The queueing delay for the next step is greater than zero but less than ϵ, and we have

$$\begin{aligned} p_{i1}^a &= \Pr\{0 < T_{\mathrm{q}}^{(i)} < \epsilon | (i-1)\epsilon \leq T_{\mathrm{q}}^{(i-1)} < i\epsilon\} \\ &= F_{T_\Delta^{(i)}}[(2-i)\epsilon] - F_{T_\Delta^{(i)}}[(1-i)\epsilon]. \end{aligned}$$

  - In another sub-case, the task i will not need to be queued at the edge cloud, and we have

$$\begin{aligned} p_{i1}^b &= \Pr\{T_{\mathrm{q}}^{(i)} = 0 | (i-1)\epsilon \leq T_{\mathrm{q}}^{(i-1)} < i\epsilon\} \\ &= F_{T_\Delta^{(i)}}[(1-i)\epsilon]. \end{aligned}$$

With these two sub-cases, we have $p_{i1} = p_{i1}^a + p_{i1}^b$.

When the queueing delay state for task i stays in $S_{N_\tau+1}$, the aforementioned method might not be directly applicable. This is because task i will not access the edge service when it stays in the denied state. Assume that tasks $(i-1)$, i, and $(i+1)$ belong to state S_i, $S_{N_\tau+1}$, and S_j, separately. Following our defined strategy, task i will be denied to access the edge cloud since its queueing delay is greater than τ. The transition from the denied state to a normal state cannot be easily estimated since no such "queueing delay" metric can be defined for task i. As a solution, the state for the previous state should be referred when task i is denied service, and the transition probability can be re-written as

$$p_{N_\tau+1,j} = \frac{\sum_{i=1}^{N_\tau} p_{i,N_\tau+1} \cdot p_{i,N_\tau+1,j}}{\sum_{i=1}^{N_\tau} p_{i,N_\tau+1}}, \tag{26}$$

where $p_{i,N_\tau+1,j}$ denotes the transition probability from state S_i to $S_{N_\tau+1}$ to S_j, where $i, j \in \{1, 2, \ldots, N_\tau\}$. The summation operation is to clarify the state transmitted before the denied state $S_{N_\tau+1}$, which is needed to calculate the transition probability from the denied state to a normal state. Following the principle in (23)-(4.3.3), and with the queueing delay difference characteristic derived in (28), we have

$$p_{i,N_\tau+1,j} = \begin{cases} F_{T_{2\Delta}^{(i)}}[(2-i)\epsilon], & j = 1 \\ 1 - F_{T_{2\Delta}^{(i)}}(\tau), & j = N_\tau + 1 \\ F_{T_{2\Delta}^{(i)}}[(j-i+1)\epsilon] - F_{T_{2\Delta}^{(i)}}[(j-i)\epsilon], & \text{otherwise}, \end{cases} \tag{27}$$

where $T_{2\Delta}^{(i+1)}$ denotes the queueing delay difference for tasks $(i-1)$ and $(i+1)$, where task i is denied with the edge service, that is, $T_{\mathrm{q}}^{(i)} > \tau$. We have the CDF of the queueing delay difference $T_{2\Delta}^{(i+1)}$ as

$$\begin{aligned} \Pr\{T_{2\Delta}^{(i+1)} \leq t\} &= \Pr\{T_{\mathrm{q}}^{(i+1)} - T_{\mathrm{q}}^{(i-1)} \leq t\} \\ &= \Pr\{T' + T_{\mathrm{o}}^{(i+1)} - T_{\mathrm{o}}^{(i-1)} \geq \tau_{\mathrm{e}} - t\}, \end{aligned} \tag{28}$$

where T' denotes the time interval between every other tasks, e.g., tasks $(i-1)$ and $(i+1)$. That is, $T' = T_1 + T_2$, where T_1 and T_2 represent the task generation time interval between the three consecutive tasks $(i-1)$, n and $(i+1)$ as shown in Fig. 4. As the summation of two independent and identically exponentially distributed variables, T' is Gamma distributed and we have $F_{T'}(t) = 1 - e^{-\lambda t} - \lambda t e^{-\lambda t}$. Similarly, we can obtain the CDF and pdf expressions for the queueing delay difference $T_{2\Delta}^{(i+1)}$.

With the derivations in (23), (24), (25), (26), (27), and (28), we can achieve tractable results on state transition probability p_{ij} in **P**.

4.3.4 Analysis of Edge Serviceability

Besides the task processing time, we also define the edge serviceability to show the processing performance of the edge cloud server.

Definition 1. Edge serviceability *is defined as the probability that a newly arrived task's queueing time at the edge cloud is less than τ, where τ denotes the queueing delay threshold above which the task is unable to finish in time.*

Let p_{s} denote the edge serviceability, and we have

$$p_{\mathrm{s}} = \Pr\{T_{\mathrm{q}}^{(i)} \leq \tau\}, \forall n \in \{m^*, m^*+1, \cdots\}, \tag{29}$$

where m^* represents the task index when the queueing delay at the edge cloud server is convergent.

Then we can estimate the edge serviceability by

$$\hat{p}_{\mathrm{s}} = \Pr\{T_{\mathrm{q}}^{(m^*)} \leq \tau\} = \sum_{i=1}^{N_\tau} \pi_i^*. \tag{30}$$

When a node with computation tasks to be offloaded connects with an edge cloud server, the edge serviceability metric can help make the task offloading decision.

4.4 Extension to MEC with Group Task Arrival

In a practical MEC system, computation tasks generated by users may arrive at the edge cloud server in groups. Suppose that the arrival of group tasks follows a Poisson process with rate λ. The group size is denoted by the random variable G with probability distribution

$$g_k = \Pr\{G = k\}, \quad k = 0, 1, 2, \cdots. \tag{31}$$

Note that we also allow zero-size groups to arrive. As an extension of Proposition 2, we have the following proposition.

Proposition 4. *In a stable MEC system with group arrived tasks, we have the average task waiting time as*

$$\mathrm{E}(T_{\mathrm{w}}) = \frac{p_D/\theta + \lambda\lambda_g/\mu^2 + \lambda_g/(2\mu)}{1 - \lambda\lambda_g/\mu}. \tag{32}$$

Proof. Let r_k denote the probability that the task is the kth task served in his group. To find r_k we first determine the probability h_n that the task is a member of a group of size n. Since it is more likely that the target task belongs to a large group than to a small one, it follows that h_n is proportional to the group size n as well as the frequency of

such groups. Thus we have

$$h_n = nCg_n, \tag{33}$$

where C is a constant to normalize this distribution as

$$C^{-1} = \sum_{n=1}^{\infty} ng_n = \mathrm{E}(G). \tag{34}$$

Hence

$$h_n = \frac{ng_n}{\mathrm{E}(G)}, \quad n = 1, 2, \cdots, \tag{35}$$

Given that the target task is a member of a group of size n, it will be with probability $1/n$ the kth task in its group going into service ($n \geq k$). Then we have

$$r_k = \sum_{n=k}^{\infty} h_n \cdot \frac{1}{n} = \frac{1}{\mathrm{E}(G)} \sum_{n=k}^{\infty} g_n. \tag{36}$$

Then, we have the expected task waiting time at the edge cloud server as

$$\mathrm{E}(T_\mathrm{w}) = \underbrace{p_D \cdot \frac{1}{\theta}}_{\text{Part I}} + \underbrace{(\rho + \mathrm{E}(L))\mathrm{E}(T_\mathrm{e})}_{\text{Part II}} + \underbrace{\sum_{k=1}^{\infty} r_k(k-1)\mathrm{E}(T_\mathrm{e})}_{\text{Part III}}, \tag{37}$$

where ρ is the server utilization, so $\rho = \lambda \mathrm{E}(G)\mathrm{E}(T_\mathrm{e})$; Part I denotes the disconnected time period between the task initiator and the edge cloud; Part II corresponds to the mean waiting time of the whole group; Part III represents the mean waiting time due to the servicing of members in his own group, and we have

$$\begin{aligned} \sum_{k=1}^{\infty} r_k(k-1)\mathrm{E}(T_\mathrm{e}) &= \frac{\mathrm{E}(T_\mathrm{e})}{\mathrm{E}(G)} \sum_{k=1}^{\infty} \sum_{n=k}^{\infty} g_n(k-1) \\ &= \frac{\mathrm{E}(T_\mathrm{e})}{\mathrm{E}(G)} \sum_{n=1}^{\infty} \sum_{k=1}^{n} g_n(k-1) \\ &= \frac{\mathrm{E}(T_\mathrm{e})}{\mathrm{E}(G)} \sum_{n=1}^{\infty} \frac{1}{2} n(i-1) g_n \\ &= \frac{[\mathrm{E}(G^2) - \mathrm{E}(G)]\mathrm{E}(T_\mathrm{e})}{2\mathrm{E}(G)}. \end{aligned} \tag{38}$$

Using Little's law (i.e., $\mathrm{E}(L) = \lambda \mathrm{E}(G)\mathrm{E}(T_\mathrm{w})$), we have

$$\mathrm{E}(T_\mathrm{w}) = \frac{p_D/\theta + \lambda \mathrm{E}(G)[\mathrm{E}(T_\mathrm{e})]^2 + [\mathrm{E}(G^2)/\mathrm{E}(G) - 1]\mathrm{E}(T_\mathrm{e})/2}{1 - \lambda \mathrm{E}(G)\mathrm{E}(T_\mathrm{e})}. \tag{39}$$

Recall that the group size is Poisson distributed, i.e., $g_k = \lambda_g^k e^{-\lambda_g}/k!$, where λ_g denotes the average group size. Moreover, since $\mathrm{E}(T_\mathrm{e}) = 1/\mu$, we have

$$\mathrm{E}(T_\mathrm{w}) = \frac{p_D/\theta + \lambda\lambda_g/\mu^2 + \lambda_g/(2\mu)}{1 - \lambda\lambda_g/\mu}.$$

This completes the proof for Proposition 4. □

5 Experimental Validation

In this section, we perform a series of simulations to validate the efficiency of our analytical models.

5.1 Experiment Settings

For the purpose of model validation, we build a discrete-event simulator to simulate the scenarios of MEC with various network conditions. The generation of computation tasks is assumed to follow a Poisson distribution with parameter λ, which is set to 0.001 event per second according to [44]. In our experiments, the capacity of each edge cloud server is set as 1 Giga CPU cycles per second, and each task has the computation requirement of ten to fifty Giga CPU cycles. Each user can generate at most one active task of any application in a slot, and each edge cloud server can host at most one application task in any slot in the simulation. Further, a user does not offload its task to the edge cloud server that has buffered tasks to be executed, in order to improve the processing efficiency on the computation resource limited scenarios. We repeat each experiment multiple times and calculate the average value to increase the confidence interval.

In the following, we verify the performance of the proposed analytical models on both synthetic and real-world user mobility traces.

5.2 Validation on Synthetic Traces

We first consider a setting with synthetic user movements, where the MEC service links are synthetically generated. Similar to the assumptions made in [28], [29], [30], we assume that the link connection time follows an exponential distribution with parameter η, and the link disconnection time follows an exponential distribution with parameter θ. Different settings on these values simulate various network conditions, and we initially set $\eta = 0.5$ and $\theta = 0.05$ times per second.

5.2.1 Expectation of Task Processing Time

Figs. 7 and 8 illustrate the average task processing time and the average task queueing time versus different task arrival rate λ and service rate μ. It can be found that the analytical results match well with that from the simulations. Furthermore, both the task processing time and the task queueing time increase with the task arrival rate λ, and conversely decrease with the task service rate μ. This is reasonable

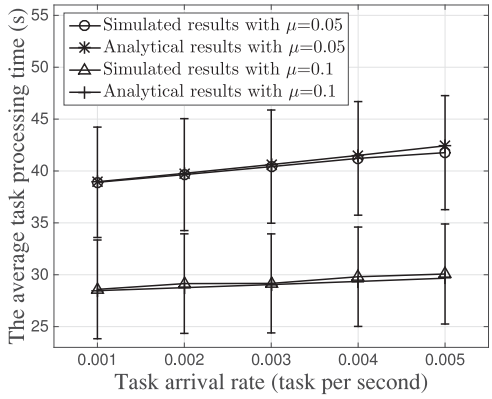


Fig. 7. The average task processing time versus different task arrival rate λ and service rate μ, where $1/\eta = 2$ s and $1/\theta = 20$ s.

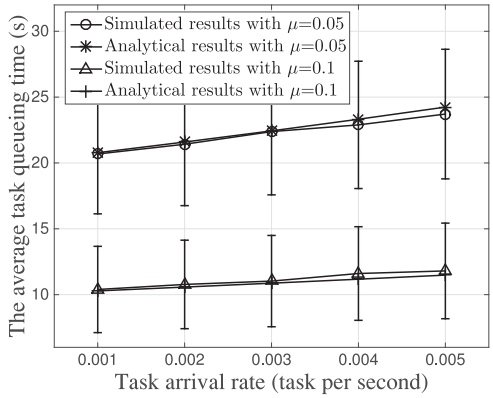


Fig. 8. The average task queueing time $\mathsf{E}(T_{\mathrm{q}})$ versus different task arrival rate λ and service rate μ, where $1/\eta = 2$ s and $1/\theta = 20$ s.

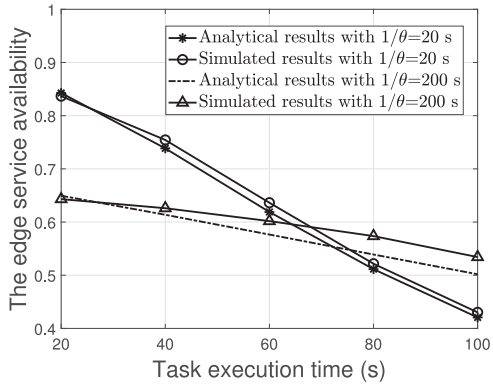


Fig. 10. The edge serviceability in an MEC system, where $1/\theta = 20$ s, $1/\eta = 10$ s, $\tau = 10$ s, and $N_s = 10$.

because an increased task arrival rate λ creates more burden on the MEC system, while the decrease of task service rate μ slows down the efficiency of the MEC system.

5.2.2 Distribution of Task Processing Time

Fig. 9 illustrates the comparison on the task queueing time versus the task generation rate λ. Again, the simulation results for the task queueing probability match the analytical results well, which verifies the accuracy of our analytical model. In addition, it can be observed that the task queueing probability increases with λ. The reason is that a small value of λ will enlarge the generation interval between two consecutive tasks, resulting in a relatively low probability of task queueing. Moreover, the task queueing time will not exceed 50 seconds with probability 95 percent under our synthetic simulation settings.

Fig. 10 illustrates a comparison of the edge serviceability p_{s} in the offloading process from our proposed analytical model and the synthetic simulations. As well, the analytical results show a good match with the simulation results, which verifies the accuracy of our analysis on the edge serviceability. As expected, we can see that the edge serviceability decreases with the task execution time τ_{e} at the edge cloud. This is due to the fact that the task will be more likely to be queued with a higher task execution time τ_{e} at the edge cloud. Moreover, when θ varies from 0.05 to 0.005, the edge serviceability declines dramatically when τ_{e} is small. Instead, when τ_{e} is greater than 70 seconds, the edge serviceability is even better than that with a more frequent link connection. This is because that a large task execution time τ_{e} will reduce the proportion of the task queueing time in the overall task processing time, which introduces a higher p_{s} given a constant connection rate η.

5.2.3 Cases with Group Task Arrivals

Fig. 11 verifies the derived analytical results on the average task processing time at the edge cloud through comparison with simulation results. Again, we can observe a good match between the analytical results and the simulation results. When the initiated tasks on the device are generated in groups (e.g., $\lambda_g = 2$), the task processing time is greater than that in the single task arrival scenario ($\lambda_g = 1$). This is reasonable because that the actual task arrival rate is doubled when $\lambda_g = 2$. Note that the task processing time in the group arrival case is lower than the twice of that in the single arrival case. This is due to that the tasks other than the first task in the group will not be affected by the intermittent connectivity.

5.3 Validation on Real-World Traces

Besides studies on synthetical traces, we also conduct studies on real-world traces measured in [45]. In the real-world traces, the maximum number of edge cloud servers (i.e., base stations) is increased to 20, and the number of users is at most 78. The service range of each edge cloud server is set as 30 meters, and a user can access edge computing resources

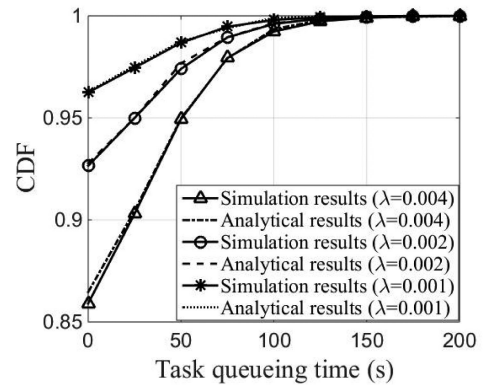


Fig. 9. The CDF of the task queueing time T_q with uniform distributed task execution time assumption, where $\tau_{\mathrm{e}} = 50$ s and $1/\theta = 20$ s.

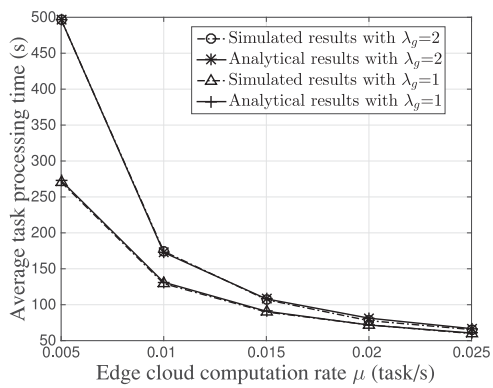
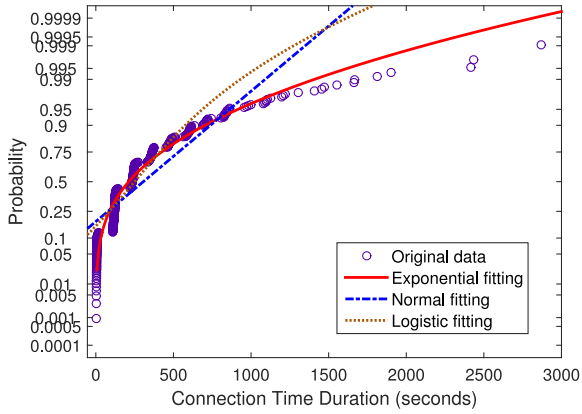
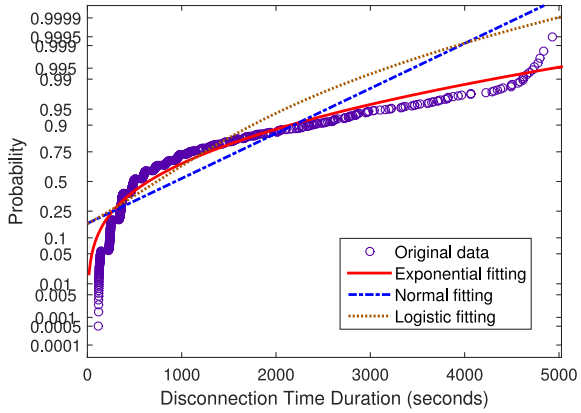


Fig. 11. The task processing time at the edge cloud with different group size, where $\lambda = 0.001$ tasks per second, $1/\eta = 2$ s, and $1/\theta = 20$ s.



(a) Connection statistics



(b) Disconnection statistics

Fig. 12. The probability density of the connection/disconnection periods.

when its distance to the edge cloud server is less than 30 meters. For the tractability of the following validation, we eliminate the users who are disconnected with the edge cloud server for a long time, e.g., 5000 seconds. The other parameter settings are identical to that for the simulations on synthetic user traces.

5.3.1 User-Server Contact Patterns

Based on the real-world traces, we plot contact patterns between mobile devices (users) and their access points (edge cloud servers) to study the performance of the MEC system. As shown in Fig. 12, link connection/disconnection time period can be well captured by exponential distribution, which implies that link connection/disconnection behaviors can be modeled as a Poisson process as assumed in the synthetic simulations.

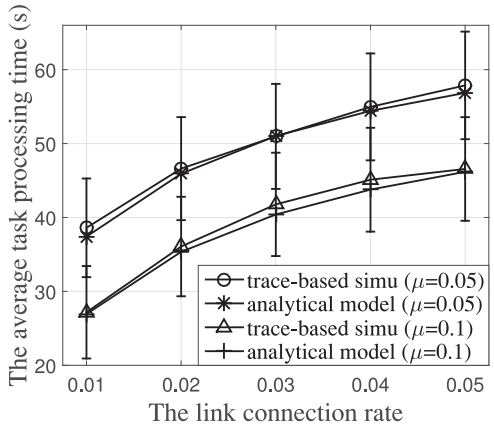


Fig. 13. The average task processing time versus the link connection rate in real-world trace based simulation.

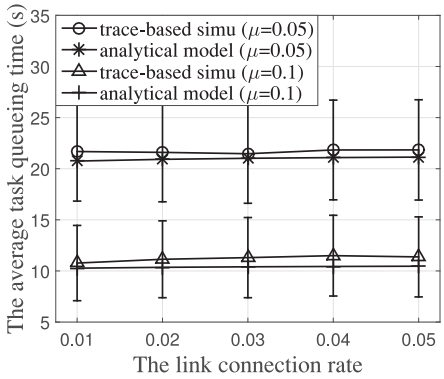


Fig. 14. The average task queueing time $\mathrm{E}(T_{\mathrm{q}})$ versus the link connection rate in real-world trace based simulation.

5.3.2 Expectation of Task Processing Time

Figs. 13 and 14 illustrate the average task processing time and the average task queueing time versus different task arrival rate λ and link connection rate η. Statistically, with the increased number of edge cloud servers, we have an increased value on the link connection rate η and a decreased value on the link disconnection rate θ. Again, the analytical results match well with that from the simulations. Furthermore, both the task processing time and the task queueing time increase with the task arrival rate λ, and the link connection rate η. This is reasonable since an increased task arrival rate λ puts more burden on the MEC system, while a long link connection time period $1/\eta$ also improves the efficiency of the MEC system.

5.3.3 Distribution of Task Processing Time

Fig. 15 presents the task queueing time versus the link connection rate η measured from real-world traces. Again, the simulation results for the task queueing probability match the analytical results well, which verifies the accuracy of our analytical model. In addition, the task queueing probability decreases with the value of link connection rate η. The reason is that a large value of the link connection time $1/\eta$

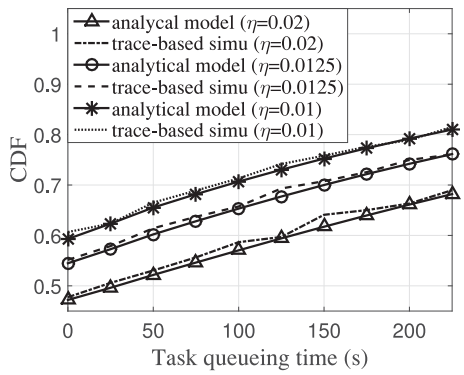


Fig. 15. Comparison on the CDF of the task queueing time T_q, where $\tau_{\mathrm{e}} = 50$ s and $\eta = 0.01, 0.0125, 0.02$ measured from real-world traces.

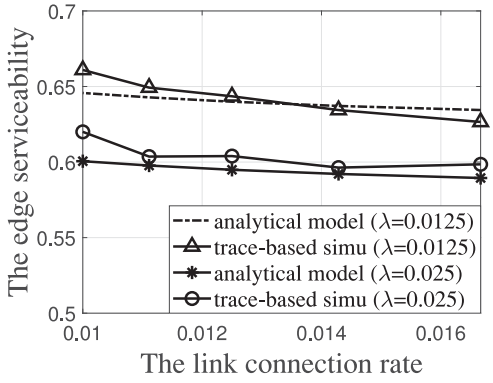


Fig. 16. The edge serviceability versus the link connection rate.

will promote the probability that a task could be finished in one contact period, resulting in a relatively low probability of task queueing. Moreover, compared with the results in Fig. 9, we find a greater task queueing time, which might be due to a relatively long link disconnection time in real-world traces.

Fig. 16 illustrates a comparison of the edge serviceability p_s versus the link connection rate η. As well, the analytical results show a good match with the simulation results, which verifies the accuracy of our analysis on the edge serviceability. As expected, we can see that the edge serviceability decreases with the connection rate for the link between the user and the edge cloud server. This is due to the fact that the task will be more likely to be queued with a shorter link connection time period.

5.4 Discussion

One major function of the proposed model is to allow a mobile user to obtain an accurate estimation on the performance of task processing before conducting actual task offloading. The results can help a mobile user determine where is the best venue to perform computation (e.g., local device, edge cloud server, or remote cloud server) and how to better schedule all the tasks. In case that the communication link to the edge cloud server is congested or intermittently connected, a mobile user can easily quantify the impact of network conditions beforehand with our model, and make offloading decisions accordingly.

Contrast to the previous models, our model is more complete, which considers all stages of task processing in MEC (including task generation, task offloading, task queueing, task execution, and task result retrieval). Moreover, instead of conducting estimation of task processing time by heuristics, our analytical model is more parameterized, which allows us to explore more design dimensions in different scenarios. The model itself can also be integrated with existing MEC task offloading algorithms and provide guidelines for offloading decisions.

In terms of the practicality and importance of our model, we believe that our paper makes important contributions in the following aspects: (1) our analytical model allows system designers to directly explore the relationship between MEC performance measures (e.g., task processing time) and network conditions. Otherwise, it requires extensive simulations to examine all the possible scenarios. (2) our model provides guidelines on how to set a few key system parameters when performing task offloading (e.g., timeout value for task retrieval, task re-request threshold). Our model enables us to derive the probability that the task result could be retrieved back within a time period.

6 Conclusions

In this paper, we developed an analytical framework to investigate the influence of various network conditions on the performance of MEC. In particular, we studied the impact of intermittent connectivity and random task generation. The proposed model allows us to derive the closed-form expressions on the task processing time and the model can be extended to account for the case with group task arrivals. We also derived the distribution of task queueing time by using a discrete-time Markov chain. We conducted simulations to validate the proposed theoretical models with both synthetic and real-world mobility traces. Our theoretical results can provide useful guidelines on the design of efficient task offloading algorithms in real MEC systems.

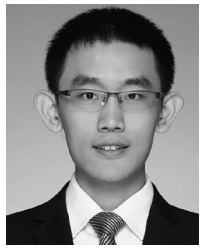
Acknowledgments

This work is supported in part by the National Key R&D Program of China (No. 2018YFB0204100), in part by the Nature Science Foundation of China (Nos. 61802452, 61572538, and U1801266), in part by the Natural Science Foundation of Guangdong (No. 2018A030310079), in part by the Guangdong Special Support Program (No. 2017TX04X148), and in part by the China Postdoctoral Science Foundation (No. 2018M631025).

References

[1] W. Shi and S. Dustdar, "The promise of edge computing," *Comput.*, vol. 49, no. 5, pp. 78–81, May 2016.
[2] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[3] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *CoRR*, vol. abs/1809.07857, pp. 1–10, 2018.
[4] L. Wang, L. Jiao, T. He, J. Li, and M. Muhlhuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 468–476.
[5] P. Ren, X. Qiao, J. Chen, and S. Dustdar, "Mobile edge computing: A booster for the practical provisioning approach of web-based augmented reality," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 349–350.
[6] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.
[7] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 965–984, Apr. 2018.
[8] L. Xu, X. Guo, Y. Lu, S. Li, O. C. Au, and L. Fang, "A low latency cloud gaming system using edge preserved image homography," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2014, pp. 1–6.
[9] K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," in *Proc. IEEE Int. Conf. Fog Mobile Edge Comput.*, May 2017, pp. 68–73.
[10] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 181–11 191, Nov. 2018.
[11] Z. Deng, Y. Zhou, D. Wu, G. Ye, M. Chen, and L. Xiao, "Utility maximization of cloud-based in-car video recording over vehicular access networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 5213–5226, Dec. 2018.

[12] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in *Proc. IEEE 7th Int. Symp. Service-Oriented Syst. Eng.*, Mar. 2013, pp. 320–323.
[13] M. Sheraz, F. Wang, and Y. Li, "Enhancing mobile user performance through data caching over edge computing wireless networks," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf.*, Jun. 2018, pp. 1148–1153.
[14] G. Lewis, S. Echeverra, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *Proc. IEEE Military Commun. Conf.*, Oct. 2014, pp. 1440–1446.
[15] Z. Cao, M. French, R. Krishnan, J. Ng, D. Talmage, and Q. Zhang, "Content-oriented mobile edge technology system integration framework and field evaluation," in *Proc. IEEE Military Commun. Conf.*, Oct. 2014, pp. 1405–1410.
[16] Z. Lu, J. Zhao, Y. Wu, and G. Cao, "Task allocation for mobile cloud computing in heterogeneous wireless networks," in *Proc. Int. Conf. Comput. Commun. and Netw.*, Aug. 2015, pp. 1–9.
[17] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[18] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[19] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A dynamic service-migration mechanism in edge cognitive computing," *ACM Trans. Internet Technol.*, Vol. 19, No. 2, pp. 1–15, 2018.
[20] N. Fernando, S. W. Loke, and W. Rahayu, "Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 329–343, Apr.-Jun. 2019.
[21] D. T. Nguyen, L. B. Le, and V. K. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Trans. Cloud Comput.*, pp. 1–15, 2018, doi: 10.1109/TCC.2018.2844379.
[22] R. Mahajan, J. Zahorjan, and B. Zill, "Understanding WiFi-based connectivity from moving vehicles," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 321–326.
[23] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc. ACM Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 209–222.
[24] P. Deshpande, X. Hou, and S. R. Das, "Performance comparison of 3G and metro-scale WiFi for vehicular network access," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 301–307.
[25] B. Li, Z. Liu, Y. Pei, and H. Wu, "Mobility prediction based opportunistic computational offloading for mobile device cloud," in *Proc. IEEE Int. Conf. Comput. Sci. Eng.*, Dec. 2014, pp. 786–792.
[26] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2014, pp. 287–296.
[27] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.
[28] X. Zhang and G. Cao, "Efficient data forwarding in mobile social networks with diverse connectivity characteristics," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2014, pp. 31–40.
[29] Z. Lu, X. Sun, and T. L. Porta, "Cooperative data offloading in opportunistic mobile networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[30] N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment: Waiting or downloading now?" in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[31] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1779–1782, Oct. 2014.
[32] C. Wang, Y. Li, D. Jin, and S. Chen, "On the serviceability of mobile vehicular cloudlets in a large-scale urban environment," *IEEE Intell. Transp. Syst. Mag.*, vol. 17, no. 10, pp. 2960–2970, Oct. 2016.
[33] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?" in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2014, pp. 1060–1068.
[34] J. P. Champati and B. Liang, "Semi-online algorithms for computational task offloading with communication delay," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1189–1201, Apr. 2017.
[35] Z. Chen, R. Klatzky, D. Siewiorek, M. Satyanarayanan, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, and K. Ha, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. ACM/IEEE Symp. Edge Comput.*, 2017, pp. 1–14.
[36] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Learning driven computation offloading for asymmetrically informed edge computing," *IEEE Trans. Parallel Distrib. Syst.*, pp. 1–14, 2019, doi: 10.1109/TPDS.2019.2893925.
[37] T. P. Pham, J. J. Durillo, and T. Fahringer, "Predicting workflow task execution time in the cloud using a two-stage machine learning approach," *IEEE Trans. Cloud Comput.*, pp. 1–12, 2018, doi: 10.1109/TCC.2017.2732344.
[38] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Jul.-Sep. 2017.
[39] Y. Chi, H. Hacígümüş, W.-P. Hsiung, and J. F. Naughton, "Distribution-based query scheduling," *Proc. VLDB Endowment*, vol. 6, no. 9, pp. 673–684, Jul. 2013.
[40] H. Cao and M. Lin, "Mining smartphone data for app usage prediction and recommendations: A survey," *Pervasive Mobile Comput.*, vol. 37, pp. 1–22, 2017.
[41] R. E. Barlow and L. C. Hunter, "Reliability analysis of a one-unit system," *Operations Res.*, vol. 9, no. 2, pp. 200–208, Mar. 1961.
[42] W. L. Smith, "Asymptotic renewal theorems," *Proc. Roy. Soc. Edinburgh. Section A. Math. Phys. Sci.*, vol. 64, 1953, pp. 9–48.
[43] I. Adan and J. Resing, *Queueing Systems*. The Netherlands: Eindhoven Univ. Technol., 2015, pp. 1–182.
[44] M. Harchol-Balter, M. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," in *Proc. 10th Int. Conf. Comput. Perform. Eval.: Modelling Techn. Tools*, 1998, pp. 231–242.
[45] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 606–620, Jun. 2007.



Miao Hu (S'13-M'17) received the BS and PhD degrees in communication engineering from Beijing Jiaotong University, Beijing, China, in 2011 and 2017, respectively. He is currently an associate research fellow in computer science with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China. From September 2014 to September 2015, he was a visiting scholar with the Pennsylvania State University, PA. His research interests include edge computing, multimedia system, and software defined network. He is a member of the IEEE.

Di Wu (M'06-SM'17) received the BS degree from the University of Science and Technology of China, Hefei, China, in 2000, the MS degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and the PhD degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2007. He was a post-doctoral researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, from 2007 to 2009, advised by Prof. K. W. Ross. He is currently a professor and the assistant dean of the School of Data and Computer Science with Sun Yat-sen University, Guangzhou, China. His research interests include cloud computing, multimedia communication, Internet measurement, and network security. He was a co-recipient of the IEEE INFOCOM 2009 Best Paper Award. He has served as an editor of the Journal of Telecommunication Systems (Springer), the Journal of Communications and Networks, Peer-to-Peer Networking and Applications (Springer), Security and Communication Networks (Wiley), and the KSII Transactions on Internet and Information Systems, and a guest editor of the *IEEE Transactions on Circuits and Systems* for Video Technology. He has also served as the MSIG chair of the Multimedia Communications Technical Committee in the IEEE Communications Society from 2014 to 2016. He served as the TPC co-chair of the IEEE Global Communications Conference - Cloud Computing Systems, and Networks, and Applications in 2014, the Chair of the CCF Young Computer Scientists and Engineers Forum - Guangzhou from 2014 to 2015, and a member of the Council of China Computer Federation. He is a senior member of the IEEE.

Weigang Wu received the BSc and MSc degrees from Xi'an Jiaotong University, China, in 1998 and 2003, and the PhD degree in computer science from Hong Kong Polytechnic University, in 2007. He is currently a full professor with the school of data and computer science, Sun Yat-sen University, China. His research interests include distributed systems, cloud computing, big data processing, and blockchain. He has published more than 100 papers in major conferences and journals. He has served as an organizing/program committee member for many international conferences.

Julian Cheng (S'96-M'04-SM'13) received the BEng degree (Hons.) in electrical engineering from the University of Victoria, Victoria, BC, Canada, in 1995, the MSc(Eng.) degree in mathematics and engineering from Queens University, Kingston, ON, Canada, in 1997, and the PhD degree in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2003. He is currently a full professor with the School of Engineering, Faculty of Applied Science, The University of British Columbia, Kelowna, BC, Canada. He was with Bell Northern Research and NORTEL Networks. His current research interests include digital communications over fading channels, statistical signal processing for wireless applications, optical wireless communications, and 5G wireless networks. He was the co-chair of the 12th Canadian Workshop on Information Theory in 2011, the 28th Biennial Symposium on Communications in 2016, and the 6th EAI International Conference on Game Theory for Networks (GameNets 216). He currently serves as an area editor for the *IEEE Transactions on Communications*, and he was a past associate editor of the *IEEE Trsactions on Communications*, the *IEEE Transactions on Wireless Communications*, the *IEEE Communications Letters*, and the *IEEE Access*. He served as a guest editor for a Special Issue of the IEEE Journal On Selected Areas In Communications on Optical Wireless Communications. He is also a Registered Professional Engineer with the Province of British Columbia, Canada. Currently he serves as the President of the Canadian Society of Information Theory as well as the Secretary for the Radio Communications Technical Committee of IEEE Communications Society. He is a senior member of the IEEE.

Min Chen (SM'09) has been a full professor with the School of Computer Science and Technology at the Huazhong University of Science and Technology since 2012. He is the chair of the IEEE Computer Society STC on Big Data. His Google Scholars Citations reached more than 17,000 with an h-index of 64. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017, and the IEEE Jack Neubauer Memorial Award in 2019. His research focuses on IoT sensing, 5G networks, healthcare big data, and cognitive computing. He is a senior member of the IEEE.