Queue-Length-based Offloading for Delay Sensitive Applications in Federated Cloud-Edge-Fog Systems

Ren-Hung Hwang[1], Yuan-Cheng Lai[2], and Ying-Dar Lin[3]
[1]College of Artificial Intelligence, National Yang Ming Chiao Tung University, Taiwan
[2]Department of Information Management, National Taiwan University of Science and Technology, Taiwan
[3]Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan
rhhwang@nycu.edu.tw, laiyc@cs.ntust.edu.tw, ydlin@cs.nctu.edu.tw

2024 IEEE 21st Consumer Communications & Networking Conference (CCNC) | 979-8-3503-0457-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/CCNC51664.2024.10454697

Abstract—**Delay-sensitive applications demand ultra-low latency, which can be achieved by leveraging edge and fog computing to provide computation services closer to users. However, the server capacity limitation of edge and fog computing necessitates offloading to balance the computation load across cloud, edge, and fog servers. While previous works typically focus on the average delay of users' requests and employ probabilistic offloading schemes based on certain probabilities, our study introduces a novel approach that considers the QoS violating probability and offloads users' requests based on the queue length of computing servers. We propose an approximate model with closed-form solutions to determine the near-optimal offloading thresholds of the queue lengths at fog and edge servers. Although the performance results of the approximate queueing model do not precisely match the simulation results, our numerical findings demonstrate that they share the same trend, and the approximate queueing model can provide the optimal queue length threshold in most cases. Moreover, our numerical results reveal that the QoS violating probability of the queue-length-based offloading is significantly lower than that of the probabilistic offloading scheme, with potential reductions of up to 60% in QoS violations in large-scale network scenarios.**

Index Terms—**cloud-edge-fog, federation, offloading, queueing analysis**

I. Introduction

Numerous innovative applications have emerged, leveraging the new technologies and services offered by 5G networks. Many of these cutting-edge applications heavily rely on high bandwidth and low latency. Notably, certain applications are extremely sensitive to delays and demand the Ultra-Reliable and Low Latency Communications (URLLC) capabilities of 5G, such as autonomous driving and telemedicine. For delay-sensitive applications, it is not only imperative to maintain low latency but also to ensure that the latency does not exceed a specific hard deadline, for example, 20 ms for cooperative driving, particularly in vehicle platooning scenarios with a high degree of automation [1].

To achieve low latency, Mobile Edge Computing (MEC) was introduced by the European Telecommunications Standards Institute (ETSI) [2] and later adopted by the 3rd Generation Partnership Project (3GPP). MEC utilizes edge servers physically close to users to provide low latency services. In contrast to traditional two-tier mobile cloud computing, which consists of a cloud server and User Equipment (UEs), MEC employs a three-tier architecture comprising federated cloud servers, edge servers, and UEs. This design reduces both computation and communication loads on the core network and cloud servers.

In a multi-tier architecture, offloading to where, i.e., cloud server, edge server, or UE itself, to execute a request is an important issue. There were many previous studies focusing on the offloading decision problem in a two-tier architecture [3-7] or in a three-tier architecture [8-13]. Most of these studies investigated how to minimize cost with latency constraint [3][6], latency [7][9][10], energy consumption[3][6][8], computation overhead by offloading[4][5], or resource allocation[9][12][13].

Previous papers focusing on latency reduction only considered the average latency. However, what matters most for each request is whether its Quality of Service (QoS) requirement can be met—specifically, if its latency can be kept below a given threshold necessary for the service [1]. Consequently, while certain offloading algorithms or mechanisms in these studies may reduce mean latency, they might not guarantee QoS satisfaction. In this work, our emphasis is on the QoS violating probability, defined as the probability that the latency of a request exceeds a given threshold. This latency includes both communication and computation delays.

In our previous work [14], we considered a probabilistic offloading (PLO) strategy, which offloads a request to a higher-tier server based on an optimal probability to minimize the QoS violating probability. In this study, we propose a novel offloading mechanism called queue-length-based offloading (QLO). With QLO, we set a threshold for the queue length on both the fog and edge servers. When a computation task request arrives at the fog server, it offloads the task to the edge server if the current queue length meets or exceeds the threshold. Similarly, the edge server offloads an incoming task to the cloud server if its queue length also meets or exceeds its threshold.The advantage of QLO over PLO is that it performs offloading based on the current server status, enabling real-time, on-demand offloading decisions. To estimate the optimal thresholds for the fog and edge servers, we develop an approximate analytical model. Through simulations on both small-scale and large-scale networks, our results demonstrate that QLO significantly reduces the QoS violation probability compared to PLO.

The main contributions of this study are:

(1) The QLO offloading scheme is proposed which outperforms the traditional PLO.
(2) We present a tractable analytical model for computing the distribution of the end-to-end delay of a computation task and the associated QoS violating probability. While the model assumes independent queues and Poisson arrival, making it less accurate, it remains valuable for estimating optimal thresholds for queue lengths of fog and edge servers.

The remainder of this paper is organized as follows: Section II describes the system architecture and system model. Closed-form solutions for calculating the QoS violating probability and optimal thresholds are derived based on queueing models. Section III presents a simple iterative search algorithm (ISA) for finding the optimal thresholds. Numerical results are then presented in Section IV. Finally, the conclusions and future works are summarized in Section V.

II. Problem Formulation

A. System Architecture

The system architecture of the three-tier federated cloud, edge, and fog computation system is shown in Fig. 1 [14]. UEs are considered as fog servers, enabling a computation task generated at a UE to be served either by the UE itself or offloaded to a nearby edge server or the cloud server. In this work, we focus on a homogeneous federated system with one cloud server pool. The cloud server is associated with *N* edge servers, and each edge server is associated with *M* UEs. Although our model is initially designed for homogeneous systems, it can be easily extended to handle heterogeneous setups. We assume computation tasks arrive at a UE according to a Poisson process. When a task arrives at a UE, it is offloaded to the edge server if the queue length of the UE meets or exceeds a threshold denoted by TH_U. Similarly, hen a task arrives at an edge server, it is offloaded to the cloud server if the edge server's queue length meets or exceeds a threshold denoted by TH_E. Additionally, we assume the cloud server has abundant service capacity, ensuring that a computation task arriving at the cloud server will always be served. We refer to this offloading strategy as queue-length-based offloading (QLO). The objective of optimal QLO is to determine the optimal threshold values, TH_U and TH_E, to minimize the overall QoS violating probability.

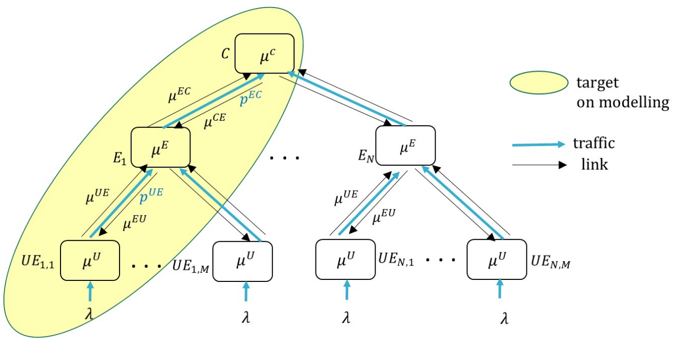

Fig. 1. Architecture of the three-tier federated Cloud, Edge, and Fog computation offloading system [14]

In line with our previous work [14], we also consider the communication overhead when offloading a task to higher-tier servers. We assume that when a task is offloaded to a higher-tier computing server, it is sent through a communication link (uplink), which introduces some communication delay. Similarly, after the computing server completes the computation, the task's result is sent back through a downlink, incurring another communication delay. Specifically, if the task is computed at the edge server, it experiences an uplink communication delay from the UE to the edge server and a downlink communication delay in the opposite direction. On the other hand, if the task is processed at the cloud server, it encounters two uplink communication delays and two downlink communication delays: one between the UE and the edge server and the other between the edge server and the cloud server, respectively.

B. Queueing Model

In this section, our goal is to select a tractable queueing model to determine the optimal threshold values TH_U and TH_E. A straightforward approach is to model the whole system as a multi-dimensional continuous-time Markov Chain. However, given the system's complexity with *N*M* UEs, *N* edge servers, *N* UE-to-edge uplink and downlink communication servers, 1 cloud server, 1 edge-to-cloud uplink and downlink communication server, and the bound of each queue to be *T*, the number of states will be $T^{NM+3N+3}$ which grows exponentially in terms of *N* and *M*. Even for a small system with $T = N = M = 10$, the number of states will be 10^{133}, making this approach computationally infeasible. The second approach treats each server or communication link as an independent queue. In this work, we choose to model UEs and edge servers as M/M/1/K queues, and the cloud server and communication links as M/M/1 queues. This choice is made to maintain the tractability of computing the end-to-end delay distribution.

Although the output process of the M/M/1/K queue is not a Poisson process, and the analytical results may not be entirely accurate, our numerical results show that the proposed analytical model's outcomes are consistent with simulation results. The model may underestimate the QoS violating probability due to the output process being burstier than a Poisson process. Nevertheless, the analytical model remains applicable in determining the optimal threshold values for QLO.

In the following, we will derive the end-to-end delay distribution for a computation task depends on which tier of server it is served. Based on the delay distribution, we can derive the QoS violating probability for a given delay constraint. Table I shows the notations used in our analysis. Fig. 2 shows the queueing network of the three-tier federated Cloud, Edge, and Fog computation system.

Case 1: the task is served by UE

By assuming the arrival of computation tasks to a UE follows a Poisson process with rate $\lambda^U = \lambda$, the service time follows an exponential distribution with rate μ^U, and the queue length bound $K = TH_U$, it forms an M/M/1/K queue. The steady-state probability for queue length equals *i* is given by

$$P_U(i) = \frac{\rho_U^i\,(1-\rho_U)}{1-\rho_U^{TH_U+1}}, \tag{1}$$

where $\rho_U = \lambda^U/\mu^U$ and the queue length includes the task in service if *i*>0.

The probability of offloading the task from UE to the edge server is given by

$$p^{UE} = \frac{\rho_U^{TH_U}(1-\rho_U)}{1-\rho_U^{TH_U+1}}. \quad (2)$$

TABLE I
Table of Notations

Notation	Meaning
μ^X ($\mu^C, \mu^E, \mu^U, \mu^{UE}, \mu^{EC}, \mu^{CE}, \mu^{EU}$)	Service capacity of the server or communication link X where X is either C (cloud), E (edge), U (UE), UE (uplink from UE to E), EC (uplink from edge to cloud), CE (downlink from cloud to edge), EU (downlink from edge to UE).
λ^X ($\lambda^C, \lambda^E, \lambda^U, \lambda^{UE}, \lambda^{EC}, \lambda^{CE}, \lambda^{EU}$)	The arrival rate of the server or communication link X where X is either C (cloud), E (edge), U (UE), UE (uplink from UE to E), EC (uplink from edge to cloud), CE (downlink from cloud to edge), EU (downlink from edge to UE).
λ	External task arrival rate to each UE.
TH_U	A computation task will be offloaded from UE to the edge server when its queue length is greater than or equal to this threshold. (In reality, the queue length will not be greater than this threshold as a result of admission control.)
TH_E	A computation task will be offloaded from the edge server to the cloud server when its queue length is greater than or equal to this threshold.
$\overline{TH}_U$	Upper bound of the queue length of UE.
$\overline{TH}_E$	Upper bound of the queue length of the edge server.
υ_X	The difference between service rate and arrival rate of X, i.e., $\upsilon_X = \mu^X - \lambda^X$, where X is either C, UE, EC, CE, or EU. The queuing delay of an M/M/1 server X is exponentially distributed with parameter υ_X.
p^X (p^{UE}, p^{EC})	The offloading probability from UE to the edge server (x=UE) or from the edge server to the cloud server (x=EC).
θ	Delay constraint of a computation task
$P_X^U(\theta)$ $P_X^E(\theta)$ $P_X^C(\theta)$	The probability of the delay of a task served by UE, edge server, or cloud server exceeds the delay constraint θ respectively.
$P_X(\theta)$	The overall probability of the delay of a task exceeds the delay constraint θ.

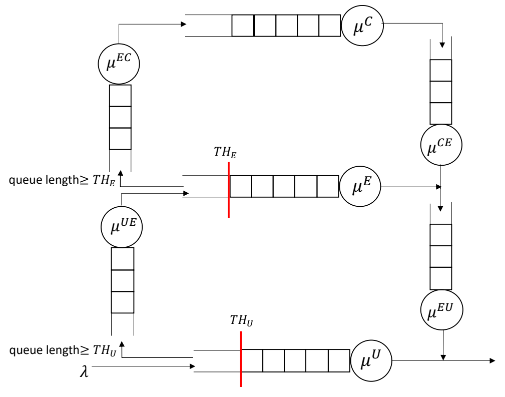

Fig. 2. Queueing network of the federated three-tier computation system.

The delay distribution of this M/M/1/K queue at the UE can be calculated based on the Poisson Arrivals See Time Averages (PASTA) theorem [15]. That is, upon an arrival sees i tasks in the queue, the delay distribution experienced by this arrival follows the Erlang(i+1, μ^U) distribution. Thus, the probability of the delay of a task exceeds the delay constraint θ is given by

$$P_X^U(\theta) = \sum_{i=0}^{TH_U-1} \frac{\rho_U^i(1-\rho_U)}{1-\rho_U^{TH_U+1}} \times \left(\sum_{j=0}^{i} \frac{(\mu^U\theta)^j}{j!} e^{-\mu^U\theta}\right). \quad (3)$$

In (3), the first term is the probability that an arrival sees i tasks in the queue, and the second term follows the Erlang(i+1, μ^U) distribution. Note that when i=0, Erlang(1, μ^U) becomes an exponential distribution, and when $i = TH_U$, the task is offloaded to the edge server, thus the summation is from i=0 to $TH_U - 1$.

For ease of explanation, in this paper, we consider a homogeneous scenario where all UEs have the same arrival and service rate, and similar assumptions apply to edge servers and each type of communication link. However, it is important to note that our results can be extended to a heterogeneous case.

Case 2: the task is served by the edge server

The delay consists of three parts: the delay of the uplink communication from UE to the edge server, the delay at the edge server, and the delay of the downlink communication from the edge server to the UE. Since an edge server is associated with M UEs, the arrival rate of these three queues is the same, given by

$$\lambda^{UE} = \lambda^E = \lambda^{EU} = M \times p^{UE} \times \lambda. \quad (4)$$

As aforementioned, although the output process of an M/M/1/K queue is not a Poisson process, to make the analysis tractable, we assume the arrival process to the queues at the edge server and two communication links between UE and the edge server are Poisson processes. The edge server is modeled as an M/M/1/K queue where $K = TH_E$. The two communication links are modeled as M/M/1 queues.

The model of the edge server is similar to that of UE. The steady-state probability for queue length equals i is given by

$$P_E(i) = \frac{\rho_E^i(1-\rho_E)}{1-\rho_E^{TH_E+1}}, \quad (5)$$

where $\rho_E = \lambda^E/\mu^E$. The probability of offloading the task from the edge server to the cloud server is given by

$$p^{EC} = \frac{\rho_E^{TH_E}(1-\rho_E)}{1-\rho_E^{TH_E+1}}. \quad (6)$$

The delay distribution of the edge server seen by an arrival can also be derived by the PASTA theorem. That is, if an arrival sees i tasks in the queue, the delay follows the Erlang(i+1, μ^E) distribution. On the other hand, the delay distribution of an M/M/1 queue with arrival rate λ and service rate μ is an exponential distribution with parameter $(\mu - \lambda)$.

The end-to-end delay of a task served by an edge server is derived using the Laplace transform. Let X_E be the random variable of the delay. For ease of explanation, let the delay distribution of the edge server be Erlang(r, μ_1), where $\mu_1 = \mu^E$. Let the delay of the communication link from UE to edge be exponentially distributed with parameter $\mu_2 = \mu^{UE} - \lambda^{UE}$ and that of the link from edge to UE be exponentially distributed with parameter $\mu_3 = \mu^{EU} - \lambda^{EU}$.

Case 2-1: $\mu_1 \neq \mu_2 \neq \mu_3$

The end-to-end delay is a summation of an Erlang distribution and two exponential distributions with different parameters. The Laplace transform of X_E is given by

$$S^*(s) = \frac{{\mu_1}^r \times \mu_2 \times \mu_3}{(s+\mu_1)^r \times (s+\mu_2) \times (s+\mu_3)}$$
$$= {\mu_1}^r \times \mu_2 \times \mu_3 \times$$
$$\left(\begin{array}{c} \sum_{i=0}^{r-1} \frac{(-1)^{r-i+1}}{(\mu_2-\mu_1)^{r-i}} \times \left(\begin{array}{c} \sum_{j=0}^{i} \frac{(-1)^{i-j}}{(\mu_3-\mu_1)^{i-j+1}} \times \frac{1}{(s+\mu_1)^{j+1}} \\ + \frac{(-1)^{i+1}}{(\mu_3-\mu_1)^{i+1}} \times \frac{1}{s+\mu_3} \end{array}\right) \\ + \frac{(-1)^r}{(\mu_2-\mu_1)^r \times (\mu_3-\mu_2)} \times \frac{1}{s+\mu_2} + \frac{(-1)^{r+1}}{(\mu_2-\mu_1)^r \times (\mu_3-\mu_2)} \times \frac{1}{s+\mu_3} \end{array}\right). \quad (7)$$

The delay distribution is obtained by finding the inverse Laplace transform, which is given by

$$f_{X_E}(x) = {\mu_1}^r \times \mu_2 \times \mu_3 \times$$
$$\Bigg(\sum_{i=0}^{r-1} \frac{(-1)^{r-i+1}}{(\mu_2-\mu_1)^{r-i}} \sum_{j=0}^{i} \frac{(-1)^{i-j}}{(\mu_3-\mu_1)^{i-j+1}} \times \left(\frac{x^i}{i!} \times e^{-\mu_1 x}\right) +$$
$$\sum_{i=0}^{r-1} \frac{(-1)^{r-i+1}}{(\mu_2-\mu_1)^{r-i}} \times \frac{(-1)^{i+1}}{(\mu_3-\mu_1)^{i+1}} \times e^{-\mu_3 x} + \frac{(-1)^r}{(\mu_2-\mu_1)^r \times (\mu_3-\mu_2)} \times$$
$$e^{-\mu_2 x} + \frac{(-1)^{r+1}}{(\mu_2-\mu_1)^r \times (\mu_3-\mu_2)} \times e^{-\mu_3 x}\Bigg). \quad (8)$$

Finally, the probability of the delay of a task exceeds the delay constraint θ is given by

$$P_{X_E}(r,\theta) = {\mu_1}^r \times \mu_2 \times \mu_3 \times$$
$$\left(\begin{array}{c} \sum_{i=0}^{r-1} \frac{(-1)^{r-i+1}}{(\mu_2-\mu_1)^{r-i}} \times \sum_{j=0}^{i} \frac{(-1)^{i-j}}{(\mu_3-\mu_1)^{i-j+1}} \times \frac{e^{-\mu_1 x}}{{\mu_1}^{i+1}} \times \sum_{j=0}^{i} \frac{(\mu_1\theta)^j}{j!} \\ + \sum_{i=0}^{r-1} \frac{(-1)^{r-i+1}}{(\mu_2-\mu_1)^{r-i}} \times \frac{(-1)^{i+1}}{(\mu_3-\mu_1)^{i+1}} \times \frac{1}{\mu_3} \times e^{-\mu_3\theta} \\ \frac{(-1)^r}{(\mu_2-\mu_1)^r \times (\mu_3-\mu_2)} \times \frac{e^{-\mu_2\theta}}{\mu_2} + \frac{(-1)^{r+1}}{(\mu_2-\mu_1)^r \times (\mu_3-\mu_2)} \times \frac{e^{-\mu_3\theta}}{\mu_3} \end{array}\right). \quad (9)$$

Case 2-2: $\mu_1 \neq \mu_2, \mu_2 = \mu_3$

The delay distribution becomes the summation of two Erlang distributions, i.e., Erlang(r, μ_1)+Erlang($2, \mu_2$). Due to the space limitation of the paper, we only present the probability of the delay of a task exceeds the delay constraint θ as follows

$$P_{X_E}(r,\theta) = {\mu_1}^r \times \mu_2^2 \times$$
$$\left(\begin{array}{c} \sum_{i=0}^{r-1} (-1)^{r+1-i} \times \frac{r-i}{(\mu_2-\mu_1)^{r+1-i}} \times \frac{e^{-\mu_1\theta}}{{\mu_1}^{i+1}} \times \sum_{j=0}^{i} \frac{(\mu_1\theta)^j}{j!} \\ + (-1)^r \times \frac{r}{(\mu_2-\mu_1)^{r+1}} \times \frac{e^{-\mu_2\theta}}{\mu_2} \\ + (-1)^r \times \frac{1}{(\mu_2-\mu_1)^r} \times \frac{1}{\mu_2^2} \times \left(e^{-\mu_2\theta} + \mu_2\theta e^{-\mu_2\theta}\right) \end{array}\right). \quad (10)$$

Case 2-3: $\mu_1 = \mu_2, \mu_2 \neq \mu_3$; or $\mu_1 = \mu_3, \mu_2 \neq \mu_3$

The delay distribution is the summation of Erlang($r+1, \mu_1$) and exponential(μ_3) (or exponential(μ_2) in the latter case). The probability of the delay of a task exceeds the delay constraint θ is given by

$$P_{X_E}(r,\theta) = {\mu_1}^{r+1} \times \mu_2 \times$$
$$\left(\sum_{i=0}^{r} \frac{(-1)^{r-i+2}}{(\mu_2-\mu_1)^{r-i+1}} \times \frac{e^{-\mu_1\theta}}{{\mu_1}^{i+1}} \times \left(\sum_{j=0}^{i} \frac{(\mu_1\theta)^j}{j!}\right) + \frac{(-1)^{r+1}}{(\mu_2-\mu_1)^{r+1}} \times \frac{e^{-\mu_2\theta}}{\mu_2}\right). \quad (11)$$

Case 2-4: $\mu_1 = \mu_2 = \mu_3$

The delay distribution is Erlang($r+2, \mu_1$) and the probability of the delay of a task exceeds the delay constraint θ is given by

$$P_{X_E}(r,\theta) = \sum_{i=0}^{r+1} \frac{(\mu_1\theta)^i}{i!} \times e^{-\mu_1\theta} \quad (12)$$

Case 2 summary:

With probability $P_E(i)$, an arrival sees i tasks in the edge server's queue, the delay distribution is Erlang($i+1, \mu_1$). Thus the overall probability of the delay of a task exceeds the delay constraint θ is given by

$$P_X^E(\theta) = \sum_{i=0}^{TH_E-1} P_E(i) \times P_{X_E}(i+1,\theta), \quad (13)$$

where $P_E(i)$ is defined by (5) and $P_{X_E}(i+1,\theta)$ is defined by either (9), (10), (11), or (12).

Case 3: the task is served by the cloud server

The analytical model for delay distribution is the same as we presented in our previous work[14]. The readers are referred to our previous work for details. In the following, we summarize the results from [14].

The delay of a task served by the cloud server consists of five parts: the delay of the uplink communication from UE to the edge server, the delay of the uplink communication from the edge server to the cloud server, the delay at the cloud server, the delay of the downlink communication from the cloud server to the edge server, and the delay of the downlink communication from the edge server to the UE. Since the cloud server is associated with N edge servers, the arrival rates to the cloud server, the uplink from the edge server to the cloud server, and the downlink from the cloud server to the edge server are the same and given by

$$\lambda^{EC} = \lambda^C = \lambda^{CE} = N \times M \times p^{EC} \times p^{UE} \times \lambda, \quad (13)$$

while recall that $\lambda^{UE}, \lambda^{EU}$ are given in equation (4).

We have a tandem of five M/M/1 queues and the delay of each queue is exponentially distributed. Let us denote the parameter of the delay distribution by $\upsilon_i, i = 1, \ldots, 5$. For example, $\upsilon_1 = \mu^{UE} - \lambda^{UE}$ and $\upsilon_2 = \mu^{EC} - \lambda^{EC}$. Depending on whether there are some $\upsilon_i's$ have the value, we have 7 sub-cases, denoted by (1,1,1,1,1), (5), (4,1), (3,1,1), (3,2), (2,1,1,1), (2,2,1), where each number denotes how many $\upsilon_i's$ are the same. For example, (3,2) denotes the case where 3 $\upsilon_i's$ have the same value while the other 2 $\upsilon_i's$ have the same value. In the following, we only show the result for the first case. For the rest 6 cases, the readers are referred to [14].

Case 3-1: $\upsilon_i \neq \upsilon_j$ if $i \neq j$, denoted by (1,1,1,1,1)

The probability of the delay of a task exceeds the delay constraint θ is given by

$$P_X^C(\theta) = P(X_C \geq \theta) = \prod_{i=1}^{5} \upsilon_i \times$$
$$\left[\begin{array}{l} \frac{1}{\upsilon_5-\upsilon_1} \times \frac{1}{\upsilon_4-\upsilon_1} \times \frac{1}{\upsilon_3-\upsilon_1} \times \frac{1}{\upsilon_2-\upsilon_1} \times \frac{1}{\upsilon_1} \times e^{-\upsilon_1\theta} \\ - \frac{1}{\upsilon_5-\upsilon_2} \times \frac{1}{\upsilon_4-\upsilon_2} \times \frac{1}{\upsilon_3-\upsilon_2} \times \frac{1}{\upsilon_2-\upsilon_1} \times \frac{1}{\upsilon_2} \times e^{-\upsilon_2\theta} \\ + \frac{1}{\upsilon_5-\upsilon_3} \times \frac{1}{\upsilon_4-\upsilon_3} \times \frac{1}{\upsilon_3-\upsilon_2} \times \frac{1}{\upsilon_3-\upsilon_1} \times \frac{1}{\upsilon_3} \times e^{-\upsilon_3\theta} \\ - \frac{1}{\upsilon_5-\upsilon_4} \times \frac{1}{\upsilon_4-\upsilon_3} \times \frac{1}{\upsilon_4-\upsilon_2} \times \frac{1}{\upsilon_4-\upsilon_1} \times \frac{1}{\upsilon_4} \times e^{-\upsilon_4\theta} \\ + \frac{1}{\upsilon_5-\upsilon_4} \times \frac{1}{\upsilon_5-\upsilon_3} \times \frac{1}{\upsilon_5-\upsilon_2} \times \frac{1}{\upsilon_5-\upsilon_1} \times \frac{1}{\upsilon_5} \times e^{-\upsilon_5\theta} \end{array}\right]. \quad (15)$$

Summary:

Given the probability of the delay of a task exceeds the delay constraint θ in three cases, the overall probability of violating the delay constraint is given by

$$P_X(\theta) = P(X \geq \theta) = (1-p^{UE}) \times P_X^U(\theta)$$
$$+(1-p^{EC}) \times p^{UE} \times P_X^E(\theta) + p^{UE} \times p^{EC} \times P_X^C(\theta) \quad (16)$$

C. Problem Statement

Given following system parameters, how to configure TH_U and TH_E such that $P_X(\theta)$ could be minimized? The system parameters include number of UEs per edge server (M), number of edge servers per cloud server (N), external task arrival rate (λ), the service rate of each server and communication link ($\mu^C, \mu^E, \mu^U, \mu^{UE}, \mu^{EC}, \mu^{CE}, \mu^{EU}$), and the delay constraint θ.

III. Iterative Search Algorithm (ISA)

In this section, we proposed a search algorithm for finding the optimal TH_U and TH_E which minimizes $P_X(\theta)$ based on an iterative algorithm. In section II, we have showed how to compute $P_X(\theta)$ when TH_U and TH_E are given. Since TH_U and TH_E are integer values with an upper bound, a straightforward search algorithm is the exhaustive search that explores all possible pairs of TH_U and TH_E. The complexity of the exhaustive search algorithm is $O(\overline{TH}_U \times \overline{TH}_E)$ where $\overline{TH}_U$ and $\overline{TH}_E$ are the upper bounds of the queue length of the UE and the edge server. To avoid exhaustive search, we propose an iterative search algorithm, as shown in Fig. 3, which first searches for the optimal TH_U when TH_E is fixed, and then fixed TH_U to current optimal value and search for the optimal TH_E. It then repeats these two steps until TH_U and TH_E converges to fixed points. In our experiments, ISA converges within 2 iterations in most cases.

```
Iterative Search Algorithm
Input: M, N, $\lambda$, $\mu^C, \mu^E, \mu^U, \mu^{UE}, \mu^{EC}, \mu^{CE}, \mu^{EU}$, θ, $\overline{TH}_U, \overline{TH}_E$
Output: optimal $TH_U, TH_E$
Begin
    $TH_U = \overline{TH}_U/2$; $TH_E = \overline{TH}_E/2$; //initialize $TH_U, TH_E$
    Repeat {
        $old\_TH_U = TH_U$; $old\_TH_E = TH_E$; $minp = 1.0$;
        For ($TH_U = 0, 1, \ldots, \overline{TH}_U$){ //search for optimal $TH_U$
            calculate $P_X(\theta)$ using equations (3)-(16)
            If ($P_X(\theta)$<$minp$) {
                $minp$=$P_X(\theta)$; $optimal\_TH_U$=$TH_U$; }
        }
        $TH_U = optimal\_TH_U$; $minp$ =1.0;
        For ($TH_E = 0, 1, \ldots, \overline{TH}_E$){ //search for optimal $TH_E$
            calculate $P_X(\theta)$ using equations (3)-(16)
            If ($P_X(\theta)$< $minp$) {
                $minp$=$P_X(\theta)$; $optimal\_TH_E$=$TH_E$; }
        }
        $TH_E = optimal\_TH_E$
    } until ($old\_TH_U == TH_U$ AND $old\_TH_E$== $TH_E$);
    Output $TH_U, TH_E$
End
```

Fig. 3. Pseudocode for the ISA algorithm.

IV. Numerical Results

A. Parameter Setting

We evaluate the system performance using two scenarios: a small-scale network and a large-scale network. The setting of system parameters for the small-scale network is shown in Table II, which is mainly used to compare the performance of the proposed QLO with that of PLO, as proposed in [14].

B. Analysis vs. Simulation

We first validate our analytical results by comparing with the simulation results, as shown in Fig. 4. In Fig. 4(a), TH_E is set to 2 and TH_U varies from 0 to 4, while in (b), TH_U is set to 1, and TH_E varies from 0 to 5. As expected, the analytical results do not match the simulation results very well. However, the trend of both simulation and analysis results is similar. The analytical model still provides a good mechanism to estimate the optimal queue length thresholds, namely TH_U and TH_E. Therefore, in the following experiments, when comparing the performance of QLO with that of PLO, we first use the analytical model to determine the approximate optimal TH_U and TH_E values. Subsequently, we run simulations to evaluate the actual performance of QLO using these approximate optimal values and then compare its performance with PLO. Each simulation is run 30 times with a simulation time of 10,000 time units, and the 95% confidence interval is less than 1% of the mean value.

TABLE II

Small network's system parameters (rates are per time unit) (M=N=5, $\theta = 1.2$)

λ=2	μ^C=25	μ^E=8	μ^U=1.5	$\overline{TH}_U = 4$
μ^{UE}=12	μ^{EC}=22	μ^{CE} =21	μ^{EU}=11	$\overline{TH}_E = 5$

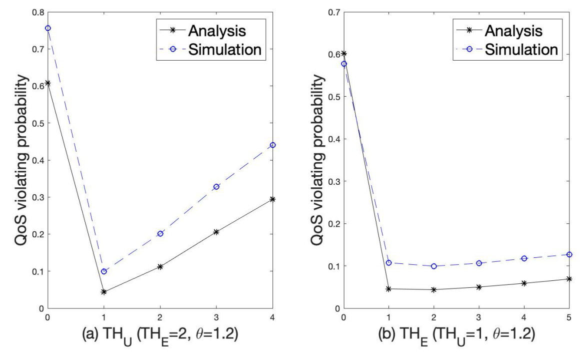

Fig. 4. Comparison of simulation and analytical results (small network).

For the small network case, the analytical model estimates the optimal thresholds are $TH_U = 1, TH_E = 2$. Table III shows the QoS violating probability for all possible combinations of TH_U and TH_E. As we can see that both analytical and simulation results indicate that $TH_U = 1$ and $TH_E = 2$ are indeed the optimal thresholds, although the values of analytical and simulation do not match well.

TABLE III

Simulation and analytical results for all combination of TH_U and TH_E (small network)

TH_U / TH_E	0		1		2		3		4	
	S	A	S	A	S	A	S	A	S	A
0	1.0	1.0	0.58	0.60	0.62	0.54	0.50	0.32	0.51	0.33
1	0.65	0.67	0.11	0.05	0.20	0.11	0.33	0.20	0.44	0.29
2	0.76	0.61	0.10	0.04	0.20	0.11	0.33	0.21	0.44	0.29
3	0.70	0.54	0.11	0.05	0.21	0.11	0.33	0.21	0.44	0.30
4	0.71	0.54	0.12	0.06	0.21	0.12	0.33	0.21	0.44	0.30
5	0.73	0.56	0.13	0.07	0.21	0.12	0.34	0.21	0.45	0.30

S: Simulation results; A: Analytical results

The system parameters of the large network are shown in Table IV. This setting emulates a real-world system with 1600 UEs, where each edge server is associated with 40 UEs, and the cloud server is associated with 40 edge servers. Fig. 5 illustrates the comparison of analytical results with simulation results for the large network scenario. Similar to Fig. 4, they do not match well, but they show the same trend. In summary, the analytical model underestimates the QoS violating probability due to the invalid Poisson arrival assumption; however, it is still useful to estimate the optimal TH_U and TH_E.

C. POL vs. QOL

Fig. 6 compares the QoS violating probability of POL and QOL. As we can observe, QOL significantly outperforms POL. The rationale is that QOL leverages the advantage of offloading a task to the upper-tier server by checking the current queue length status. This real-time on-demand approach allows tasks to be offloaded to the upper tier only when necessary. On the other hand, POL blindly offloads tasks to upper-tier servers randomly with a pre-defined probability. However, we also notice that the curve of the QoS violating probability is not smooth in Fig. 6(b). This is due to the fact that the value of the threshold can only be a discrete (integer) value. One possible solution to this problem is to combine queue length-based offloading with probabilistic offloading. That is, when the queue length is greater than or equal to the threshold, the task is offloaded to the upper tier with some probability less than 1.

TABLE IV

Large network's system parameters (M=N=40, θ = 2)

λ=1	μ^{C}=200	μ^{E}=20	μ^{U}=1	$\overline{TH}_U = 5$
μ^{UE}=40	μ^{EC}=800	μ^{CE} =800	μ^{EU}=40	$\overline{TH}_E = 10$

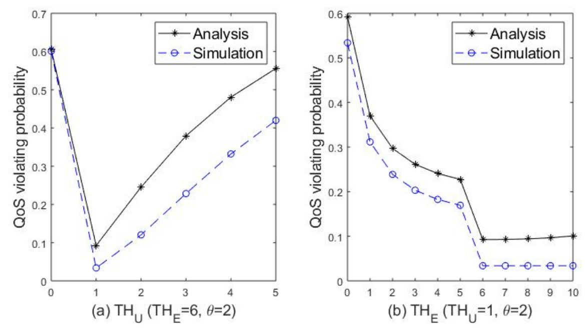

Fig. 5. Comparison of simulation and analytical results (large network).

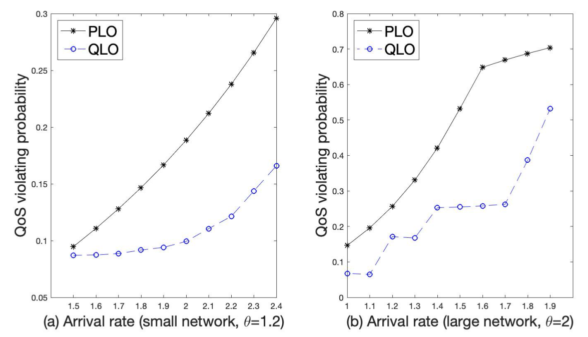

Fig. 6. Comparison of QoS violating probability of PLO and QLO

V. Conclusions

This paper investigates queue-length-based offloading under a delay constraint in a three-tier computation system comprising a cloud server, edge servers, and fog servers (mobile devices). We propose an approximate analytical model by assuming the independence of all queues in the system. While the analytical model's results do not precisely match those of simulations due to an invalid assumption of the input process as a Poisson process, both sets of results share the same trend. Thus, the analytical model still provides a good estimation of the optimal thresholds for the queue lengths of the edge server and mobile devices (UE). By setting the thresholds based on the analytical model, our simulation results demonstrate that the queue-length-based offloading scheme can achieve a better QoS violating probability compared to the probabilistic offloading scheme.

There are several areas that require further investigation. First, we can consider modeling the output process of the M/M/1/K queue as an MMPP process and deriving the delay distribution of a tandem of MMPP/M/1 queues. Secondly, as evident from Fig. 6, the performance of QLO is limited due to the integer value constraint of the queue length threshold. To address this, we are exploring an approach that combines QLO and PLO. Specifically, when a task arrives, if the queue length equals the threshold minus one (e.g., $TH_U - 1$), the task will be offloaded to the upper-tier server with a certain probability. As a result, we have four parameters in the 3-tier system: thresholds for UE and edge server queue lengths, and the probability of offloading when the queue length equals the threshold minus one. We can derive analytical models and design a search algorithm to find the optimal thresholds and offloading probabilities.

REFERENCE

[1] 3GPP TS 22.186, v16.2.0, Enhancement of 3GPP support for V2X scenarios, Stage 1 (Release 16), Jun. 2019.

[2] Y. H. Milan Patel, Patrice Hédé (September 2014). Mobile-Edge Computing – Introductory Technical White Paper. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

[3] Y.-D. Lin, J.-C. Hu, B. Kar, L.-H. Yen, "Cost Minimization with Offloading to Vehicles in two-Tier Federated Edge and Vehicular-Fog Systems," *IEEE VTC2019-Fall*, Honolulu, HI, USA, Sep. 22-25, 2019.

[4] I. Kovacevic, *et al.*, "Cloud and Edge Computation Offloading for Latency Limited Services," *IEEE Access*, vol. 9, pp. 55764 - 55776, Apr. 2022.

[5] Y. Xiao, L. Wei, J. Feng, W. En, "Two-tier end-edge collaborative computation offloading for edge computing," *Journal of Computational Methods in Sciences and Engineering*, vol. 22, no. 2, pp. 677–688, Jan. 2022.

[6] B. Kar, Y.-D. Lin, and Y. C. Lai, "Cost optimization of omnidirectional offloading in two-tier cloud–edge federated systems," *Journal of Network and Computer Applications*, vol. 215, Jun. 2023.

[7] W. B. Qaim, *et al.*, "Understanding the Performance of Task Offloading for Wearables in a Two-Tier Edge Architecture," *IEEE 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Brno, Czech, 25-27 Oct. 2021.

[8] L. Li, and H. Zhang, "Delay Optimization Strategy for Service Cache and Task Offloading in Three-Tier Architecture Mobile Edge Computing System," *IEEE Access*, vol. 8, Sep. 2020.

[9] H. Dinh, *et al.*, "Deep Reinforcement Learning-based Offloading for Latency Minimization in 3-tier V2X Networks," *IEEE Wireless Communications and Networking Conference (WCNC)*, Austin, TX, USA, Apr. 2022.

[10] M. Bolourian and H. Shah-Mansouri, "Energy-Efficient Task Offloading for Three-Tier Wireless-Powered Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 10(12), pp. 10400-10412, Jun. 2023.

[11] Z. Gao, W. Hao, and S. Yang, "Joint Offloading and Resource Allocation for Multi-User Multi-Edge Collaborative Computing System," *IEEE Transactions on Vehicular Technology*, vol. 71(3), pp. 3383-3388, Mar. 2022.

[12] F. You, W. Ni, J. Li, and A. Jamalipour, "New Three-Tier Game-Theoretic Approach for Computation Offloading in Multi-Access Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 71(9), pp. 9817-9829, Sep. 2022.

[13] C.-C. Wang, Y.-D. Lin, J.-J. Wu, P.-C. Lin, R.-H. Hwang, "Towards Optimal Resource Allocation of Virtualized Network Functions for Hierarchical Datacenters," IEEE Transactions on Network and Service Management, Vol. 15(4), Dec. 2018, pp. 1532-1544.

[14] R.-H. Hwang, Y.-C. Lai and Y.-D. Lin, "Offloading Optimization with Delay Constraint in the 3-tier Federated Cloud, Edge, and Fog Systems," 2021 IEEE Globecom, Madrid, Spain, Dec. 7-11, 2021.

[15] R. W. Wolff, "Poisson Arrivals See Time Averages," Operations Research, vol. 30(2), pp. 223-414, Mar.-Apr. 1982.