Telecommunication Systems
https://doi.org/10.1007/s11235-021-00864-0

Analyzing vertical and horizontal offloading in federated cloud and edge computing systems

Kohei Akutsu[1] · Tuan Phung-Duc[2] · Yuan-Cheng Lai[3] · Ying-Dar Lin[4]

Accepted: 13 October 2021

Abstract
Mobile Edge Computing architecture is one of the most promising architectures that can satisfy different quality of Services required by various applications. In this paper, we model mobile edge computing architecture with queue-length thresholds at user equipments and edges to determine whether the task is offloaded or not in federated cloud and edge computing systems. We propose two models as vertical default & vertical (VDV) model and vertical default & horizontal shortest (VDHS) model. The former only does vertical offloading, meaning that the edge can offload tasks to the cloud, while the latter does vertical offloading and horizontal offloading, meaning that the edge can offload tasks to other edges. However, it is very difficult to directly derive the performance metrics in our models, so we approximate them. Based on these approximations, we determine the optimal queue-length thresholds of UEs and edges. Experiment results show that analytical and simulation results match very well. Also VDHS can reduce the mean task sojourn time by 30% at most and increase delay satisfaction ratio by 11% at most compared with VDV.

Keywords Mobile edge computing · Vertical offloading · Horizontal offloading

1 Introduction

In the past few years, various services using wireless communications such as Virtual Reality (VR) and Augmented Reality (AR) have been developed. These services demand massive data communication and the constraint of sojourn time in wireless communication. 5th generation (5G) mobile communication system networks are considered to be able to achieve these targets [1]. Compared with earlier networks, 5G networks present enhanced Mobile Broadband (eMBB), massive Machine Type Communications (mMTC) and Ultra-Reliable and Low Latency Communications (URLLC) [2]. It is expected that 5G networks will impact hugely on people's lives.

In order to realize 5G networks to meet different Quality of Services (QoS) required by various applications, it is necessary not only to evolve wireless communication technology, but also to create a network architecture that allows many users to carry out simultaneous connections and low-latency communications [3]. Nowadays, the network architecture used in wireless communications is an architecture called Cloud Computing. This architecture is composed of User Equipments (UEs) such as smartphones or tablet terminals and a cloud server. UEs which have limited computation capacity offload their tasks to the cloud server. However, considering the distances between the UEs and the cloud server, it takes several hundred milliseconds latency to transmit a task from UEs to the cloud server. For that reason, it is difficult to satisfy the low latency required by URLLC services [4].

✉ Yuan-Cheng Lai
laiyc@cs.ntust.edu.tw

Kohei Akutsu
akutsu.kohei.sp@alumni.tsukuba.ac.jp

Tuan Phung-Duc
tuan@sk.tsukuba.ac.jp

Ying-Dar Lin
ydlin@cs.nctu.edu.tw

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan

[2] Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Japan

[3] Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan

[4] Department of Computer Science, National Yang-Ming Chiao-Tung University, Hsinchu, Taiwan

Published online: 15 January 2022

To achieve such targets in 5G networks, one of the most promising architectures is Mobile Edge Computing (MEC) [5]. MEC has a similar architecture to Cloud Computing, but differs from Cloud Computing in that the installation of network edges is close to the UEs. MEC servers can be deployed collocated with base station or be attached in the core network of cellular system [6], which can significantly reduce the communication latency. However, MEC has a smaller computation capacity than the cloud. Therefore, the federation between the cloud and edges is carried out to enlarge a system's capacity and accommodate different service requirements [7].

There are many earlier studies that have focused on MEC offloading [8–18]. The authors of [8] and [9] discussed MEC offloading in an IoT system with the objective of minimizing the sojourn time and energy. Yousefpour et al. [11] and Lee et al. [10] considered the waiting time of tasks and proposed an algorithm to select the edges for offloading destinations. In these studies, the authors did not consider transmission latency. To address the transmission latency, Yousefpour et al. [11] proposed a new model. Taking into consideration transmission latency, the waiting time of tasks at edge servers and at a cloud server, these authors proposed a model for optimally allocating computing resources. However, this research focused on the mean sojourn time of the tasks and did not consider the QoS of the sojourn time of each task. As noted above, in 5G networks, URLLC is set as a critical goal. We can therefore not neglect the QoS of the sojourn time of each task. In considering this issue, Hwang et al. [12] derived the distribution of sojourn time in MEC. In this paper, a method of allocating tasks was proposed, based on the fixed probabilities at UEs and edge servers, and used a Sub-Gradient Descent (SGD) algorithm to derive optimal probabilities.

Some studies [13–18] used machine learning (ML) to carry out task offloading in a cellular system. Elgendy et al. [13] proposed an offloading and caching algorithm based on Deep Reinforcement Learning (DRL) that would minimize the sojourn time and energy use of MEC for real-time video applications. Instead of considering the sojourn time, the authors of [14] considered the task's deadline by carrying out offloading with DRL. Wang et al. [15] extended MEC with unmanned aerial vehicles whose trajectories were optimized by DRL to serve offloaded traffic from UEs. The authors of [16] allocated offloaded tasks of multiple clouds to MECs and UEs to minimize the mean sojourn time by using a multi-agent Reinforcement Learning (RL) approach. The authors of [17] proposed an ML-based offloading strategy for an IoT-Edge system and examined offloading accuracy. Task priority and redundancy were addressed in offloading decisions that would improve the Quality of Experience (QoE) in [18]. The authors of [13–15] only considered vertical offloading from UEs to edges. The authors of [16–18] considered cloud-edge federation, but only carried out vertical offloading between edges and a cloud.

Two concerns exist in previous studies: (1) Most papers addressed vertical offloading, but did not address horizontal offloading; and (2) they considered many parameters and adopted some complex mechanisms to determine whether offloading or not. However, since each offloading decision is a data-plane action, it must be fast enough.

Therefore, in this paper, we adopt queue-length-based offloading strategy, which does the offloading when the queue length exceeds a threshold. To address a queue-length-based offloading strategy, we modelled MEC with a threshold of the number of tasks at UEs and edge servers using queueing models. Our model consists of three layers, UEs, edge servers, and the cloud server. We consider transmission latency and computation time at UEs, edge servers or the cloud server. The sum of transmission latency and computation time is regarded as the task's sojourn time. We propose two models, Vertical Default & Vertical (VDV) model and Vertical Default & Horizontal Shortest (VDHS) model, which differ in the offloading selection of edges and the cloud. VDV only does vertical offloading, meaning that the edge can offload tasks to the cloud, while VDHS not only does vertical offloading, but also does horizontal offloading, meaning that the edge can offload tasks to the targeted edge which owns the shortest queue. Analyzing these models, we derive the performance metrics (mean and distribution of the sojourn time) and find the optimal queue-length thresholds at UEs and edge servers in these models.

One of the closest references to our work is [21] in which the authors consider the same architecture and also handle the optimization problem. However, there are three main differences between [21] and our paper: (1) they assume that each component is an M/M/1 or an M/M/c queue while we consider all processing components in a three-tier architecture as a single stochastic process and thus the dependency between components is considered; (2) they use probability offloading, meaning that the offloading decision is according to a pre-calculated probability while we use queue-length offloading, meaning that each device (UE or edge) offloads its tasks when its current queue length exceeds a threshold; and (3) they calculate only the mean sojourn time while we also calculate the sojourn time distribution, which is difficultly obtained but can be used to calculate an important performance metric, the delay satisfaction ratio.

The rest of this paper is organized as follows. In Sect. 2, we explain the architectures and behaviors of our models. Section 3 presents stability conditions and stationary distributions in the models. Performance metrics in our models are provided in Sect. 4 and we compare them and simulation results in Sect. 5. Section 6 concludes the paper.

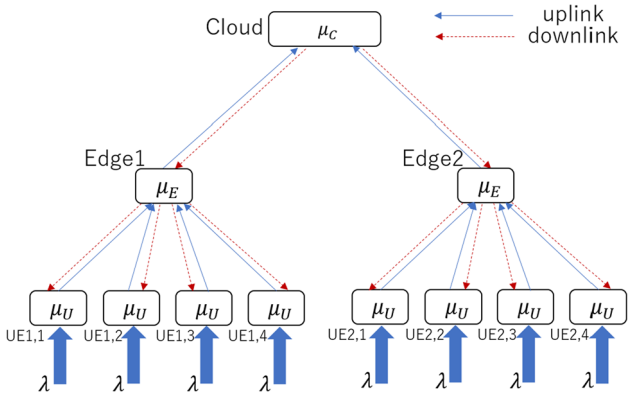

Fig. 1 Architecture in the VDV model, $M = 2$ and $N = 4$

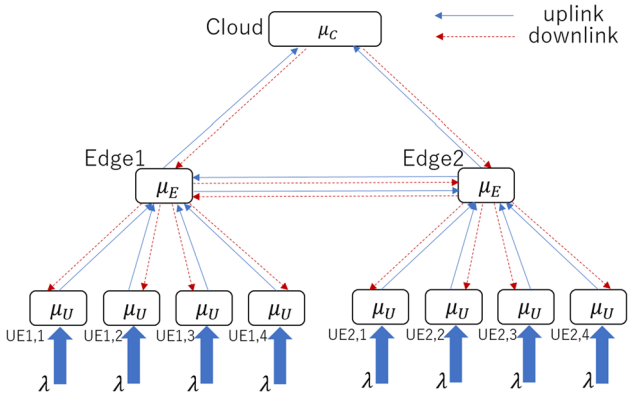

Fig. 2 Architecture in the VDHS model, $M = 2$ and $N = 4$

2 Model

2.1 Architecture

First, we assume that the cloud server is associated with M edge servers and each edge server is associated with N UEs. Figures 1 and 2, taking $M = 2$ and $N = 4$ as an example in each model, represent the VDV and VDHS models, respectively. We can analyze these models as a continuous-time Markov chains. However, it is difficult to express the infinitesimal generators of these models for arbitrary M and N. Therefore, we assume that the cloud server is associated with 2 edge servers and each edge server is associated with 1 UE when we analyze these models. We furthermore assume that the UEs and edge servers have thresholds of the number of existing tasks in their queues.

2.2 Behaviors

At the beginning, a computation task is generated at a UE. The number of existing tasks at the UE determines either the task is served locally or offloaded to an edge or the cloud. If the number of tasks at the UE is less than a threshold, the task is served at the UE. On the other hand, if the number of tasks reaches the threshold of the UE, the task is offloaded to the edge. When the tasks are offloaded, two models differ in which edge handles the task (this edge is called as the targeted edge). In the VDV model, a task is offloaded to the targeted edge server, which is associated with the UEs. In the VDHS model, the task is offloaded to the edge associated with UEs. If the number of existing tasks at this edge reaches its threshold, and the number of tasks at other edges has not reached their threshold, the task is offloaded to the targeted edge, which is the edge with the shortest queue. In these models, if the number of tasks at the targeted edge is less than its threshold, the task is served at that edge; otherwise, the task is offloaded to the cloud to be served there.

Using queueing models, we can express the behaviors in these models. Tasks arrive at UEs according to the Poisson process with parameter λ. If, on arrival, the number of tasks at the UE is less than its threshold, $TH^{(U)}$, the task is served at the UE for an exponentially distributed time with parameter $\mu_{(U)}$. After completion of the service at the UE, the task leaves the system. Otherwise, when the number of tasks at the UE equals $TH^{(U)}$ upon arrival of a task, the task is offloaded. The time spent on the uplink transmission from a UE to an edge is $d_{(UE)}$. After the transmission has been completed, the task arrives at the edge. Note that, if horizontal offload takes place in the VDHS model, the transmission latency is given by $d_{(EE)}$.

After the horizontal offloading in VDHS or no horizontal offloading in VDV, the targeted edge is determined. If the number of existing tasks at the targeted edge does not exceed the threshold, $TH^{(E)}$, the task is served at the edge for an exponentially distributed time with parameter $\mu_{(E)}$. After the computation at the edge has been completed, the task is sent back to the UE via the downlink. The latency in the downlink from an edge to a UE is $d_{(EU)}$. After transmission has been completed, the task leaves the system. By contrast, if the number of existing tasks at the targeted edge reaches $TH^{(E)}$, after spending the transmission latency $d_{(EC)}$, the task is offloaded from the edge to the cloud, and is then served at the cloud for an exponentially distributed time with parameter $\mu_{(C)}$. After the computation has been completed, the task is sent back through the downlink from the cloud to the UE, spending the transmission latency $d_{(CE)} + d_{(EU)}$. Figures 3 and 4 show the processing flows when tasks arrive at UE1 in the VDV and VDHS models, respectively, when $M = 2$ and $N = 1$. Table 1 shows the parameters used in our models.

3 Analysis

In this section, we derive the performance metrics of the models developed in the Sect. 2. As mentioned in Sect. 2, we assumed that the cloud server is associated with 2 edge

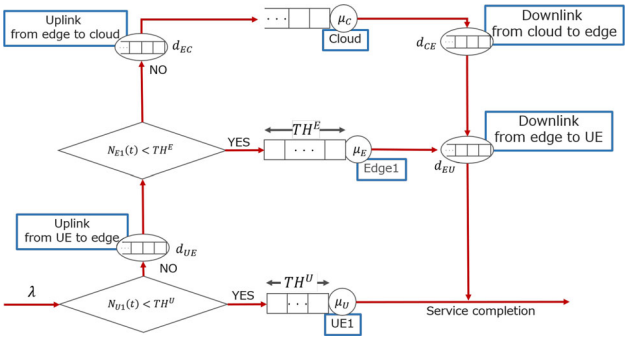

Fig. 3 Flow in the VDV model, $M = 2$ and $N = 1$

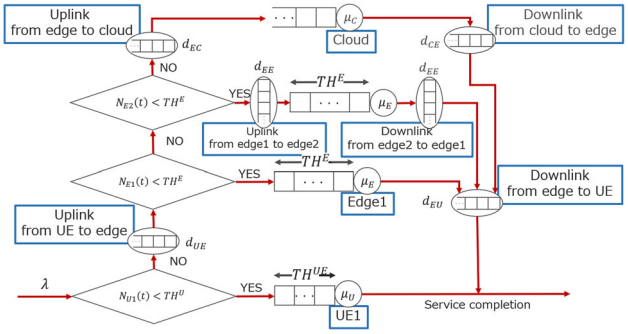

Fig. 4 Flow in the VDHS model, $M = 2$ and $N = 1$

servers and each edge server is associated with 1 UE. $\boldsymbol{I}$ and $\boldsymbol{O}$ are the identity matrix and the zero matrix of an appropriate dimension, and $\boldsymbol{e}$ and $\boldsymbol{0}$ represent the column vectors of an appropriate dimension of 1's and 0's, respectively. Furthermore, we define the element in the matrix as follows:

$$\boldsymbol{Y} := (\boldsymbol{Y}^{(i,j)}) = \begin{pmatrix} \boldsymbol{Y}^{(0,0)} & \boldsymbol{Y}^{(0,1)} & \boldsymbol{Y}^{(0,2)} & \cdots \\ \boldsymbol{Y}^{(1,0)} & \boldsymbol{Y}^{(1,1)} & \boldsymbol{Y}^{(1,2)} & \ddots \\ \boldsymbol{Y}^{(2,0)} & \boldsymbol{Y}^{(2,1)} & \boldsymbol{Y}^{(2,2)} & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix},$$

where $\boldsymbol{Y}^{(i,j)}$ can be either a matrix or a scalar.

3.1 Random variables

The random variables are defined as follows. Let $N_{U1}(t)$, $N_{U2}(t)$, $N_{E1}(t)$, $N_{E2}(t)$ and $N_C(t)$ denote the number of tasks at UE1, UE2, edge1, edge2 and cloud at time t, respectively. Note that each random variable includes the task in service. In the current setting, without considering transmission latency, i.e., $d_{UE} = d_{EC} = d_{EU} = d_{CE} = d_{EE} = 0$, we can construct the continuous-time Markov chain $\boldsymbol{X}(t) := \{(N_{U1}(t), N_{U2}(t), N_{E1}(t), N_{E2}(t), N_C(t)) | t \geq 0\}$ on the state space $\mathcal{S} := \mathcal{S}' \times \mathbb{N}_0$, where $\mathcal{S}' := \mathcal{S}_U \times \mathcal{S}_U \times \mathcal{S}_E \times \mathcal{S}_E$, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$, $\mathcal{S}_U := \{0, 1, \ldots, TH^U\}$, $\mathcal{S}_E := \{0, 1, \ldots, TH^E\}$.

Table 2 summarizes the notations that are frequently used in our analysis.

3.2 Infinitesimal generator in VDV

First, we analyze the VDV model: the infinitesimal generator of this model is given as

$$\boldsymbol{Q}_R = \begin{pmatrix} \boldsymbol{H}_R & \boldsymbol{A}_R & \boldsymbol{O} & \cdots \\ \boldsymbol{D}_R & \boldsymbol{S}_R & \boldsymbol{A}_R & \ddots \\ \boldsymbol{O} & \boldsymbol{D}_R & \boldsymbol{S}_R & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix}, \tag{1}$$

where $\boldsymbol{D}_R, \boldsymbol{S}_R, \boldsymbol{A}_R$ and $\boldsymbol{H}_R$ are square matrices of order $(TH^U + 1)^2(TH^E + 1)^2$, as

$$\boldsymbol{H}_R = (\boldsymbol{H}_R^{(i,j)})_{i,j \in \mathcal{S}_E} = \begin{cases} \boldsymbol{K}_R, & i = j - 1, \\ \boldsymbol{H}_{R,1}, & i = j = 0, \\ \boldsymbol{H}_{R,2}, & i = j \neq 0, \\ \mu_E \boldsymbol{I}, & i = j + 1, \\ \boldsymbol{O}, & otherwise, \end{cases}$$

$$\boldsymbol{D}_R = \mu_C \boldsymbol{I}, \boldsymbol{S}_R = \boldsymbol{H}_R - \boldsymbol{D}_R,$$

$$\boldsymbol{A}_R = diag(\boldsymbol{A}_{R,1}, \ldots, \boldsymbol{A}_{R,1}, \boldsymbol{A}_{R,2}).$$

Table 1 The parameters in our model

Indicator	Definition
λ	Arrival rate of task at each UE
μ_X	Service rate at X where X is either U (UE), E (edge) or C (cloud).
d_{UE}	Transmission latency from a UE to an edge
d_{EC}	Transmission latency from an edge to the cloud
d_{EE}	Transmission latency from an edge to an other edge
d_{CE}	Transmission latency from the cloud to an edge
d_{EU}	Transmission latency from an edge to a UE
TH^U	Threshold of the number of existing tasks at UE
TH^E	Threshold of the number of existing tasks at the edges

Table 2 The notations in our analysis

Indicator	Definition
$N_{U1}(t)$	The number of tasks at UE1 at time t
$N_{U2}(t)$	The number of tasks at UE2 at time t
$N_{E1}(t)$	The number of tasks at edge1 at time t
$N_{E2}(t)$	The number of tasks at edge2 at time t
$N_C(t)$	The number of tasks at the cloud at time t
$\boldsymbol{I}$	Identity square matrix with an appropriate dimension
$\boldsymbol{O}$	Zero square matrix withan appropriate dimension
$\boldsymbol{e}$	Column vector of ones with an appropriate dimension
$\boldsymbol{0}$	Column vector of zeros with an appropriate dimension
R	The subscript for notations of the VDV model
H	The subscript for notations of the VDHS model
T_R	Sojourn time in the VDV model
T_H	Sojourn time in the VDHS model
$f_R(x)$	Probability density function of T_R
$f_H(x)$	Probability density function of T_H
$F_R(x)$	Cumulative distribution function of T_R
$F_H(x)$	Cumulative distribution function of T_H

$\boldsymbol{A}_{R,1}$, $\boldsymbol{A}_{R,2}$, $\boldsymbol{K}_R$, $\boldsymbol{H}_{R,1}$ and $\boldsymbol{H}_{R,2}$ are square matrices of order $(TH^U+1)^2(TH^E+1)$, as

$$\boldsymbol{A}_{R,1} = diag(\boldsymbol{O},\ldots,\boldsymbol{O},\boldsymbol{L}_{R,1}),$$
$$\boldsymbol{A}_{R,2} = diag(\boldsymbol{L}_{R,2},\ldots,\boldsymbol{L}_{R,2},\boldsymbol{L}_{R,1}+\boldsymbol{L}_{R,2}),$$
$$\boldsymbol{K}_R = diag(\boldsymbol{L}_{R,2},\ldots,\boldsymbol{L}_{R,2}),$$
$$\boldsymbol{H}_{R,1} = (\boldsymbol{H}_{R,1}^{(i,j)})_{i,j\in\mathcal{S}_E} = \begin{cases} \boldsymbol{L}_{R,1}, & i=j-1,\\ \boldsymbol{G}_{R,0}, & i=j=0,\\ \boldsymbol{G}_{R,1}, & i=j\neq 0,\\ \mu_E\boldsymbol{I}, & i=j+1,\\ \boldsymbol{O}, & otherwise,\end{cases}$$
$$\boldsymbol{H}_{R,2} = (\boldsymbol{H}_{R,2}^{(i,j)})_{i,j\in\mathcal{S}_E} = \begin{cases} \boldsymbol{L}_{R,1}, & i=j-1,\\ \boldsymbol{G}_{R,1}, & i=j=0,\\ \boldsymbol{G}_{R,2}, & i=j\neq 0,\\ \mu_E\boldsymbol{I}, & i=j+1,\\ \boldsymbol{O}, & otherwise.\end{cases}$$

$\boldsymbol{L}_{R,1}$, $\boldsymbol{L}_{R,2}$, $\boldsymbol{G}_{R,0}$, $\boldsymbol{G}_{R,1}$ and $\boldsymbol{G}_{R,2}$ are square matrices of order $(TH^U+1)^2$, as

$$\boldsymbol{L}_{R,1} = diag(\boldsymbol{L}_{R,11},\ldots,\boldsymbol{L}_{R,11}),\ \boldsymbol{L}_{R,2} = diag(\boldsymbol{O},\ldots,\boldsymbol{O},\boldsymbol{L}_{R,21}),$$
$$\boldsymbol{G}_{R,0} = (\boldsymbol{G}_{R,0}^{(i,j)})_{i,j\in\mathcal{S}_U} = \begin{cases} \lambda\boldsymbol{I}, & i=j-1,\\ \boldsymbol{M}_R, & i=j=0,\\ \boldsymbol{M}_R-\mu_U\boldsymbol{I}, & i=j\neq 0,\\ \mu_U\boldsymbol{I}, & i=j+1,\\ \boldsymbol{O}, & otherwise,\end{cases}$$
$$\boldsymbol{G}_{R,1} = \boldsymbol{G}_{R,0}-\mu_E\boldsymbol{I},\ \boldsymbol{G}_{R,2} = \boldsymbol{G}_{R,0}-2\mu_E\boldsymbol{I}.$$

$\boldsymbol{L}_{R,11}$, $\boldsymbol{L}_{R,21}$ and $\boldsymbol{M}_R$ are square matrices of order (TH^U+1), as

$$\boldsymbol{L}_{R,11} = diag(0,\ldots,0,\lambda),\ \boldsymbol{L}_{R,21} = \lambda\boldsymbol{I},$$
$$\boldsymbol{M}_R = (\boldsymbol{M}_R^{(i,j)})_{i,j\in\mathcal{S}_U} = \begin{cases} \lambda, & i=j-1,\\ -2\lambda, & i=j=0,\\ -2\lambda-\mu_U, & i=j\neq 0,\\ \mu_U, & i=j+1,\\ 0, & otherwise.\end{cases}$$

3.3 Infinitesimal generator in VDHS

The infinitesimal generator $\boldsymbol{Q}_H$ is derived as

$$\boldsymbol{Q}_H = \begin{pmatrix} \boldsymbol{H}_H & \boldsymbol{A}_H & \boldsymbol{O} & \cdots\\ \boldsymbol{D}_H & \boldsymbol{S}_H & \boldsymbol{A}_H & \ddots\\ \boldsymbol{O} & \boldsymbol{D}_H & \boldsymbol{S}_H & \ddots\\ \vdots & \ddots & \ddots & \ddots\end{pmatrix}, \tag{2}$$

where $\boldsymbol{D}_H$, $\boldsymbol{A}_H$, $\boldsymbol{S}_H$ and $\boldsymbol{H}_H$ are square matrices of order $(TH^U+1)^2(TH^E+1)^2$, as

$$\boldsymbol{D}_H = \mu_C\boldsymbol{I},\ \boldsymbol{S}_H = \boldsymbol{H}_H-\boldsymbol{D}_H,$$
$$\boldsymbol{A}_H = diag(\boldsymbol{O},\ldots,\boldsymbol{O},\boldsymbol{L}_{H,1}+\boldsymbol{L}_{H,2}),$$

$$\boldsymbol{H}_H = (\boldsymbol{H}_H^{(i,j)})_{i,j\in\mathcal{S}_E} = \begin{cases} \boldsymbol{K}_H, & i = j-1, \\ \boldsymbol{H}_{H,0}, & i = j = 0, \\ \boldsymbol{H}_{H,1}, & i = j,\ 1 \le i \le TH^E - 1, \\ \boldsymbol{H}_{H,2}, & i = j = TH^E, \\ \mu_E \boldsymbol{I}, & i = j+1, \\ \boldsymbol{O}, & otherwise. \end{cases}$$

$\boldsymbol{K}_H$, $\boldsymbol{H}_{H,0}$, $\boldsymbol{H}_{H,1}$ and $\boldsymbol{H}_{H,2}$ are square matrices of order $(TH^U+1)^2(TH^E+1)$, as

$$\boldsymbol{K}_H = diag(\boldsymbol{L}_{H,1}, \ldots, \boldsymbol{L}_{H,1}, \boldsymbol{L}_{H,1} + \boldsymbol{L}_{H,2}),$$

$$\boldsymbol{H}_{H,0} = (\boldsymbol{H}_{H,0}^{(i,j)})_{i,j\in\mathcal{S}_E} = \begin{cases} \boldsymbol{L}_{H,2}, & i = j-1, \\ \boldsymbol{G}_{H,0}, & i = j = 0, \\ \boldsymbol{G}_{H,1}, & i = j,\ 1 \le i \le TH^E, \\ \mu_E \boldsymbol{I}, & i = j+1, \\ \boldsymbol{O}, & otherwise, \end{cases}$$

$$\boldsymbol{H}_{H,1} = (\boldsymbol{H}_{H,1}^{(i,j)})_{i,j\in\mathcal{S}_E} = \begin{cases} \boldsymbol{L}_{H,2}, & i = j-1, \\ \boldsymbol{G}_{H,1}, & i = j = 0, \\ \boldsymbol{G}_{H,2}, & i = j,\ 1 \le i \le TH^E, \\ \mu_E \boldsymbol{I}, & i = j+1, \\ \boldsymbol{O}, & otherwise, \end{cases}$$

$$\boldsymbol{H}_{H,2} = (\boldsymbol{H}_{H,2}^{(i,j)})_{i,j\in\mathcal{S}_E}$$
$$= \begin{cases} \boldsymbol{L}_{H,2}, & i = j-1,\ 0 \le i \le TH^E - 2, \\ \boldsymbol{L}_{H,1} + \boldsymbol{L}_{H,2}, & i = TH^E - 1,\ j = TH^E, \\ \boldsymbol{G}_{H,1}, & i = j = 0, \\ \boldsymbol{G}_{H,2}, & i = j,\ 1 \le i \le TH^E, \\ \mu_E \boldsymbol{I}, & i = j+1, \\ \boldsymbol{O}, & otherwise. \end{cases}$$

$\boldsymbol{L}_{H,1}$, $\boldsymbol{L}_{H,2}$, $\boldsymbol{G}_{H,0}$, $\boldsymbol{G}_{H,1}$ and $\boldsymbol{G}_{H,2}$ are square matrices of order $(TH^U+1)^2$, as

$$\boldsymbol{L}_{H,1} = diag(\boldsymbol{O}, \ldots, \boldsymbol{O}, \boldsymbol{L}_{H,11}),$$
$$\boldsymbol{L}_{H,2} = diag(\boldsymbol{L}_{H,11}, \ldots, \boldsymbol{L}_{H,11}, \lambda \boldsymbol{I}),$$

$$\boldsymbol{G}_{H,0} = (\boldsymbol{G}_{H,0}^{(i,j)})_{i,j\in\mathcal{S}_U} = \begin{cases} \lambda \boldsymbol{I}, & i = j-1, \\ \boldsymbol{M}_H, & i = j = 0, \\ \boldsymbol{M}_H - \mu_U \boldsymbol{I}, & i = j \ne 0, \\ \mu_U \boldsymbol{I}, & i = j+1, \\ \boldsymbol{O}, & otherwise, \end{cases}$$

$$\boldsymbol{G}_{H,1} = \boldsymbol{G}_{H,0} - \mu_E \boldsymbol{I},\ \boldsymbol{G}_{H,2} = \boldsymbol{G}_{H,0} - 2\mu_E \boldsymbol{I}.$$

$\boldsymbol{L}_{H,11}$ and $\boldsymbol{M}_H$ are square matrices of order (TH^U+1), as

$$\boldsymbol{L}_{H,11} = diag(0, \ldots, 0, \lambda),$$

$$\boldsymbol{M}_H = (\boldsymbol{M}_H^{(i,j)})_{i,j\in\mathcal{S}_U} = \begin{cases} \lambda, & i = j-1, \\ -2\lambda, & i = j = 0, \\ -2\lambda - \mu_U, & i = j \ne 0, \\ \mu_U, & i = j+1, \\ 0, & otherwise. \end{cases}$$

3.4 Stability condition and stationary distribution

Based on the above results, the stability condition and stationary distribution in each model are given as follows. From (1) or (2), it can be seen that $\boldsymbol{X}(t)$ becomes a continuous time quasi birth-and-death process by taking $N_C(t)$ as the level and $(N_{U1}(t), N_{U2}(t), N_{E1}(t), N_{E2}(t))$ as the phase. We define $\boldsymbol{B}_X := \boldsymbol{D}_X + \boldsymbol{S}_X + \boldsymbol{A}_X$. Note that X expresses R or H. From the results in Latouche and Ramaswami [19], we can obtain the stationary distribution $\boldsymbol{\eta}_X$ satisfying $\boldsymbol{\eta}_X \boldsymbol{B}_X = \boldsymbol{0}$, $\boldsymbol{\eta}_X \boldsymbol{e} = 1$. Using $\boldsymbol{\eta}_X$, we then obtain the stability condition as

$$\boldsymbol{\eta}_X (\boldsymbol{A}_X - \boldsymbol{D}_X)\boldsymbol{e} < 0. \tag{3}$$

Assuming that (3) holds, we can then obtain the stationary distribution of $\boldsymbol{X}(t)$ [19]. For each state $(i_1, i_2, j_1, j_2, k) \in \mathcal{S}$, we define the stationary distribution of $\{\boldsymbol{X}(t) \mid t \ge 0\}$ as

$$\pi^X_{i_1,i_2,j_1,j_2,k} := \mathbb{P}(\boldsymbol{X}(t) = (i_1, i_2, j_1, j_2, k)).$$

Furthermore, we define $\boldsymbol{\pi}^X_k$ and $\boldsymbol{\pi}^X$ below.

$$\boldsymbol{\pi}^X_k := (\pi^X_{i_1,i_2,j_1,j_2,k})_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},\ \boldsymbol{\pi}^X := (\boldsymbol{\pi}^X_k)_{k\in\mathbb{N}_0}.$$

Using [19], we can obtain $\boldsymbol{\pi}^X_k$ as

$$\boldsymbol{\pi}^X_k = \begin{cases} \boldsymbol{\pi}^X_0, & k = 0, \\ \boldsymbol{\pi}^X_1 \boldsymbol{R}_X^{k-1}, & k \in \mathbb{N}, \end{cases} \tag{4}$$

where $\boldsymbol{R}_X$ is the minimal non-negative solution of

$$\boldsymbol{A}_X + \boldsymbol{R}_X \boldsymbol{S}_X + \boldsymbol{R}_X^2 \boldsymbol{D}_X = \boldsymbol{O}. \tag{5}$$

Using the method in [20], we can compute $\boldsymbol{R}_X$ and $\boldsymbol{\pi}^X_0$ numerically.

4 Approximation of performance metrics

We then derive the performance metrics with transmission latency. However, it is difficult to derive them directly because the transmission latency is constant and thus we cannot directly construct Markov chains for our models. We approximate them as follows: First, we derive performance metrics of the models without transmission latency which

are Markov chains presented in Sect. 3. Second, applying the performance metrics, and assuming communication links are infinite-server queues, we approximate the performance metrics with transmission latency. In particular, we derive the following performance metrics:

- Mean sojourn time.
- Probability Density Function (PDF) of the sojourn time.
- Cumulative distribution function of the sojourn time.

Our models in this paper are symmetry. Therefore, by focusing on the tasks arriving at UE1, we can obtain performance metrics for an arbitrary task (at UE2 also). To simplify the formulas, we define

$$
\begin{aligned}
d_{\mathrm{E1}} &:= d_{\mathrm{UE}} + d_{\mathrm{EU}},\\
d_{E2} &:= d_{UE} + d_{EE} + d_{EU},\\
d_C &:= d_{UE} + d_{EC} + d_{CE} + d_{EU}.
\end{aligned}
$$

4.1 Performance metrics in VDV

We then derive the performance metrics of the VDV model. We define the state space as

$$
\begin{aligned}
\mathcal{S}_{R,U} &:= \{i \in \mathcal{S}_U | i \neq TH^U\},\\
\mathcal{S}_{R,E} &:= \{j \in \mathcal{S}_E | j \neq TH^E\},\\
\mathcal{S}_{R,U}(i_1) &:= \{(i_1, i_2, j_1, j_2) \in \mathcal{S}' | i_1 \neq TH^U\},\\
\mathcal{S}_{R,E}(j_1) &:= \{(i_1, i_2, j_1, j_2) \in \mathcal{S}' | i_1 = TH^U, j_1 \neq TH^E\},\\
\mathcal{S}_{R,C} &:= \{(i_1, i_2, j_1, j_2) \in \mathcal{S}' | i_1 = TH^U, j_1 = TH^E\}.
\end{aligned}
$$

Note that, in the above state space, we pay attention to the tasks which arrive at the UE1 and classify the situations. Based on the classifications, we define the row vectors $\boldsymbol{e}_{R,U}(i_1), \boldsymbol{e}_{R,E}(j_1), \boldsymbol{e}_{R,C} \in \{0,1\}^{(TH^U+1)^2(TH^E+1)^2}$ as

$$
\begin{aligned}
\boldsymbol{e}_{R,U}(i_1) &:= (1_{\mathcal{S}_{R,U}(i_1)}(i_1, i_2, j_1, j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},\\
\boldsymbol{e}_{R,E}(j_1) &:= (1_{\mathcal{S}_{R,E}(j_1)}(i_1, i_2, j_1, j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},\\
\boldsymbol{e}_{R,C} &:= (1_{\mathcal{S}_{R,C}}(i_1, i_2, j_1, j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},
\end{aligned}
$$

where $1_A(i)$ represents the indicator function on set A.

First, we derive the mean sojourn time $\mathbb{E}[T_R]$. In this system, tasks can be served at UEs, edges or cloud. Hence, we can represent $\mathbb{E}[T_R]$ as

$$
\mathbb{E}[T_R] = T_{R,U} + T_{R,E} + T_{R,C}, \tag{6}
$$

where $T_{R,U}$, $T_{R,E}$ and $T_{R,C}$ represent the mean of the sojourn time at UEs, edges and the cloud respectively, as

$$
T_{R,U} := \sum_{i\in\mathcal{S}_{R,U}} T_{R,U}(i)\mathbb{P}(N_{U1}(t) = i), \tag{7}
$$

$$
T_{R,E} := \sum_{j\in\mathcal{S}_{R,E}} T_{R,E}(j)\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = j), \tag{8}
$$

$$
\begin{aligned}
T_{R,C} := \sum_{k\in\mathbb{N}_0} T_{R,C}(k)\mathbb{P}(&N_{U1}(t) = TH^U, N_{E1}(t) = TH^E,\\
&N_C(t) = k).
\end{aligned} \tag{9}
$$

$T_{R,U}(i)$ shows the mean sojourn time that are experienced by a task that sees i tasks at UE1 upon arrival. Similarly, $T_{R,E}(j)$ and $T_{R,C}(k)$ represent the mean sojourn time experienced by as task that sees j tasks at edge1 or k tasks at the cloud upon arrival to edge1 or the cloud, as

$$
\begin{aligned}
T_{R,U}(i) &:= \mathbb{E}[T_R | N_{U1}(t) = i],\\
T_{R,E}(j) &:= \mathbb{E}[T_R | N_{U1}(t) = TH^U, N_{E1}(t) = j],\\
T_{R,C}(k) &:= \mathbb{E}[T_R | N_{U1}(t) = TH^U, N_{E1}(t) = TH^E, N_C(t) = k].
\end{aligned}
$$

We consider the case where there are i existing tasks at UE1 upon the arrival of a task and the task is served at UE1. The service time of each task at UE1 follows an i.i.d. exponential distribution with parameter μ_U. Therefore, the sojourn time follows an Erlang distribution with parameter $i+1$ and μ_U. Thus, we obtain

$$
T_{R,U}(i) = \tfrac{i+1}{\mu_U},\ i \in \mathcal{S}_{R,U}. \tag{10}
$$

In the same way, we obtain

$$
T_{R,E}(j) = \tfrac{j+1}{\mu_E} + d_{E1},\ j \in \mathcal{S}_{R,E}, \tag{11}
$$

$$
T_{R,C}(k) = \tfrac{k+1}{\mu_C} + d_C,\ k \in \mathbb{N}_0. \tag{12}
$$

Furthermore, using (4), we can obtain each probability as

$$
\mathbb{P}(N_{U1}(t) = i) = (\boldsymbol{\pi}_0^R + \boldsymbol{\pi}_1^R(\boldsymbol{I} - \boldsymbol{R}_R)^{-1})\boldsymbol{e}_{R,U}(i),\ i \in \mathcal{S}_{R,U}, \tag{13}
$$

$$
\begin{aligned}
&\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = j)\\
&\quad = (\boldsymbol{\pi}_0^R + \boldsymbol{\pi}_1^R(\boldsymbol{I} - \boldsymbol{R}_R)^{-1})\boldsymbol{e}_{R,E}(j),\ j \in \mathcal{S}_{R,E},
\end{aligned} \tag{14}
$$

$$
\begin{aligned}
&\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = TH^E, N_C(t) = k)\\
&\quad = \begin{cases} \boldsymbol{\pi}_0^R \boldsymbol{e}_{R,C}, & k = 0,\\ \boldsymbol{\pi}_1^R \boldsymbol{R}_R^{k-1} \boldsymbol{e}_{R,C}, & k \in \mathbb{N}. \end{cases}
\end{aligned} \tag{15}
$$

Substituting (10)–(15) to (7)–(9), we can represent

$$
T_{R,U} = \sum_{i\in\mathcal{S}_{R,U}} \tfrac{i+1}{\mu_U}(\boldsymbol{\pi}_0^R + \boldsymbol{\pi}_1^R(\boldsymbol{I} - \boldsymbol{R}_R)^{-1})\boldsymbol{e}_{R,U}(i), \tag{16}
$$

$$
T_{R,E} = \sum_{j\in\mathcal{S}_{R,E}} (\tfrac{j+1}{\mu_E} + d_{E1})(\boldsymbol{\pi}_0^R + \boldsymbol{\pi}_1^R(\boldsymbol{I} - \boldsymbol{R}_R)^{-1})\boldsymbol{e}_{R,E}(j), \tag{17}
$$

$$T_{R,C} = [\frac{1}{\mu_C}[\boldsymbol{\pi}_0^R + \boldsymbol{\pi}_1^R\{(\boldsymbol{I} - \boldsymbol{R}_R)^{-1} + \{(\boldsymbol{I} - \boldsymbol{R}_R)^{-1}\}^2\}] + d_C(\boldsymbol{\pi}_0^R + \boldsymbol{\pi}_1^R(\boldsymbol{I} - \boldsymbol{R}_R)^{-1})]\boldsymbol{e}_{R,C}. \quad (18)$$

Next, we derive the PDF $f_R(x)$. In the same way as we derived the mean sojourn time, we focus on the tasks which arrive at UE1 and can derive $f_R(x)$ as

$$f_R(x) = f_{R,U}(x) + f_{R,E}(x) + f_{R,C}(x), \ x \geq 0, \quad (19)$$

where $f_{R,U}(x)$, $f_{R,E}(x)$ and $f_{R,C}(x)$ represent the PDF of the sojourn time at UEs, edges and the cloud respectively, as

$$f_{R,U}(x) := \sum_{i \in \mathcal{S}_{R,U}} f_{R,U}(x,i)\mathbb{P}(N_{U1}(t) = i), \quad (20)$$

$$f_{R,E}(x) := \sum_{j \in \mathcal{S}_{R,E}} f_{R,E}(x,j)\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = j), \quad (21)$$

$$f_{R,C}(x) := \sum_{k \in \mathbb{N}_0} f_{R,C}(x,k)\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = TH^E, N_C(t) = k). \quad (22)$$

$f_{R,U}(x,i)$ shows the PDF of the sojourn time of a task at UE1 that sees i tasks at UE1 upon arrival. Similarly, $f_{R,E}(x,j)$ and $f_{R,C}(x,k)$ represent the PDF of the sojourn time that there are j tasks at edge1 or there are k tasks at the cloud when the task arrives at edge1 or the cloud, respectively, as

$$f_{R,U}(x,i) := f_R(x|N_{U1}(t) = i),$$
$$f_{R,E}(x,j) := f_R(x|N_{U1}(t) = TH^U, N_{E1}(t) = j),$$
$$f_{R,C}(x,k) := f_R(x|N_{U1}(t) = TH^U, N_{E1}(t) = TH^E, N_C(t) = k).$$

In the same way as we derived the mean sojourn time, using $\boldsymbol{R}_R$, we can express the PDF of the sojourn time. However, the numerical analysis of this method is unstable. We therefore truncate the infinite vectors $(\boldsymbol{\pi}_k^R)_{k\in\mathbb{N}_0}$ to the vector of their first (k_R^*+1) elements. The constant k_R^* is determined as

$$k_R^* := \inf\{n \in \mathbb{N}_0 | 1 - \sum_{k=0}^{n} |\boldsymbol{\pi}_k^R| < 10^{-6}\}, \quad (23)$$

where $|\boldsymbol{\pi}_k^R| := \sum_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'} \pi_{i_1,i_2,j_1,j_2,k}^R$. If the sum of $(\boldsymbol{\pi}_k^R)_{k\geq k_R^*+1}$ is sufficient small, we use $(\boldsymbol{\pi}_k^R)_{k\in\{0,1,\ldots,k_R^*\}}$ to approximate the PDF of the sojourn time. We define $\mathcal{K}_R := \{0,1,\ldots,k_R^*\}$.

First, we derive $f_{R,U}(x,i)$. As noted in the derivation of the mean sojourn time, when there are i tasks at a UE on the arrival of a task, the sojourn time at the UE of the arriving task follows the Erlang distribution with parameters $i+1$ and μ_U. The same discussion could be applied to the task arriving at the edge or the cloud. Thus, we can express

$$f_{R,U}(x,i) = \frac{(\mu_U x)^i}{i!}\mu_U e^{-\mu_U x}, \ i \in \mathcal{S}_{R,U}, x \geq 0, \quad (24)$$

$$f_{R,E}(x,j) = \frac{(\mu_E(x-d_{E1}))^j}{j!}\mu_E e^{-\mu_E(x-d_{E1})}, \ j \in \mathcal{S}_{R,E}, x \geq d_{E1}, \quad (25)$$

$$f_{R,C}(x,k) = \frac{(\mu_C(x-d_C))^k}{k!}\mu_C e^{-\mu_C(x-d_C)}, \ k \in \mathcal{K}_R, x \geq d_C. \quad (26)$$

Furthermore, using $(\boldsymbol{\pi}_{*,k}^R)_{k\in\mathcal{K}_R}$, we can express

$$\mathbb{P}(N_{U1}(t) = i) = \sum_{k\in\mathcal{K}_R} \boldsymbol{\pi}_{*,k}^R \boldsymbol{e}_{R,U}(i), \ i \in \mathcal{S}_{R,U}, \quad (27)$$

$$\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = j) = \sum_{k\in\mathcal{K}_R} \boldsymbol{\pi}_{*,k}^R \boldsymbol{e}_{R,E}(j), \ j \in \mathcal{S}_{R,E}, \quad (28)$$

$$\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = TH^E, N_C(t) = k) = \boldsymbol{\pi}_{*,k}^R \boldsymbol{e}_{R,C1}, \ k \in \mathcal{K}_R. \quad (29)$$

Using the results in (24)–(29), we can get (20)–(22). Moreover, substituting (19) with (20)–(22), we can get the PDF of the sojourn time in the whole system.

Finally, using the PDF in (19), we obtain the cumulative distribution function for the sojourn time in VDV as

$$F_R(\alpha) = P(T_R \leq \alpha) = \int_0^\alpha f_R(x)dx. \quad (30)$$

$F_R(\alpha)$ represents the delay satisfaction ratio (DSR) of the VDV model, i.e., the ratio of tasks that have the sojourn time within a delay constraint, α. The performance metric, DSR, is important for providing service level agreement between the provider and customers.

4.2 Performance metrics in VDHS

To derive performance metrics in the VDHS model, we define the state space as

$$\mathcal{S}_{H,U} := \{i \in \mathcal{S}_U | i \neq TH^U\},$$
$$\mathcal{S}_{H,E} := \{j \in \mathcal{S}_E | j \neq TH^E\},$$
$$\mathcal{S}_{H,U}(i_1) := \{(i_1,i_2,j_1,j_2) \in \mathcal{S}' | i_1 \neq TH^U\},$$
$$\mathcal{S}_{H,E1}(j_1) := \{(i_1,i_2,j_1,j_2) \in \mathcal{S}' | i_1 = TH^U, j_1 \neq TH^E\},$$
$$\mathcal{S}_{H,E2}(j_2) := \{(i_1,i_2,j_1,j_2) \in \mathcal{S}' | i_1 = TH^U, j_1 = TH^E, j_2 \neq TH^E\},$$
$$\mathcal{S}_{H,C} := \{(i_1,i_2,j_1,j_2) \in \mathcal{S}' | i_1 = TH^U, j_1 = j_2 = TH^E\}.$$

As with the VDV model, in the above state space, we focus on the tasks which arrive at UE1 and classify the situations.

Based on classification, we define the row vectors $\boldsymbol{e}_{H,U}(i_1)$, $\boldsymbol{e}_{H,E1}(j_1)$, $\boldsymbol{e}_{H,E2}(j_2)$, $\boldsymbol{e}_{H,C} \in \{0,1\}^{(TH^{UE}+1)^2(TH^E+1)^2}$ as

$$\boldsymbol{e}_{H,U}(i_1) := (1_{\mathcal{S}_{H,U}(i_1)}(i_1,i_2,j_1,j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},$$
$$\boldsymbol{e}_{H,E1}(j_1) := (1_{\mathcal{S}_{H,E1}(j_1)}(i_1,i_2,j_1,j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},$$
$$\boldsymbol{e}_{H,E2}(j_2) := (1_{\mathcal{S}_{H,E2}(j_2)}(i_1,i_2,j_1,j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'},$$
$$\boldsymbol{e}_{H,C} := (1_{\mathcal{S}_{H,C}}(i_1,i_2,j_1,j_2))'_{(i_1,i_2,j_1,j_2)\in\mathcal{S}'}.$$

First, we derive the mean sojourn time $\mathbb{E}[T_H]$. In the same way, we can represent $\mathbb{E}[T_H]$ as

$$\mathbb{E}[T_H] = T_{H,U} + T_{H,E} + T_{H,C}, \tag{31}$$

where

$$T_{H,U} := \sum_{i\in\mathcal{S}_{H,U}} T_{H,U}(i)\mathbb{P}(N_{U1}(t)=i), \tag{32}$$

$$T_{H,E} := \sum_{j_1\in\mathcal{S}_{H,E}} T_{H,E1}(j_1)\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=j_1) + \sum_{j_2\in\mathcal{S}_{H,E}} T_{H,E2}(j_2)\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=TH^E, N_{E2}(t)=j_2), \tag{33}$$

$$T_{H,C} := \sum_{k\in\mathbb{N}_0} T_{H,C}(k)\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=N_{E2}(t)=TH^E, N_C(t)=k),$$
$$T_{H,U}(i) := \mathbb{E}[T_H|N_{U1}(t)=i],$$
$$T_{H,E1}(j_1) := \mathbb{E}[T_H|N_{U1}(t)=TH^U, N_{E1}(t)=j_1],$$
$$T_{H,E2}(j_2) := \mathbb{E}[T_H|N_{U1}(t)=TH^U, N_{E1}(t)=TH^E, N_{E2}(t)=j_2],$$
$$T_{H,C}(k) := \mathbb{E}[T_H|N_{U1}(t)=TH^U, N_{E1}(t)=N_{E2}(t)=TH^E, N_C(t)=k]. \tag{34}$$

We obtain

$$T_{H,U}(i) = \tfrac{i+1}{\mu_U},\ i\in\mathcal{S}_{H,U}, \tag{35}$$
$$T_{H,E1}(j_1) = \tfrac{j_1+1}{\mu_E} + d_{E1},\ j_1\in\mathcal{S}_{H,E}, \tag{36}$$
$$T_{H,E2}(j_2) = \tfrac{j_2+1}{\mu_E} + d_{E2},\ j_2\in\mathcal{S}_{H,E}, \tag{37}$$
$$T_{H,C}(k) = \tfrac{k+1}{\mu_C} + d_C,\ k\in\mathbb{N}_0, \tag{38}$$

and

$$\mathbb{P}(N_{U1}(t)=i) = (\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,U}(i),\ i\in\mathcal{S}_{H,U}, \tag{39}$$

$$\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=j_1) = (\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,E}(j_1),\ j_1\in\mathcal{S}_{H,E}, \tag{40}$$

$$\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=TH^E, N_{E2}(t)=j_2) = (\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,E}(j_2),\ j_2\in\mathcal{S}_{H,E}, \tag{41}$$

$$\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=N_{E2}(t)=TH^E, N_C(t)=k) = \begin{cases}\boldsymbol{\pi}_0^H\boldsymbol{e}_{H,C}, & k=0,\\ \boldsymbol{\pi}_1^H\boldsymbol{R}_H^{k-1}\boldsymbol{e}_{H,C}, & k\in\mathbb{N}.\end{cases} \tag{42}$$

Substituting (32)-(34) with (35)-(42), we can represent

$$T_{H,U} = \sum_{i\in\mathcal{S}_{H,U}} \tfrac{i+1}{\mu_U}(\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,U}(i), \tag{43}$$

$$T_{H,E} = \sum_{j_1\in\mathcal{S}_{H,E}} (\tfrac{j_1+1}{\mu_E} + d_{E1})(\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,E}(j_1) + \sum_{j_2\in\mathcal{S}_{H,E}} (\tfrac{j_2+1}{\mu_E} + d_{E2})(\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,E}(j_2), \tag{44}$$

$$T_{H,C} = \tfrac{1}{\mu_C}[\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H\{(\boldsymbol{I}-\boldsymbol{R}_H)^{-1} + \{(\boldsymbol{I}-\boldsymbol{R}_H)^{-1}\}^2\}]\boldsymbol{e}_{H,C} + d_C(\boldsymbol{\pi}_0^H + \boldsymbol{\pi}_1^H(\boldsymbol{I}-\boldsymbol{R}_H)^{-1})\boldsymbol{e}_{H,C}. \tag{45}$$

From (43)-(45), we obtain $\mathbb{E}[T_H]$. We next derive the PDF $f_H(x)$, which we can express in the same way as the VDV model:

$$f_H(x) = f_{H,U}(x) + f_{H,E}(x) + f_{H,C}(x),\ x\geq 0, \tag{46}$$

where

$$f_{H,U}(x) := \sum_{i\in\mathcal{S}_{H,U}} f_{H,U}(x,i)\mathbb{P}(N_{U1}(t)=i), \tag{47}$$

$$f_{H,E}(x) := \sum_{j\in\mathcal{S}_{H,E}} \{f_{H,E1}(x,j)\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=j) + f_{H,E2}(x,j)\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=TH^E, N_{E2}(t)=j)\}, \tag{48}$$

$$f_{H,C}(x) := \sum_{k\in\mathbb{N}_0} f_{H,C}(x,k)\mathbb{P}(N_{U1}(t)=TH^U, N_{E1}(t)=N_{E2}(t)=TH^E, N_C(t)=k), \tag{49}$$

$$f_{H,U}(x,i) := f_H(x|N_{U1}(t)=i), \tag{50}$$

$$f_{H,E1}(x,j) := f_H(x|N_{U1}(t)=TH^U, N_{E1}(t)=j), \tag{51}$$

$$f_{H,E2}(x,j) := f_H(x|N_{U1}(t)=TH^U, N_{E1}(t)=TH^E, N_{E2}(t)=j), \tag{52}$$

$$f_{H,C}(x,k) := f_H(x|N_{U1}(t)=TH^U, N_{E1}(t)=N_{E2}(t)=TH^E, N_C(t)=k). \tag{53}$$

We then define the constant k_H^* as

$$k_H^* := \inf\{n\in\mathbb{N}_0 | 1-\sum_{k=0}^{n}|\boldsymbol{\pi}_k^H| < 10^{-6}\}. \tag{54}$$

Using $(\boldsymbol{\pi}_k^H)_{k\in\{0,1,\ldots,k_H^*\}}$, we can approximate the PDF of the sojourn time. We define $\mathcal{K}_H := \{0, 1, \ldots, k_H^*\}$. First, we derive $f_{H,U}(x,i)$, $f_{H,E1}(x,j)$, $f_{H,E2}(x,j)$, $f_{H,C}(x,k)$. In the same way as the VDV model, we can then express

$$f_{H,U}(x,i) = \frac{(\mu_U x)^i}{i!}\mu_U e^{-\mu_U x},\ i \in \mathcal{S}_{H,U}, x \geq 0, \tag{55}$$

$$f_{H,E1}(x,j) = \frac{(\mu_E(x-d_{E1}))^j}{j!}\mu_U e^{-\mu_U(x-d_{E1})},\ j \in \mathcal{S}_{H,E}, x \geq d_{E1}, \tag{56}$$

$$f_{H,E2}(x,j) = \frac{(\mu_E(x-d_{E2}))^j}{j!}\mu_U e^{-\mu_U(x-d_{E2})},\ j \in \mathcal{S}_{H,E}, x \geq d_{E2}, \tag{57}$$

$$f_{H,C}(x,k) = \frac{(\mu_C(x-d_C))^k}{k!}\mu_C e^{-\mu_C(x-d_C)}, k \in \mathcal{K}_H, x \geq d_C. \tag{58}$$

And, using $(\pi_k^H)_{k\in\mathcal{K}_H}$, we can express

$$\mathbb{P}(N_{U1}(t) = i) = \sum_{k\in\mathcal{K}_H} \boldsymbol{\pi}_{*,k}^H \boldsymbol{e}_{H,U}(i),\ i \in \mathcal{S}_{H,U}, \tag{59}$$

$$\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = j) = \sum_{k\in\mathcal{K}_H} \boldsymbol{\pi}_{*,k}^H \boldsymbol{e}_{H,E1}(j),\ j \in \mathcal{S}_{H,E}, \tag{60}$$

$$\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = TH^E, N_{E2}(t) = j) = \sum_{k\in\mathcal{K}_H} \boldsymbol{\pi}_{*,k}^H \boldsymbol{e}_{H,E2}(j),\ j \in \mathcal{S}_{H,E}, \tag{61}$$

$$\mathbb{P}(N_{U1}(t) = TH^U, N_{E1}(t) = N_{E2}(t) = TH^E, N_C(t) = k) = \boldsymbol{\pi}_{*,k}^H \boldsymbol{e}_{H,C}, k \in \mathcal{K}_H. \tag{62}$$

Using the results in (55)–(62), we can get (47)–(49). Moreover, substituting (46) with (47)–(49), we can get the PDF of the sojourn time in the whole system.

Finally, using the PDF in (46), we obtain the cumulative distribution functions for the sojourn time in VDHS as

$$F_H(\alpha) = P(T_H \leq \alpha) = \int_0^\alpha f_H(x)dx. \tag{63}$$

$F_H(\alpha)$ represents the DSR of the VDHS model, i.e., the ratio of tasks that have the sojourn time within a delay constraint, α.

5 Evaluations

In this section, we give analytical results of the performance metrics for each model as derived in the above section. The performance metrics investigated in the evaluation include the mean sojourn time and DSR. We calculated the mean sojourn time based on (6) for VDV and (31) for VDHS. We calculated DSR based on (30) for VDV and (63) for VDHS.

To verify the accuracy of the derivations, we also carried out simulations. The simulation is needed because we could not incorporate the uplink transmission latency and the downlink transmission latency in our stochastic models. The latency is fixed and thus it does not have the Markovian property. We only model the stochastic behaviors of tasks at a UE, an edge and the cloud and compute the sojourn time distribution in these three components. Using these sojourn time distributions, we approximate the sojourn time distribution in the whole system by adding the uplink and downlink transmission latency. That is, in our stochastic process, once a task is offloaded from a UE to an edge, the task immediately enters the edge. However, in a realistic environment, that task must spend an uplink transmission latency before it reaches the edge. To verify our approximation, we carried out simulation that considers the real behaviors of tasks. The simulator is our developed program which is a Monte Carlo simulation using Python.

Table 3 Parameter setting

μ_U	μ_E	μ_C	d_{UE}	d_{EC}	d_{CE}	d_{EU}	d_{EE}
3	7	100	0.05	0.5	0.5	0.05	0.05

The simulation results are an average of 30 identical experiments with random numbers. In each experiment, we ran 110,000 tasks and discarded the first 10,000 tasks for stationary behavior. The parameter setting is given in Table 3. The unit of service rates, μ_U, μ_E, and μ_C, is 1/ms while the unit of the delay, d_{UE}, d_{EC}, d_{CE}, d_{EU}, and d_{EE}, is ms.

5.1 Analysis versus simulation

This subsection covers the verification of the analysis by simulation. Figure 5 shows the mean sojourn time for the VDV and VDHS models with analysis and simulation when the task arrival rate λ is changed. It is very reasonable that the mean sojourn time increases as the task arrival rate increases. The results show that there is little difference between the analytical and simulation results in the VDV model while there is an observable gap in the VDHS model. In addition, it can be easily seen that the mean sojourn time in VDHS is less than that in VDV when setting up the same threshold. The main reason is that VDV can only carry out vertical offloading, i.e., a heavily-loaded edge only offloads tasks to the cloud. However, other than the vertical offloading, VDHS can also carry out horizontal offloading, that is, a heavily-loaded edge can offload tasks to another lightly-loaded edge. When the arrival rate is 5, the improvement on the mean sojourn time of VDHS can achieve 30%, compared with that of VDV.

Figure 6 shows DSR when the task arrival rate λ is changed. We observe that DSR decreases as the arrival rate λ increases. We also confirm that VDHS has a higher DSR than VDV, representing that VDHS outperforms VDV in terms of

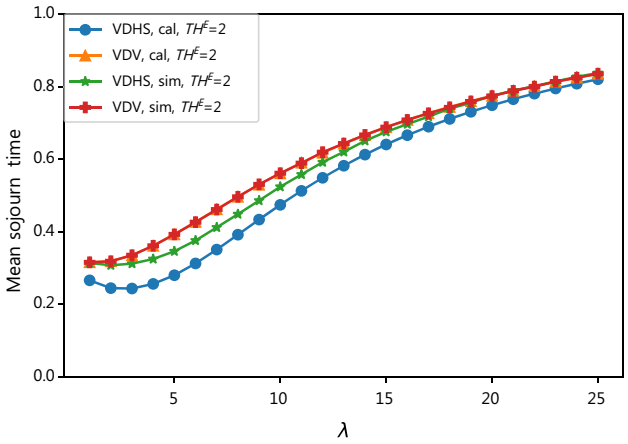

Fig. 5 Analytical and simulation results concerning the mean sojourn time

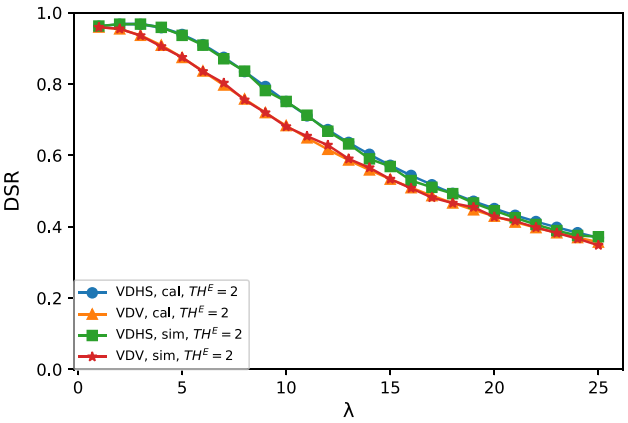

Fig. 6 Analytical and simulation results concerning DSR

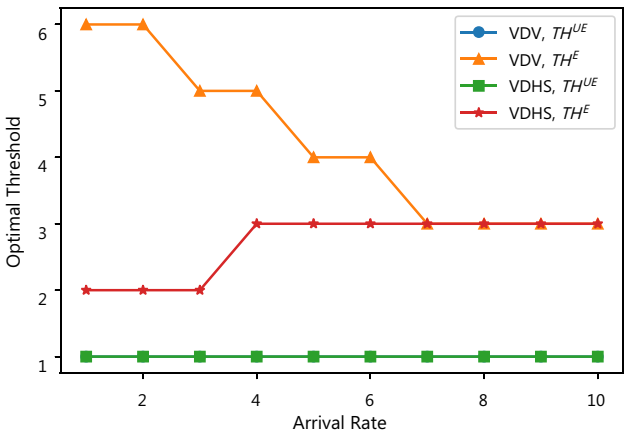

Fig. 7 The optimal threshold vs. arrival rate

satisfying the delay constraint. When the arrival rate is 5, the improvement on DSR of VDHS can achieve 11%, compared with that of VDV. Also the results show that the analytical and simulation results match well for the VDV and VDHS models, verifying that the correctness of our analysis on DSR.

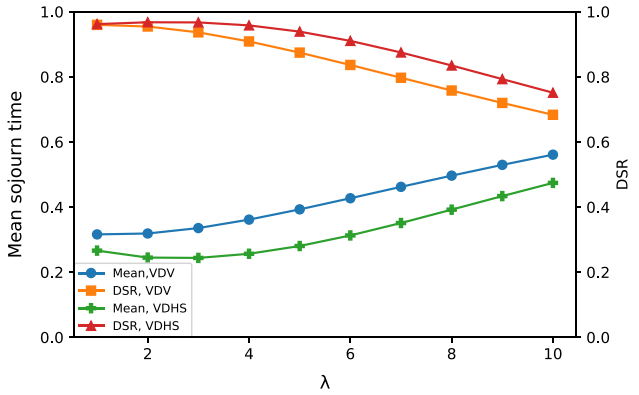

Fig. 8 The mean sojourn time and DSR vs. arrival rate

5.2 The effects of task arrival rate

Figure 7 shows the optimal threshold when the task arrival rate λ is changed. We can see that the optimal threshold TH^{UE} is always equal to one, no matter what model is adopted. That is, a newly-arriving task which encounters a busy UE should be offloaded to the edge/cloud, because the UE has low service capacity. Thus, the sojourn time, which is composed of the waiting time and the service time, in this busy UE will be longer than the time to be offloaded to the edge/cloud. On the other hand, both models have high thresholds in edges, TH^E. However, the optimal threshold TH^E of VDV decreases while that of VDHS increases as the task arrival rate increases. For VDV, when the task arrival rate is low, the edge can serve its tasks without the helps of the cloud, resulting in a high threshold TH^E. When the task arrival rate is high, the edge can not accommodate the load of many arriving tasks, so it will try to offload some tasks to the cloud, resulting in the decrease of TH^E. On the other hand, for VDHS, when the task arrival rate is low, and if an edge's load is higher than the load of other edges, tasks can be offloaded to other edges for a better sojourn time, resulting in a low threshold. When the task arrival rate increases, all edges become heavily-loaded, so it is very unlikely to offload tasks to other edges and more likely to offload tasks to the cloud. Therefore, in this case, VDHS will behave like VDV, resulting in the same optimal threshold TH^E. This causes VDV to have a higher TH^E than VDHS when the task arrival rate is low, and they have the same TH^E when it is high.

Figure 8 shows the mean sojourn time and DSR as the task arrival rate λ is changed when optimal thresholds (Fig. 7) are given. As a result the mean sojourn time of both models increases as the task arrival rate increases because the overall system load increases. The mean sojourn time of VDV is larger than that of VDHS because the former only carries out vertical offloading while the latter carries out vertical and horizontal offloading. VDHS can offload some tasks to other

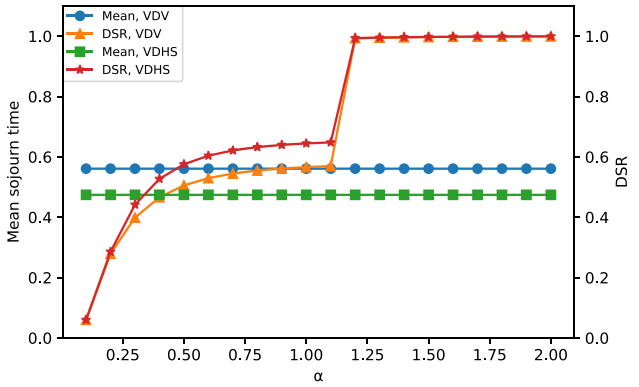

Fig. 9 The mean sojourn time and DSR vs. delay constraint

Table 4 Offloading ratio and mean sojourn time at each server in VDV and VDHS

Model	Role UE	Default edge	Targeted edge	Cloud
VDV(simulation)	13%	34%	0%	53%
	0.334	0.460	NA	1.114
VDV(analysis)	13%	34%	0%	53%
	0.333	0.459	NA	1.114
VDHS(simulation)	13%	25%	10%	52%
	0.335	0.620	0.656	1.123
VDHS(analysis)	13%	21%	14%	52%
	0.333	0.624	0.681	1.115

lightly-loaded edges to reduce the mean sojourn time. We observe that the DSR decreases as the arrival rate increases. It means that a higher task arrival rate generates a smaller DSR because more tasks cause more serious congestion.

5.3 The effects of delay constraint

Figure 9 shows the mean sojourn time and DSR when the delay constraint α is changed. We observe that the mean sojourn time of VDV and VDHS are stable, no matter which value of the delay constraint. It is reasonable because the delay constraint will not affect the mean sojourn time. However, it is obvious that DSR increases as the delay constraint increases because more tasks can satisfy a higher delay constraint. Also DSR of VDHS is higher than that of VDV, representing that VDHS outperforms VDV in terms of DSR. The reason is the same as that mentioned in Figs. 5 and 6. Furthermore, Fig. 9 shows an interesting jump in both VDV and VDHS models at $\alpha = 1.00$. This is due to the tasks that are processed in the cloud. These tasks need an uplink transmission latency from an edge to the cloud and a downlink transmission latency from the cloud to an edge. The total transmission latency for uplink and downlink is $d_{EC} + d_{CE} = 1.00$ ms in our parameter setting. Thus, when the delay constraint exceeds 1, some tasks can be offloaded to the cloud, so DSR has a big jump around $\alpha = 1.00$.

5.4 Offloading ratios

Table 4 shows the offloading ratio and the mean sojourn time at each server in the VDV and VDHS models, derived by analyses and simulations. We set the arrival rate as $\lambda = 20$ and the threshold of the VDHS model $(TH^U, TH^E) = (1, 4)$, and the queue-length threshold of the VDV model $(TH^U, TH^E) = (1, 3)$. In each cell of the table, the upper left is the offloading ratio, and the lower right is the mean sojourn time at each server. From this table, we can see that the offloading ratios in each layer of a UE, an edge, and the cloud are almost the same in the two models. It can also be seen that the mean sojourn time at the server in the upper tier is larger than that in the lower tier.

6 Conclusion

In this paper, we modelled MEC architecture with the queue-length thresholds at UEs and edges considering the transmission latency. It is difficult to directly derive the performance metrics in our models. We therefore approximated them. Based on the approximation, we evaluated the optimal thresholds of UEs and edges.

We can summary some interesting observations from results: (1) The analytical and simulation match well, so the correctness of our analysis can be verified; (2) VDHS always has lower sojourn time than VDV since the former does both vertical and horizontal offloading while the latter only does vertical offloading. VDHS can improve 30% at most on the mean sojourn time and 11% at most on DSR; (3) The threshold TH^U is always 1, no matter which model, because the UE's capacity is low in our experiment, so the task will be offloaded to an edge if it encounters a busy UE; and (4) TH^E in VDHS is always less than TH^E in VDV because VDHS can earlier offload the tasks to other lightly-loaded edges.

In future works, we look to proposing other approximation methods for our models. In this paper, we analyzed MEC using a matrix analysis method. However, it is difficult to analyze as the number of dimensions in the Markov chain increases. We then need to find other approximation method.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Hu, F., Deng, Y., Saad, W., Bennis, M., & Aghvami, A. H. (2020). Cellular-connected wireless virtual reality: Requirements, challenges, and solutions. *IEEE Communications Magazine., 58*, 105–111.
2. Ahmadi, S. (2019). *5G NR: architecture, technology, implementation, and operation of 3GPP new radio standards* (pp. 22–32). Cambridge: Academic Press.
3. Chataut, R., & Akl, R. (2020). Massive MIMO systems for 5G and beyond networks-overview, recent trends, challenges, and future research direction. *Sensors, 20*, 2753.
4. Pham, Q. V., Fang, F., Ha, V. N., Piran, M. J., Le, M., Le, L. B., & Ding, Z. (2020). A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access, 8*, 116974–117017.
5. Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing-A key technology towards 5G. *ETSI White Paper*. pp. 1–16.
6. Giust, F., Verin, G., Antevski, K., Chou, J., Fang, Y., Featherstone, W., & Zhou, Z. (2018). MEC deployments in 4G and evolution towards 5G. *ETSI White paper.* pp. 1–24.
7. Kar, B., Lin, Y. D., & Lai, Y. C. (2020). OMNI: Omni-directional dual cost optimization of two-tier federated cloud-edge systems. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. pp. 1–7.
8. Rafiq, A., Ping, W., Min, W., Hong, S. H., & Josbert, N. N. (2021). Optimizing energy consumption and latency based on computation offloading and cell association in MEC enabled Industrial IoT environment. In *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)* (pp. 10–14). IEEE.
9. Qin, M., Cheng, N., Jing, Z., Yang, T., Xu, W., Yang, Q., & Rao, R. R. (2020). Service-oriented energy-latency tradeoff for iot task partial offloading in mec-enhanced multi-rat networks. *IEEE Internet of Things Journal, 8*, 1896–1907.
10. Lee, G., Saad, W., & Bennis, M. (2017). An online secretary framework for fog network formation with minimal latency. In *Proceedings of 2017 IEEE International Conference on Communications*. pp. 1–6.
11. Yousefpour, A., Ishigaki, G., Gour, R., & Jue, J. P. (2018). On reducing IoT service delay via fog offloading. *IEEE Internet of Things Journal, 5*, 998–1010.
12. Hwang, R., Lai, Y., & Lin, Y. *Offloading Optimization with Delay Distribution in the 3-tier Federated Cloud, Edge, and Fog Systems. preprint.*
13. Elgendy, I. A., Zhang, W. Z., He, H., Gupta, B. B., & Abd El-Latif, A. A. (2021). Joint computation offloading and task caching for multi-user and multi-task MEC systems: Reinforcement learning-based algorithms. *Wireless Networks, 27*, 2023–2038.
14. Li, Z., Chang, V., Ge, J., Pan, L., Hu, H., & Huang, B. (2021). Energy-aware task offloading with deadline constraint in mobile edge computing. *EURASIP Journal on Wireless Communications and Networking, 56*, 1–24.
15. Wang, L., Wang, K., Pan, C., Xu, W., Aslam, N., & Hanzo, L. (2020). Multi-agent deep reinforcement learning based trajectory planning for multi-UAV assisted mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking, 7*, 73–84.
16. Zhang, Y., Di, B., Zheng, Z., Lin, J., & Song, L. (2019). Joint data offloading and resource allocation for multi-cloud heterogeneous mobile edge computing using multi-agent reinforcement learning. In *2019 IEEE Global Communications Conference (GLOBECOM)* pp. 1–6.
17. Sun, W., Liu, J., & Yue, Y. (2019). AI-enhanced offloading in edge computing: When machine learning meets industrial IoT. *IEEE Network, 33*, 68–74.
18. He, X., Lu, H., Huang, H., Mao, Y., Wang, K., & Guo, S. (2020). QoE-based cooperative task offloading with deep reinforcement learning in mobile edge networks. *IEEE Wireless Communications, 27*, 111–117.
19. Latouche, G., & Ramaswami, V. (1999). *Introduction to matrix analytic methods in stochastic modeling (5)*. New Delhi: SIAM.
20. Takine, T. (2014). Beyond M/M/1—invitation to quasi-birth-and-death processes. *Communications of the Operations Research Society of Japan, 59*(4), 179–184. (**in Japanese).**
21. Thai, M. T., Lin, Y. D., Lai, Y. C., & Chien, H. T. (2019). Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading. *IEEE Transactions on Network and Service Management, 17*(1), 227–238.