# Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing

Tao Ouyang, Zhi Zhou, *Member, IEEE*, and Xu Chen, *Member, IEEE*

***Abstract*— Mobile edge computing is a new computing paradigm, which pushes cloud computing capabilities away from the centralized cloud to the network edge. However, with the sinking of computing capabilities, the new challenge incurred by user mobility arises: since end users typically move erratically, the services should be dynamically migrated among multiple edges to maintain the service performance, i.e., user-perceived latency. Tackling this problem is non-trivial since frequent service migration would greatly increase the operational cost. To address this challenge in terms of the performance-cost tradeoff, in this paper, we study the mobile edge service performance optimization problem under long-term cost budget constraint. To address user mobility which is typically unpredictable, we apply Lyapunov optimization to decompose the long-term optimization problem into a series of real-time optimization problems which do not require *a priori* knowledge such as user mobility. As the decomposed problem is NP-hard, we first design an approximation algorithm based on Markov approximation to seek a near-optimal solution. To make our solution scalable and amenable to future fifth-generation application scenario with large-scale user devices, we further propose a distributed approximation scheme with greatly reduced time complexity, based on the technique of the best response update. Rigorous theoretical analysis and extensive evaluations demonstrate the efficacy of the proposed centralized and distributed schemes.**

***Index Terms*— Mobile edge computing, service placement, Lyapunov optimization, Markov approximation, game theory.**

## I. Introduction

With the explosive growth of mobile devices, the recent years have witnessed an unprecedented shift of user preferences from traditional desktops and laptops to smartphones and other connected devices. Subsequently, more and more new mobile applications, as exemplified by augmented reality and interactive gaming [2], emerge and catch public attention. In general, these kinds of applications demand intensive computation resources and high energy consumption for real-time processing. However, due to the physical size constraint, the end device can not efficiently support theses applications alone within our expectation. The tension between resource-hungry applications and resource-limited end devices yields a huge challenge for the next generation network development.

A proliferation of powerful and reliable cloud computing, together with widespread fourth/fifth generation (4G/5G) Long Term Evolution (LTE) networks and WiFi access, has brought rich cloud-hosted mobile services [3] to end users. This approach indeed tackles the resource limitation problem of mobile devices. Unfortunately, the long communication latency to the centralized cloud data center (typically hundreds of milliseconds), such as Amazon EC2 and Windows Azure, can far exceed the stringent timeliness requirement (typically tens of milliseconds) of these mission-critical mobile applications. It will significantly deteriorate the user quality of experience. Furthermore, reducing the delay in the wide area network is not tractable.

To satisfy these mission-critical mobile applications that require ultra-low latency, mobile edge computing (MEC) [4], [5] has been proposed as an extension of centralized cloud computing, which deploys a cloud computing platform at the edge of radio access network (RAN) *in close proximity to mobile devices and users*. Here an edge is typically a micro-data center or cluster of servers that can host cloud applications [6], attached to a base station (BS) or an access point, and available for use by nearby devices. In the paradigm of MEC, as user workload is served by a nearby edge node rather than the remote cloud, the end-to-end latency is significantly reduced [7].

Although the computation capacity of a mobile user is dramatically augmented by edge cloud, a new challenge arises by unpredicted user mobility in the wireless network. With the presence of user mobility [8], enhancing low-latency and smoothing user experience are far more than simply pushing the cloud capabilities to the network edge. To guarantee service continuity when users travel across different edges, an efficient mobility management scheme should be employed in the network edge. An emerging technique, software-defined network (SDN) [9] is proposed to provide seamless and transparent mobility support to users. In the SDN based fog computing architecture [10], the routing logic and intelligent management logic are deployed on the SDN controller, which dramatically simplifies network operation and management.

Manuscript received May 5, 2018; revised August 19, 2018; accepted August 28, 2018. Date of publication September 13, 2018; date of current version November 28, 2018. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1001703, in part by the National Science Foundation of China under Grants U1711265 and 61802449, in part by the Fundamental Research Funds for the Central Universities under Grant 17lgjc40, in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X355, and in part by the Guangdong Natural Science Funds under Grant 2018A030313032. This paper was presented in part at the IEEE/ACM IWQoS 2018 [1]. *(Corresponding author: Xu Chen.)*

The authors are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: ouyt9@mail2.sysu.edu.cn; zhouzhi9@mail.sysu.edu.cn; chenxu35@mail.sysu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2018.2869954

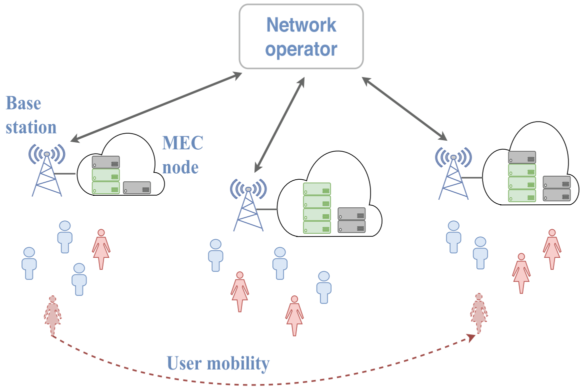


Fig. 1. An example of dynamic service placement when a user roams throughout the network in MEC.

Let us consider a practical scenario as shown in Fig. 1, when a mobile user is within the geographical coverage of the left MEC node, it is clear that if we want to minimize the user-perceived latency, the user should be served by the nearest edge, i.e., the left one. Considering the user mobility and assuming that after a while, the aforementioned mobile user moves to the coverage of the right MEC node. Then, if the service profile of this user is still placed at the left MEC node to serve this user, his perceived latency would greatly deteriorate due to the extended network distance. This example demonstrates that, to optimize the user experience of MEC, the service profiles of mobile users should be dynamically re-placed among edges to *follow the mobility* of users.

However, the dynamic service profile placement problem is non-trivial. On one hand, the user-perceived latency is jointly determined by the communication delay and computing delay [11], [12]. Therefore, if the service profile of each user is placed aggressively at the nearest MEC node, then some MEC nodes could be overloaded, leading to increased computing latency. On the other hand, following the user mobility requires frequent service migration among multiple MEC nodes. In return, such frequent service migration incurs additional operational cost such as usage of the expensive wide-area-network (WAN) bandwidth and system energy consumption [13]. As a result, an effective dynamic service placement strategy should carefully (1) cooperate the communication delay and computing delay to minimize the user-perceived latency, and (2) navigate the *performance-cost trade-off* in a cost-efficient manner.

Following the above two guidances on dynamic service placement for MEC, in this paper, we propose a *mobility-aware dynamic service placement framework for cost-efficient MEC*. In particular, to strike a nice balance between the service performance and the operational cost incurred by cross-edge service migration, we propose to minimize the user-perceived latency over the long run, under the constraint of a long-term migration cost budget which is pre-defined monthly or yearly by the network operator in practice. By applying Lyapunov optimization technique to the formulated stochastic optimization problem, our framework can effectively incorporate the long-term migration cost budget into real-time optimizations, and make online decisions on dynamic service placement, *without requiring any a priori future information (e.g., user mobility)*. To address the challenge of the NP-hardness of the resulted real-time optimizations, we further design centralized and distributed approximation schemes to seek near-optimal solutions respectively. Both rigorous theoretical analysis and extensive trace-driven evaluations demonstrate the cost-efficiency of the proposed mobility-aware online service placement framework.

The rest of this paper is organized as follows. Section II reviews related work. The system model and problem formulation are presented in Section III. Section IV proposes two online service placement algorithms under the centralized and decentralized mechanisms to seek near-optimal strategies respectively. Section V presents the theoretical analysis of the proposed framework. Performance evaluation is carried out in Section VI. Section VII concludes this paper.

## II. Related Work

Service placement is not a new topic, as it has been extensively studied in the paradigm of cloud computing. Specifically, the goal of service placement in cloud computing can be categorized into: (1) consolidating the services to a smaller set of physical servers to improve the resource utilization and reduce the operational cost [14], (2) placing the services to a set of heterogeneous nodes to leverage the heterogeneities on energy efficiency or cost efficiency [15], and (3) placing the services to different nodes to perform network load balancing [16]. However, as we have discussed in Section I, the goal of service placement in MEC is to follow the user mobility and thus to reduce the user-perceived latency adaptively.

A key challenge towards efficient service placement in MEC is to follow the mobility of users and devices. In addressing this challenge, some work is based on the assumption of perfect predictability on future information. For example, Nadembega *et al.* [17] tackle the trade-off between the execution overhead and latency, with a mobility-based prediction scheme which estimates the data transfer throughput, handoff time and VM migration management in advance. Moreover, the recent work [18] further studies how to place service by predicting the future cost incurred by data transmission, processing and service migration. Aissioui *et al.* [19] propose a FMeC-based framework in an automated driving use case, which captures the trade-off between reducing service migration cost and maintaining the end-to-end QoS based on the vehicle mobility pattern update analysis. But this work does not consider the load balancing among multiple edge servers in the multi-user case. Unfortunately, the future information such as user mobility is extremely challenging to accurately predict in realistic environments.

In response to the challenge that user mobility may not be readily predictable in practice, another stream of recent work resorts to a milder assumption that the user mobility follows a Markovian process, and then applies the technique of Markov Decision Process (MDP). Specifically, a preliminary research in [20] explores how service migration impacts the perceived latency of mobile users, via utilizing Markov chains to analyze whether to migrate services or not. Ksentini *et al.* [21] and Wang *et al.* [22] try to determine an optimal threshold decision policy on service migration based on MDP. Further, Tarik Taleb and Frangoudis [23] extend service migration decision algorithm [21] to capture 2D mobility scenarios. In [24], the optimal service migration strategy is devised by formulating the service placement problem as a sequential decision-making problem. In comparison, our online service placement strategy does not make any assumption on the user mobility, yet can achieve a performance that can be arbitrarily close to the offline optimum. Moreover, all these works do not consider the practical operational cost constraint for dynamic service placement.

Without requiring the future user mobility as a priori knowledge, Kiani and Ansari [25] design a two-time scale approach to maximize the profit of a service provider, while satisfying users' QoS by allocating computing and communications resources in hierarchical mobile edge computing. A pronounced difference is that our work considers a long-term cost budget constraint, where our algorithm can be continuously adjusted to accommodate system dynamics. A closely related work [26] proposes an energy-aware mobility management scheme to minimize the total delay (including both communication and computation delay) under the long-term energy consumption constraint. However, it is worth noting that our work substantially differs from and complements to [26] in at least the following three aspects: (1) the study in [26] only considers a single-user service placement scenario, while we consider a more practice-relevant multi-user case. (2) We consider the efficient allocation of the limited edge resource to multiple users, and thus to coordinate computation and communication delay to minimize the total latency. (3) To avoid excessive operational cost incurred by frequent service migration, we navigate the performance-cost tradeoff in a cost-efficient manner.

This work significantly extends the preliminary work [1]. To improve the service performance in a large scale and ultra dense network, we propose a distributed service placement scheme with faster convergence rate in this work. Rather than optimizing the service performance based on Markov approximation in a cooperative manner, the distributed scheme minimizes the cost in a non-cooperative manner. To evaluate the efficiency of distributed algorithms, we introduce a dynamic placement policy with mobility pattern update analysis [19] and more detailed comparison between two proposed algorithm (such as the running time of decision making with different dense works) in the experiment.

## III. System Model and Problem Formulation

As shown in Fig.1, we consider a network operator running a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of MEC nodes to serve a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of mobile users. Each MEC node is attached to a local base station or wireless access point, via high-speed local-area network (LAN). Inspired by the recent work [27], [28] on resource allocation for MEC, in this paper we adopt a device-oriented service model for MEC, rather than the traditional application-oriented service model for cloud computing [29]. Specifically, the service profile and environment for the applications run on each mobile device (rather than each application) is assigned to a dedicated virtual machine [30] or container [31]. To better capture the user mobility, the system is assumed to operate in a slotted structure and its timeline is discretized into time frames $t \in \mathcal{T} = \{0, 1, 2 \ldots, T\}$. At all discrete time slots, each mobile user sends a service request to the local MEC node, then the network operator (i.e., SDN controller) will gather all request information and determine the optimal MEC node to serve corresponding user based on the current global system information. Table I summarizes the key parameter notations in our paper.

TABLE I
Key Notations in Our Model

| Notation | Definition |
|---|---|
| $x_i^k(t)$ | Whether the service profile of user k is placed at MEC node i (=1) or not (=0) |
| $R^k(t)$ | The amount of computation capacity required by user $k$ |
| $N_i(t)$ | The number of services served by MEC node $i$ |
| $F_i$ | The maximum computing capacity of MEC node $i$ |
| $D^k(t)$ | The computing delay for user $k$ |
| $H_i^k(t)$ | The communication delay when the service profile of user k is placed on MEC node $i$ |
| $L^k(t)$ | The communication delay for user $k$ |
| $T^k(t)$ | The total perceived latency for user $k$ |
| $E_{ji}^k(t)$ | The cost of migrating service of user $k$ from source MEC node $j$ to destination MEC node $i$ |
| $E(t)$ | The total migration cost for all users |
| $E_{avg}$ | The long-term time-averaged cost budget |
| $V$ | Lyapunov control parameter |
| $\beta$ | Markov approximation control parameter |
| $\mu$ | The approximation ratio of distributed scheme |

### A. Service Placement Model

To maintain satisfactory Quality-of-Service (QoS), i.e., low service latency for mobile users which typically move erratically, the service profile of each user should be dynamically migrated across multiple edges to follow the user mobility. Here we take a binary indicator $x_i^k(t)$ to denote the dynamic service placement decision variable. Let $x_i^k(t) = 1$ if the service profile of user $k \in \mathcal{N}$ is placed at the MEC node $i \in \mathcal{M}$ at time slot $t$, and $x_i^k(t) = 0$ else. Note that at a given time slot, since each user is served by one and only one MEC node, we have the following constraints for service placement decision $x_i^k(t)$:

$$\sum_{i=1}^{M} x_i^k(t) = 1, \quad \forall k, t. \tag{1}$$

$$x_i^k(t) \in \{0, 1\}, \quad \forall i, k, t. \tag{2}$$

Based on the above defined service placement decision, we are now ready to formulate the user-perceived latency which is determined by the service placement.

### B. QoS Model

In the paradigm of MEC, the QoS, i.e., user-perceived latency is jointly determined by the computing delay and communication delay.

*1) Computing Delay:* At each MEC node, multiple mobile users will simultaneously share the computing resource to process their applications. However, when confronted with service request surge, the small-scale MEC node may not guarantee to provide satisfactory service for all served users. A more efficient load balancing across multiple MEC nodes can be achieved by dynamic service placement. In this paper, we use $R^k(t)$ to denote the amount of computing capacity (in terms of CPU cycles) required by the service request of user $k$ at time slot $t$. Taking video stream analytics as an instance [26], the amount of required computation capacity is determined by the input data size of the video and the corresponding computation intensity of the analytic task. We consider an equal resource allocation case, i.e., each user evenly shares computing resource of the serving MEC node.[1] Then, the computing delay for mobile user $k$ at time slot $t$ is given by $D^k(t) = \sum_{i=1}^{M} x_i^k(t)R^k(t)N_i(t)/F_i$, where $N_i(t)$ is the number of users served by MEC node $i$ during the time slot $t$, which follows $N_i(t) = \sum_{k=1}^{N} x_i^k(t)$. Moreover, $F_i$ represents the maximum computing capacity (in CPU cycles per second) of MEC node $i$.

*2) Communication Delay:* In MEC, the communication delay between a mobile device and the MEC node generally contains the network propagation delay and the data transmission delay. In particular, when a data packet passes through the intermediate network devices along the targeted path between service-served MEC node and local connected MEC node, the network propagation delay is majorly determined by the network distance (i.e., the hop count), such as in [23]. While the data transmission delay is jointly determined by the amount of data transferred $d^k$ and the link bandwidth $b_i^k$. Then the delay can be denoted as $\gamma\frac{d^k}{b_i^k}$, where $\gamma$ is a positive coefficient. Since the network condition (i.e., hop distance and bandwidth) and data transmission information in current time slot are available from the system-level perspective, we can extend the above cases into a general model, which we do not impose structural assumption on. Given the service request information as well as the current location of user $k$, the communication delay to MEC node $i$ can be characterized by a general model $H_i^k(t)$. When considering the service placement decision $x_i^k(t)$, the communication latency experienced by user $k$ can be further expressed as $L^k(t) = \sum_{i=1}^{M} x_i^k(t)H_i^k(t)$.

By combining the computing delay $D^k(t)$ and communication delay $L^k(t)$, we denote the total latency experienced by user $k$ at time $t$ as

$$T^k(t) = D^k(t) + L^k(t). \tag{3}$$

### C. Migration Cost Model

While dynamic service placement empowers satisfactory QoS by migrating service profiles among edges to follow the user mobility, it is worth noting that cross-edge service migration would incur additional operational cost. Specifically, when transferring the service profile of each user across edges, enormous usage of the scarce and expensive wide-area-network (WAN) bandwidth would be caused. In additional, cross-edge transferring also increases the energy consumption of network devices such as routers and switches. To model the operational cost incurred by cross-edge service migration, we use $E_{ji}^k(t)$ to denote the cost of migrating the service profile of user $k$ from source MEC node $j$ to destination MEC node $i$. Without loss of generality, we assume that $E_{ji}^k(t) = 0, \forall j = i$. Then, given the service placement decision $x_i^k(t-1)$ at time slot $t-1$, and $x_i^k(t)$ at time slot $t$, the service migration cost of user $k$ at time slot $t$ can be computed by $\sum_{i=1}^{M}\sum_{j=1}^{M} x_j^k(t-1)x_i^k(t)E_{ji}^k(t)$. Considering all the $N$ users, the total service migration cost at time slot $t$ can be further denoted as

$$E(t) = \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{M} x_j^k(t-1)x_i^k(t)E_{ji}^k(t).$$

With the presence of user mobility, it is intuitive that to ensure a desirable level of QoS, the service profile should be actively migrated to follow the user mobility. However, frequent migration would incur excessive operational cost in return. Then, a natural question is how to navigate such a performance-cost trade-off in a cost-efficient manner.

### D. Navigating the Performance-Cost Trade-Off

To optimize multiple conflicting objectives in a balanced manner, the most commonly adopted approach is to assign different weights to those conflicting objectives and then optimize the weighted sum of them. Unfortunately, in our problem, how to properly defining the weights of performance and cost in a realistic environment is not straightforward. In response, considering the fact that network providers generally operate within a long-term (e.g., yearly) cost budget, we propose to optimize the long-term performance under the predefined long-term cost budget. Specifically, we introduce $E_{avg}$ to denote the long-term time-averaged cost budget over a time span of $T$ time slots, which satisfies:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T} E(t) \le E_{avg}. \tag{4}$$

Then, our problem of minimizing the long-term time-average service latency under the constraints of long-term cost budget can be formulated as the following stochastic optimization:

$$\begin{aligned}\mathcal{P}1: \min_{c(t)}\ &\frac{1}{T}\lim_{T\to\infty}\sum_{t=1}^{T}\sum_{k=1}^{N} T^k(t)\\ &\text{s.t. } (1)-(4).\end{aligned} \tag{5}$$

In general, the derivation of the optimal long-term policy $\mathcal{P}1$ is not a one-shot operation but needs to be continuously adjusted

[1] Other resource allocation models, such as weighted resource allocation, are also applicable to Markov approximation based scheme.

to accommodate system dynamics such as erratic user mobility and requested service pattern. This is because predicting accurate user behavior (mobility and requested service pattern) and network condition over a long run is extremely hard. Moreover, even though the long-term service placement optimization has been decomposed into the real-time decoupling problem, preventing frequent service migration with the long-term migration cost constraint is non-trivial. In the current literature, some approaches have been proposed to handle this problem. For example, in [18], by finding an optimal look-ahead window size, the long-term optimization problem can be approximately discretized into a series of equivalent shortest-path problems. However, the near-future information cannot be predicted accurately for dynamic mobile wireless network. Fortunately, in the queuing theory [32], the long-term migration budget constraint (4) in this optimization problem can be regarded as the queue stability control, i.e., the time-averaged migration $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}E(t)$ is beneath the long-term budget $E_{avg}$. Moreover, Lyapunov optimization technique provides an efficient approach to decouple the long-term problems. It does not require any a priori system information while maintaining the queue stability in an online way. Hence, we propose an online algorithm that transforms the original problem into a series of real-time minimization problems.

## IV. Online Service Placement Algorithm

In this section, we describe a novel framework that makes online service placement decisions. To solve the $\boldsymbol{\mathcal{P}1}$, we first convert the original problem to a queue stability control problem based on Lyapunov optimization.

### A. Problem Transformation via Lyapunov Optimization

*1) The Construction of Virtual Queue for Long-Term Service Placement Cost:* Due to the dynamic and stochastic property of the system (e.g., time-varying and uncertainly user mobility and request arrival process), a prime challenge of $\boldsymbol{\mathcal{P}1}$ is to navigate the performance-cost trade-off in a cost-efficient manner without global information over the long run. A key idea of Lyapunov optimization is to strike a desirable balance between current perceived latency and migration cost while maintaining the cost queue stable by introducing a virtual queue for the long-term budget. First, we define a virtual queue as a historical measurement of the exceeded migration cost and assume that initial queue backlog is 0 (i.e., $Q(0) = 0$).

$$Q(t+1) = max[Q(t) + E(t) - E_{avg}, 0], \tag{6}$$

where $Q(t)$ is the queue length at time slot $t$, which represents the exceeded cost of executed service migration by the end of time slot $t$.

Intuitively, the value of $Q(t)$ can be regarded as an evaluation criteria to assess the migration cost condition. A large value of $Q(t)$ implies the cost has far exceeded the long-term budget $E_{avg}$ since carrying out online service placement algorithm. In order to guarantee that the time-averaged service migration cost is lower than budget $E_{avg}$, i.e., inequality (4) holds, the virtual queue $Q(t)$ must be stable, i.e., $\lim_{T\to\infty}\mathbb{E}\{Q(T)\}/T = 0$. Furthermore, by total summing the inequality $Q(t+1) \geq Q(t)+E(t)-E_{avg}$ derived from equation (6) and rearranging it, we can gain:

$$\frac{Q(T)-Q(0)}{T} + E_{avg} \geq \frac{1}{T}\sum_{t=0}^{T-1}E(t).$$

For $Q(0) = 0$, we can take expectations of the above inequality and have

$$\lim_{T\to\infty}\frac{\mathbb{E}\{Q(T)\}}{T} + E_{avg} \geq \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{E(t)\}.$$

Hence, the stability of the virtual queue can ensure that the time-averaged migration cost does not exceed the budget.

*2) Queue Stability:* To stabilize the virtual queue, we first define a quadratic Lyapunov function and Lyapunov drift function [32] respectively as follows:

$$L(\Theta(t)) \triangleq \frac{1}{2}Q(t)^2. \tag{7}$$

This represents a scalar measure of cost queue congestion. For instance, a small value of $L(\Theta(t))$ implies the queue backlog is small. Thus, if a policy consistently pushes the quadratic Lyapunov function towards a bounded level, it implies that the virtual queue is stable.

To remain the virtual queue stable, we introduce the *one-step conditional Lyapunov drift* to push the quadratic Lyapunov function towards a lower congestion region:

$$\Delta(\Theta(t)) \triangleq \mathbb{E}\Big[L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\Big]. \tag{8}$$

The drift $\Delta(\Theta(t))$ denotes the migration cost queue change in the Lyapunov function over a one-time slot. It generates an important term that includes a product of queue backlog and migration cost, which helps the algorithm adjust to accommodate the system dynamics [32].

*3) Joint Lyapunov Drift and User-Perceived Latency Minimization:* After constructing the virtual cost queue, the original problem has been decomposed into a series of real-time optimization problems. Our goal is to find a current placement policy to coordinate the perceived latency and migration cost. By incorporating queue stability into delay performance, we define a *Lyapunov drift-plus-penalty* function to solve the real-time problem.

$$\Delta(\Theta(t)) + V\sum_{k=1}^{N}T^k(t), \tag{9}$$

where $V$ is a non-negative control parameter that adjusts the trade-off between delay performance and migration cost queue backlogs. It shows the attention on the delay performance compared to migration cost budget. Moreover, the following lemma provides the performance guarantee of the drift-plus-penalty function.

*Lemma 1:* For all possible values of $\Theta(t)$ by using any placement schedule over all time slots, the following statement holds:

$$\Delta(\Theta(t))+V\sum_{k=1}^{N}T^k(t) \le B+\sum_{k=1}^{N}V\mathbb{E}\Big[T^k(t)|\Theta(t)\Big] + Q(t)\mathbb{E}\Big[E(t)-E_{avg}|\Theta(t)\Big], \quad (10)$$

where $B = \frac{1}{2}(E_{avg}^2 + E_{max}^2)$ is a constant value for all time slots, and $E_{max} = \max_{t\in\mathcal{T}} E(t)$.

The detailed proof is given in the technique report [33]. Based on the *Lemma 1*, the *drift-plus-penalty* function has a supremum bound at every time slot $t$.

### B. Online Service Placement Algorithm

In this section, we convert the problem $\boldsymbol{\mathcal{P}1}$ to a series of real-time drift-plus-penalty supremum bound minimizations. While the *drift-plus-penalty* expression involves the $max[*]$ term in equation (6), which complicates reaching solution to placement issue. Following the lemma 1, we observe that minimizing the right side of inequality (10) can approximate the supremum bound closely, which is equivalent to minimizing the *drift-plus-penalty*. Therefore, based on the aforementioned parameters definition, we rearrange it for a concise form and obtain an optimal service placement policy $c^*(t)$.

$$\sum_{k=1}^{N}V\mathbb{E}\Big[T^k(t)|\Theta(t)\Big] + Q(t)\mathbb{E}\Big[E(t)-E_{avg}|\Theta(t)\Big] \le \sum_{k=1}^{N}\sum_{i=1}^{M}x_i^k(t)\Big(\frac{VR^k(t)\sum_{k=1}^{N}x_i^k(t)}{F_i} + VH_i^k(t)+\rho_i^k(t)\Big), \quad (11)$$

where $\rho_i^k(t) = \sum_{j=1}^{M}x_j^k(t-1)Q(t)E_{ji}^k$ is a constant at every time slot $t$, which does not affect the placement decision-making. Thus, the major part of our online service placement algorithm is to solving following $\boldsymbol{\mathcal{P}2}$ to minimize the real-time supremum bound for the *drift-plus-penalty* function.

$$\boldsymbol{\mathcal{P}2} : \min_{c(t)} \sum_{k=1}^{N}\sum_{i=1}^{M}x_i^k(t)\Big(\frac{VR^k(t)\sum_{k=1}^{N}x_i^k(t)}{F_i} + VH_i^k(t)+\rho_i^k(t)\Big) \quad \text{s.t. } (1)-(4). \quad (12)$$

For simplify the formulation, we use $U(\boldsymbol{c},t)$ to replace the objective function of problem $\boldsymbol{\mathcal{P}2}$, where $\boldsymbol{c}$ is feasible service placement policy. In Algorithm 1, we describe the implementation of the online service placement algorithm. In each time slot $t$, a close-to-optimal service placement schedule can be obtained when solving $\boldsymbol{\mathcal{P}2}$, and the migration cost virtual queue will be updated subsequently for next time slot calculation.

Unfortunately, this real-time optimization problem is NP-hard in general [34], due to its combinatorial nature. To address this challenge, we apply Markov approximation [35] to obtain a near-optimal solution for this real-time problem.

**Algorithm 1** Online Service Placement Algorithm

```
Initialization: We set the cost queue backlog $Q(0) = 0$ at
    beginning.
End initialization
for each time slot $t = 1, 2, \ldots, \infty$ do
    Solve the problem $\mathcal{P}2$: $\mathbf{c}^*(t) = arg\min(12)$.
    Update the virtual queue: run (6) based on $\mathbf{c}^*(t)$.
end for
```

### C. Markov Approximation Method

In this subsection, we design a centralized service placement optimization that can obtain the minimum solution approximatively. The problem $\boldsymbol{\mathcal{P}2}$ is a combinatorial optimization of finding the optimal service placement policy, we leverage the idea of Markov approximation in [35] to optimize the policy. To proceed, then we can convert the problem $\boldsymbol{\mathcal{P}2}$ to the following equivalent problem:

$$\min \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}}(t)U(\boldsymbol{c},t) \quad s.t. \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}}(t) = 1, \forall t \in \mathcal{T}, \quad (13)$$

where $q_{\mathbf{c}}(t)$ is a decision variable, which means the probability of the placement policy $\mathbf{c}$ is adopted at current time slot $t$; $c(t)$ is the collection of all feasible placement policies. Obviously, the optimal solution to problem (13) is to choose the minimum cost placement policy with probability one. The problem can be approximately treated as the following *convex log-sum-exp* problem [35].

$$\min \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}}(t)U(\boldsymbol{c},t) + \frac{1}{\beta}\sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}}(t)\log q_{\mathbf{c}}(t) \quad s.t. \sum_{\mathbf{c}\in c(t)} q_{\mathbf{c}}(t) = 1, \forall t \in \mathcal{T}, \quad (14)$$

where $\beta$ is a positive constant that charges the approximation ratio of the entropy term. When $\beta \to \infty$, the problem (14) becomes the original problem (13). If handling the problem with the Karush-Kuhn-Tucker (KKT) [36], we can obtain the optimal solution to problem (14)

$$q_{\mathbf{c}}^*(t) = \frac{exp\big(-\beta U(\boldsymbol{c},t)\big)}{\sum_{\mathbf{c'}\in c(t)} exp\big(-\beta U(\boldsymbol{c'},t)\big)}, \quad \forall \mathbf{c} \in c(t),\ \forall t \in \mathcal{T}. \quad (15)$$

According to the probability $q_{\mathbf{c}}^*(t)$, we can gain the current optimal policy. Then, we design a service placement algorithm that constantly updates the placement policy $c$ to form a discrete-time Markov chain [35]. Once the Markov chain achieves to the stationary distribution as shown in (16), the optimal placement profile which minimizes the real-time supremum bound in (13) can be derived by setting parameter $\beta$ as large as possible. In this algorithm, the Markov chain is irreducible, which traverses all feasible states under different placement policies. Besides, designing a desired time-reversible Markov chains needs to hold the following

**Algorithm 2** Markov Approximation Based Placement Policy Search

```
Initialization: Initialize the service placement policy c as
    randomly assigning a MEC node for each service.
End initialization
loop for each service placement update iteration
    Choose a service $k$ randomly and carry out the following operations:
        Calculate the bound $U(\boldsymbol{c}',t)$ for any other feasible
            service placement policy.
        Select a placement policy acc. to (17) probabilistically.
        Update the service placement policy by placing the
            service to the new MEC node.
    Record the placement policy c* with the smallest
        $U(\boldsymbol{c}*,t)$, found up to now.
end loop
```

balance equation:

$$q_{\mathbf{c}}^*(t)q_{\mathbf{c},\mathbf{c}'}(t) = q_{\mathbf{c}'}^*(t)q_{\mathbf{c}',\mathbf{c}}(t), \quad \forall \mathbf{c}, \mathbf{c}' \in c(t),\ \forall t \in \mathcal{T}, \tag{16}$$

where $q_{\mathbf{c},\mathbf{c}'}(t)$ is the probability of the placement policy update from **c** to **c**′.

The Markov approximation based service placement policy algorithm is described in Algorithm 2, which can be implemented in the network operator that can gather sufficient network information and computing capability for real-time decision making. In the algorithm, a random service will be picked to update its placement policy for each update iteration. In this situation, a state transition of services from **c** to **c**′ only occurs if only one user service is migrated. Since knowing the targeting migration policy performance (i.e., supposing we migrate service from MEC node $a$ to MEC node $c$, the new joint cost of perceived latency and migration in (12) can be calculated easily), the probability of each feasible migration adjustment is directly proportional to the difference of the total cost under two placement policies **c** and **c**′, denoted as follows:

$$q_{\mathbf{c},\mathbf{c}'}(t) = \alpha \exp\Big(-\frac{1}{2}\beta\big(U(\boldsymbol{c}',t) - U(\boldsymbol{c},t)\big)\Big). \tag{17}$$

Note that during each policy iteration, the network operator will record the best policy found up to now. As shown in [35], by proper parameter tuning Markov approximation algorithm can converge in a super-linear rate. Next, we analyze the complexity of the Markov approximation algorithm. For each update iteration, the system chooses arbitrarily a mobile device to update its service placement. During the process of calculating the total cost $U(\boldsymbol{c}',t)$ for all feasible placement policies, the possible placement configurations enumerates at most $MN$. Assuming that this algorithm needs to be executed $I$ iterations to achieve the convergence, then the total time complexity of Algorithm 2 is $O(IMN)$.

### D. Best Response Update Method

To dramatically reduce running time of placement decision-making, we apply the best response update technique to construct a distributed mechanism for a faster service placement search. Different from the centralized method based on Markov approximation where the near-optimal placement decision is achieved by centralized probabilistic policy explorations collectively by all the users, in distributed service policy update, each user $k$ is generally greedy and adopts the best response to optimize its own placement decision in a deterministic manner. That is, best response update method emphasizes more on the exploitation of individual efficient decision instead of randomized decision exploration, leading to significant running time reduction.

As aforementioned description about edge resource allocation $D^k(t)$, the resource competition among multiple users will influence the service performance. Inspired by the application of game theory to non-cooperative AP channel selection [37], we consider the problem $\boldsymbol{\mathcal{P}2}$ as a congestion game [38] with user-specific cost function. Let $\mathbf{c}_{-k} = \{c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_N\}$ be service placement decisions made by all users except for user $k$. Given the placement policies of all other users $\mathbf{c}_{-k}$, the placement problem confronted by user $k$ is to select a proper MEC node to minimize its cost in terms of perceived latency and migration cost, i.e.,

$$c_k = arg \min_{c_k \in \mathcal{M}} U_k(c_k, \mathbf{c}_{-k}, t), \quad \forall k \in \mathcal{N}. \tag{18}$$

The non-cooperative nature of the service placement problem leads to a formulation based on game theory, where each placement decision is finally executed by the user device in a mutually acceptable way, i.e., a *Nash equilibrium*, which is defined as follows:

*Definition 1 (Nash Equilibrium):* A placement policy profile $\mathbf{c}^* = (c_1^*, \ldots, c_N^*)$ achieves a Nash equilibrium when no user can minimize its cost further by unilaterally updating its placement policy, i.e.,

$$U_k(c_k^*, \mathbf{c}_{-k}^*, t) \leq U_k(c_k, \mathbf{c}_{-k}^*, t), \quad \forall k \in \mathcal{N},\ \forall \mathbf{c}_k \in \mathcal{M}. \tag{19}$$

To study the existence of the Nash equilibrium of multiple service placements, we introduce the best response update first.

*Definition 2 (Best Response Update):* Given the service placement profile $\mathbf{c}_{-k}^*$ for all other users, the placement decision of user $k$ is the best response if

$$U_k(c_k^*, \mathbf{c}_{-k}, t) \leq U_k(c_k, \mathbf{c}_{-k}, t), \quad \forall c_k \in \mathcal{M}. \tag{20}$$

Similar to the n-player congestion game in [38], through a finite best response update execution for service migration, our distributed service placement policy search can reach a Nash equilibrium by induction, i.e., Suppose that once a user sends the service request to the connected MEC node, the system will allocate a unique ID for its service. Then, we can update all service placement profiles according to the random order of assigned IDs. Let a service $k$, which is the current smallest index among the pending update users, be assigned to a preferable MEC node to achieve its current performance cost minimization by the best response update. Hence we can formulate the service placement update process as follows:

$$\begin{aligned} c_k(r+1) = arg\min U_k\big(c_k, \{&c_1(r+1), \ldots, c_{k-1}(r+1),\\ &c_{k+1}(r), \ldots, c_N(r)\}, t\big), \end{aligned} \tag{21}$$

where $r$ is the policy update round. In the current service placement profile, the services with smaller indexes have updated (i.e., $c_1(r+1), \ldots, c_{k-1}(r+1)$), while the strategies of the ones with larger indexes are kept unchanged. By adopting the asynchronous best response update strategy, the service migration profile will be gradually converged. The detailed implementation is summarized in the Algorithm 3.

**Algorithm 3** Best Response Update Based Placement Policy Search

```
1: Initialization: Initialize the service placement profile $\mathbf{c}(0) = (c_1(1), c_2(0), \ldots, c_N(0))$ as randomly assigning a MEC node for each service and the update iteration round as $r = 0$.
2: End initialization
3: while c(r) does not reach a Nash equilibrium do
4:     for indexed service $k$ = 1 to $N$ do
5:         Select the proper MEC node where user $k$ can minimizes the its own cost acc. to (21) and gain the corresponding placement policy $c_k$
6:     end for
7:     Set service migration profile as $\mathbf{c}(r+1) = (c_1(r+1), \ldots, c_N(r+1))$ and the update iteration round $r = r+1$
8: end while
```

*Theorem 1:* There exists a Nash equilibrium of the distributed service placement that can be achieved within at most $M\binom{N+1}{2}$ best response update steps.

The detailed proof is given in the technique report [33]. Note that the proposed best response update based distributed policy search approach explores the possible service improvement update paths and terminates when achieving to a Nash equilibrium. Due to the weakly acyclic property (i.e., there must exist a finite improvement update path) [38], it can make our policy coverage into a Nash equilibrium.

Next, we evaluate the computational complexity of the algorithm 3. As shown in line 4 to 6, for each update iteration, the system will update service profile of users placement, which involves N minimization operations and each minimization operation can be achieved by sorting over at most M values. Hence this procedure has the complexity of $O(NM\log M)$. Line 7 has a complexity of O(1). Assuming that the algorithm 3 needs $I$ times update iteration to be converged to a Nash equilibrium. Then the total computational complexity of the algorithm 3 is $O(INM\log M)$. Surprisingly, the total computational complexity of distributed placement update seems to be higher than Markov approximation. While, in practical process, the update iteration round is a critical factor to increase the time overhead. In the later simulate experiment, we can find that the distributed scheme reduces dramatically running time of placement decision-making compared with the Markov approximation.

## V. Performance Analysis

In this section, we analyze theoretically the performance of our two mobility-aware dynamic service placement algorithms for MEC. First, we discuss the optimality gap in the Markov approximation based scheme and best response update based scheme respectively. Then, we compare the performance of our two online algorithms (i.e., Markov approximation and best response update in the Lyapunov framework) with the offline optimum.

### A. Markov Approximation

With above description of Markov approximation algorithm, the probability of a service placement state switch from $\mathbf{c}$ to $\mathbf{c}'$ in the Markov chain is denoted in (17). It is obvious that our algorithm can be converged to a distinctive stationary distribution for its time reversibility.

*Theorem 2:* There exists a distinctive stationary distribution for the service placement algorithm as stated in equation (16).

The detailed proof is given in the technique report [33]. As shown in Theorem 1, we can obtain the minimal supremum bound for the *drift-plus-penalty* function as the parameter $\beta$ increasing to a large enough value in our service placement algorithm. We denote the minimal supremum bound and expected supremum bound by proposed algorithm as $S^* = \min \sum_{\mathbf{c}\in c(t)} U(\boldsymbol{c},t)$ and $\widetilde{S} = \sum_{\mathbf{c}\in c(t)} q^*_{\mathbf{c}} U(\boldsymbol{c},t)$ respectively.

*Theorem 3:* For the algorithm, the optimality gap is given as follows:

$$0 \le \widetilde{S} - S^* \le \frac{1}{\beta} \ln|\delta|, \tag{22}$$

where $|\delta|$ is the amount of feasible service placement policies of all mobile users at time slot $t$.

The detailed proof is given in the technique report [33]. By Theorem 2, the error of worse-case solution in our algorithm is no more than $\frac{1}{\beta}\ln|\delta|$. Thus, if setting the value of parameter $\beta$ as large as possible, we can approach an almost equivalent solution to the minimal supremum bound. Fortunately, the value of $\beta$ is usually large enough in an acceptable scope, the performance deviation of the optimum is quite small [35].

### B. Approximation Ratio for Best Response Update

With above description of best response update, we can quantify the efficiency ratio of our distributed placement mechanism in the worst-case equilibrium over the optimal centralized one. Let $\Gamma$ be the set of equilibria of the service placement profile. Then the approximation ratio for best response update can be expressed as follows:

$$\mu = \frac{\max_{\mathbf{c}\in\Gamma} \sum_{k\in\mathcal{N}} U_k(c,t)}{\min_{\mathbf{c}\in c(t)} \sum_{k\in\mathcal{N}} U_k(c,t)}, \tag{23}$$

Obviously, the lower bound of approximation ratio $\mu$ is 1. A larger approximation ratio denotes that the worst performance of our distributed algorithm is less efficient than using the centralized optimum as a benchmark. Let $F_{max}$ and $F_{min}$ be the maximum and minimum computing capacity of all MEC nodes respectively. Similarly, Let $H^k_{max} = \max_{i\in\mathcal{M},t\in\mathcal{T}} H^k_i(t)$, $H_{min} = \min_{i\in\mathcal{M},t\in\mathcal{T}} H^k_i(t)$, $R^k_{max} = \max_{t\in\mathcal{T}} R^k(t)$, $R^k_{min} = \min_{t\in\mathcal{T}} R^k(t)$ and $\rho_{max} = \max_{i\in\mathcal{M},t\in\mathcal{T}} \rho^k_i(t)$. Thus, we can have:

*Lemma 2:* In the distributed service placement search, the joint cost performance of each user $k$ at an equilibrium is no more than $\frac{VR^k_{max}(M+N-1)}{MF_{min}} + VH^k_{max} + \rho^k_{max}$.

The detailed proof is given in the technique report [33]. According to Lemma 2, we can gain the upper bound of the approximation ratio $\mu$ as follows:

*Theorem 4:* The the approximation ratio $\mu$ of the distributed service placement search is at most

$$\mu \leq \frac{\frac{VR^k_{max}(M+N-1)}{MF_{min}} + VH^k_{max} + \rho^k_{max}}{\sum_{k=1}^{N}\left(\frac{VR^k_{min}}{F_{max}} + VH^k_{min}\right)}. \tag{24}$$

The approximation ratio $\mu$ demonstrates the worse-case performance of our distributed scheme in an equilibrium. Numerical results in the next section show the algorithm is efficient compared with the centralized approximation.

### C. Optimality Analysis

As we have mentioned, the transformed problem $\boldsymbol{\mathcal{P}2}$ is NP-hard. Fortunately, the minimization error of the $\boldsymbol{\mathcal{P}2}$ is acceptable under the control of our online algorithm. We use $\widetilde{T}^k(t)$ and $T^{opt}$ to respectively denote the delay performance in time slot $t$ by the proposed algorithm and the infimum time average performance delay with the overall information. Then the following theorems will give a supremum bound of the time-averaged delay performance and the migration cost queue backlogs for our two approximation algorithms. The former one is the Markov approximation based centralized scheme, and the later one is the best response update based distributed scheme.

*Theorem 5:* For any non-negative control parameter $V$, the long-term delay performance implemented by proposed two online algorithms satisfy that

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{N}\mathbb{E}\{\widetilde{T}^k(t)\} \leq T^{opt} + \frac{B}{V} + \frac{1}{\beta V}\ln|\delta|. \tag{25}$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{N}\mathbb{E}\{\widetilde{T}^k(t)\} \leq \mu T^{opt} + \frac{B}{V} \tag{26}$$

*Theorem 6:* Assuming that $E_{avg} > 0$ and initializing the migration cost queue backlog is 0, thus for all time slots we having the following bound:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{Q(t)\} \leq \frac{B + VT^{opt}}{\varepsilon} + \frac{1}{\beta\varepsilon}\ln|\delta|. \tag{27}$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{Q(t)\} \leq \frac{B + \mu VT^{opt}}{\mu\varepsilon}. \tag{28}$$

The detailed proof is given in the technique report [33]. Where $\varepsilon > 0$ is a finite constant that represents the distance between the time-averaged migration cost by some control policy and long-term cost budget. From Theorem 1, it is known that the delay performance of the online algorithm can be approached closely to the offline optimum with the adjustable control parameter increasing $V$. Besides, the bound of migration cost queue backlog is also determined by the parameter V. In short, a performance-cost trade-off of $[O(1/V), O(V)]$ exists in our online algorithm, where we can set the parameter $V$ to a desirable value to achieve the balance of the long-term delay performance and migration cost.

## VI. Performance Evaluation

In this section, we conduct numerical studies to evaluate the time-averaged perceived latency performance under the long-term migration cost constraint of the proposed algorithms and to verify the derived theoretical results.

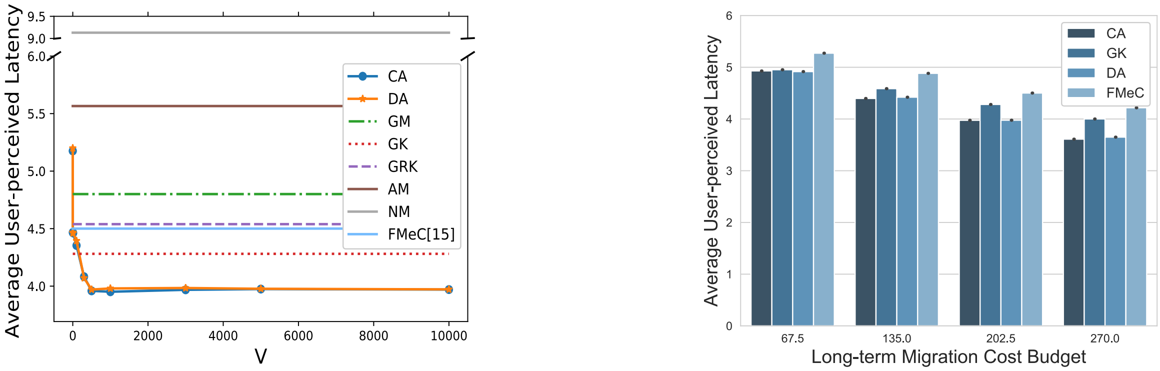
### A. Simulation Setup

We adopt the ONE simulator [39] to conduct system simulation, where mobile devices move along the roads or streets based on Shortest Path Map-Based Movement Model [39] in a downtown Helsinki, Finland. For simplicity, we divide the whole area into 63 square parts. Each part occupies 500×500 $m^2$, endowed with one MEC node to provide mobile services. Each MEC sever is equipped with multiple CPU cores, and the maximum computing capacity $F_i = 25GHz$. Besides, considering diverse mobility patterns in realistic environment, we choose two typical kinds of mobile users: about $85.7\%$ $(\frac{6}{7})$ of the mobile users is pedestrians with speed uniformly distributed in $[0.5, 1.5]$ m/s, the remaining users are drivers with speed uniformly distributed in $[2.7, 11.1]$ m/s. The hop distance between two MEC nodes is calculated by Manhattan distance. We simulate 2000 time slots for our system, and the interval of a time slot is 5 minutes. During each time slot $t$, we assume that the placement for service profile of users and wireless connections between user and edge are unchanged. The request arrival process in the interval $R^k(t)$ for each user $k$ is uniformly distributed within $[0.6, 1]$ Mbps and its processing density is 2640 cycle/bit (such as 400 frame video game in [40]). To simplify the problem, we assume that the maximum computing capacity of all MEC nodes is the same and the current communication delay follows uniform distribution within $[1, 1.35]$ of the optimal delay, which is 0.6 min per hop for every service. It is the same as migration cost, perturbed by timing a random parameter in $[1, 1.35]$. The difference is that one hop migration takes 1 unit cost, and plus 0.5 unit cost in the end, which is allied to the request arrival process.

### B. Performance Benchmark

We consider two representative situations and four approaches as a benchmark to evaluate our algorithms. One situation is no matter what the distribution of mobile user is, the service VM is always migrated to execute on its nearest MEC node, i.e., "Always migration" (AM) strategy. On the contrary, another is always to keep the initial assignment policy unchanged ("No migration") strategy. Furthermore, the four algorithms are described as follows:

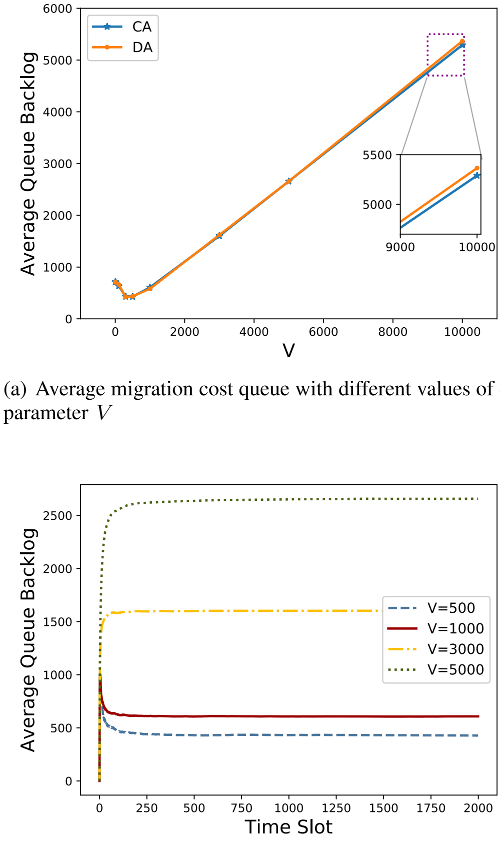
1) *GM*: this algorithm migrates the request services to the nearest MEC node at every opportunity over a long period of time.



(a) Average perceived latency performance with different values of control parameter $V$ under different placement policies

(b) Average perceived latency performance with different long-term cost budgets $E_{avg}$

Fig. 2. Optimality analysis.



(a) Average migration cost queue with different values of control parameter $V$

(b) Average migration cost queue with different values of control parameter $V$ at different time slots

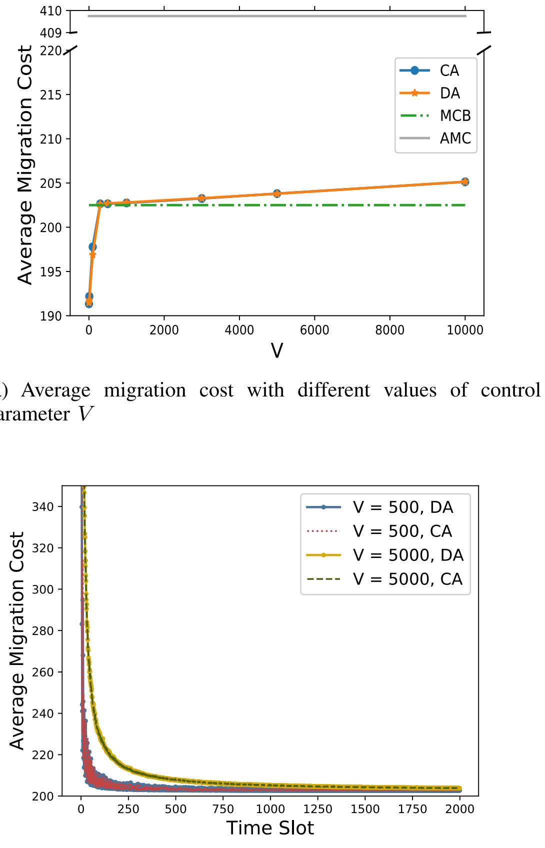
Fig. 3. Queue stability.

2) *GRK*: this algorithm randomly picks up different $K$ services and migrates them to the current optimal MEC nodes at every opportunity over a long run.
3) *GK*: this algorithm migrates $K$ services in descending order by time cost to the current optimal MEC nodes at every opportunity over a long run.
4) *FMeC* [19]: this algorithm migrates services to the current optimal MEC node based on the mobility pattern update analysis at every opportunity over a long run. Since the one-dimensional (1D) mobility model with one direction of traffic flow was assumed in [19], we assume the estimated direction of velocity in next time slot is similar to the current time slot in our simulation.

### *C. Latency Cost Trade-Off*

Obviously, the optimization of the user-perceived latency and migration trade-off in a cost-efficient manner is the key to the long-term service placement problem, which guides the following analysis for our proposed two algorithms: Markov approximation based centralized algorithm (CA) and best response update based distributed algorithm (DA).

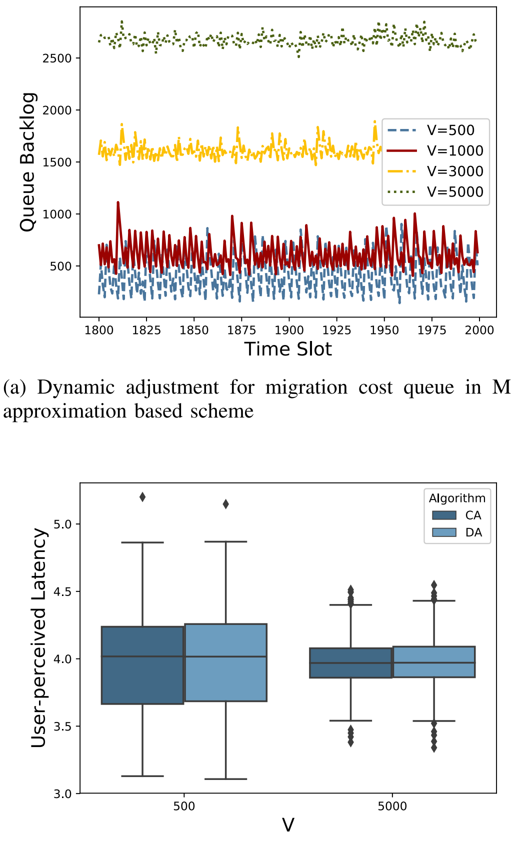
*Average user-perceived latency optimality.* To analyze key elements to influence the user-perceived latency, we formulate a standard of comparison, where 315 mobile users move in the city and the long-term time-averaged migration cost budget for the network operator is set as 202.5 cost units, approximation control parameter $\beta$ is set as 0.1. Fig. 2(a) shows the average latency with different values of control parameter $V$ under various online algorithms. We can observe that the average latency decreases with $V$ increasing, and gradually approaches a minimum value in both two proposed algorithms (i.e., CA and DA). This confirms Theorem 5 we have mentioned in the theoretical analysis that the time-averaged latency performance is proportional to the $1/V$. Besides, compared with the benchmark, our two algorithms do have remarkable improvements in average latency performance, around from 8% to 56% improvement with $V = 1000$. For the AM strategy, the major reason for the poor performance is the low utilization of edge resources. In general, only almost two-thirds of the MEC nodes provide all user services during every time slot $t$. Even though GRK and GK make up this deficiency of the inefficient utilization, the unreasonable migration policy still exists since every migration selection update is a local optimization. Furthermore, we find the particular migration sequence, such as descending order, can alleviate the performance gap to some extent. The overload among MEC nodes and inaccurate mobility pattern analysis deteriorate the latency performance of the FMeC algorithm. Intuitively, a larger migration cost budget can provide more opportunities for further enhancements on the placement optimization. As illustrated in Fig. 2(b), with



(a) Average migration cost with different values of control parameter $V$

(b) Average migration cost with different values of control parameter $V$ at different time slots

Fig. 4. Convergence of average migration cost.



(a) Dynamic adjustment for migration cost queue in Markov approximation based scheme

(b) Distribution of user-perceived latency with different values of control parameter $V$

Fig. 5. Adaptability to system dynamics.

the cost budget increasing, CA and DA have more notable improvements compared with GK and FMeC algorithms.

*Queue stability.* Fig. 3 (a) compares the time-averaged migration cost queue between CA and DA algorithms with different values of control parameter $V$. Broadly, as $V$ increases, the time-averaged backlog queue increases in a linear fashion, which is matched in Theorem 6. Besides, the CA scheme has a slightly better performance in queue backlog with a large value of $V$. Along with Fig. 2(a), the performance of time-averaged latency and migration cost follows the $[O(1/V), O(V)]$ trade-off. As shown in Fig. 3(b), the varying curve of average migration backlog queue gradually becomes stable in our algorithm no matter what V is. It implies our proposed algorithms will satisfy the long-term cost budget, and the detailed discussion is presented later.

*Convergence of average migration cost.* Fig. 4(a) plots the average migration cost with different values of $V$ under two proposed algorithms. Note that the migration cost budget (MCB) is almost half of the all services migration cost (AMC). In this situation, a large value of $V$ makes system care more about user-perceived latency, which may violate the long-term cost budget in finite time slots, such as $V = 5000$. While in Fig. 4(b), as time slot increases, the average migration cost decreases remarkably and gradually converges to the migration cost budget under different values of control parameter $V$. The reason for this problem is insufficient time slots. As we have discussed, if the migration cost queue is stable, i.e., $\lim_{T\to\infty} \mathbb{E}\{Q(T)\}/T = 0$, the actual migration cost would not violate the budget. In Fig. 3(b), we know that all migration backlog queues gradually converge to some certain finite value. Thus, if increasing time slots, the long-term constraint can be satisfied.

*Adaptability to system dynamics.* To further explore the dynamic adjustment of our algorithm to minimize the average latency performance under the control of migration cost budget, we depict a part of real-time fluctuation of the migration backlog queue and the distribution of latency performance with different values of $V$. As shown in Fig. 5(a), for a clear fluctuation exposition, we select a partial time snippet from the long run. It can be observed that the real-time migration backlog queue fluctuates frequently, which means the system will adjust migration policy frequently. When the current migration backlog queue is large and thus the remaining available migration cost is relatively scarce, the CA algorithm will endeavor to reduce the queue backlog to prevent the over-budget. Contrarily, when the current migration backlog queue is small, minimizing the time cost is the prime goal since the remaining available migration cost is abundant. Surprisingly, the large value of $V$ reduces the fluctuation range of the migration backlog queue, which makes system real-time latency performance more stable. Fig. 5(b) compares the distribution latency performance between two algorithms with

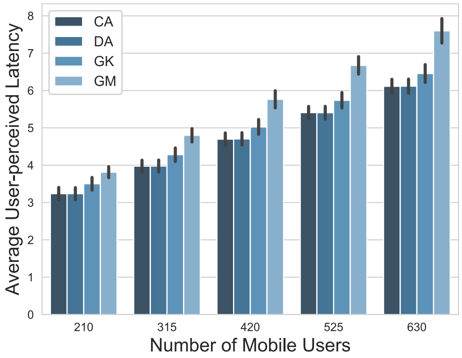


Fig. 6. Average perceived latency performance in different dense networks.

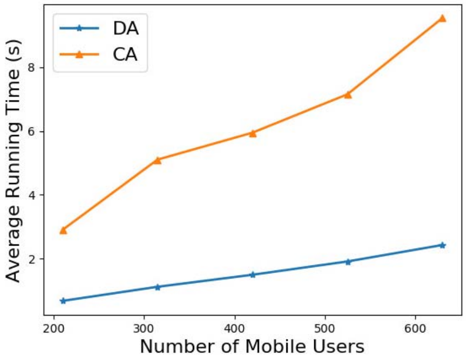


Fig. 7. The average running time for placement update in different dense networks.

different values of $V$. It is obvious that the distribution of latency performance is more centralized to the median as the control parameter $V$ increasing, which is consistent with the dynamic adjustment of migration backlog queue.

### *D. Efficient on Different Dense Networks*

Fig. 6 suggests that our two algorithms still work efficiently in different dense networks (i.e., the percentage ratio of user amount to MEC node amount). Higher user dense network will lead to the rapid growth of computation delay, which is the main factor of perceived latency increasing. Our algorithm can balance the edge load by migrating services to slow the total perceived latency growth. Nevertheless, the computation delay has been growing significantly faster than total perceived latency, while the network propagation delay is not affected by user amount. To maintain the original quality of service, the network operator should improve the computation capacity of edges accordingly.

### *E. Process Time for Placement Decision-Making*

We evaluate the proposed algorithms on Intel Core i7-6700 CPU (4 Cores @ 3.4G) computer with PyCharm. Fig. 7 suggests that the distributed scheme (DA) can dramatically reduce the running time of placement decision-making compared with the Markov approximation based scheme (CA), especially when the amount of users is large. The critical reason is that the asynchronous best response update can converge faster into an equilibrium. This result demonstrates the distributed scheme is more efficient in a large-scale and ultra dense network. Besides, the trend of curves for both two algorithms is consistent with their total time complexities.

## VII. Conclusion

In this paper, we study the mobile edge service performance optimization problem with long-term time-averaged migration cost budget. We design a novel mobility-aware online service placement framework to achieve a desirable balance between time-averaged user-perceived latency and migration cost. To tackle the unavailable future system information, which involves mobility pattern and request arrival processes, we utilize Lyapunov optimization technique to incorporate the long-term budget into a series of real-time optimization problems. Since the decomposed optimization is an NP-hard problem, we develop two efficient heuristic schemes based on the Markov approximation and best response update techniques to approach a near-optimal solution. Furthermore, we provide the theoretical proof of our proposed algorithm performance. Besides, extensive simulation demonstrates the effectiveness of our online algorithm while maintaining the long-term migration cost constraint. Future research direction includes combining online network selection in ultra dense networks. To gain a lower end-to-end latency, a user will choose an optimal one (such as a less congestion one) out of multiple access networks (e.g., cellular macrocell, femtocell, and WiFi networks). Further, we aim to extend the existing federated edge cloud framework by incorporating the Device-to-Device (D2D) collaboration, where mobile users can beneficially share their computation and communication resource in a closer proximity.

## References

[1] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, 2018, pp. 1–10.

[2] M. Patel, Y. Hu, P. Hede, J. Joubert, C. Thornton, and B. Naughton, "Mobile edge computing–introductory technical white paper," ESTI, Sophia Antipolis, France, White Paper 1, 2014.

[3] "Cisco global cloud index: Forecast and methodology, 2014–2019," Cisco, San Jose, CA, USA, White Paper 5471110-15, 2013.

[4] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2016, pp. 1–8.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, 2015.

[6] X. Chen, Q. Shi, L. Yang, and J. Xu, "Thriftyedge: Resource-efficient edge computing for intelligent IoT applications," *IEEE Netw.*, vol. 32, no. 1, pp. 61–65, Jan./Feb. 2018.

[7] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun. (MECOMM SIGCOMM)*, Budapest, Hungary, Aug. 2018, pp. 31–36.

[8] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, "Mobility management for femtocells in LTE-advanced: Key aspects and survey of handover decision algorithms," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 64–91, 1st Quart., 2014.

[9] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[10] Y. Bi, G. Han, C. Lin, Q. Deng, L. Guo, and F. Li, "Mobility support for fog computing: An SDN approach," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 53–59, May 2018.

[11] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.

[12] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8769–8780, Sep. 2018.

[13] C. Shen, C. Tekin, and M. van der Schaar, "A non-stochastic learning approach to energy efficient mobility management," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3854–3868, Dec. 2016.

[14] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. INFOCOM*, vol. 12, Mar. 2012, pp. 2876–2880.

[15] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proc. IEEE*, vol. 102, no. 1, pp. 11–31, Jan. 2014.

[16] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[17] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[18] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[19] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[20] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 1291–1296.

[21] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1350–1354.

[22] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2014, pp. 835–840.

[23] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, to be published.

[24] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Netw. Conf. (IFIP Networking)*, May 2015, pp. 1–9.

[25] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2082–2091, Dec. 2017.

[26] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[27] R. Urgaonkar *et al.*, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, Sep. 2015.

[28] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[29] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 872–880.

[30] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, and D. Towsley, "Fast transparent virtual machine migration in distributed edge clouds," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, Art. no. 10.

[31] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, 2017, Art. no. 11.

[32] M. J. Neely, "Stochastic network optimization with application to communication and queuing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[33] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," Sun Yat-sen Univ., Guangzhou, China, Tech. Rep., 2018. [Online]. Available: https://tinyurl.com/y98pq2gl

[34] L. Blumrosen and S. Dobzinski, "Welfare maximization in congestion games," in *Proc. 7th ACM Conf. Electron. Commerce*, Ann Arbor, MI, USA, Jun. 2006, pp. 52–61.

[35] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[37] X. Chen and J. Huang, "Database-assisted distributed spectrum sharing," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2349–2361, Nov. 2013.

[38] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games Econ. Behav.*, vol. 13, no. 1, pp. 111–124, 1996.

[39] A. Kerönen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proc. 2nd Int. Conf. Simulation Tools Techn.*, 2009, Art. no. 55.

[40] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

**Tao Ouyang** is currently pursuing the master's degree with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include mobile edge computing, online learning, and optimization.

**Zhi Zhou** received the B.S., M.E., and Ph.D. degrees from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2012, 2014, and 2017, respectively. In 2016, he joined the University of Gottingen as a Visiting Scholar. He is currently a Research Associate Fellow with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include edge computing, cloud computing, and distributed systems.

**Xu Chen** received the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2012. He was a Post-Doctoral Research Associate with Arizona State University, Tempe, AZ, USA, from 2012 to 2014, and a Humboldt Scholar Fellow with the Institute of Computer Science, University of Gottingen, Germany, from 2014 to 2016. He is currently a Full Professor with Sun Yat-sen University, Guangzhou, China, and the Vice Director of the National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He was a recipient of the 2014 Hong Kong Young Scientist Runner-Up Award, the 2016 Thousand Talents Plan Award for Young Professionals of China, the 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, and the 2017 IEEE ComSoc Young Professional Best Paper Award. He received the prestigious Humboldt Research Fellowship awarded by the Alexander von Humboldt Foundation, Germany, the Honorable Mention Award from the 2010 IEEE International Conference on Intelligence and Security Informatics, the Best Paper Runner-Up Award from the 2014 IEEE International Conference on Computer Communications, and the Best Paper Award from the 2017 IEEE International Conference on Communications. He is currently an Associate Editor of the IEEE INTERNET OF THINGS JOURNAL and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Network Softwarization and Enablers.