Dual-Side Optimization for Cost-Delay Tradeoff in Mobile Edge Computing

Article *in* IEEE Transactions on Vehicular Technology · October 2017

DOI: 10.1109/TVT.2017.2762423

3 authors, including:

Jeongho Kwak
Trinity College Dublin
18 PUBLICATIONS **187** CITATIONS

SEE PROFILE

Song Chong
Korea Advanced Institute of Science and Technology
155 PUBLICATIONS **4,540** CITATIONS

SEE PROFILE

Dual-Side Optimization for Cost-Delay Tradeoff in Mobile Edge Computing

Yeongjin Kim, *Student Member, IEEE*, Jeongho Kwak, *Member, IEEE*, and Song Chong, *Member, IEEE*

***Abstract*—Mobile code offloading (MCO) is a technology that offloads computing tasks from mobile devices to remote servers managed by code offload service providers (CSPs). Nowadays, it is recommended that the remote servers be placed on the edge of the network to support modern applications, e.g., augmented reality, which demand stringent and ultralow latency or high bandwidth. To date, previous studies have independently addressed code offloading policy in mobile devices and pricing/server provisioning policies in the CSP; moreover, the system models for both user side and CSP side have not adequately reflected their practical aspects. This paper designs a practical model for the both sides and takes account of them in an integrated MCO framework simultaneously. By leveraging Lyapunov drift-plus-penalty technique, we propose code offloading, local CPU clock frequency, and network interface selection policies for mobile users, and propose MCO service pricing and server provisioning policies for the CSP in each of a competition scenario and a cooperation scenario. In the competition scenario, we propose a Com-UC algorithm for mobile users and a Com-PC algorithm for the CSP with the aim to minimize each cost for each queue stability constraint. In the cooperation scenario, we propose a Coo-JC algorithm with the aim to minimize their sum cost for both mobile users and CSP. Via trace-driven simulations, we demonstrate that Com-UC saves at most 71% of its cost and Com-PC attains 82% profit gain for the same delay compared to existing algorithms; moreover, the cooperation between mobile users and CSP additionally reduces costs and delays.**

***Index Terms*—Cost minimization, competition scenario and cooperation scenario, code offloading service provider (CSP), edge computing, mobile code offloading, mobile users.**

I. Introduction

MOBILE code offloading (MCO) is arising as a prominent technology where mobile devices offload processing-heavy tasks such as voice/image/video processing to remote servers. The remote servers are typically located in the data centers which have extremely higher computing capacity than the mobile devices whereas they are geometrically far from the mobile devices. The task computing in these data centers is called cloud computing. Although cloud computing enables convenient access of MCO service, it frequently cannot satisfy the requirements of latency-sensitive or mobility-aware applications [2], [3]. Moreover, due to a growing amount of distributed data generated by pervasive mobile devices and Inter of Things (IoT) devices, a core network of current network infrastructure (especially in cellular network) is highly congested which makes it hard to transport all the data to the cloud data center [4], [5]. To overcome these limitations, edge computing [2], [3] is recently emerging as one of promising technologies in next-generation networks. The main idea of the edge computing is to place remote servers for MCO at the edge of network, which is closer to end users [6], as opposed to cloud computing. Thereby, the edge computing provides low latency and mobility-aware MCO services to the end users.

IDC (International Data Corporation) reports that more than 40% of data will be stored, processed, analyzed and acted upon close to, or at the edge of, the network by 2019 [7]. Moreover, many consortia have recently proposed new edge computing architectures, e.g., OFC [8], MEC [9] and Cloudlets [10]. According to ETSI (European Telecommunications Standards Institute) [11], edge servers will be located on an LTE macro BS, a Radio Network Controller (RNC) or a multi-technology cell aggregation site (e.g., company, campus and stadium) for future 5G cellular networks.

Mobile devices would consume significant amount of energy by processing a lot of tasks using the local CPU because the local CPU power exponentially increases with respect to the increasing clock frequency [12]. Mobile users are able to use the MCO service to save processing energy and enhance computing performance by offloading heavy tasks to edge servers. However, the mobile devices not only consume networking energy (instead of processing energy) to transfer computing tasks to the edge servers, but also should make a payment for a usage of the MCO service and cellular network. This tradeoff causes code offloading decision, i.e., either processing the computing tasks locally or transferring them to the edge servers, clock frequency adjustment and network interface selection (between cellular and Wi-Fi) problems in a mobile device. Control parameters of three problems should be decided by taking account of the amount of computing tasks, processing

Manuscript received January 23, 2017; revised July 5, 2017; accepted October 1, 2017. Date of publication October 12, 2017; date of current version February 12, 2018. This work was supported by the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2015-0-00557, Resilient/Fault-Tolerant Autonomic Networking Based on Physicality, Relationship and Service Semantic of IoT Devices, No. B0717-17-0034, Versatile Network System Architecture for Multidimensional Diversity). This paper was presented in part at the WiOpt 2015, Mumbai, India [1]. The review of this paper was coordinated by Prof. J.-M. Chung. (*Corresponding author: Jeongho Kwak.*)

Y. Kim and S. Chong are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail: yj.kim@netsys.kaist.ac.kr; songchong@kaist.edu).

J. Kwak was with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea. He is now with the CONNECT, Trinity College Dublin, Dublin 2, Ireland (e-mail: jeongho.kwak@tcd.ie).

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/TVT.2017.2762423

density[1], processing capability, network speed and energy consumption of mobile devices and service prices for the MCO and cellular network.

Code offloading service provider (CSP) is able to make a profit on providing MCO service to subscribers whereas it makes a payment for a usage of electricity to operate edge servers; hence there is a tradeoff between revenue (from the subscribers) and expenditure (to the electricity provider). This tradeoff causes MCO service pricing and server provisioning, i.e., activation and allocation of edge servers, problems in a CSP. Control parameters of these problems should be decided by taking account of willingness to pay of mobile users, processing density of computing tasks and electricity bill. Because some factors for mobile users (e.g., network speed [13]) and CSP (e.g., electricity bill [14]) are time-varying in mobile environments, the mobile users and CSP should carefully design their policies in the light of varying conditions. Moreover, the joint problem of mobile users and CSP is much more challenging because they are intertwined with each other.

The CSP can be one of commercial corporations such as a network operator (e.g., AT&T) and a third-party OTT (Over-the-top) service vendor (e.g., Microsoft Azure [15]) or one of public institutions such as a university and a hospital [16][2]. We should be noted that the public institutions are likely to provide free code offloading service to their members[3]. Hence, we can classify realistic edge computing scenarios into a competition scenario and a cooperation scenario between CSP and mobile users. In the competition scenario, an aim of the CSP (e.g., network operator) is to maximize the profit whereas mobile users aim to minimize their costs (e.g., energy consumption and monetary costs) by using the MCO service with a payment to the CSP. On the other hand, in the cooperation scenario, a unified objective of the CSP (e.g., university) and mobile users (e.g., students) is to maximize their common benefits.

In this paper, we formulate dual-side control[4] problems in a mobile edge computing system under two different scenarios as follows. *(i)* Competition scenario: users and CSP compete, i.e., they aim to minimize their own costs for queue stability constraints, respectively. *(ii)* Cooperation scenario: users and CSP cooperate aiming to minimize the social cost for queue stability constraints. We leverage "Lyapunov drift-plus-penalty" technique [17] to design dual-side control algorithms which do not require future task arrivals, network states and billing information. To capture the realistic MCO systems, we take account of the interaction between the mobile users and CSP, finite computing capacity of edge servers, heterogeneity of processing density and time-varying electricity price of the active edge servers. We execute trace-driven simulations based on real-world measurement parameters such as LTE/Wi-Fi data rates and processing/networking energy consumption in popular smartphone models.

The contributions of this paper are summarized as follows.

1) We design a realistic model for the strategies and environments of mobile users and code offloading service provider (CSP) in a unified edge computing framework.
2) We introduce competition and cooperation scenarios between mobile users and a CSP. In the competition scenario, mobile users and a CSP are selfish entities, e.g., the CSP is a commercial corporation. In the cooperative scenario, a CSP can be considered as a public institution such as school; hence mobile users and the CSP cooperate with each other to minimize their total cost.
3) We propose user-side (Com-UC) and CSP-side (Com-PC) algorithms aiming to minimize costs of the mobile users and CSP, respectively, while ensuring finite processing time under a competition scenario. More specifically, Com-UC jointly controls code offloading policy, CPU clock speed and wireless interfaces, and Com-PC jointly controls the MCO service price and the number of active edge servers. In addition, we propose a dual-side algorithm, namely Coo-JP, which aims to minimize social cost while ensuring finite processing time under a cooperation scenario.
4) We rigorously evaluate our algorithms using actual parameters measured by us in real network environment and collected parameters given by various high-impact references. Then we demonstrate that Com-UC saves 71% of cost by trading 6 MB of average queue backlogs (60 seconds delays) and Com-PC attains 82% of profit gain for the same delay compared to existing schemes; moreover, the users-CSP cooperation enables them to attain 28% of additional social cost reduction and 30% of additional delay reduction.

In the rest of this paper, we summarize related work on mobile code offloading in Section II. We describe system model in Section III. In Section IV, we propose control algorithms under competition and cooperation scenarios, respectively. Next, in Section V, we evaluate proposed algorithms via trace-driven simulations. Finally, we conclude our paper in Section VI.

II. Related Work

User-side mobile code offloading: Since the seminal work of Cuervo *et al.* [18], there have been extensive studies on the user-side MCO [19], [20] aiming to minimize energy consumption and processing delay. Kosta *et al.* [19] proposed a code offloading algorithm that takes account of offloading costs, energy consumption and execution time priorities. However, it was a heuristic algorithm which is far from optimal. Wen *et al.* [20] proposed an optimal code offloading scheme aiming to minimize energy consumption of a mobile device for a given processing delay constraint. However, they overlooked the change of computing task size, network selection or a multi-homed transmission of mobile devices. By way of exception, Kwak *et al.* [13] considered them in a mobile code offloading framework with

[1]This is defined as required CPU cycles to process one bit of tasks. Each computing task generally requires different cycles in processing the same amount of data in bits.

[2]The author in [16] provided concrete use cases of mobile edge computing in the public institutions.

[3]Similarly, some university provides free Wi-Fi service to it's students within a campus.

[4]This means that we jointly control user-side and CSP-side.

heterogeneous types of tasks. However, the processing capacity of the edge server was assumed to be infinite, thus the processing delay at CSP-side was not considered. Moreover, they only focused on a minimization of device energy consumption, except for monetary cost such as cellular data charges and MCO service charges. Chen *et al.* [21] studied code offloading decision among multiple mobile users who share common wireless access based on game theoretical approach.

CSP-side mobile code offloading: Commercial CSPs such as Windows Azure have adopted a fixed pricing for unit computing resources in the MCO service [15]. There have been studies to address the CSP-side MCOs [22]–[28], that aim to minimize CSP costs, e.g., electric bills for operating remote servers, or maximize CSP's revenue obtained from subscribers. Lin *et al.* [22] proposed a dynamic server sizing algorithm to reduce date center energy consumption by deactivating servers for pre-defined periods. Liu *et al.* [23] considered profit maximization of CSP by controlling admission and server allocation of requested tasks. As an extension of [23], Chen *et al.* [24] considered additional control dimension, frequency scaling of server CPUs, to achieve additional energy saving. Ren *et al.* [25] considered load balancing of computing tasks among geo-distributed data centers aiming to minimize energy consumption of data centers and guarantee throughput fairness among different client groups. However, these works [22]–[24] overlooked intertwined relations between pricing control for MCO services and offloading decision of end users. In other words, these studies assumed that the amounts of offloading requests from end users are already given to the CSP, regardless of policies of the CSP and end users.

To capture the heterogeneous offloading decisions of end users depending on the MCO service pricing, Wang *et al.* [26] proposed an auction-based pricing algorithm aiming to increase revenue of the CSP and allow users to evenly compete in order to access computing resources. Ren *et al.* [27] and Zhao *et al.* [28] developed a joint computing task scheduling and pricing technique aiming to maximize CSP's profit by leveraging Lyapunov optimization [17]. Even though dynamic pricing which controls the amount of offloading requests is considered in [26]–[28], their system models did not capture various user-side factors, which affect user's offloading decision, such as network status, processing capacity of local device and amount of remaining computing tasks.

Cooperative mobile code offloading: Menache *et al.* [29] proposed a pricing and computing resource allocation algorithm so as to regulate admissions of subscribers, where the objective is to maximize social welfare of mobile users and CSP by exploiting mobile code offloading system. However, they overlooked the local processing adjustment and energy consumption of mobile devices. Song *et al.* [30] proposed a cooperative code offloading scheme among mobile users by sharing local computing resources among them under WLAN environments. Chen *et al.* [31] studied cooperative code offloading among individual edge servers deployed at small base stations (SBSs) to reduce the latency of MCO services. However, these works [30], [31] ignored collaboration between mobile devices and CSP.

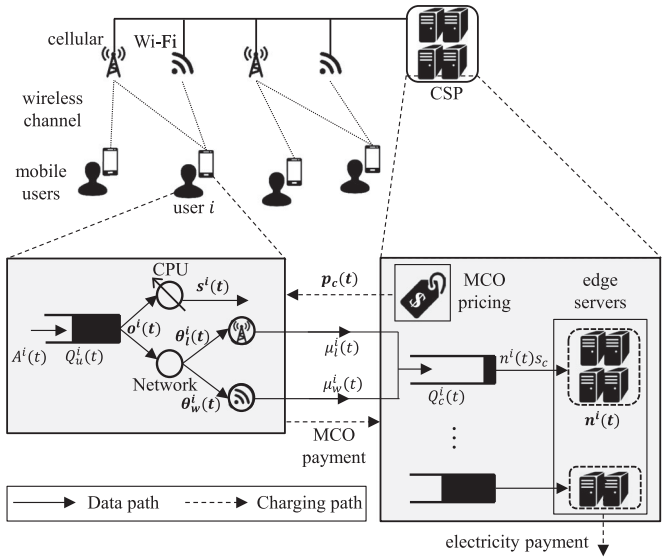


Fig. 1. Mobile code offloading in an edge computing system.

III. System Models

We illustrate mobile code offloading (MCO) mechanism in an edge computing system as shown in Fig. 1. We consider a code offloading service provider (CSP) which manages multiple edge servers located closer to end users (e.g., multi-technology cell aggregation site [16]) and supports code offloading services. We denote $i \in \mathcal{I}$ by a mobile user registered in the CSP for the MCO service. The user i informs the CSP of his/her device profile (e.g., smartphone's model number or specifications) and using applications. We consider a time-slotted system, indexed by $t \in \mathcal{T} = \{0, 1, ...T-1\}$. From now on, we describe detailed models for several parts of the edge computing system in Fig. 1.

A. Computing Task and Arrival Models

At each time slot t, $A^i(t)$ (in bits) amount of computing tasks are requested from user i. We assume that $A^i(t)$ is an i.i.d.[5] process and bounded every time slot, i.e., $A^i(t) \leq A^i_{\max}$, for all $i \in \mathcal{I}$. Each user executes a different type of application characterized by a processing density γ^i (in cycles/bit) which is defined as required CPU clock cycles to process a unit bit of computing task of user i. The application of user i is assumed to be pre-installed in both device of user i and edge servers. Target computing tasks in our system are able to be separated into independent sub-tasks, and have delay-tolerant features[6] (examples of such kind of applications can be found in many code offloading studies, e.g., [18] and references are therein). The arrival tasks are buffed into the user-side queue $Q_u^i(t)$, defined in Section III-D, and can be processed by two types of processing resources: *(i)* the local CPU resources in each mobile device, and *(ii)* the CPU resources in the remote edge servers.

[5]Independent and identically distributed.
[6]They do not have instantaneous delay constraints.

B. Mobile User Resource Model

Typically, modern smartphones have DVFS (Dynamic Voltage and Frequency Scaling) capability [32][7], so user i can adjust CPU clock speeds, i.e., $s^i(t) \in \mathcal{S}$ (in cycles/time slot), every time slot where $\mathcal{S} = \{0, s_1, s_2, \ldots, s_{\max}\}$ is defined as the set of adjustable CPU clock speeds of mobile device, and $s^i(t) = 0$ means that user i stops processing the computing tasks and let the CPU be in the idle state.

We consider two types of wireless interfaces, i.e., cellular and Wi-Fi, to transfer data from a smartphone to edge servers. Due to the mobility and traffic load fluctuation, network availabilities and achievable rates of cellular and Wi-Fi networks vary over time. We denote $\mu_l^i(t) \in [0, \mu_{l,\max}^i]$ and $\mu_w^i(t) \in [0, \mu_{w,\max}^i]$ (in bits/time slot) by the amount of computing tasks that can be transferred through cellular and Wi-Fi network interfaces of user i at time slot t, respectively, e.g., if the Wi-Fi is not available, $\mu_w^i(t) = 0$. Because the edge servers are located closer to the end users, $\mu_l^i(t)$ and $\mu_w^i(t)$ are predictable by means of online and offline estimations using received signal strength indicator (RSSI) and history of past data rates [33].

At each time slot, a user i firstly decides whether to process the computing tasks locally or to transfer the computing tasks to the edge server, namely offloading policy denoted by $o^i(t)$.

$$o^i(t) = \begin{cases} 1, & \text{if the computing tasks in a mobile device are} \\ & \text{transferred to the edge servers,} \\ 0, & \text{if the computing tasks in a mobile device are} \\ & \text{processed in the local CPU.} \end{cases}$$

If local computing is decided, the user i selects local CPU clock speed $s^i(t)$. Otherwise, the user i decides active network interface between cellular and Wi-Fi to transfer the computing tasks to the edge servers as follows[8].

$$\theta_l^i(t) = \begin{cases} 1, \text{ if cellular interface is activated,} \\ 0, \text{ otherwise.} \end{cases}$$

$$\theta_w^i(t) = \begin{cases} 1, \text{ if Wi-Fi interface is activated,} \\ 0, \text{ otherwise.} \end{cases}$$

In addition, we consider state-of-the-art multi-homing technology of mobile devices which is able to use cellular and Wi-Fi networks simultaneously, $\boldsymbol{\theta}^i(t) = (\theta_l^i(t), \theta_w^i(t)) = (1, 1)$. It boosts the network speed through MPTCP [34], [35] or HTTP aggregation [36].

C. CSP Resource Model

Every time slot, the CSP decides a computing price for unit CPU cycle $p_c(t) \geq 0$ (in \$/cycle) to provide the MCO services. In our system model, the CSP adopts time-dependent pricing scheme, i.e., the computing price can be changed over time according to the CSP's policy [28]. The CSP has $n_{\max}(t)$ available edge servers where $n_{\max}(t) \in \{0, 1, ..., N\}$ can be changed every time slot [27]. The finite processing capacity of an edge server is denoted by s_c (in cycles/slot). Then, the CSP decides the number of active edge servers $n^i(t)$ for processing computing tasks requested by user i, namely server provisioning. Because the total number of active edge servers cannot exceed the number of available ones, we consider a constraint as follows.

$$\sum_{i \in \mathcal{I}} n^i(t) \leq n_{\max}(t).$$

D. Queueing Model

For a mobile user $i \in \mathcal{I}$, we denote $Q_u^i(t)$ by a user-side queue of computing task (in bits), and for CSP, we denote $Q_c^i(t)$ by the CSP-side queue (in bits) at time slot t. These queues represent a volume of unfinished tasks requested by user i. For MCO service, $Q_u^i(t)$ and $Q_c^i(t)$ are commonly shared by user i and the CSP[9]. We model the queueing dynamics at the user-side and the CSP-side, respectively, as follows.

$$Q_u^i(t+1) = \left[Q_u^i(t) - \left(1 - o^i(t)\right) \frac{s^i(t)}{\gamma^i} - o^i(t) R^i\left(\boldsymbol{\theta}^i(t)\right) + A^i(t) \right]^+, \tag{1}$$

$$Q_c^i(t+1) = \left[Q_c^i(t) - n^i(t) \frac{s_c}{\gamma^i} + o^i(t) R^i\left(\boldsymbol{\theta}^i(t)\right) \right]^+, \forall i \in \mathcal{I}, \tag{2}$$

where $[x]^+ = \max(x, 0)$, and $R^i(\boldsymbol{\theta}^i(t)) \doteq \theta_l^i(t)\mu_l^i(t) + \theta_w^i(t)\mu_w^i(t)$ denotes the amount of computing tasks transferred to the edge servers when user i determines $\boldsymbol{\theta}^i(t)$. The amount of departure from the user-side queue depends on the control parameters $(o^i(t), \theta_l^i(t), \theta_w^i(t), s^i(t))$. The amount of arrival to and departure from the CSP-side queue depend on the control parameters $(o^i(t), \theta_l^i(t), \theta_w^i(t))$ and $n^i(t)$, respectively. Because the unit of $s^i(t)$ in the local CPU of mobile device (and s_c in the remote CPU of edge servers) is cycles/time slot and that of $Q_u^i(t)$ (and $Q_c^i(t)$) is bits, the amount of serviced workloads (in bits) from the queue is $s^i(t)$ (and s_c) divided by γ^i for a unit agreement.

E. Cost Model

Usage of the local CPU resources to process computing tasks in a mobile device consumes the CPU energy $E_s(s^i(t))$ (in J/time slot)[10] that is a function of CPU clock speed. We assume that $E_s(\cdot)$ is a convex, differentiable and increasing function of the clock speeds $s^i(t)$ [12][11]. On the other hand, if the computing tasks are processed by computing resources of edge servers, various types of costs are required for a mobile user as follows: *(i)* networking energy cost (E_l for cellular and E_w for Wi-Fi) in transferring computing tasks to the edge server (in J/time slot), *(ii)* the MCO service cost ($p_c(t)\gamma^i\mu_l^i(t)$ for cellular and

[7] In the typical DVFS scheme [32], CPU speed is set to be maximized when the amount of workloads is greater than a threshold, and proportional to workloads when it is less than the threshold which can be controllable manually.

[8] Notice that if an interval of network selection is long enough, e.g., 1 min, then transition delay or cost between Wi-Fi and cellular interfaces can be ignored.

[9] It is possible if users and CSP contract with each other in advance.

[10] We do not consider commonly using energy caused by display and peripheral devices in this paper. The sum proportion of energy consumption of CPU and network interface of typical smartphones is known as about 49% of total energy [37].

[11] We showed that the CPU energy consumption of smartphones is well fitted to the cubic function on the clock speed by real measurements in Section V.

$p_c(t)\gamma^i\mu_w^i(t)$ for Wi-Fi) of which the user has to pay to the CSP and *(iii)* a data usage cost $p_l\mu_l^i(t)$ applied only to the cellular network where p_l (in \$/bit) is a cellular data price per bit. We assume that the usage of Wi-Fi networks is free of charge. On the other side, the CSP makes the same amount of money from mobile users by processing tasks in the edge servers instead of the mobile devices. However, the CSP should pay electricity bill to the electric utility grid for operating edge servers. We denote $e(t)$ by an electricity price (in \$) required to activate one edge server during time slot t. In summary, the costs (in \$) of user $h_u^i(t)$ and the CSP $h_c(t)$ at time slot t are presented as follows.

$$\begin{aligned}\text{User } i\text{: } h_u^i(t) =\ & \left(1-o^i(t)\right)\alpha^i E_s\left(s^i(t)\right)\\ & + o^i(t)\left(\theta_l^i(t)D_l^i(t)+\theta_w^i(t)D_w^i(t)\right),\\ \text{CSP: } h_c(t) =\ & \sum_{i\in\mathcal{I}} n^i(t)e(t) - \sum_{i\in\mathcal{I}} o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\gamma^i p_c(t),\end{aligned}$$

where $D_l^i(t) \doteq \alpha^i E_l + p_c(t)\gamma^i\mu_l^i(t) + p_l\mu_l^i(t)$ and $D_w^i(t) \doteq \alpha^i E_w + p_c(t)\gamma^i\mu_w^i(t)$ are the incurred costs when the user i offloads the computing tasks through cellular and Wi-Fi networks, respectively. For user i, α^i (in \$/J) is a weight parameter which transforms device energy consumption into money. It is a human factor decided by sensitiveness on money and energy, and we assume that each user determines α^i in advance.[12]

F. Structure of the System Models

We describe the system flow based on competition scenario between mobile users and CSP. In this scenario, mobile users and CSP are independent entities, e.g., the CSP is a commercial corporation such as a network operator and a third-party OTT service vendor.

At the beginning of each time slot, the CSP is aware of the states of queue backlogs and networks (E, B), and makes a decision of pricing and server provisioning (C). Then, each mobile user makes a decision of its offloading policy, local CPU clock speed and network interface activation in response to the service price given by the CSP (B). In accordance to the decisions of mobile users and CSP, computing tasks are offloaded to edge servers or directly serviced in each device where queueing model (D) captures the dynamic arrival and departure of offloaded or serviced amount of computing tasks. In addition, energy consumption and payment (or exchange) of monetary cost stated by the cost model (E) are also affected by the control parameters (i.e., decisions of mobile users and CSP). Finally, new computing tasks are requested by mobile users (A) and buffered to the user-side queue by the queueing model (D).

On the other hand, in a cooperation scenario, a CSP is a public institution such as school or hospital; hence the CSP provides mobile users with complimentary MCO service. In this scenario, the system flow is about the same as the competition case except the decision on pricing of CSP, i.e., the charging path in Fig. 1 is disappeared. Table I summarizes the notations in this section for ease of understanding.

[12]In a simulation part, we set α^i based on a price for fast charging of smartphone battery in South Korea.

TABLE I
SUMMARY OF NOTATIONS.

Notation	Meaning
I	Set of mobile users
$A^i(t)$	Arrival of computing tasks for user i at t
γ^i	Processing density of computing task for user i
$\mu_l^i(t)$	Amount of computing tasks of user i that can be transferred through cellular interface at t
$\mu_w^i(t)$	Amount of computing tasks of user i that can be transferred through Wi-Fi interface at t
$o^i(t)$	Offloading policy of user i at t
$\theta_l^i(t)$	Cellular interface activation of user i at t
$\theta_w^i(t)$	Wi-Fi interface activation of user i at t
$s^i(t)$	Local CPU clock speed of user i at t
s_c	CPU clock speed of a edge server.
$n^i(t)$	# of allocated edge severs for user i's tasks at t
$n_{\max}(t)$	Available number of edge servers at t
$p_c(t)$	Price of MCO service per unit cycle at t
p_l	Price of unit cellular data
$e(t)$	Price of unit electricity at t
$Q_u^i(t)$	User-side queue backlog of user i at t
$Q_c^i(t)$	CSP-side queue backlog of user i at t
$E_s(s^i(t))$	CPU energy consumption for the CPU clock speed $s^i(t)$
E_l	Energy consumption to activate cellular interface
E_w	Energy consumption to activate Wi-Fi interface
α^i	Weight parameter between energy and money of user i

IV. DUAL-SIDE CONTROL ALGORITHMS

In this section, we formulate two optimization problems where the objective is to minimize costs subject to queue stability under *(i)* a competition scenario (i.e., an aim of users and the CSP is to minimize their own costs, respectively) and *(ii)* a cooperation scenario (i.e., the users and the CSP cooperate aiming to minimize the sum of their costs, namely social cost). As an example of the competition scenario, we take account of a competition between network operator, which is in charge of a CSP, and subscribers, which are in charge of mobile users, where the entities (i.e., network operator and subscribers) have conflict objectives, respectively. As an example of the cooperation scenario, we take account of a free MCO service in a campus where a university is in charge of a CSP, and students are in charge of mobile users as mentioned in [16]. For the competition scenario, we design a user-side control algorithm, namely Com-UC and a CSP-side control algorithm, namely Com-PC. For the cooperation scenario, we design a joint control algorithm, namely Coo-JC. Finally, we demonstrate theoretical analysis of proposed Coo-JC in this section.

A. Problem Formulation

Our objectives and constraints under the competition scenario (Com-U and Com-P) and the cooperation scenario (Coo-UP) are summarized in Table II, where the control parameters $(\boldsymbol{o}, \boldsymbol{\theta}, \boldsymbol{s}, \boldsymbol{p}_c, \boldsymbol{n})$ are defined as follows.

$$\begin{cases}\boldsymbol{o} = (o^i(t), \forall i\in\mathcal{I}, t\in\mathcal{T}),\ \boldsymbol{\theta} = (\theta_l^i(t), \theta_w^i(t), \forall i\in\mathcal{I}, t\in\mathcal{T}),\\ \boldsymbol{s} = (s^i(t), \forall i\in\mathcal{I}, t\in\mathcal{T}),\\ \boldsymbol{p}_c = (p_c(t), \forall t\in\mathcal{T}), \qquad \boldsymbol{n} = (n^i(t), \forall i\in\mathcal{I}, t\in\mathcal{T}).\end{cases}$$

The objective of (Com-U) is to minimize the average cost of user i by controlling $(\boldsymbol{o}^i, \boldsymbol{\theta}^i, \boldsymbol{s}^i)$ for given CSP's policy. It is constrained by the queue stability of both user-side and CSP-side

TABLE II
OBJECTIVES AND CONSTRAINTS

Problem	Objective	Constraints
(Com-U)	$\min\limits_{(\boldsymbol{o}^i,\boldsymbol{\theta}^i,\boldsymbol{s}^i)} \left[\lim\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\left\{h_u^i(t)\right\}\right], \forall i$	$\limsup\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\left[Q_u^i(t)+Q_c^i(t)\right]<\infty,$ $o^i(t),\theta_l^i(t),\theta_w^i(t)\in\{0,1\},\forall i\in\mathcal{I},t\in\mathcal{T},$ $s^i(t)\in\mathcal{S},\forall i\in\mathcal{I},t\in\mathcal{T},$
(Com-P)	$\min\limits_{(\boldsymbol{p}_c,\boldsymbol{n})} \left[\lim\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\left\{h_c(t)\right\}\right],$	$\limsup\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\left[\sum\limits_{i\in\mathcal{I}}Q_c^i(t)\right]<\infty,$ $p_c(t)\geq 0,\forall t\in\mathcal{T},$ $\sum\limits_{i\in\mathcal{I}}n^i(t)\leq n_{\max}(t),\forall t\in\mathcal{T},$
(Coo-UP)	$\min\limits_{(\boldsymbol{o},\boldsymbol{\theta},\boldsymbol{s},\boldsymbol{n})} \left[\lim\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\left\{\sum\limits_{i\in\mathcal{I}}h_u^i(t)+h_c(t)\right\}\right],$	$\limsup\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\left[\sum\limits_{i\in\mathcal{I}}\left(Q_u^i(t)+Q_c^i(t)\right)\right]<\infty,$ $o^i(t),\theta_l^i(t),\theta_w^i(t)\in\{0,1\},\forall i\in\mathcal{I},t\in\mathcal{T},$ $s^i(t)\in\mathcal{S},\forall i\in\mathcal{I},t\in\mathcal{T},$ $\sum\limits_{i\in\mathcal{I}}n^i(t)\leq n_{\max}(t),\forall t\in\mathcal{T},$

in order to process all the requested computing tasks within a finite time. The objective of (Com-P) is to minimize the average cost of CSP by controlling $(\boldsymbol{p}_c,\boldsymbol{n})$ for given policy of the users. It is constrained by the queue stability of CSP-side because the CSP has to process the received tasks from users within a finite time. The objective of (Coo-UP) is to minimize the average social cost by controlling $(\boldsymbol{o},\boldsymbol{\theta},\boldsymbol{s},\boldsymbol{n})$ with the queue stability constraints of both user-side and CSP-side. The social cost at time t is defined as $\sum_{i\in\mathcal{I}}h_u^i(t)+h_c(t)$ which contains all the costs generated by the users and CSP. Because the MCO price $\boldsymbol{p}_c$ does not affect the social cost (i.e., the benefits of CSP from users is the same as the payment of users to the CSP), we do not take $\boldsymbol{p}_c$ into account in the cooperation scenario.

B. Algorithm Design

We design control algorithms for the problems (Com-U), (Com-P) and (Coo-UP) by leveraging "Lyapunov drift-plus-penalty" framework [17] where the theoretical meaning is to minimize costs (or maximize profits) by trading delay without loss of capacity (satisfying queueing stability). This technique has advantages in the sense that it does not require future information about the task arrivals, network status and electricity price, but only demands current information of their status.

Making slot-by-slot objective: First, we define Lyapunov functions and their Lyapunov drift functions as follows.

$$\text{(Com-U): } L(t)=\frac{1}{2}\left[\left(Q_u^i(t)\right)^2+\left(Q_c^i(t)\right)^2\right], \tag{3}$$

$$\Delta L(t)=\mathbb{E}\left[L(t+1)-L(t)\middle|\boldsymbol{Q}^i(t)\right], \tag{4}$$

$$\text{(Com-P): } L(t)=\frac{1}{2}\left[\sum_{i\in\mathcal{I}}\left(Q_c^i(t)\right)^2\right], \tag{5}$$

$$\Delta L(t)=\mathbb{E}\left[L(t+1)-L(t)\middle|\boldsymbol{Q}_c(t)\right], \tag{6}$$

$$\text{(Coo-UP): } L(t)=\frac{1}{2}\left[\sum_{i\in\mathcal{I}}\left\{\left(Q_u^i(t)\right)^2+\left(Q_c^i(t)\right)^2+\left(Q_u^i(t)+Q_c^i(t)\right)^2\right\}\right], \tag{7}$$

$$\Delta L(t)=\mathbb{E}\left[L(t+1)-L(t)\middle|\boldsymbol{Q}(t)\right], \tag{8}$$

where

$$\begin{cases}\boldsymbol{Q}_u(t)=(Q_u^1(t),Q_u^2(t),\ldots,Q_u^I(t)),\\ \boldsymbol{Q}_c(t)=(Q_c^1(t),Q_c^2(t),\ldots,Q_c^I(t)),\\ \boldsymbol{Q}^i(t)=(Q_u^i(t),Q_c^i(t)),\quad \boldsymbol{Q}(t)=(\boldsymbol{Q}_u(t),\boldsymbol{Q}_c(t)).\end{cases}$$

In the Lyapunov function (3), $(Q_u^i(t))^2$ is designed to stabilize the user-side queue and $(Q_c^i(t))^2$ is designed to avoid congestion at the CSP-side. The Lyapunov functions in (5) and (7) are designed to fairly stabilize queues among users. The function (5) aims to stabilize the CSP-side queue. In the function (7), $(Q_u^i(t))^2$ and $(Q_c^i(t))^2$ are designed to stabilize the user-side and CSP-side queues, respectively, and $(Q_u^i(t)+Q_c^i(t))^2$ is designed to stabilize the system-level queue for user i. Then, minimizations of Lyapunov drifts in (4), (6) and (8) for each objective function have the same meaning as the constraints in Table II, respectively.

Next, we define "Lyapunov drift-plus-penalty" functions where the penalty function is the expected cost during time slot t as follows.

$$\text{(Com-U): } \Delta L(t)+V\mathbb{E}\left[h_u^i(t)\middle|\boldsymbol{Q}^i(t)\right], \tag{9}$$

$$\text{(Com-P): } \Delta L(t)+V\mathbb{E}\left[h_c(t)\middle|\boldsymbol{Q}_c(t)\right], \tag{10}$$

$$\text{(Coo-UP): } \Delta L(t) + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\big|\boldsymbol{Q}(t)\right], \quad (11)$$

where V is a non-negative parameter that adjusts the cost and delay tradeoff, i.e., how much we take care of the cost reduction compared to the processing delay reduction[13]. We note that by Littles law [39], the average delay can be transformed as the average queue backlogs divided by the average arrival rate. Therefore, from now on, we will use average delay and average queue backlog interchangeably for a given average arrival rate. Then, our original long-term minimization objectives in Table II for (Com-U), (Com-P) and (Coo-UP) are transformed into minimizing the short-term functions (9)∼(11) in every time slot t, respectively. The key derivation step is to derive upper-bound on the Lyapunov drift-plus-penalty. **Deriving a upper bound.** We derive upper bounds of (9)∼(11) using queuing dynamics (1), (2) and bounds of computing task arrivals, CPU and network speeds assumed in Section III.

Lemma 1: Under any possible control variables $(\boldsymbol{o}(t), \boldsymbol{\theta}(t), \boldsymbol{s}(t), \boldsymbol{n}(t), p_c(t))$, we have:

$$\begin{aligned}
&\text{(Com-U):}\Delta L(t) + V\mathbb{E}\left[h_u^i(t)\big|\boldsymbol{Q}^i(t)\right]\\
&\le B_1 + V\mathbb{E}\left[h_u^i(t)\big|\boldsymbol{Q}^i(t)\right]\\
&-\mathbb{E}\left[\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i}+o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)-A^i(t)\right]Q_u^i(t)\big|\boldsymbol{Q}^i(t)\right]\\
&-\mathbb{E}\left[\left[n^i(t)\frac{s_c}{\gamma^i}-o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\right]Q_c^i(t)\big|\boldsymbol{Q}^i(t)\right],\\
&\text{(Com-P): : }\Delta L(t) + V\mathbb{E}\left[h_c(t)\big|\boldsymbol{Q}_c(t)\right]\\
&\le B_2 + V\mathbb{E}\left[h_c(t)\big|\boldsymbol{Q}_c(t)\right]\\
&-\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left[n^i(t)\frac{s_c}{\gamma^i}-o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\right]Q_c^i(t)\big|\boldsymbol{Q}_c(t)\right],\\
&\text{(Com-UP):}\Delta L(t) + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\big|\boldsymbol{Q}(t)\right]\\
&\le B_3 + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\big|\boldsymbol{Q}(t)\right]\\
&-\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left[n^i(t)\frac{s_c}{\gamma^i}-o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\right]Q_c^i(t)\big|\boldsymbol{Q}(t)\right]\\
&-\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i}+o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)-A^i(t)\right]Q_u^i(t)\big|\boldsymbol{Q}(t)\right]\\
&-\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i}+n^i(t)\frac{s_c}{\gamma^i}-A^i(t)\right]\left(Q_u^i(t)+Q_c^i(t)\right)\big|\boldsymbol{Q}(t)\right], \quad (12)
\end{aligned}$$

where

$$\begin{cases}
B_1 = \frac{1}{2}\left[(A^i_{\max})^2 + \frac{(s_{\max})^2}{(\gamma^i)^2} + 2(\mu^i_{l,\max}+\mu^i_{w,\max})^2 + \frac{(s_c)^2}{(\gamma^i)^2}\right],\\
B_2 = \frac{1}{2}\sum_{i\in\mathcal{I}}\left[\frac{(s_c)^2}{(\gamma^i)^2} + (\mu^i_{l,\max}+\mu^i_{w,\max})^2\right],\\
B_3 = \sum_{i\in\mathcal{I}}\left[(A^i_{\max})^2 + \frac{(s_{\max})^2}{(\gamma^i)^2} + (\mu^i_{l,\max}+\mu^i_{w,\max})^2 + \frac{(s_c)^2}{(\gamma^i)^2} + \frac{s_{\max}s_c}{(\gamma^i)^2}\right].
\end{cases}$$

Proof: The proof is presented in Appendix A. ■

C. Control Algorithms for Cost Minimization in a Competition Scenario

For the competition scenario between users and CSP, we develop Com-UC and Com-PC algorithms by finding (Com-U): $(o^i(t), \theta^i_l(t), \theta^i_w(t), s^i(t))$ for user i and (Com-P): $(\boldsymbol{n}(t), p_c(t))$ for the CSP which minimize the upper bounds of (9) and (10) every time slot, respectively.

User-side control algorithm (Com-UC): Each user considers queue backlogs of user-side and CSP-side, network status, and MCO price at time slot t. Before a decision of offloading policy $o^i(t)$, the user considers two cases when user chooses *(i)* local processing and *(ii)* code offloading.

1) If local processing is selected ($o^i(t) = 0$), the local CPU clock selection problem can be written as follows.

$$\min_{s^i(t)\in\mathcal{S}}\left[V\alpha^i E_s\left(s^i(t)\right) - \frac{s^i(t)}{\gamma^i}Q_u^i(t)\right] = \Omega^i_{s,t}. \quad (13)$$

Because $E_s(\cdot)$ is a convex, differentiable function on the CPU clock speed, the best CPU clock speed $s^i_u(t)^{\#}$ from the continuous set of clock speed can be easily found by differentiation.

$$s^i_u(t)^{\#} = (\dot{E}_s)^{-1}\left(\frac{Q^i_u(t)}{V\gamma^i\alpha^i}\right). \quad (14)$$

Because the set of adjustable CPU clock speed $\mathcal{S}_u$ is discretized, we have to check two points of adjustable clock speed which are the nearest ones in the right and left from the $s^i_u(t)^{\#}$ and choose the clock speed $s^i_u(t)$ which minimizes (13). In (13), as $s^i(t)$ becomes faster, more computing tasks can be processed by paying much energy of a mobile device. In (14), we can see that the best clock speed increases when the many computing tasks remain at user-side ($Q^i_u(t)$) and the worth of battery energy becomes low (α^i).

2) Else if code offloading is selected ($o^i(t) = 1$), the network interface selection problem can be decomposed into cellular and Wi-Fi on/off problems as follows.

$$\min_{\theta^i_l(t)\in\{0,1\}}\theta^i_l(t)\left[VD^i_l(t) - \mu^i_l(t)\left(Q^i_u(t) - Q^i_c(t)\right)\right] = \Omega^i_{l,t}(p_c(t)),$$

$$\min_{\theta^i_w(t)\in\{0,1\}}\theta^i_w(t)\left[VD^i_w(t) - \mu^i_w(t)\left(Q^i_u(t) - Q^i_c(t)\right)\right] = \Omega^i_{w,t}(p_c(t)).$$

[13]We refer readers to [33], [38] for the practical V control. The authors of these works dynamically control V depending on the instantaneous delay.

We can directly derive the network interface selection algorithm $(\theta_l^i(t), \theta_w^i(t))$ as follows.

$$\theta_l^i(t) = \begin{cases} 1, & \text{if } VD_l^i(t) < \mu_l^i(t)\left(Q_u^i(t) - Q_c^i(t)\right), \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

$$\theta_w^i(t) = \begin{cases} 1, & \text{if } VD_w^i(t) < \mu_w^i(t)\left(Q_u^i(t) - Q_c^i(t)\right), \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

In (15), the network interfaces become more likely to be activated as the many computing tasks remain at user-side $(Q_u^i(t))$, the processing queue backlog becomes lower at CSP-side $(Q_s^i(t))$ and the computing price $(p_c(t))$ gets cheaper. Also, when the computing price and the cellular data price are cheap enough compared to the queue difference between user-side and CSP-side, high network speeds ($\mu_l^i(t)$ and $\mu_w^i(t)$) make the network interfaces be activated. We note that the cellular network is more conservative to be activated than the Wi-Fi network due to the additional data cost $p_l \mu_l^i(t)$.

Based on the above results, user decides offloading policy $o^i(t)$ as follows.

$$o^i(t) = \begin{cases} 1, & \text{if } \Omega_{s,t}^i \geq \Omega_{l,t}^i\left(p_c(t)\right) + \Omega_{w,t}^i\left(p_c(t)\right), \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

To decide offloading policy (17), the user considers which decision is beneficial between local processing and code offloading in terms of cost reduction and queue stabilization. We observe that the user prefers code offloading ($o^i(t) = 1$) when the user-side queue backlogs are much larger than CSP-side one $(Q_u^i(t) \gg Q_c^i(t))$, network channel states are good and computing price is low. This implies the user distributes the computing tasks opportunistically to the user-side and the CSP-side. We note that the complexity of user-side control algorithm is low because each control variable is decided between two choices by comparing their values.

CSP-side control algorithm (Com-PC): At the CSP-side, the minimization problem can be decomposed into sub-problems for (a) server provisioning and (b) MCO service pricing.

1) The server provisioning problem can be written as follows.

$$\min_{\boldsymbol{n}(t)} \left[\sum_{i\in\mathcal{I}} n^i(t)\left\{Ve(t) - \frac{s_c}{\gamma^i} Q_c^i(t)\right\}\right], \quad \text{subject to } \sum_{i\in\mathcal{I}} n^i(t) \leq n_{\max}(t). \tag{18}$$

To solve the problem (18), the CSP defines the user set $\mathcal{J}$ as follows.

$$\mathcal{J} = \left\{ i \in \mathcal{I} | Ve(t) < \frac{s_c}{\gamma^i} Q_c^i(t) \right\}. \tag{19}$$

For all users $i \in \mathcal{J}$, the CSP prefers to allocate edge servers to process user i's computing tasks to reduce the delay because many tasks remain in the CSP-side queue $Q_c^i(t)$. However, the total number of available servers is limited by $n_{\max}(t)$, the CSP allocates $\min(\frac{Q_c^i(t)\gamma^i}{s_c}, n_{\max}(t))$ servers to the user whose $\frac{Q_c^i(t)}{\gamma^i}$ is the highest in the set $\mathcal{J}$. This process is iterated in a greedy manner until no available server exists or $|\mathcal{J}|$ iterations are finished. We can observe that the CSP fairly serves CSP-side queues in terms of queueing delay among users. We note that the complexity of server provisioning algorithm at the CSP-side is $O(|\mathcal{I}|^2)$ because the CSP has to sort users based on $\frac{Q_c^i(t)}{\gamma^i}$.

2) The MCO service pricing problem can be written as follows.

$$\min_{p_c(t)\geq 0} \left[\sum_{i\in\mathcal{I}} o^i(t) R^i\left(\boldsymbol{\theta}^i(t)\right)\left[Q_c^i(t) - V p_c(t)\gamma^i\right]\right] = \min_{p_c(t)\geq 0}\left[\sum_{i\in\mathcal{I}} \Upsilon^i(p_c(t))\right], \tag{20}$$

where $o^i(t)$, $\theta_l^i(t)$ and $\theta_w^i(t)$ depend on $p_c(t)$. In (20), there exists a tradeoff for the CSP that tries to make much money from users and to reduce the increment of the CSP-side queue, simultaneously. Unfortunately, it is not able to separate the problem (20) into each user and has to check all the $p_c(t) \geq 0$ to find an optimal MCO service price which requires high complexity. However, we introduce a way to find the price easily without loss of optimality using Lemma 2 as follows.

Lemma 2: We can find an optimal MCO service price $p_c(t)$ by checking a set of transition prices $\mathcal{P}_{\text{trans}}$ where $|\mathcal{P}_{\text{trans}}| \leq 2|\mathcal{I}|$.

Proof: To prove Lemma 2, we show that there are at most two transition prices for each mobile user which make the user change his/her policy (e.g., offloading policy related to $o^i(t)$ or the number of activated network interfaces related to $R^i(\boldsymbol{\theta}^i(t))$ in (20)). Then, we can show that the function $\Upsilon^i(p_c(t))$ for user i in (20) is a truncated, partially decreasing and linear function on $p_c(t)$, where the truncated points are the transition prices. In the same manner, for $\mathcal{I}$ set of mobile users, the properties of $\sum_{i\in\mathcal{I}} \Upsilon^i(p_c(t))$ are the same as those of $\Upsilon^i(p_c(t))$, the CSP can find a maximizer of $\sum_{i\in\mathcal{I}} \Upsilon^i(p_c(t))$ by checking at most $2|\mathcal{I}|$ transition prices. The detailed proof is presented in Appendix B. ■

Then, our Com-UC and Com-PC algorithms for a competition scenario can be described as follows.

D. Control Algorithms for Cost Minimization in a Cooperation Scenario

For the cooperation scenario between users and CSP, we develop a Coo-JC algorithm by finding $(\boldsymbol{o}(t), \boldsymbol{\theta}(t), \boldsymbol{s}(t), \boldsymbol{n}(t))$ for the users and the CSP which minimizes the upper bounds of (11) every time slot. Fortunately, we can decompose the Coo-JC algorithm into user-side and CSP-side, independently, but each control variable is manipulated based on the social cost and delay compared to the competition scenario.

Algorithm 1: Algorithm description for a competition scenario.

At each time slot t,

CSP-side control (Com-PC):

1: Find the user set $\mathcal{J}$ by (19).
2: Initialize $n^i(t)=0$ for all $i\in\mathcal{I}$ and $n_{\text{use}}=0$.
3: **while** $\mathcal{J}\neq\phi$ and $n_{\text{use}}\neq n_{\max}(t)$,
4: $j=\arg\min_{i\in\mathcal{J}}\frac{Q_c^i(t)}{\gamma^i}$ and $n^j(t)=\min(\frac{Q_c^j(t)\gamma^j}{s_c}, n_{\max}(t)-n_{\text{use}})$.
5: $n_{\text{use}}=n_{\text{use}}+n^j(t)$ and $\mathcal{J}=\mathcal{J}-\{j\}$.
6: **end while**
7: Find the set of transition prices $\mathcal{P}_{\text{trans}}$ by Lemma 2.
8: $p_c(t)=\underset{p_c(t)\in\mathcal{P}_{\text{trans}}\cup\{\infty\}}{\arg\min}\left[\sum_{i\in\mathcal{I}}\Upsilon^i(p_c(t))\right]$.

User-side control (Com-UC): for each user i,

1: **if** $\Omega^i_{s,t}<\Omega^i_{l,t}(p_c(t))+\Omega^i_{w,t}(p_c(t))$,
2: Select $o^i(t)=0$, $\theta^i_l(t)=0$ and $\theta^i_w(t)=0$. Select $s^i(t)$ based on (14).
3: **else**
4: Select $o^i(t)=1$, $s^i(t)=0$. Select $\theta^i_l(t)$ and $\theta^i_w(t)$ based on (15) and (16), respectively.
5: **end if**

Update the queues $Q^i_u(t), Q^i_c(t)$ for all $i\in\mathcal{I}$ according to the queueing dynamics (1) and (2).

For the CSP-side, the server provisioning problem can be written as follows.

$$\min_{\boldsymbol{n}(t)}\left[\sum_{i\in\mathcal{I}}n^i(t)\left\{Ve(t)-\frac{s_c}{\gamma^i}\left(Q^i_u(t)+2Q^i_c(t)\right)\right\}\right],$$
$$\text{subject to}\ \sum_{i\in\mathcal{I}}n^i(t)\le n_{\max}(t).$$

Compared to the server provisioning problem for a competition scenario (18), $Q^i_c(t)$ term is replaced by $Q^i_u(t)+2Q^i_c(t)$. The reason is that the CSP takes care of not only CSP-side delay, but also system-level delay. Then, the set $\tilde{\mathcal{J}}$ is defined as

$$\tilde{\mathcal{J}}=\left\{i\in\mathcal{I}|Ve(t)<\frac{s_c}{\gamma^i}\left(Q^i_u(t)+2Q^i_c(t)\right)\right\},\tag{21}$$

and the priority of allocating server is changed by $\frac{Q^i_u(t)+2Q^i_c(t)}{\gamma^i}$ in the set $\tilde{\mathcal{J}}$. This means that the CSP will allocate the server to the user with high load at the user-side queue even if there are few computing tasks in the CSP-side queue.

For user i, the control algorithm behaves similar to the competition scenario, but the $\Omega^i_{s,t}$, $\Omega^i_{l,t}(p_c(t))$ and $\Omega^i_{w,t}(p_c(t))$ in the competition scenario are replaced by $\tilde{\Omega}^i_{s,t}, \tilde{\Omega}^i_{l,t}(p_c(t))$ and $\tilde{\Omega}^i_{w,t}(p_c(t))$ defined as follows.

$$\min_{s^i(t)\in\mathcal{S}}\left[V\alpha^iE_s(s^i(t))-\frac{s^i(t)}{\gamma^i}\left(2Q^i_u(t)+Q^i_c(t)\right)\right]=\tilde{\Omega}^i_{s,t},$$
$$\min_{\theta^i_l(t)\in\{0,1\}}\theta^i_l(t)\left[V\tilde{D}^i_l(t)-\mu^i_l(t)\left(Q^i_u(t)-Q^i_c(t)\right)\right]=\tilde{\Omega}^i_{l,t}(p_c(t)),$$
$$\min_{\theta^i_w(t)\in\{0,1\}}\theta^i_w(t)\left[V\tilde{D}^i_w(t)-\mu^i_w(t)\left(Q^i_u(t)-Q^i_c(t)\right)\right]=\tilde{\Omega}^i_{w,t}(p_c(t)),$$

where $\tilde{D}^i_l(t)\doteq\alpha^iE_l+p_l\mu^i_l(t)$ and $\tilde{D}^i_w(t)\doteq\alpha^iE_w$.

Then the offloading policy of user i is derived as follows.

$$o^i(t)=\begin{cases}1, & \text{if } \tilde{\Omega}^i_{s,t}\ge\tilde{\Omega}^i_{l,t}(p_c(t))+\tilde{\Omega}^i_{w,t}(p_c(t)),\\ 0, & \text{otherwise.}\end{cases}$$

If the local processing is selected ($o^i(t)=0$), the CPU clock speed $s^i_u(t)$ is decided by the same mechanism with the competition scenario where $s^i_u(t)^{\#}$ is replaced by $s^i_u(t)^{\#}$ as follows.

$$s^i_u(t)^{\#}=(E'_s)^{-1}\left(\frac{2Q^i_u(t)+Q^i_c(t)}{V\gamma^i\alpha^i}\right).\tag{22}$$

The local CPU clock speed also depends on the congestion at the CSP-side queue while it was independent for the competition scenario. The reason is that the user takes care of not only user-side delay, but also system-level delay for the cooperation scenario.

On the other hand, if the code offloading is selected ($o^i(t)=1$), the network interface activations $(\theta^i_l(t),\theta^i_w(t))$ are decided as follows.

$$\theta^i_l(t)=\begin{cases}1, & \text{if } V\tilde{D}^i_l(t)<\mu^i_l(t)\left(Q^i_u(t)-Q^i_c(t)\right),\\ 0, & \text{otherwise,}\end{cases}\tag{23}$$

$$\theta^i_w(t)=\begin{cases}1, & \text{if } V\tilde{D}^i_w(t)<\mu^i_w(t)\left(Q^i_u(t)-Q^i_c(t)\right),\\ 0, & \text{otherwise.}\end{cases}\tag{24}$$

These are similar with (15), but the MCO service price terms $p_c(t)\gamma^i\mu^i_l(t)$ and $p_c(t)\gamma^i\mu^i_w(t)$ disappear, respectively. It means that the user decides offloading policy and network interface selection by considering only energy cost, data communication cost and queueing stability without a consideration of MCO service price. The reason why these phenomena occur is that the user behaves to minimize the social cost. We note that the complexity of Coo-JC algorithm is the same as Com-UC and Com-PC algorithms. Then, Coo-JC algorithm for a cooperation scenario can be described as follows.

E. Performance Analysis of Coo-JC

We show that the time-averaged social cost and the time-averaged sum queue backlogs of user-side and CSP-side can be upper-bounded by the following Theorem 2 when the Coo-JC algorithm is adopted. Before showing that, we present the existence of optimal policy that is independent of queue backlogs and optimizes the social cost with satisfying queueing stability in Theorem 1.

Theorem 1: For any arrival rate vector of computing tasks $\boldsymbol{\lambda}$ within the capacity region[14] $\boldsymbol{\Lambda}$ where $\lambda^i=\mathbb{E}\left[A^i(t)\right],\forall i\in\mathcal{I}$, there exists a stationary randomized control policy $(\boldsymbol{o}(t)^*,\boldsymbol{\theta}(t)^*,\boldsymbol{s}(t)^*,\boldsymbol{n}(t)^*)$ which is independent in current queue backlogs every time slot t with satisfying the followings: for all $i\in\mathcal{I}$,

$$\mathbb{E}\left[\left(1-o^i(t)^*\right)\frac{s^i(t)^*}{\gamma^i}+o^i(t)^*R^i\left(\boldsymbol{\theta}^i(t)^*\right)\right]\ge\lambda^i,\tag{25}$$

[14]Note that the capacity region means a set of all task arrivals of which the system is able to process within finite time.

Algorithm 2: Algorithm description for a cooperation scenario (Coo-JC).

At each time slot t,

CSP-side control:

1: Find the user set $\tilde{\mathcal{J}}$ by (21).
2: Initialize $n^i(t)=0$ for all $i\in\mathcal{I}$ and $n_{\text{use}}=0$.
3: **while** $\tilde{\mathcal{J}}\neq\phi$ and $n_{\text{use}}\neq n_{\max}(t)$,
4: $j=\arg\min_{i\in\tilde{\mathcal{J}}}\frac{Q_u^i(t)+2Q_c^i(t)}{\gamma^i}$ and
$n^j(t)=\min\left(\frac{Q_c^j(t)\gamma^j}{s_c},n_{\max}(t)-n_{\text{use}}\right)$.
5: $n_{\text{use}}=n_{\text{use}}+n^j(t)$ and $\tilde{\mathcal{J}}=\tilde{\mathcal{J}}-\{j\}$.
6: **end while**

User-side control: for each user i,

1: **if** $\tilde{\Omega}_{s,t}^i<\tilde{\Omega}_{l,t}^i(p_c(t))+\tilde{\Omega}_{w,t}^i(p_c(t))$,
2: Select $o^i(t)=0$, $\theta_l^i(t)=0$ and $\theta_w^i(t)=0$.
Select $s^i(t)$ based on (22).
3: **else**
4: Select $o^i(t)=1$, $s^i(t)=0$.
Select $\theta_l^i(t)$ and $\theta_w^i(t)$ based on (23) and (24), respectively.
5: **end if**

Update the queues $Q_u^i(t),Q_c^i(t)$ for all $i\in\mathcal{I}$ according to the queueing dynamics (1) and (2).

$$\mathbb{E}\left[n^i(t)^*\frac{s_c}{\gamma^i}\right]=\mathbb{E}\left[o^i(t)^*R^i\left(\boldsymbol{\theta}^i(t)^*\right)\right], \tag{26}$$

$$\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left[\left(1-o^i(t)^*\right)\alpha^i E_s\left(s^i(t)^*\right)+o^i(t)^*\left\{\theta_l^i(t)^*\tilde{D}_l^i(t)+\theta_w^i(t)^*\tilde{D}_w^i(t)\right\}+n^i(t)^*e(t)\right]\right]=\Psi(\boldsymbol{\lambda}), \tag{27}$$

where $\Psi(\boldsymbol{\lambda})$ is the minimum time average expected social cost that can be achieved by any control policy that stabilize the system when the arrival rate vector is $\boldsymbol{\lambda}\in\boldsymbol{\Lambda}$. It can be similarly proven with [12] using Caratheodory's theorem [40].

Theorem 2: Let $t\in\{0,1,...,T-1\}$ and the arrival rate vector of computing tasks is interior to the capacity region $\boldsymbol{\lambda}\in\Lambda$. Then, under Coo-JC algorithm, we have:

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{I}}h_u^i(t)+h_c(t)\right]\leq\Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon})+\frac{B_3}{V},$$

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left\{Q_u^i(t)+Q_c^i(t)\right\}\right]\leq\frac{B_3+V\Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon})}{\epsilon},$$

where $\epsilon>0$ such that $\boldsymbol{\lambda}+\boldsymbol{\epsilon}=(\lambda^1+\epsilon,\lambda^2+\epsilon,...,\lambda^I+\epsilon)\in\Lambda$. Theorem 2 means that for given cost-delay tradeoff parameter V, Coo-JC algorithm achieves $O(1/V)$ gap of the optimal average social cost with maintaining $O(V)$ of average-sum queue backlogs. Note that if we set V and ϵ closer to infinity and zero, respectively, Coo-JC algorithm achieves optimal social cost.

Proof: We derive the bound of short-term objective function in Lemma 1 from the stationary randomized control policy in Theorem 1. Then, using queue-independent property of the randomized policy and law of iterated expectations, we can prove Theorem 2. The detailed proof is presented in Appendix C. ■

V. Trace-Driven Simulation

A. Measurement and Traces

Energy measurement: We measure the energy consumptions of Galaxy Note [41] smartphone with LTE chipset using Monsoon power monitor [42]. We measure the CPU energy consumptions for several clock speeds (0.1∼1.4 GHz). Then we fit the parameters (κ,φ,ρ) to the typical CPU energy consumption model in [12] where Δt denotes time duration for one time slot in sec[15] as follows.

$$E_s(s^i(t))=(\kappa(s^i(t))^{\varphi}+\rho)\Delta t.$$

The measured energy consumptions of LTE and Wi-Fi networks are 2605 mJ/sec and 1225 mJ/sec, respectively, and the CPU energy parameters are $(\kappa,\varphi,\rho)=(0.33,3.00,0.10)$.

Real traces: We collect the eight traces which contain availabilities and uplink throughputs of LTE and Wi-Fi network. These are measured for three network operators and diverse mobility scenarios, including fixed location, walking in downtown and driving on highway in South Korea. The uplink throughput traces are measured by uploading 5 MB dummy file to our private server and checking the transmission time every 1 minute. The measured average uplink throughputs of LTE and Wi-Fi are 5.9 Mbps and 6.4 Mbps, respectively, and the temporal coverage of Wi-Fi APs is 63%. To run the simulation, we use one of the uplink throughput and availability traces of LTE and Wi-Fi for each user. To generate computing task arrivals of users, we use a data set of YouTube video size distribution [43] and re-scale it, where the average video size is 12.6 MB. We use the real-world traces of electricity bills in California, USA [44] where the time granularity is 5 minutes. For the LTE data price, we use the flat pricing plan from one of network operators in South Korea ($p_l=1.16\times10^{-9}$ \$/bit) where the price of 1 GB LTE data usage is \$10 [45].

B. Simulation Setup

We consider a scenario when mobile users who have Galaxy Note go around within LTE coverage and an intermittent Wi-Fi coverage. They execute applications which generate code offloadable computing tasks where the tasks arrive as a Bernoulli process. The processing densities of mobile users are set to be from 200 to 6000 cycles/bit[16]. We mainly execute simulations for the 3000 cycles/bit of processing density that is generally acceptable for the chess game, video transcoding and face

[15] All measurements are done with disabling other components such as GPS and display (and network in CPU measurement).

[16] That range includes processing densities of current mobile applications [13].

TABLE III
SIMULATION SETTINGS

Number of users	80
Experimental time [hour]	6
Avg. arrival rate [Mbps]	0.96
Processing density [cycles/bit]	$200 \sim 6000$ [13]
Energy consumption (LTE, Wi-Fi) [mJ/sec]	(2605, 1225) [13]
CPU energy consumption parameters (κ, φ, ρ)	(0.33, 3.00, 0.10) [12]
Avg. uplink throughput (LTE, Wi-Fi) [Mbps]	(5.9, 6.4) [1]
Price of LTE data [$/bit]	1.16×10^{-9} [45]
Energy-money weight parameter [$/J]	2.44×10^{-4}
Avg. electricity price per unit server [$/sec]	5.5×10^{-5} [44]
Available # of servers at CSP	$uniform[1, 80]$
Processing capacity of mobile device [GHz]	1.4 [41]
Processing capacity of one edge server [GHz]	3.0 [15]

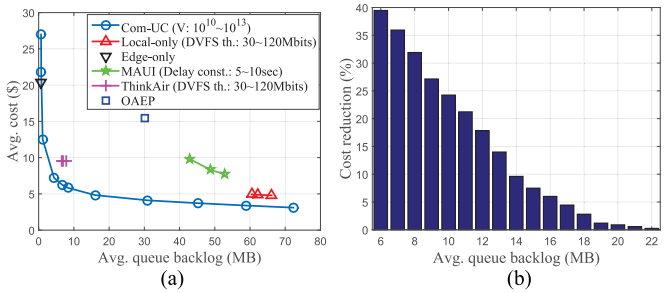


Fig. 2. Simulation results of user-side control algorithms. (a) Avg. user cost and sum of avg. queue backlogs of users and CSP in cases of Com-UC and existing user-side control algorithms. (b) Percentage of cost reduction for user when multi-homing technology is available compared to the case when it is not available.

recognition applications [13]. For ease of analysis, we set the energy-money weight parameter of all users to be the same as $\alpha^i = 2.44 \times 10^{-4}$ \$/J. It is calculated by 5 times of fast charging price of a smartphone at a convenient store in South Korea divided by battery capacity of Galaxy Note. For the CSP-side, we assume that the CSP has edge servers where the available number is uniformly distributed in the range $[1, |\mathcal{I}|]$ for every time slot. The electricity bill is also time-varying [44] where the average is $\$5.5 \times 10^{-5}$ to activate one server for 1 second [44]. Detailed simulation parameters are summarized in Table III.

As performance metrics, we observe average costs and average queue backlogs for the users and CSP. For a competition scenario, we compare our Com-UC with existing code offloading algorithms at user-side [18]–[20]. Also, we compare our Com-PC with a baseline algorithm which is currently deployed by commercial CSPs at CSP-side. For all comparing algorithms at user-side, a CPU speed is selected by a DVFS scheme [32]. For *MAUI* [18], code offloading is selected only when the current tasks can be transferred within a specified delay. For *ThinkAir* [19], code offloading is selected only when both delay and energy of networking are less than those of local processing. For *OAEP*[17] [20], code offloading is selected when the energy efficiency (speed/energy) of networking is greater than that of local processing. *Local-only* and *Edge-only* policies are always using the local processing and code offloading using all the available networks, respectively. For the CSP-side, the baseline algorithm adopts a fixed pricing policy and does not defer the task processing on purpose (i.e., activate edge servers to process the computing tasks as much as possible). We also compare our Coo-JC algorithm with (Com-UC+Com-PC) to verify the advantage of cooperation between the users and CSP.

C. Simulation Results 1—Competition Scenario

Cost-delay tradeoff: Fig. 2(a) depicts the average cost of users and the sum of average stabilized queue backlogs of user-side and CSP-side for Com-UC and existing algorithms when the CSP adopts Com-PC. First, we observe *the 71% cost savings for Com-UC by trading only 6 MB queue backlogs (50 seconds delays)* which means there are efficient rooms for drastic cost saving while paying small amount of delay (see a circle-dotted line). The cost saving comes from that *(i)* it saves the CPU energy by adjusting local CPU clock speed depending on the current queue backlogs and *(ii)* it opportunistically transfers computing tasks to the CSP by taking into account time-varying MCO environments. We should be noted that the efficient cost-delay tradeoff region is from 4 MB to 10 MB of average queue backlog (see the slope of Fig. 2(a)). Moreover, the mobile user can adjust cost-delay tradeoff parameter V according to the delay-tolerance of the target application.

In the extreme cases of user-side control algorithm, *Local-only and Edge-only (with different DVFS thresholds) show low cost with high delay and high cost with low delay, respectively*. These results represent that a smart user-side control algorithm is needed to regulate the cost-delay tradeoff according to the user's demand. The weaknesses of other comparing algorithms are as follows. MAUI does not take care of cost of a smartphone (energy consumption, monetary cost) and always choose local processing when the delay becomes larger even if network speed is faster than local CPU speed. ThinkAir is not able to dynamically put the weight between cost and delay, according to the current backlog of computing tasks. The reason is that it selects code offloading only when both of the energy and the delay of networking are better than local processing which is not flexible. OAEP selects code offloading as long as it is more energy-efficient than local processing even though the MCO price is high. We observe that *the Com-UC outperforms existing algorithms where the cost reductions are 60% (at the 43 MB queue backlogs) for MAUI, 35% (at the 6.7 MB queue backlogs) for ThinkAir, and 73% (at the 30 MB queue backlogs) for OAEP*. The reason why Com-UC outperforms existing algorithms is that it optimizes offloading policy, CPU clock selection and network selection jointly by exploiting the opportunisms of time-varying network states, prices of MCO and LTE data, and remaining queue backlogs.

Impact of multi-homing technology: We quantify a ripple effect of multi-homing technology for mobile users in MCO environment. To this end, we simulate a Com-UC without multi-homing availability, which does not have an option to select both of cellular and Wi-Fi interfaces for the transmission, and

[17]We denote by OAEP the optimal application execution policy in [20].

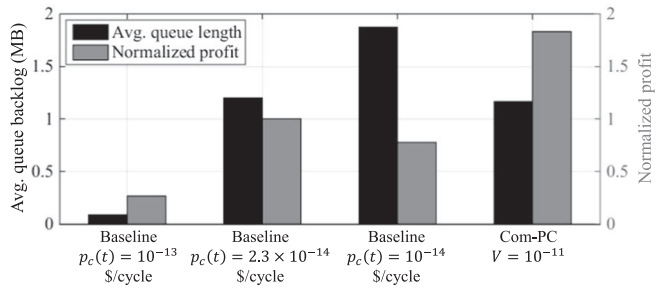


Fig. 3. Avg. queue backlogs of CSP-side and avg. profit of CSP for Com-PC and flat pricing algorithms.

compare with the original Com-UC algorithm. Fig. 2(b) demonstrates the percentage of cost reduction for mobile user, when multi-homing is available compared to the case when it is unavailable. The cost reduction rapidly decreases as the average queue backlog becomes larger, i.e., the percentage of cost reduction is under 5% from 17 MB of average queue backlogs. The reason is that the user does not need to activate cellular and Wi-Fi interfaces simultaneously to boost the transfer speed with high costs (energy and money) when the user can tolerate long processing delay. However, when the user requires a short processing delay, the multi-homing technology drastically reduces the user cost, i.e., 39% at 6 MB of average queue backlogs. It implies the multi-homing technology has an significant influence on the performance of delay-sensitive applications.

Impact of dynamic pricing and server provisioning: To evaluate our Com-PC algorithm, we compare Com-PC with the baseline algorithm when mobile users adopt Com-UC. The baseline algorithm adopts flat pricing for the MCO service and activates edge servers to process the computing tasks as much as possible regardless of the electricity bill. Fig. 3 depicts the average queue backlogs and the profits (revenue from subscribers minus electricity bill) of the CSP. For the baseline algorithm, as the fixed MCO price becomes cheaper, the CSP-side queue backlog gets larger due to the increasing requests of computing tasks from the users. However, the profit of the CSP forms a parabola curve to the MCO price and the saddle point (the highest profit achievable price) is 2.3×10^{-14} \$/cycle that we found in our simulation. *The Com-PC achieves 82% of the profit gain compared to the baseline algorithm with a fixed MCO price 2.3×10^{-14} \$/cycle, while maintaining similar average queue backlogs (1.2 MB).* The reason is that Com-PC fully exploits the different willingness to pay of users by time-dependent pricing. Also, it utilizes opportunities of time-varying electricity bills which can save the operating cost of edge servers with satisfying the queue stability.

Impact of user-dependent parameters: To observe microscopic aspects, we analyze the processed computing tasks for each user with different user parameters. Fig. 4 shows the ratio of computing tasks processed by local CPU, offloaded via cellular networks and offloaded via Wi-Fi networks for each mobile user, namely offloading ratio. In Fig. 4(a), we simulate in the case that the temporal coverage and average data rate of Wi-Fi networks are the same (63% and 6.4 Mbps, respectively) for all mobile users and plot the offloading ratio where the user indexes are sorted by average cellular data rate in a descending order (max: 8.5 Mbps, min: 4.9 Mbps). As the average cellular data rate goes higher, the ratio of computing tasks offloaded to edge servers (green+blue) increases because the energy efficiency of the cellular interface becomes higher. Meanwhile, the Wi-Fi offloading ratio (green) is similar for all mobile users, which means that Wi-Fi networks are oftentimes more preferable than cellular networks even through the average cellular data rate is high enough due to the fact that the Wi-Fi offloading requires lower energy and monetary cost than cellular offloading in our simulation settings.

In Fig. 4(b), we simulate in the case that average data rates of cellular networks are the same (5.9 Mbps) for all mobile users and plot the offloading ratio where the user indexes are sorted by average Wi-Fi data rate in a descending order (max: 10 Mbps, min: 3 Mbps). We observe that as a mobile user has higher average Wi-Fi data rate, the sum of cellular and Wi-Fi offloading ratios goes higher due to the similar reason in Fig. 4(a). However, the cellular offloading ratio (blue) goes lower because a good Wi-Fi connection is always better than cellular networks in terms of data rate, energy efficiency and monetary cost. Nevertheless, because the Wi-Fi networks have intermittent connections, the ratio of cellular offloading remains 30% for the mobile users under the best Wi-Fi conditions.

In Fig. 4(c), we simulate in case that average data rates of cellular and Wi-Fi networks are the same (5.9 Mbps, 6.4 Mbps, respectively) for all mobile users and plot the offloading ratio where the user indexes are sorted by processing density of computing tasks in a descending order (min: 200 cycles/bit, max: 6000 cycles/bit). For the mobile user which has tasks with low processing density, almost of computing tasks are processed locally (yellow) because the local CPU with a low clock speed is the most cost-efficient method in a processing-light task scenario. On the other hand, when the processing density of the computing tasks is high, a burden of local processing increases due to high CPU clock speed while per-cycle (energy and data) cost to transfer the computing tasks to the edge servers becomes lower. Then, the ratio of computing tasks offloaded to edge servers becomes higher. We observe that 90% of computing tasks are offloaded from the users with the highest processing density (6000 cycles/bit[18]).

D. Simulation Results 2—Cooperation Gains

Cost-delay tradeoff: We compare two proposed dual-side control algorithms when the users and the CSP cooperate (Coo-JC) or not (Com-UC+ Com-PC). Fig. 5(a) and (b) demonstrate the social cost reduction and the queue backlog reduction of the Coo-JC compared to the (Com-UC+Com-PC) for the same average queue backlogs and social costs, respectively. *The cooperation achieves 28% of cost reduction when the average queue backlog is 10 MB and 30% of queue backlog reduction when the social cost is \$20, compared to the competition scenario.* These gains of cooperation mainly come from the fact that mobile users transfer the computing tasks to the CSP without a burden of the MCO service price. In addition, the CSP actively processes the

[18]Some virus scanning and image recognition applications require above 10000 cycles/bit of processing densities [13].

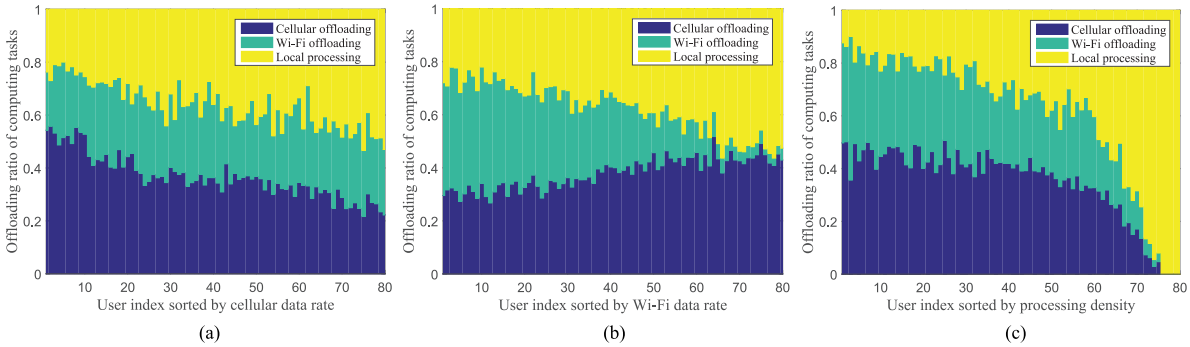


Fig. 4. Offloading ratio of mobile users for different user parameters for the competition scenario (Com-UC+Com-PC). (a) Offloading ratio of mobile users for different avg. data rate of cellular network. (b) Offloading ratio of mobile users for different avg. data rate of Wi-Fi network. (c) Offloading ratio of mobile users for different processing density of computing tasks.

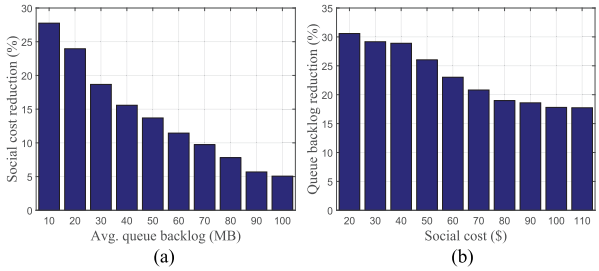


Fig. 5. Avg. social cost and queue backlog reductions from the competition scenario (Com-UC+Com-PC) to the cooperation scenario (Coo-JC). (a) Percentage of social cost reduction for the same average queue backlogs. (b) Percentage of queue backlog reduction for the same social costs.

CSP-side queues by taking account of the remaining tasks at the both user-side and CSP-side. As the queue backlog (the social cost) becomes larger, the social cost reduction (the queue backlog reduction) gets smaller because both control algorithms operate similarly when the cost-delay tradeoff parameter V is extremely large or small. When V becomes extremely large, the mobile users try to minimally process the computing tasks by only local processing with lowest CPU clock speed. On the other hand, when V goes extremely small, the mobile users try to maximally process the computing tasks by only code offloading by exploiting all the available network interfaces.

VI. Concluding Remark

In this paper, we explored the cost-delay tradeoffs for mobile users and code offloading service provider (CSP) in a mobile edge computing system. We proposed algorithms for energy/monetary cost minimization while ensuring finite processing delay. In a competition scenario, we proposed a user-side control algorithm, namely Com-UC which controls offloading policy, local CPU clock frequency and network interface activation. We also proposed a CSP-side control algorithm, namely Com-PC which controls the server provisioning and mobile code offloading (MCO) service price. Moreover, we proposed a joint users-CSP control algorithm, namely Coo-JC in a cooperation scenario. Trace-driven simulations based on the real measurements demonstrated that our control algorithms attain drastic cost saving for users and CSP compared to existing algorithms. This paper will provide CSPs with guidelines how to manage the MCO service, and mobile users with guidelines how to efficiently exploit local and edge computing resources in the competition scenario and the cooperation scenario, respectively. Our work can be extended to multiple CSPs environment where each CSP is located at different place (which leads to different electricity price), and has different computing capacity. Then, we can imagine a scenario that several CSPs compete with each other, and mobile users have additional decision making, e.g., choosing a CSP which provides high quality of offloading service and low price. We leave it as a future work.

Appendix

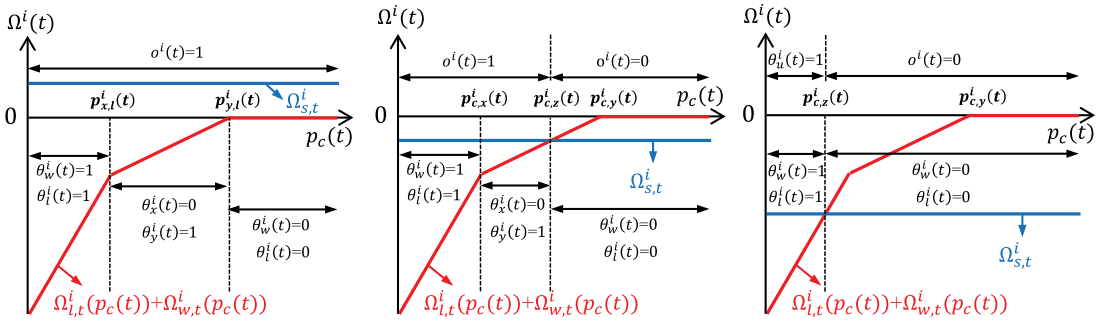
A. Proof of Lemma 1

Proof: Let us consider user-side queueing dynamics (Q_u^i) in (1). By taking square on (1) and using the fact that $([X]^+)^2 \leq X^2$, we have:

$$\begin{aligned}
&\left(Q_u^i(t+1)\right)^2 \\
&\leq \left[Q_u^i(t) - \left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i} - o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right) + A^i(t)\right]^2 \\
&\leq \left(Q_u^i(t)\right)^2 - 2\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i} + o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right) - A^i(t)\right]Q_u^i(t) \\
&\quad + (A_{\max}^i)^2 + \frac{(s_{\max})^2}{(\gamma^i)^2} + (\mu_{l,\max}^i + \mu_{w,\max}^i)^2 \qquad (28)
\end{aligned}$$

where the second inequality in (28) is from the bounds of task arrival rate, CPU clock speed and network speeds. Then, by rearranging (28), we have:

$$\begin{aligned}
&\left(Q_u^i(t+1)\right)^2 - \left(Q_u^i(t)\right)^2 \\
&\leq -2\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i} + o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right) - A^i(t)\right]Q_u^i(t) \\
&\quad + \left(A_{\max}^i\right)^2 + \frac{(s_{\max})^2}{(\gamma^i)^2} + \left(\mu_{l,\max}^i + \mu_{w,\max}^i\right)^2. \qquad (29)
\end{aligned}$$



(a) $\Omega^i_{s,t} > 0$ (b) $\Omega^i_{y,t}(p^i_{c,x}(t)) \le \Omega^i_{s,t} \le 0$ (c) $\Omega^i_{s,t} < \Omega^i_{y,t}(p^i_{c,x}(t))$

Fig. 6. Three cases of $\Omega^i_{l,t}(p_c(t)) + \Omega^i_{w,t}(p_c(t))$ and $\Omega^i_{s,t}$. (a) $\Omega^i_{s,t} > 0$. (b) $\Omega^i_{y,t}(p^i_{c,x}(t)) \le \Omega^i_{s,t} \le 0$. (c) $\Omega^i_{s,t} < \Omega^i_{y,t}(p^i_{c,x}(t))$.

Similarly, we obtain the followings by repeating for CSP-side (Q^i_c) and dual-side ($Q^i_u + Q^i_c$).

$$\begin{aligned}&\left(Q^i_c(t+1)\right)^2 - \left(Q^i_c(t)\right)^2\\ &\le -2\left[n^i(t)\frac{s_c}{\gamma^i} - o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\right]Q^i_c(t) + \frac{(s_c)^2}{(\gamma^i)^2}\\ &\quad + (\mu^i_{l,\max} + \mu^i_{w,\max})^2. \end{aligned} \tag{30}$$

$$\begin{aligned}&\left(Q^i_u(t+1) + Q^i_c(t+1)\right)^2 - \left(Q^i_u(t) + Q^i_c(t)\right)^2\\ &\le -2\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i} + n^i(t)\frac{s_c}{\gamma^i} - A^i(t)\right]\left(Q^i_u(t) + Q^i_c(t)\right)\\ &\quad + (A^i_{\max})^2 + \frac{(s_{\max})^2}{(\gamma^i)^2} + \frac{(s_c)^2}{(\gamma^i)^2} + \frac{2s_{\max}s_c}{(\gamma^i)^2}. \end{aligned} \tag{31}$$

By $\frac{1}{2}\mathbb{E}\left[(29)+(30)|\boldsymbol{Q}^i(t)\right]$ for (Com-U), $\frac{1}{2}\mathbb{E}\left[\sum_{i\in\mathcal{I}}(30)|\boldsymbol{Q}_c(t)\right]$ for (Com-P) and $\frac{1}{2}\mathbb{E}\left[\sum_{i\in\mathcal{I}}((29)+(30)+(31))\,|\boldsymbol{Q}(t)\right]$ for (Coo-UP), we obtain the upper bounds of following Lyapunov drift in Lemma 1. ■

B. Proof of Lemma 2

Proof: Based on $\Omega^i_{s,t}$, $\Omega^i_{l,t}(p_c(t))$ and $\Omega^i_{w,t}(p_c(t))$ in the Com-UC algorithm of user i, we firstly define $p^i_{c,l}(t)$ and $p^i_{c,w}(t)$ as follows.

$$p^i_{c,l}(t) = \frac{\mu^i_l(t)\left(Q^i_u(t) - Q^i_c(t)\right) - V\left(\alpha^i E_l + p_l\mu^i_l(t)\right)}{V\gamma^i\mu^i_l(t)} \tag{32}$$

$$p^i_{c,w}(t) = \frac{\mu^i_w(t)\left(Q^i_u(t) - Q^i_c(t)\right) - V\alpha^i E_w}{V\gamma^i\mu^i_w(t)}, \tag{33}$$

where $p^i_{c,l}(t)$ and $p^i_{c,w}(t)$ are the maximum MCO prices making user i selects $\theta^i_l(t) = 1$ and $\theta^i_w(t) = 1$, respectively, when the offloading policy is given by $o^i(t) = 1$. Then, $\Omega^i_{l,t}(p_c(t)) + \Omega^i_{w,t}(p_c(t))$ can be drawn as a red line in Fig. 6 where $x = \arg\min_{k\in\{l,w\}} p^i_{c,k}(t)$ and $y = \arg\max_{k\in\{l,w\}} p^i_{c,k}(t)$. Because the offloading policy $o^i(t)$ of user i is determined by comparison between $\Omega^i_{l,t}(p_c(t)) + \Omega^i_{w,t}(p_c(t))$ and $\Omega^i_{s,t}$ in (17), three cases are possible described in Fig. 6.

(i) If $\Omega^i_{s,t} > 0$ (for the case Fig. 6(a)), the user decisions are separated by two prices $p^i_{c,x}(t)$ and $p^i_{c,y}(t)$ at most. *(ii)* If $\Omega^i_{y,t}(p^i_{c,x}(t)) \le \Omega^i_{s,t} \le 0$ (for the case Fig. 6(b)), the user decisions are separated by two prices $p^i_{c,x}(t)$ and $p^i_{c,z}(t)$ at most where

$$p^i_{c,z}(t) = \frac{\Omega^i_{s,t} + \mu^i_y(t)\left(Q^i_u(t) - Q^i_c(t)\right) - V\left(\alpha^i E_y + p_y\mu^i_y(t)\right)}{V\gamma^i\mu^i_y(t)}$$

$$\text{and} \quad p_w = 0.$$

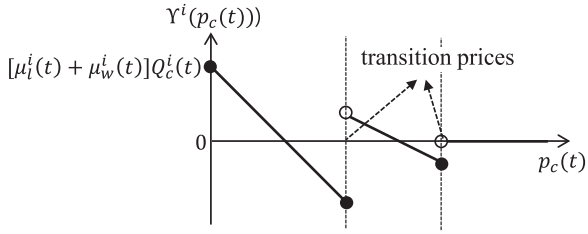


Fig. 7. Example of user behavior and $\Upsilon^i(p_c(t))$.

(iii) If $\Omega^i_{s,t} < \Omega^i_{y,t}(p^i_{c,x}(t))$ (for the case Fig. 6(c)), the user decisions are separated by one price $p^i_{c,z}(t)$ at most where

$$p^i_{c,z}(t) = \frac{\Omega^i_{s,t} + (\mu^i_l(t) + \mu^i_w(t))\left(Q^i_u(t) - Q^i_c(t)\right) - V\left(\alpha^i(E_l + E_w) + p_l\mu^i_l(t)\right)}{V\gamma^i(\mu^i_l(t) + \mu^i_w(t))}.$$

We call these prices as *transition prices* and each user i has maximal two transition prices. Note that transition prices sometimes can be a negative value according to the system state at time slot t. Then, there are at most three feasible intervals divided by transition prices where user i's decisions $(o^i(t), \theta^i_l(t), \theta^i_w(t))$ are not changed within each interval. The function $\Upsilon^i(p_c(t))$ in (20) linearly decreases on $p_c(t)$ within each interval. Fig. 7 shows an example of $\Upsilon^i(p_c(t))$ when all the three intervals are in the feasible region of $p_c(t)$.

When we consider $\mathcal{I}$, which is set of users, there exist maximum $2|\mathcal{I}|$ transition prices. We index transition prices in ascending order (i.e., $p^1_{\text{trans}} \le p^2_{\text{trans}} \le \ldots$) and let $\mathcal{P}_{\text{trans}}$ be a set of transition prices. For each interval $(p^k_{\text{trans}}, p^{k+1}_{\text{trans}}]$, it is enough to check p^{k+1}_{trans} to find minimum $\sum_{i\in\mathcal{I}}\Upsilon^i(p_c(t))$ value because $\sum_{i\in\mathcal{I}}\Upsilon^i(p_c(t))$ is a superposition of linearly decreasing function. Therefore, we need to check only $2|\mathcal{I}|$ transition prices at most (i.e., $O(|\mathcal{I}|)$ complexity), to find $p_c(t)$.

To determine the MCO price $p_c(t)$, the CSP requires user-side information in (32). Fortunately, the CSP is able to directly estimate all the information except α^i because the CSP already knows the device profiles (e.g., energy profile, local resource profile) and shares the application profiles (e.g., processing density, queue) with each user. Moreover, the CSP is also able to indirectly estimate the weight parameter α^i by pricing and observing user behavior for few time slots. ■

C. Proof of Theorem 2.

Proof: For a given computing task arrival rate vector $\boldsymbol{\lambda}$ which is interior to the capacity region Λ, there exists $\epsilon > 0$ such that $\boldsymbol{\lambda} + \boldsymbol{\epsilon} = (\lambda^1 + \epsilon, \lambda^2 + \epsilon, ..., \lambda^I + \epsilon) \in \Lambda$. Then, for arrival rate vector $\boldsymbol{\lambda} + \boldsymbol{\epsilon}$, there exists a stationary randomized control policy $(\boldsymbol{o}(t)^*, \boldsymbol{\theta}(t)^*, \boldsymbol{s}(t)^*, \boldsymbol{n}(t)^*)$ with satisfying (25)∼(28) by Theorem. 1.

From the upper bound of Lyapunov drift-plus-penalty function of (Coo-UP) (12), we have:

$$
\begin{aligned}
&\Delta L(t) + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\middle|\boldsymbol{Q}(t)\right] \\
&\le B_3 + V\mathbb{E}\left[\sum_{i\in\mathcal{I}}\Big[\left(1-o^i(t)\right)\alpha^i E_s\left(s^i(t)\right)\right. \\
&\quad \left. + o^i(t)\left\{\theta_l^i(t)\tilde{D}_l^i(t) + \theta_w^i(t)\tilde{D}_w^i(t)\right\} + n^i(t)e(t)\Big]\middle|\boldsymbol{Q}(t)\right] \\
&\quad -\sum_{i\in\mathcal{I}}\mathbb{E}\left[\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i} + o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\right]Q_u^i(t)\middle|\boldsymbol{Q}(t)\right] \\
&\quad +\sum_{i\in\mathcal{I}}\mathbb{E}\left[A^i(t)Q_u^i(t)\middle|\boldsymbol{Q}(t)\right] \\
&\quad -\sum_{i\in\mathcal{I}}\mathbb{E}\left[\left[n^i(t)\frac{s_c}{\gamma^i} - o^i(t)R^i\left(\boldsymbol{\theta}^i(t)\right)\right]Q_c^i(t)\middle|\boldsymbol{Q}(t)\right] \\
&\quad -\sum_{i\in\mathcal{I}}\mathbb{E}\left[\left[\left(1-o^i(t)\right)\frac{s^i(t)}{\gamma^i} + n^i(t)\frac{s_c}{\gamma^i}\right]\left(Q_u^i(t)+Q_c^i(t)\right)\middle|\boldsymbol{Q}(t)\right] \\
&\quad +\sum_{i\in\mathcal{I}}\mathbb{E}\left[A^i(t)\left(Q_u^i(t)+Q_c^i(t)\right)\middle|\boldsymbol{Q}(t)\right]. \qquad (34)
\end{aligned}
$$

Because our Coo-JC algorithm minimizes the RHS of (34), we can derive following equation by plugging the stationary randomized control policy $(\boldsymbol{o}(t)^*, \boldsymbol{\theta}(t)^*, \boldsymbol{s}(t)^*, \boldsymbol{n}(t)^*)$ into the RHS of (34). Note that this policy makes decisions independent of queue backlogs:

$$
\begin{aligned}
&\Delta L(t) + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\middle|\boldsymbol{Q}(t)\right] \\
&\le B_3 + V\mathbb{E}\left[\sum_{i\in\mathcal{I}}\Big[\left(1-o^i(t)^*\right)\alpha^i E_s\left(s^i(t)^*\right)\right. \\
&\quad \left. + o^i(t)^*\left\{\theta_l^i(t)^*\tilde{D}_l^i(t) + \theta_w^i(t)^*\tilde{D}_w^i(t)\right\} + n^i(t)^*e(t)\Big]\right] \\
&\quad -\sum_{i\in\mathcal{I}}\mathbb{E}\left[\left[(1-o^i(t)^*)\frac{s^i(t)^*}{\gamma^i} + o^i(t)^*R^i\left(\boldsymbol{\theta}^i(t)^*\right)\right]Q_u^i(t)\right] \\
&\quad +\sum_{i\in\mathcal{I}}Q_u^i(t)\lambda^i - \sum_{i\in\mathcal{I}}\mathbb{E}\left[\left[n^i(t)^*\frac{s_c}{\gamma^i} - o^i(t)^*R^i\left(\boldsymbol{\theta}^i(t)^*\right)\right]Q_c^i(t)\right] \\
&\quad -\sum_{i\in\mathcal{I}}\mathbb{E}\left[\left[(1-o^i(t)^*)\frac{s^i(t)^*}{\gamma^i} + n^i(t)^*\frac{s_c}{\gamma^i}\right]\left(Q_u^i(t)+Q_c^i(t)\right)\right] \\
&\quad +\sum_{i\in\mathcal{I}}\left(Q_u^i(t)+Q_c^i(t)\right)\lambda^i,
\end{aligned}
$$

By applying the property of policy $(\boldsymbol{o}(t)^*, \boldsymbol{\theta}(t)^*, \boldsymbol{s}(t)^*, \boldsymbol{n}(t)^*)$, we have:

$$
\begin{aligned}
&\Delta L(t) + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\middle|\boldsymbol{Q}(t)\right] \\
&\le B_3 + V\Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon}) - \epsilon\sum_{i\in\mathcal{I}}Q_u^i(t) - \epsilon\sum_{i\in\mathcal{I}}\left(Q_u^i(t)+Q_s^i(t)\right). \qquad (35)
\end{aligned}
$$

By taking expectations of (35) and using the law of iterated expectations, we have:

$$
\begin{aligned}
&\mathbb{E}\left[L(t+1)\right] - \mathbb{E}\left[L(t)\right] + V\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\right] \\
&\quad \le B_3 + V\Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon}) - \epsilon\sum_{i\in\mathcal{I}}\mathbb{E}\left[2Q_u^i(t) + Q_c^i(t)\right]. \qquad (36)
\end{aligned}
$$

By summing (37) over $t \in \{0, 1, ..., T-1\}$, we have:

$$
\begin{aligned}
&\mathbb{E}\left[L(T)\right] - \mathbb{E}\left[L(0)\right] + V\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\right] \\
&\quad \le B_3 T + VT\Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon}) - \epsilon\sum_{t=0}^{T-1}\sum_{i\in\mathcal{I}}\mathbb{E}\left[2Q_u^i(t) + Q_c^i(t)\right]. \qquad (37)
\end{aligned}
$$

By rearranging terms, neglecting negative quantities and $T \to \infty$, we can show following inequalities.

$$
\begin{aligned}
&\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{I}} h_u^i(t) + h_c(t)\right] \le \Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon}) + \frac{B_3}{V}, \\
&\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left\{Q_u^i(t) + Q_c^i(t)\right\}\right] \\
&\le \limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{I}}\left\{2Q_u^i(t) + Q_c^i(t)\right\}\right] \le \frac{B_3 + V\Psi(\boldsymbol{\lambda}+\boldsymbol{\epsilon})}{\epsilon}.
\end{aligned}
$$

■

REFERENCES

[1] Y. Kim, J. Kwak, and S. Chong, "Dual-side dynamic controls for cost minimization in mobile cloud computing systems," in *Proc. 13th Int. Symp. Model. Optimization Mobile, Ad Hoc, Wireless Netw.*, Mumbai, India, May 2015, pp. 443–450.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[3] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog *et al.*: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.
[4] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Serv.*, Taipei, Taiwan, Jun. 2013, pp. 319–332.
[5] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020." Cisco, San Jose, CA, USA. Mar. 2017. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html
[6] "5G vision: The 5G infrastructure public private partnership: the next generation of communication networks and services." Feb. 2015. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf
[7] R. Y. Clarke, R. Westervelt, D. Vesset, M. Torchia, A. Siviero, R. Segal, K. Prouty, V. Turner, C. MacGillivray, and L. Lamy, "IDC Futurescape: Worldwide Internet of Things 2016 Predictions," IDC FutureScape, 2016.
[8] "Open Fog Consortium." [Online]. Available: https://www.openfogconsortium.org/
[9] "Mobile Edge Computing." [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing/
[10] "Cloudlets." [Online]. Available: http://elijah.cs.cmu.edu/
[11] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A. Neal "Mobile edge computing - Introductory technical white paper," White Paper, Mobile-edge Computing (MEC) industry initiative, 2014.
[12] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy-delay tradeoffs in smartphone applications," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1647–1660, Jun. 2016.
[13] 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.
[14] A.-H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 120–133, Sep. 2010.
[15] "Cloud flatform, Microsoft Azure." [Online]. Available: https://azure.microsoft.com/en-us/
[16] G. Brown, "Mobile Edge Computing Use Cases and Deployment Options, Juniper White Paper," pp. 1–9, Jul. 2016. [Online]. Available: https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf
[17] M. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems* (Synthesis Lectures on Communication Networks), vol. 3, no. 1. San Rafael, CA, USA: Morgan Claypool, 2010, pp. 1–211.
[18] E. Cuervo, A. Balasubramanian, and D. Cho, "MAUI: Making smartphones last longer with code offload," in *Proc. ACM Annu. Int. Conf. Mobile Syst., Appl., Serv.*, San Francisco, CA, USA, Jun. 2010, pp. 49–62.
[19] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.
[20] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[21] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[22] M. Lin, A. Wierman, L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.
[23] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2648–2658, Oct. 2014.
[24] Y. Chen, C. Lin, J. Huang, X. Xiang, and X. Shen, "Energy efficient scheduling and management for large-scale services computing systems," *IEEE Trans. Serv. Comput.*, vol. 10, no. 2, pp. 217–230, Mar./Apr. 2017.
[25] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Macau, China, Jun. 2012, pp. 22–31.
[26] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 936–944.
[27] S. Ren and M. van der Schaar, "Dynamic scheduling and pricing in wireless cloud computing," *IEEE Trans. Mobile Comput.*, vol. 13, no. 10, pp. 2283–2292, Oct. 2014.
[28] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. Lau, "Dynamic pricing and profit maximization for the cloud with geo-distributed data centers," in *Proc. IEEE INFOCOM*, Toronto, Canada, Apr. 2014, pp. 118–126.
[29] I. Menache, A. Ozdaglar, and N. Shimkin, "Socially optimal pricing of cloud computing resources," in *Proc. 5th Int. ICST Conf. Performance Eval. Methodologies and Tools*, Brussels, Belgium, May 2011, pp. 322–331.
[30] J. Song, Y. Cui, M. Li, J. Qiu, and R. Buyya, "Energy-traffic tradeoff cooperative offloading for mobile cloud computing," in *Proc. IEEE Int. Symp. Quality Serv.*, Hong Kong, May 2014, pp. 284–289.
[31] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," arXiv:1703.06058, 2017.
[32] "Kernel governors, modules, I/O scheduler, CPU tweaks AIO app configs." Nov. 2011. [Online]. Available: http://forum.xda-developers.com/showthread.php?t=1369817
[33] M. Ra, J. Peak, A. Sharma, R. Govindan, M. Krieger, and M. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. Annu. Int. Conf. Mobile Syst., Appl., Serv.*, San Francisco, CA, USA, Jun. 2010, pp. 255–270.
[34] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. USENIX Conf. Networked Syst. Des. Implementation*, Boston, MA, USA, Mar. 2011, pp. 99–112.
[35] "Operating Systems, iOS 6.0." [Online]. Available: http://www.apple.com/ios/
[36] "Samsung galaxy s5 download booster." [Online]. Available: http://galaxys5guide.com/samsung-galaxy-s5-features-explained/galaxy-s5-download-booster/
[37] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for android smartphone using kernel activity monitoring," in *Proc. USENIX Annu. Tech. Conf.*, San Jose, CA, USA, Jun. 2012, pp. 387–400.
[38] Y. Cui, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai, "Performance-aware energy optimization on mobile devices in cellular network," in *Proc. IEEE INFOCOM*, Toronto, Canada, Apr. 2014, pp. 1123–1131.
[39] L. Kleinrock, *Queuing Systems*. New York, NY, USA: Wiley, 1975.
[40] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–149, 2006.
[41] "Samsung Galaxy Note specifications." [Online]. Available: http://www.samsung.com/global/microsite/galaxynote/note/spec.html?type=find
[42] "Monsoon Power Monitor." [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/
[43] "Dataset for statistics and social network of YouTube videos." Jun. 2008. [Online]. Available: http://netsg.cs.sfu.ca/youtubedata/
[44] "California ISO." [Online]. Available: http://www.caiso.com/
[45] "Mobile data plans of SK telecom." [Online]. Available: http://www.tworld.co.kr/?gclid=CP7M8b_znswCFVgkvQodW6YDgw

Yeongjin Kim (S'13) received the B.S. and M.S. degrees from the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2011 and 2013, respectively. He is currently working toward the Ph.D. degree in KAIST. His research interests are in the areas of mobile opportunistic networks, content caching in 5G cellular networks, mobile cloud-edge computing, and control of multi-resource infrastructures for Software Defined Network (SDN) and Network Function Virtualization (NFV).

Jeongho Kwak (S'11–M'15) received the B.S. degree (Summa cum laude) in electrical and computer engineering from Ajou University, Suwon, South Korea, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2011 and 2015, respectively. He was a postdoctoral researcher at INRS and Western University in Canada. Currently, he is a Postdoctoral Researcher at Trinity College Dublin, Ireland. He received Marie Sklodowska-Curie Research Fellowship from European Union in 2017. His current research interests lie on storage/computing/networking resource orchestration in hybrid edge network architecture.

Song Chong (M'93) received the B.S. and M.S. degrees from Seoul National University and the Ph.D. degree from the University of Texas at Austin, all in electrical engineering. He is a Professor in the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), where he has led Network Systems Laboratory since 2000. He is the Head of Computing, Networking and Security Group of the school since 2009, and the Founding Director of KAIST 5G Mobile Communications & Networking Research Center. Prior to joining KAIST, he was with the Performance Analysis Department, AT&T Bell Laboratories, Holmdel, NJ, USA, as a Member of Technical Staff. His current research interests include wireless networks, mobile systems, performance evaluation, distributed algorithms, and data analytics. He is on the editorial boards of the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He was the Program Committee Cochair of IEEE SECON 2015 and has served on the Program Committee of a number of leading international conferences including IEEE INFOCOM, ACM MobiCom, ACM CoNEXT, ACM MobiHoc, IEEE ICNP, and ITC. He serves on the Steering Committee of WiOpt and was the General Chair of WiOpt 2009. He received two IEEE William R. Bennett Prize Paper Awards in 2013 and 2016, given to the best original paper published in IEEE/ACM Transactions on Networking in the previous three calendar years, and the IEEE SECON Best Paper Award in 2013.