Joint Optimal Pricing and Task Scheduling in Mobile Cloud Computing Systems

Hamed Shah-Mansouri, *Member, IEEE*, Vincent W.S. Wong, *Fellow, IEEE*, and Robert Schober, *Fellow, IEEE*

***Abstract*—The evolving mobile cloud computing (MCC) paradigm enables mobile users to offload their computing tasks to cloud servers. In this paper, we study the following problems in MCC systems: (i) *which tasks should be offloaded to cloud servers*? (ii) and *what is the optimal price of cloud services*? We jointly address these issues by formulating two levels of optimization problems. On the mobile users side, we formulate a utility maximization problem that takes the energy consumption, delay, and price of cloud services into account and obtain the optimal scheduling for both delay-sensitive and delay-tolerant applications. On the cloud service provider (CSP) side, we determine the optimal pricing strategy by formulating a profit maximization problem, which is non-convex in general. We further propose an algorithm using convexification and primal-dual methods to mitigate the non-convexity. Through numerical studies, we investigate the mobile users' behavior and the CSP's pricing strategy. Our results reveal that the proposed scheduler effectively balances the tradeoff between the energy consumption and delay in comparison with different schedulers proposed in the literature. Furthermore, we show that with the proposed pricing algorithm, the CSP can improve its profit by up to 25% compared to static and dynamic pricing strategies.**

***Index Terms*—Dynamic task scheduler, delay-sensitive applications, optimal pricing strategy, mobile cloud computing.**

I. Introduction

MOBILE cloud computing (MCC) reduces the computational burden of mobile devices by extending the concept of cloud computing to the mobile environment. Computation-intensive applications are rapidly developing, while the processing power of mobile devices is often limited. To overcome this problem, computational tasks can be executed remotely on cloud computing servers on behalf of mobile devices. By utilizing cloud computing services, mobile devices can benefit from powerful computing resources, save their battery power, and expedite the task execution [1]–[6]. To enable mobile computation offloading, several cloud-assisted mobile platforms have been proposed including ThinkAir [7], MAUI [8], CloneCloud [9], and cloudlets [10]. Cisco forecasts that mobile cloud traffic will grow tremendously and cloud applications will account for 90% of total mobile data traffic by 2019 [11]. Nonetheless, the decision about whether a task should be performed locally on the mobile devices or offloaded to cloud servers is still a challenging problem.

The module making the offloading decision in each mobile device is called the *task scheduler*. The task scheduler dynamically makes an offloading decision upon arrival of a task. It takes into account the energy saving obtained from task offloading, the delay each task experiences, and the price charged by the cloud service provider (CSP) to arrive at the optimal decision. On the other hand, the CSP requires an optimal pricing scheme for its computing services so as to maximize its own profit, as a static pricing strategy cannot capture the dynamic MCC environment. The pricing strategy of the CSP depends on the workload of its servers, whereas the offloading demands of mobile users depend on the price of cloud services. Thus, the optimal decisions of task schedulers and the CSP's pricing strategy are coupled together, which complicates the design of the optimal pricing strategy.

Recently, there have been several works studying task offloading for the MCC environment. Odessa [12] is a runtime system that enables task offloading to cloud servers for different mobile applications. Although it has a low complexity, its offloading strategy is sub-optimal and may degrade the benefits obtained from task offloading. An energy optimal scheduler for mobile users was proposed in [13]. The scheduler makes the offloading decision by comparing the energy consumed for executing the tasks locally and the energy required for dispatching the tasks to the cloud servers. In [14], the authors considered the case of simultaneously arriving tasks and designed an optimal task scheduler by minimizing the total energy consumption in mobile devices. The offloading strategy proposed in [15] minimizes the completion time of the tasks on mobile devices, when cloud services are available. A centralized task scheduler was proposed in [16], where the authors assumed the MCC system has a controller. The controller monitors the offloading requests of the mobile users as well as the available computing resources in the cloud servers and assigns computing resources to the mobile devices with the objective to minimize their total energy consumption. Another offloading strategy proposed in [17] considers a sequence of tasks of a mobile user and aims to minimize the energy consumption when the user collaborates with the cloud servers. In [18], the authors studied energy-efficient task execution in an MCC system. They determined the offloading decision for each task in order to minimize the energy consumption on a mobile device while meeting a latency deadline. A dynamic resource and task allocation scheme was proposed in [19]

Manuscript received on Oct. 28, 2016; revised on Mar. 21, 2017; accepted on May 8, 2017. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. The review of this paper was coordinated by Prof. Shaowei Wang.

H. Shah-Mansouri and V. W.S. Wong are with the Department of Electrical and Computer Engineering, the University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (e-mail:{hshahmansour, vincentw}@ece.ubc.ca).

R. Schober is with the Institute for Digital Communications, Friedrich-Alexander University of Erlangen–Nuremberg, Germany (email: robert.schober@fau.de)

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

Digital Object Identifier xx.xxxx/TWC.2017.xxxxxxx

that minimizes the energy consumption of the mobile users. In [20], the authors studied a dynamic MCC system and proposed an offloading mechanism for cost minimization for both mobile users and the CSP by taking the price of the CSP into account. Although the proposed mechanism addresses the energy-delay tradeoff, all computing tasks face the same tradeoff between energy consumption and delay regardless of their different sensitivity to delay[1]. In [21], the authors considered an MCC system and optimized the mobile users' offloading decisions in order to minimize the overall cost. They assumed that the mobile users share the same wireless channel while offloading their tasks to the cloud servers. In [21], a centralized controller hosted by the cloud servers decides whether to offload the tasks. A distributed offloading algorithm was proposed in [22] that aims to minimize the total energy consumption of mobile devices. An application-aware computation offloading mechanism was proposed in [23] that balances the tradeoff between energy efficiency and responsiveness of mobile applications. In [24], the authors studied energy-efficient computation offloading under a completion time deadline constraint. Given the increasing attention on mobile cloud computing in recent years, there exist several cloud services for mobile users such as AppATP [25] and eTrain [26] that leverage the tradeoff between energy consumption and delay. AppATP defers data offloading of delay-tolerant applications during poor network connectivity in order to minimize the energy consumption. In [26], the proposed eTrain mechanism also trades the delay for energy consumption for instant messaging applications.

The aforementioned offloading mechanisms demonstrate the benefits of using cloud services. However, the existing works do not take into account two important characteristics of MCC environments. In particular, they do not consider the heterogeneous latency requirements of different delay-sensitive applications. Moreover, the existing works (e.g., [12]–[17], [21]–[24]) do not consider the monetary cost (i.e., service price) that the CSP charges for providing cloud services. Different latency requirements of tasks and the price of cloud services may affect the task offloading decisions, even if offloading is beneficial in terms of energy efficiency.

Besides, the CSP is interested in maximizing its profit obtained from providing cloud services. This implies that the CSP's pricing scheme should be designed so as to admit as many service requests as the CSP can serve in order to increase the profit. Hence, a dynamic pricing strategy can significantly enhance the profit of the CSP, whereas a static approach cannot model the dynamic behavior of the MCC properly.

Dynamic pricing in mobile networks [27]–[31] as well as cloud computing systems [32]–[38] has received much attention in recent years. Several dynamic pricing schemes have been proposed for cloud computing services [32]–[38]. Among these existing works, the authors of [32] proposed a socially optimal pricing strategy to optimize the social welfare, which is the sum of the users' utilities minus the CSP's expenses. Moreover, several auction based pricing mechanisms have been proposed in [33]–[36], which aim to optimize the social cost/welfare. These works assumed that the users submit their bids to lease the virtual machines for the time period they need. In [37], the authors studied an MCC market, where multiple brokers compete to reserve computing resources from public and local clouds. The design objective in [37] is to find the allocation strategy for all brokers that minimizes the average social cost of all mobile users. However, in practice, the CSP is only interested in maximizing its own profit. In addition, users do not concern about the CSP's profit and other users' behaviors. Thus, the aforementioned existing mechanisms (e.g., [33]–[37]) that aim to optimize the social welfare/cost can neither model the optimal pricing strategy of the CSP nor the strategic behaviors of users. A pricing strategy to maximize the CSP's profit was proposed in [38] assuming that the users lease the virtual machines in cloud servers for a certain period of time. However, such pricing mechanism can be used in either Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) deployments for stationary users (e.g., personal desktop computer users or small and large enterprises), while it may not be applicable in MCC systems. Mobile users make an offloading decision upon arrival of each task and lease the cloud services during the task execution. Hence, mobile users are not interested in leasing the computing services for a certain period of time.

In this paper, we address the following technical and economical challenges in MCC system design:

1) *How does the task scheduler make the offloading decision upon arrival of each task?*
2) *What is the optimal pricing strategy of the CSP?*

To address these challenges, we jointly optimize the task scheduler for the mobile users and the pricing strategy for the CSP in a dynamic MCC market. Our design is motivated by two insights: First, the task offloading decision should not only depend on the energy consumption saving obtained by using the cloud computing services, but also be affected by the service prices. Second, a static pricing strategy can neither model a dynamic MCC market nor properly encourage mobile users to use the cloud services. An optimal strategy should take into account the dynamic workload of computing tasks which may vary due to different energy savings in mobile devices and heterogeneous delay requirements as well as the price of cloud services.

The key contributions of our work are as follows:

- *Dynamic task scheduler*: To design the task scheduler, we formulate a utility maximization problem, which takes the energy consumption, delay, and price of cloud services into account. We consider the stochastic arrival of tasks and provide a queuing analysis to address the latency requirements of both delay-sensitive and delay-tolerant applications.
- *Optimal pricing via profit maximization*: We determine the optimal pricing strategy of the CSP in a dynamic MCC market. We formulate a profit maximization prob-

[1]In [19], [20], the authors assumed that the computing tasks can be partitioned in a flexible way. The computing job arrivals are modeled as streams of bits and the offloading decisions are made for the bits arrived in fixed time slots. In practice, however, the computational tasks cannot be divided into smaller ones in an arbitrary manner [8], [14] and each indivisible computing task should either be served entirely by the local CPU of the mobile device or be offloaded to the cloud servers.

lem which takes into account both the price charged to the mobile users and the electricity price of the CSP. We model the workload of the CSP (i.e., the offloading demand arriving from the mobile users) as a function of its price to determine the CSP's profit and to reveal the interaction between the CSP and the mobile users.

- *Algorithm design*: We show that the CSP's profit maximization problem is non-convex in general. We then develop a pricing algorithm using Convexification and Primal-dual methods, namely CoPe, to cope with the non-convexity issue.
- *Numerical studies*: We investigate the performance of the proposed task scheduler for different types of tasks and show that the scheduler outperforms ThinkAir [7] and MAUI [8] in terms of energy and delay for delay-sensitive tasks, while it consumes the same amount of energy as ThinkAir for delay-tolerant tasks. We also evaluate the effect of the price on the offloading decision of the mobile users. Our results show that the mobile users may prefer local execution of a task if the price of the CSP is too high even if task offloading is beneficial in terms of energy consumption or delay. We further study the CSP's optimal pricing strategy and the CoPe algorithm and show that the CSP can obtain a significantly higher profit by employing either the optimal strategy or CoPe in comparison with static and dynamic pricing strategies.

This paper is organized as follows. In Section II, we develop the dynamic task scheduler to determine the optimal offloading strategy of the mobile users. In Section III, the optimal pricing strategy of the CSP is obtained based on a profit maximization framework and the sub-optimal pricing algorithm CoPe is developed. In Section IV, we investigate the effect of CSP's prices on the mobile users' behavior as well as the CSP's profit. Conclusions are drawn in Section V.

II. Dynamic Task Scheduler

Consider an MCC environment, which includes the CSP and mobile devices. The set of mobile devices is denoted as $\mathcal{M}$. Each mobile device uses a task scheduler that decides whether to offload a task to the cloud servers or execute it locally in its centralized processing unit (CPU). Mobile users choose their optimal strategy (i.e., offloading decision) individually based on the price announced by the CSP so as to maximize their own utilities. In this section, we first introduce the task scheduler model as well as the offloading decision of the mobile users. We then design the task scheduler using a utility maximization framework and obtain the optimal offloading strategy of the mobile users.

A. Task Scheduler Model

We model the task scheduler in each mobile device[2] by a queuing system, as illustrated in Fig. 1. We assume that in each mobile device there are two servers, namely, the CPU and the wireless interface (e.g., WiFi, Long-Term Evolution (LTE)).

[2]In the remainder of this paper, we use the terms "mobile device" and "mobile user" interchangeably.

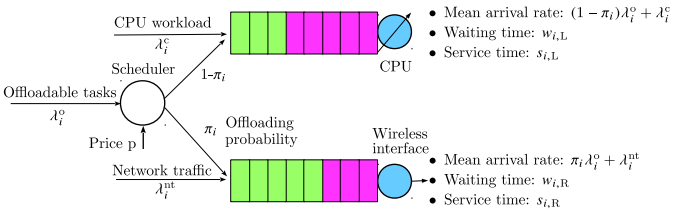


Fig. 1: The task scheduler and queuing system for mobile user i. The queuing system includes two disjoint queues served by two different servers, namely the local CPU and the wireless interface. We categorize the tasks into three types: CPU workload, offloadable computing tasks, and network traffic. The scheduler decides whether or not to offload the offloadable tasks.

The former server is used to model the local execution of the tasks in the mobile device. The latter is required for offloading the tasks to the cloud servers.

We classify the mobile user's workload into three categories: CPU workload, offloadable computing tasks, and network traffic. The CPU workload represents the tasks that have to be processed locally by the mobile device's CPU, whereas the offloadable computing tasks can either be processed by the local CPU or be offloaded to the cloud servers. The network traffic has to be transmitted over the wireless interface. For example, mobile applications such as virus scanners [39] can either be run on the local CPU or be offloaded to the cloud servers. However, there are some tasks which must be performed locally by the CPU and cannot be offloaded. Examples include but are not limited to the display control and memory and cache management.

We assume that the CPU workload, offloadable computing tasks, and network traffic independently arrive at mobile user $i \in \mathcal{M}$ according to Poisson processes with rate λ_i^{c}, λ_i^{o}, and λ_i^{nt}, respectively. The size of each task z (in bits) follows a probability density function (pdf) $f_Z(z)$. We further denote the processing density as γ (in cycles/bit), which is the number of CPU cycles required to process a unit bit of data. The value of γ depends on the application type. We model γ as a random variable that follows pdf $f_\Gamma(\gamma)$. We assume that the task size z and the processing density γ are finite. In our queuing analysis, the size of each task and the required number of CPU cycles reflect the service time that the corresponding server needs to complete the task. The service time to perform a task of size z with processing density γ in the local CPU of mobile device i is $\gamma z/C_i$, where C_i is the CPU processing capacity (in cycles/time unit)[3]. The same model for the CPU processing capacity and service time was used in [15], [23], [24]. The service time in the wireless interface server corresponds to the transmission time required to submit the computing tasks to the cloud servers. For a task of size z, the service time is z/μ, where μ (in bits/sec) is the data rate of the wireless interface. We consider a slow flat fading wireless channel model and assume that the data rate of the wireless interface at each device i remains constant during the transmission of each task. However, μ is a random variable and follows a pdf $f_{M_i}(\mu)$.

[3]CPUs in existing smartphones employ a dynamic voltage and frequency scaling method to adjust the CPU clock frequency so as to optimize the energy-speed tradeoff [13], [19]. However, for the sake of tractability of analysis, here we assume a constant clock frequency which results in a constant processing power for each CPU.

We further assume that the data rate, hence the service time, is independent of the arrival processes and is known to the user upon arrival of tasks.

The task scheduler makes the offloading decision for both delay-sensitive and delay-tolerant applications. To model the delay-sensitivity, we characterize each task by a parameter θ representing a specific application type as inspired by [27]. The value of θ varies across different types of applications, but is known to users and follows pdf $f_\Theta(\theta)$. A large value of θ represents applications with stringent delay requirements, whereas applications with $\theta = 0$ are tolerable to delay.

B. Offloading Decision

To design the task scheduler, we first introduce the offloading decision indicator and offloading probability. As illustrated in Fig. 1, each mobile device is modeled by two queues served by two different servers. For mobile user $i \in \mathcal{M}$, let $\delta_i(z,\gamma,\theta,\mu) \in \{0,1\}$ indicate whether or not a task of size z with processing density γ and delay parameter θ is offloaded, where μ is the data rate of the wireless interface and $\delta_i(z,\gamma,\theta,\mu) = 1$ indicates that the task is offloaded. Notice that the task scheduler makes its decision based on the values of z, γ, θ, and μ which are known upon arrival of the task. We further denote the probability that a task of size z with processing density γ and delay parameter θ at mobile user i with wireless data rate μ is offloaded to the cloud servers by $\pi_i(z,\gamma,\theta,\mu) \in [0,1]$. Therefore, $\delta_i(z,\gamma,\theta,\mu) = 1$ with probability $\pi_i(z,\gamma,\theta,\mu)$. Given $\{\pi_i(z,\gamma,\theta,\mu), z \geq 0, \gamma \geq 0, \theta \geq 0, \mu \geq 0\}$, the probability that a task from mobile user i is offloaded to the CSP is

$$\pi_i = \int_{\mathbb{R}_+^4} \pi_i(z,\gamma,\theta,\mu) \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu), \tag{1}$$

where $F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu)$ is the joint cumulative distribution function (cdf) of random variables z, γ, θ, and μ. We refer to π_i as the *offloading probability* of device $i \in \mathcal{M}$. To design the scheduler and determine the offloading probability, we first require a model for the utility of the users as the task scheduler evaluates the users' utility to make the offloading decisions.

C. Users' Utility

The utility reflects the benefit of mobile users from offloading the task to the cloud servers. The utility depends on the energy consumption and the delay improvement obtained by offloading the task as well as the price charged by the CSP.

To introduce the utility of user $i \in \mathcal{M}$, we first define the energy consumption saving obtained from task offloading as the energy consumed in the local CPU to execute the task minus the transmission energy required to submit the task to the cloud servers. The energy consumption in mobile user i's CPU for executing a task of size z requiring γ CPU cycles per bit can be modeled as follows [40]–[42]:

$$\left(\kappa_i \left(C_i\right)^{\varphi_i} + \varrho_i\right) \frac{\gamma z}{C_i}, \tag{2}$$

where κ_i, φ_i, and ϱ_i are user-dependent constants that depend on the CPU model. Moreover, $\gamma z / C_i$ is the time required to process the task. The energy consumption in the wireless interface is $\beta_i z/\mu$, where z/μ is the time required to transmit the task to the cloud servers and β_i depends on the type of interface and is different for WiFi and LTE. The same model for the energy consumption in the CPU and the wireless interface was used in [19], [20]. Therefore, when the data rate is μ, the energy consumption saving obtained from offloading a task of size z with processing density γ to the cloud servers, denoted by $h_i(z,\gamma,\mu)$, is as follows:

$$h_i(z,\gamma,\mu) = \left(\kappa_i \left(C_i\right)^{\varphi_i} + \varrho_i\right) \frac{\gamma z}{C_i} - \beta_i \frac{z}{\mu}. \tag{3}$$

Furthermore, we define the delay improvement obtained by using the cloud services as the difference between the time required to complete the task locally and the time spent to process the task remotely in the cloud servers. It should be noted that if offloading the task to the cloud servers imposes a longer delay than local execution, the delay improvement is negative. To determine the delay improvement, we first focus on the delay that each task experiences if it is offloaded to the cloud servers. We consider the time that the task spends in the wireless interface queue of the mobile device as well as the time the cloud servers need to perform the task. In particular, the delay for a task executed remotely by the cloud servers consists of four terms: the waiting time in the wireless interface queue of the user, the service time of the wireless interface to submit the task to the cloud servers, the processing time of the cloud servers to complete the task, and the time required to retrieve the results from the cloud servers[4]. Since the downlink rate of the mobile users is usually much higher than the uplink rate, the latter time is negligible in comparison with the time required to submit the task to the cloud servers.

From Fig. 1, the arrivals at the wireless interface queue consist of the offloaded computing tasks and the network traffic. Given the offloading probability π_i, the offloaded computing tasks arriving at this queue follow a Poisson process with arrival rate $\pi_i \lambda_i^{\mathrm{o}}$. Notice that thinning a Poisson process with a fixed probability results in a new Poisson process [43]. Moreover, combining the Poisson processes of the offloaded tasks and the network traffic forms another Poisson process [43] as they are independent. Thus, the wireless interface queue can be modeled as an $M/G/1$ queuing system with arrival rate $\pi_i \lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}}$, when π_i is a given constant. We define $w_{i,\mathrm{R}}(\pi_i)$ and $s_{i,\mathrm{R}}(z,\mu) = z/\mu$ as the waiting time and the service time in the wireless interface queue of user i, respectively. We further denote the mean service time as $\mathbb{E}[s_{i,\mathrm{R}}]$, where $\mathbb{E}[\cdot]$ denotes the expected value of a variable. Given π_i, the extended-value mean waiting time in this queue as obtained from the Pollaczek-Khinchin formula [43] is

$$\mathbb{E}[w_{i,\mathrm{R}}(\pi_i)] = \begin{cases} \frac{(\pi_i \lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}}) \mathbb{E}[s_{i,\mathrm{R}}^2]}{2\left(1 - (\pi_i \lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}}) \mathbb{E}[s_{i,\mathrm{R}}]\right)}, & \text{if } \pi_i \lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}} < \frac{1}{\mathbb{E}[s_{i,\mathrm{R}}]} \\ \infty, & \text{otherwise.} \end{cases} \tag{4}$$

[4]Similar to [15], [16], [19], [20], [24], we assume that the cloud servers are located in close proximity of the mobile devices such that the roundtrip delay between a mobile device and the cloud servers is negligible.

The queue is stable if $\pi_i\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}} < 1/\mathbb{E}[s_{i,\mathrm{R}}]$ holds. We further denote the processing time for the computing task of user i in the cloud servers as $s_{i,\mathrm{C}}(z,\gamma) = \gamma z/C_{\mathrm{R}}$, where C_{R} denotes the processing capacity (in cycles/unit time) of each cloud server. Thus, the delay caused by performing the task remotely in the cloud servers given offloading probability π_i is[5]

$$\mathbb{E}[w_{i,\mathrm{R}}(\pi_i)] + s_{i,\mathrm{R}}(z,\mu) + s_{i,\mathrm{C}}(z,\gamma).$$

We now study the CPU queue to determine the delay induced by performing the task locally. Similar to the wireless interface, when π_i is given, the CPU can be modeled as an $M/G/1$ queuing system with arrival rate $(1-\pi_i)\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{c}}$. For this queuing system, we define $w_{i,\mathrm{L}}(\pi_i)$ and $s_{i,\mathrm{L}}(z,\gamma)$ as the waiting time and the service time, respectively. According to the Pollaczek-Khinchin formula [43], given π_i, the extended-value mean waiting time in the $M/G/1$ queuing system of the local CPU with mean arrival rate $(1-\pi_i)\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{c}}$ and mean service time $\mathbb{E}[s_{i,\mathrm{L}}]$ is

$$\mathbb{E}[w_{i,\mathrm{L}}(\pi_i)] = \begin{cases} \frac{((1-\pi_i)\lambda_i^{\mathrm{o}}+\lambda_i^{\mathrm{c}})\mathbb{E}[s_{i,\mathrm{L}}^2]}{2\left(1-((1-\pi_i)\lambda_i^{\mathrm{o}}+\lambda_i^{\mathrm{c}})\mathbb{E}[s_{i,\mathrm{L}}]\right)}, & \text{if } (1-\pi_i)\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{c}} < \frac{1}{\mathbb{E}[s_{i,\mathrm{L}}]} \\ \infty, & \text{otherwise.} \end{cases} \tag{5}$$

The condition $(1-\pi_i)\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{c}} < 1/\mathbb{E}[s_{i,\mathrm{L}}]$ guarantees queue stability. Furthermore, we have $s_{i,\mathrm{L}}(z,\gamma) = \gamma z/C_i$ as the service time for a task of size z with processing density γ. Note that the service time does not depend on probability π_i. The delay introduced by performing the task locally in user i's CPU given offloading probability π_i is

$$\mathbb{E}[w_{i,\mathrm{L}}(\pi_i)] + s_{i,\mathrm{L}}(z,\gamma).$$

As mentioned earlier, the delay improvement is the time required to complete the task locally minus the time spent to process the task remotely in the cloud servers. Given offloading probability π_i, the delay improvement, denoted by $\tau_i(z,\gamma,\mu,\pi_i)$, is as follows:

$$\tau_i(z,\gamma,\mu,\pi_i) = \mathbb{E}[w_{i,\mathrm{L}}(\pi_i)] + s_{i,\mathrm{L}}(z,\gamma) - (\mathbb{E}[w_{i,\mathrm{R}}(\pi_i)] + s_{i,\mathrm{R}}(z,\mu) + s_{i,\mathrm{C}}(z,\gamma)). \tag{6}$$

In the following lemma, we show that the delay improvement $\tau_i(z,\gamma,\mu,\pi_i)$ is decreasing in π_i. We will use this property in Section II-E to determine the offloading probability.

Lemma 1. *The delay improvement $\tau_i(z,\gamma,\mu,\pi_i)$ is decreasing in π_i.*

Proof. From (4) and (5), we observe that $\mathbb{E}[w_{i,\mathrm{R}}(\pi_i)]$ is increasing in π_i, while $\mathbb{E}[w_{i,\mathrm{L}}(\pi_i)]$ decreases as π_i increases. Since the other terms in (6) do not depend on π_i, $\tau_i(z,\gamma,\mu,\pi_i)$ is decreasing. ∎

In addition to the energy consumption saving and the delay improvement, the scheduler takes also the price that the mobile user has to pay for using the cloud services into account. This payment will reduce the willingness of users to offload their tasks, since it affects their utilities. The amount of payment is proportional to the computing resources required to perform the task. The CSP charges the mobile users for offloading a task of size z with processing density γ by $p\gamma z$ (in \$), where p in \$/Giga cycle (Gcycle)[6] denotes the unit price announced by the CSP.

We now model the utility of user i by considering the energy consumption saving, delay improvement, and price. Given π_i, user i's utility obtained from offloading a task of size z with processing density γ and delay parameter θ to the cloud servers when data rate is μ is as follows[7]:

$$u_i^{\pi_i}(z,\gamma,\theta,\mu) = h_i(z,\gamma,\mu) + \theta\tau_i(z,\gamma,\mu,\pi_i) - \alpha_i p\gamma z, \tag{7}$$

where $h_i(z,\gamma,\mu)$ and $\tau_i(z,\gamma,\mu,\pi_i)$ are given by (3) and (6), respectively. In addition, α_i in (Joule (J)/\$) is a constant trade-off parameter between the payment and the energy consumed in the device of mobile user i. The value of θ reflects the application type and its sensitivity to delay. As an example, for delay-tolerant applications, θ is set to 0 to ignore the effect of delay in the utility function. A higher value of θ implies a higher sensitivity of an application to delay.

Notice that both the energy consumption saving and the delay improvement may be less than zero. Indeed, poor wireless channel conditions may introduce a large delay in dispatching the tasks to the cloud servers, which degrades the benefit of task offloading. Similarly, if the energy consumed in the local CPU is less than the energy required for transmitting the task to the cloud servers due to poor wireless channel conditions, then the energy saving is less than zero.

D. Utility Maximization Framework

Upon arrival of each task, the scheduler aims to maximize the utility of the user. The utility function given in (7) reflects the benefit of mobile user i from offloading the task. Notice that if a task is performed locally (i.e., $\delta_i(z,\gamma,\theta,\mu) = 0$), the utility is zero. Thus, the offloading decision to maximize the utility for a task of size z with processing density γ and delay parameter θ is obtained from the following problem:

$$\underset{\delta_i(z,\gamma,\theta,\mu)\in\{0,1\}}{\text{maximize}} \ \delta_i(z,\gamma,\theta,\mu)u_i^{\pi_i}(z,\gamma,\theta,\mu). \tag{8}$$

Note that problem (8) is solved upon arrival of each task. However, for the sake of clarity of problem formulation, we have removed the index of the tasks. The optimal offloading decision indicator, denoted as $\delta_i^*(z,\gamma,\theta,\mu)$, can be obtained by solving problem (8) as follows:

$$\delta_i^*(z,\gamma,\theta,\mu) = \begin{cases} 1, & \text{if } u_i^{\pi_i}(z,\gamma,\theta,\mu) \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

This offloading decision rule can be interpreted as follows. When the utility obtained from offloading a task is greater than zero, that task is executed in the cloud servers. Otherwise, the user has no incentive to offload the task and the task will be

[5]To determine the delay improvement for each task, we use the actual service time and mean waiting time experienced by the task for the sake of tractability of analysis. This approach has widely been used for queueing system analysis, e.g., [27].

[6]This unit is commonly used for the prices in MCC systems [19], [20].

[7]Similar to [15], [16], [19], [20], we assume that the charge for transmitting data over LTE does not affect the utility obtained by offloading the tasks as data is typically included in mobile users' plans.

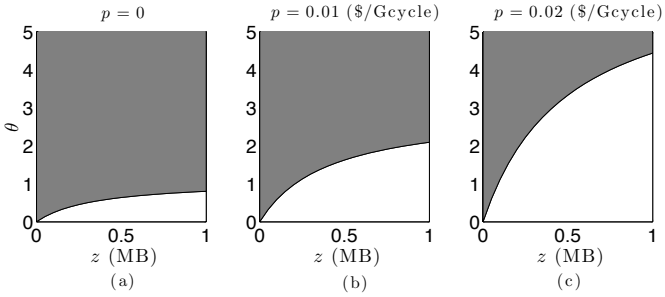


Fig. 2: Offloading region $\mathcal{O}_i$ of user i as shown in the shaded areas. A higher price set by the CSP makes user i's offloading region smaller. Here, we fixed $\gamma = 2 \times 10^3$ and $\mu = 6$ Mbps. The simulation parameters are $C_i = 1.4$ GHz, $\lambda_i^{\mathrm{o}} = 0.03$, $\lambda_i^{\mathrm{c}} = 0.08$, $\lambda_i^{\mathrm{nt}} = 0.01$, $\alpha_i = 100$ J/\$, $\kappa_i = 0.33$, $\varphi_i = 3$, $\varrho_i = 0.1$, and $\beta_i = 2605$ mJ/sec [19], [20].

performed locally. According to this rule, given z, γ, θ, and μ, the expected utility with respect to $\pi_i(z,\gamma,\theta,\mu)$ is

$$\pi_i(z,\gamma,\theta,\mu)u_i^{\pi_i}(z,\gamma,\theta,\mu). \tag{10}$$

Notice that with probability $1-\pi_i(z,\gamma,\theta,\mu)$, the task will not be offloaded and the utility is zero. We now obtain $\pi_i(z,\gamma,\theta,\mu)$ through the following proposition which can be proved based on (8).

Proposition 1. *A task of size z with processing density γ and delay parameter θ when the wireless data rate is μ is offloaded from user i to the CSP (i.e, $\delta_i(z,\gamma,\theta,\mu)=1$) with probability*

- $\pi_i(z,\gamma,\theta,\mu) = 1$, *if $u_i^{\pi_i}(z,\gamma,\theta,\mu) \geq 0$, which happens for those values of (z,γ,θ,μ) that belong to the following set:*

$$\mathcal{O}_i = \{(z,\gamma,\theta,\mu) \in \mathbb{R}_+^4 \mid u_i^{\pi_i}(z,\gamma,\theta,\mu) \geq 0\}.$$

- $\pi_i(z,\gamma,\theta,\mu) = 0$, *otherwise.*

We refer to $\mathcal{O}_i$ as the *offloading region* of mobile user i. The shaded areas in Fig. 2 illustrate $\mathcal{O}_i$ for fixed values of γ and μ and different prices announced by the CSP. When the cloud services are free (i.e., $p=0$), mobile user i evaluates only the energy consumption saving and the delay improvement upon arrival of each task. If task offloading is beneficial for the mobile user, then it submits the task to the cloud servers. Otherwise, the task will be performed locally. Therefore, even if the cloud services are free, the user may not offload the task to the CSP due to energy consumption or delay requirement issues. The offloading region shrinks as the CSP increases its price, and becomes an empty region eventually.

E. Optimal Offloading Probability

We now obtain the optimal offloading probability, denoted by π_i^*, from which we will determine the arrival rate of computing tasks from user i to the cloud servers. By substituting $\pi_i(z,\gamma,\theta,\mu)$ from Proposition 1 into (1), we have

$$\begin{aligned}\pi_i &= \int_{\mathbb{R}_+^4} \pi_i(z,\gamma,\theta,\mu)\mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu) \\ &= \int_{\mathcal{O}_i} \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu). \end{aligned}\tag{11}$$

The offloading probability can be obtained by solving equation (11). The following theorem shows that the offloading probability of user i, given price p, can be uniquely obtained.

Theorem 1. *Given the price $p \geq 0$, the optimal offloading probability of user $i \in \mathcal{M}$ is unique and can be obtained from*

$$\pi_i^*(p) = G_i(\pi_i^*(p)), \tag{12}$$

where

$$G_i(\pi_i) \triangleq \int_{\mathcal{O}_i} \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu).$$

Proof. We first form the function $F_i(\pi_i) = \pi_i - G_i(\pi_i)$ for user $i \in \mathcal{M}$ and investigate its roots. Since $F_i(0) \leq 0$ and $F_i(1) \geq 0$, to show the uniqueness of the root, we need to show that $F_i(\pi_i)$ is strictly increasing. This is equivalent to showing that function $G_i(\pi_i)$ is non-increasing. Assume that $\pi_i^{(2)} > \pi_i^{(1)}$. We rewrite the offloading region by using the utility function given in (7) as follows:

$$\begin{aligned}\mathcal{O}_i^{(2)} &= \{(z,\gamma,\theta,\mu) \in \mathbb{R}_+^4 \mid \\ &\qquad h_i(z,\gamma,\mu) + \theta\tau_i(z,\gamma,\mu,\pi_i^{(2)}) - \alpha_i p\gamma z \geq 0\} \\ &\overset{(a)}{\subseteq} \{(z,\gamma,\theta,\mu) \in \mathbb{R}_+^4 \mid \\ &\qquad h_i(z,\gamma,\mu) + \theta\tau_i(z,\gamma,\mu,\pi_i^{(1)}) - \alpha_i p\gamma z \geq 0\} \\ &= \mathcal{O}_i^{(1)},\end{aligned}$$

where (a) is due to the fact that $\tau_i(z,\gamma,\mu,\pi_i)$ decreases when π_i becomes larger according to Lemma 1. Therefore, region $\mathcal{O}_i$ becomes smaller when we increase π_i, which reduces $G_i(\pi_i)$ as well. Thus, $G_i(\pi_i^{(2)}) \leq G_i(\pi_i^{(1)})$ for any $\pi_i^{(2)} > \pi_i^{(1)}$. As a result, $F_i(\pi_i)$ is strictly increasing while we know that $F_i(0) \leq 0$ and $F_i(1) \geq 0$. Thus, $F_i(\pi_i)$ has a unique root. ■

We now study the properties of the optimal offloading probability to reveal the effect of the CSP's price on the offloading decision in mobile devices. These properties will later be used in Section III to obtain the optimal pricing strategy of the CSP. In the following lemma, we show that for each mobile user $i \in \mathcal{M}$, the offloading probability $\pi_i^*(p)$ is non-increasing in p and approaches 0 when $p \to \infty$.

Lemma 2. *The offloading probability $\pi_i^*(p)$ of user $i \in \mathcal{M}$ is a non-increasing function of p. Moreover, there exists a constant threshold $p_i^{\mathrm{th}} \geq 0$ such that $\pi_i^*(p) = 0$ for any price $p \geq p_i^{\mathrm{th}}$.*

Proof. To prove that $\pi_i^*(p)$ is non-increasing in price p, we show that for any $p^{(2)} > p^{(1)}$, we have $\pi_i^*(p^{(2)}) \leq \pi_i^*(p^{(1)})$. By contradiction, we assume that $\pi_i^*(p^{(2)}) > \pi_i^*(p^{(1)})$. According to (11), we have

$$\pi_i^*(p^{(2)}) = \int_{\mathcal{O}_i^{(2)}} \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu), \tag{13}$$

where

$$\begin{aligned}\mathcal{O}_i^{(2)} &= \{(z,\gamma,\theta,\mu) \in \mathbb{R}_+^4 \mid \\ &\qquad h_i(z,\gamma,\mu) + \theta\tau_i(z,\gamma,\mu,\pi_i^*(p^{(2)})) - \alpha_i p^{(2)}\gamma z \geq 0\} \\ &\overset{(a)}{\subseteq} \{(z,\gamma,\theta,\mu) \in \mathbb{R}_+^4 \mid\end{aligned}$$

$$
\begin{aligned}
&h_i(z,\gamma,\mu)+\theta\tau_i(z,\gamma,\mu,\pi_i^*(p^{(2)}))-\alpha_i p^{(1)}\gamma z\geq 0\}\\
&\overset{(b)}{\subseteq}\{(z,\gamma,\theta,\mu)\in\mathbb{R}_+^4\mid\\
&\quad h_i(z,\gamma,\mu)+\theta\tau_i(z,\gamma,\mu,\pi_i^*(p^{(1)}))-\alpha_i p^{(1)}\gamma z\geq 0\}\\
&=\mathcal{O}_i^{(1)}.
\end{aligned}
$$

Note that (a) is due to $\alpha_i p^{(2)}\gamma z > \alpha_i p^{(1)}\gamma z$ since $p^{(2)} > p^{(1)}$. Moreover, (b) is obtained from the contradiction assumption $\pi_i^*(p^{(2)}) > \pi_i^*(p^{(1)})$, which implies that $\tau_i(z,\gamma,\mu,\pi_i^*(p^{(1)})) > \tau_i(z,\gamma,\mu,\pi_i^*(p^{(2)}))$ due to Lemma 1. Since $\mathcal{O}_i^{(2)}\subseteq\mathcal{O}_i^{(1)}$,

$$
\begin{aligned}
\pi_i^*(p^{(2)})&=\int_{\mathcal{O}_i^{(2)}}\mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu)\\
&\leq\int_{\mathcal{O}_i^{(1)}}\mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu)=\pi_i^*(p^{(1)}),
\end{aligned}\tag{14}
$$

which contradicts the assumption. Therefore, $\pi_i^*(p)$, which can be obtained from (12), is non-increasing in price p. For a given price p, we know

$$
\begin{aligned}
\mathcal{O}_i&\overset{(c)}{\subseteq}\{(z,\gamma,\theta,\mu)\in\mathbb{R}_+^4\mid\\
&\qquad h_i(z,\gamma,\mu)+\theta\tau_i(z,\gamma,\mu,\pi_i=0)-\alpha_i p\gamma z\geq 0\},
\end{aligned}
$$

where (c) is due to the fact that $\tau_i(z,\gamma,\mu,\pi_i)$ is decreasing in π_i as stated in Lemma 1. Since $h_i(z,\gamma,\mu)$ and $\tau_i(z,\gamma,\mu,0)$ do not depend on p, the above set becomes empty for a high price p. Thus, there exists a $p_i^{\mathrm{th}}\geq 0$ such that $\mathcal{O}_i$ becomes an empty set for any $p\geq p_i^{\mathrm{th}}$, which completes the proof. ■

We will later show that the offloading probability is neither convex nor concave in p, which makes the design of an optimal pricing strategy in Section III very challenging.

III. Pricing Strategy of CSP

In this section, we analyze a monopoly market when a CSP is providing cloud services to the mobile users. We first formulate a profit maximization problem to obtain the CSP's optimal pricing strategy. We then propose a pricing algorithm, which we refer to as CoPe, to cope with the non-convexity of the formulated problem. The CSP consists of multiple cloud servers which can provide computing services to the mobile users. Upon arrival of each task, the CSP assigns the task to a cloud server for processing. We refer the cloud server as a virtual machine with dedicated processing power, which can perform one task at a time. The computing service of the CSP is specified by the pair (p, C_{R}). Recall that p is the unit price (in \$/Gcycle) and C_{R} is the processing capacity (in cycles/unit time) of each server. We assume that all cloud servers have the same processing capacity. The CSP announces the unit price p to the mobile users, according to which their task schedulers decide whether or not to offload the tasks.

A. Profit Maximization Problem

The CSP's optimal pricing strategy is determined via a profit maximization framework. We assume that the CSP has sufficient computing resources similar to [32]. As a result, the system can be viewed as a $G/G/\infty$ queuing system with infinite number of queues, each of which is stable. Note that the task arrivals to the cloud servers are not memoryless and cannot be modeled as a Poisson process. According to this $G/G/\infty$ model, the average delay spent in the cloud is the average processing time since there is no waiting delay.

The CSP's profit is the payment received from the users minus the electricity price[8]. Let $N(p)$ denote the number of offloaded tasks being processed in the servers in a time unit given the announced price p. The CSP's profit is

$$
\begin{aligned}
J(p)&=p\mathbb{E}[C_{\mathrm{R}}N(p)]-b\vartheta\mathbb{E}[C_{\mathrm{R}}N(p)]\\
&=(p-b\vartheta)C_{\mathrm{R}}\mathbb{E}[N(p)],
\end{aligned}\tag{15}
$$

where ϑ (in kWh/processing capacity cycle) is the energy consumption coefficient of the servers and b is the electricity price (in \$/kWh) that the CSP has to pay for its active servers. The electricity cost is proportional to the number of active cloud servers that are processing the assigned tasks. A lower price set by the CSP encourages more mobile users to offload their tasks and increases $N(p)$. However, it may reduce the profit of the CSP as the computation jobs offloaded to the cloud servers incur a cost to the CSP. Similarly, less users are interested in the cloud services when the CSP sets a high price. Therefore, the CSP should optimize its pricing strategy, as a static pricing strategy degrades its profit.

To obtain the CSP's profit, we first need to determine the average number of tasks being processed in the system. To do so, we use the sample-path version of Little's law [43]. The long-term average number of customers in service in a queuing system is the product of arrival rate and sojourn time[9]. The arrival rate of tasks offloaded to the cloud servers, which can be determined using the offloading probability of the mobile users, is as follows:

$$
\sum_{i\in\mathcal{M}}\pi_i^*(p)\lambda_i^{\mathrm{o}}.\tag{16}
$$

In addition, the sojourn time equals the processing time that the tasks spend in the cloud servers, since there is no waiting time in the $G/G/\infty$ system. The expected processing time of the tasks arriving from mobile user i is

$$
\frac{\mathbb{E}[z_i^{\mathrm{C}}(p)\gamma_i^{\mathrm{C}}(p)]}{C_{\mathrm{R}}},\tag{17}
$$

where $z_i^{\mathrm{C}}(p)$ denotes the size of the tasks arriving at the cloud servers from user i and $\gamma_i^{\mathrm{C}}(p)$ is the corresponding processing density. Although in mobile users, the size of the tasks, denoted by z, follows the pdf $f_Z(z)$, the offloaded tasks have a different size. The mobile users may perform tasks with small sizes locally, while they may offload large computational tasks to the cloud servers. Therefore, $z_i^{\mathrm{C}}(p)$ in user i does not follow the same distribution as z. Moreover, the price p also affects $z_i^{\mathrm{C}}(p)$. The same statement is valid for $\gamma_i^{\mathrm{C}}(p)$. To obtain the expected processing time for the tasks of user i arriving

[8]Similar to [20], [37], [38], we only consider the electricity price as the other costs such as maintenance cost are constant and do not vary with the CSP's workload.

[9]We assume that the CSP can observe and measure the arrival rate and sojourn time.

at the cloud servers, we use the analysis provided in Section II. We have

$$\mathbb{E}[z_i^{\mathrm{C}}(p)\gamma_i^{\mathrm{C}}(p)] = \int_{\mathbb{R}_+^4} z\gamma \mathrm{d}F^{\mathrm{C}}_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu), \tag{18}$$

where $F^{\mathrm{C}}_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu)$ is the conditional cdf of variables z,γ,θ, and μ given that the task with parameters (z,γ,θ,μ) is offloaded from user i to the cloud servers. Thus,

$$\begin{aligned}\mathbb{E}[z_i^{\mathrm{C}}(p)\gamma_i^{\mathrm{C}}(p)] &= \int_{\mathbb{R}_+^4} z\gamma \frac{\pi_i(z,\gamma,\theta,\mu)}{\pi_i^*(p)} \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu) \\ &= \frac{1}{\pi_i^*(p)} \int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu).\end{aligned} \tag{19}$$

Therefore, the average sojourn time of the offloaded tasks in the cloud servers is [43]

$$\sum_{i\in\mathcal{M}} \frac{\pi_i^*(p)\lambda_i^{\mathrm{o}}}{\sum_{i\in\mathcal{M}}\pi_i^*(p)\lambda_i^{\mathrm{o}}} \frac{\mathbb{E}[z_i^{\mathrm{C}}(p)\gamma_i^{\mathrm{C}}(p)]}{C_{\mathrm{R}}}. \tag{20}$$

According to the Little's law, the average number of tasks being processed in the system is

$$\mathbb{E}[N(p)] = \sum_{i\in\mathcal{M}} \pi_i^*(p)\lambda_i^{\mathrm{o}} \frac{\mathbb{E}[z_i^{\mathrm{C}}(p)\gamma_i^{\mathrm{C}}(p)]}{C_{\mathrm{R}}}. \tag{21}$$

We now determine the profit of the CSP by substituting (21) into (15). The CSP's profit is

$$J(p) = (p - b\vartheta) \sum_{i\in\mathcal{M}} \pi_i^*(p)\lambda_i^{\mathrm{o}} \mathbb{E}[z_i^{\mathrm{C}}(p)\gamma_i^{\mathrm{C}}(p)]. \tag{22}$$

The optimal price can be obtained from the following problem:

$$p^* = \arg\max_{p\geq 0} J(p), \tag{23}$$

which is not a concave maximization problem and cannot be solved directly using standard optimization techniques. We first study the existence of a local solution for problem (23).

Theorem 2. *There exists a local optimal solution for problem (23).*

Proof. Due to Lemma 2, we know that there exists a $p^{\mathrm{th}} = \max_{i\in\mathcal{M}} p_i^{\mathrm{th}}$ such that $\pi_i^* = 0$ for all $i \in \mathcal{M}$. Thus, $J(p) = 0$ for $p \geq p^{\mathrm{th}}$, while we know $J(p) \leq 0$ when $p \leq b\vartheta$. If $J(p)$ is zero for any $p \geq b\vartheta$, then the CSP will not offer any computing services. Otherwise, there exists at least one local maximum point within the interval $[b\vartheta, p^{\mathrm{th}}]$. ■

To obtain the global optimal solution of problem (23), the CSP can conduct an exhaustive search. To overcome the complexity of the exhaustive search, we proceed with a sub-optimal approach and develop a corresponding pricing algorithm. We note that the proposed algorithm achieves the global optimal solution of problem (23) under the settings that we will evaluate in Section IV.

B. Pricing Algorithm using Convexification and Primal-dual Methods

To mitigate the non-convexity of problem (23), we derive the pricing algorithm CoPe by utilizing the convexification method introduced in [44]. We first convexify and transform problem (23) into a new problem which can be solved using primal-dual methods. Since problem (23) is non-convex, using the primal-dual method makes the obtained solution sub-optimal. To convexify problem (23), we introduce p_i as the price set for user i and transform problem (23) into the following equivalent problem:

$$\begin{aligned}\text{P: } \underset{\mathbf{p},\ p\geq 0}{\text{maximize}}\ & J(\mathbf{p}) = \sum_{i\in\mathcal{M}} (p_i - b\vartheta)\pi_i^*(p_i)\lambda_i^{\mathrm{o}}\mathbb{E}[z_i^{\mathrm{C}}(p_i)\gamma_i^{\mathrm{C}}(p_i)] \\ \text{subject to } & p_i - p = 0, \quad \forall i \in \mathcal{M},\end{aligned} \tag{24}$$

where $\mathbf{p} = (p_1, \ldots, p_{|\mathcal{M}|})$. We now consider the following problem by introducing a penalty term:

$$\begin{aligned}&\underset{\mathbf{p},\ p\geq 0,\ \mathbf{q}}{\text{maximize}}\ J(\mathbf{p}) - \frac{\rho}{2}||\mathbf{q} - \mathbf{p}||_2^2 \\ &\text{subject to } p_i - p = 0, \quad \forall i \in \mathcal{M},\end{aligned} \tag{25}$$

where ρ is a fixed scalar and $\mathbf{q} = (q_1, \ldots, q_{|\mathcal{M}|})$ represents a vector of additional variables. Clearly, a vector $\mathbf{p}^*$ is a local optimum of the original problem (24) if and only if ($\mathbf{p}^*$, $\mathbf{q}^* = \mathbf{p}^*$) is a local optimum of problem (25). We now formulate the dual problem of problem (24) as:

$$\text{D: } \underset{\mathbf{q}}{\text{maximize}}\ \phi_\rho(\mathbf{q}), \tag{26}$$

where

$$\phi_\rho(\mathbf{q}) = \sup_{\substack{p_i - p = 0, i\in\mathcal{M} \\ p\geq 0}} J(\mathbf{p}) - \frac{\rho}{2}||\mathbf{q} - \mathbf{p}||_2^2, \tag{27}$$

is the dual function. For ρ sufficiently large, problem (27) has a convex structure. Through the following lemma, we show that there exists a finite ρ that makes problems (25) and (27) concave maximization problems. This guarantees the existence of a dual function which can be obtained from (27).

Lemma 3. *There exists a finite value of ρ such that problem (27) is a concave maximization problem.*

Proof. Please refer to the Appendix. ■

According to Lemma 3, problem (27) has a convex structure and, thus, dual function $\phi_\rho(\mathbf{q})$ can be obtained by means of the gradient method [45]. However, this may not result in a local solution of primal problem (24), if the duality gap is not zero. The duality gap refers to the gap between the optimal value of primal problem (24) and dual problem (26). According to [44], if the second-order condition of concavity is satisfied around a locally optimal solution of problem (24), then strong duality holds. We now propose the pricing algorithm CoPe by solving the primal and dual problems as shown in Algorithm 1. In this algorithm, $\sigma > 0$ is a constant step size. CoPe iteratively updates the dual variables $\mathbf{q}$ until convergence (Lines 4–7) and obtains price $\mathbf{p}$ by solving problem (25) (Line 8). Lemma 3 guarantees the convergence of Algorithm 1 as it states that the dual function exists. Notice that the dual problem is always a

Algorithm 1: Pricing Algorithm using Convexification and Primal-dual Methods (CoPe).

Input: $J(\mathbf{p})$, ϵ, ρ, and σ
Initialization: Randomly initialize $\mathbf{q}^{(0)}$
Initialization: $k \leftarrow 0$
do
/* Dual Update */
  $\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + \sigma \nabla \phi_\rho(\mathbf{q}^{(k)})$
  $k \leftarrow k+1$
while $||\mathbf{q}^{(k)} - \mathbf{q}^{(k-1)}||_1 > \epsilon$
/* Primal Update */
$\mathbf{p} \leftarrow \arg\max_{\substack{p_i=p, i\in\mathcal{M} \\ p\geq 0}} J(\mathbf{p}) - \frac{\rho}{2}||\mathbf{q}^{(k)} - \mathbf{p}||_2^2$
Output: $\mathbf{p}$

concave maximization problem. The price p can be obtained by solving problem (25) using the gradient method as this problem is also a concave maximization problem, when ρ is sufficiently large.

IV. Performance Evaluation

In this section, we first study the offloading strategy of the mobile users by evaluating the performance of the proposed dynamic task scheduler in comparison with ThinkAir [7] and MAUI [8]. We have chosen these two mechanisms since they also take the energy consumption and delay into account and make the offloading decision upon arrival of each task. We then evaluate the profit of the CSP under different workloads. We further study the performance of the CSP's optimal pricing strategy and CoPe, respectively, and compare them to that of static and dynamic pricing strategies.

We consider a CSP that owns cloud computing servers and offers cloud services to mobile users. We assume that the computing power of each cloud server is $C_{\mathrm{R}} = 4$ GHz [46]. We further assume that the electricity price is $b = 0.039$ \$/kWh [47], while the energy consumption coefficient is $\vartheta = 0.05$ kWh/Gcycle [48]. We assume that mobile users employ the dynamic task scheduler proposed in Section II to make the offloading decisions. Without loss of generality, we assume that each mobile device $i \in \mathcal{M}$ has a CPU with clock speed $C_i = 1.4$ GHz [19], [20]. According to the measurement results provided by [20], we set $\kappa_i = 0.33$, $\varphi_i = 3$, $\varrho_i = 0.1$, and $\beta_i = 2605$ mJ/sec to determine the energy consumption saving and delay improvement. We further set $\alpha_i = 100$ J/\$. Since the average uplink data rate measured in [19] is 5.85 Mbps, we assume that μ follows a uniform distribution in $[4.85, 6.85]$ Mbps. To generate computing jobs, unless stated otherwise, we assume that the task sizes are uniformly distributed in $[100$ B$, 1$ MB$]$, while the processing density γ is uniformly distributed in the interval $[100, 3000]$ cycles/bit. Processing densities of different applications are reported in [7]–[9]. Notice that the proposed task scheduler can be applied to any pdfs of task sizes and processing densities and any data rate distributions of the wireless interfaces. We further assume that jobs arrive at the mobile users according to independent Poisson processes with rates $\lambda_i^{\mathrm{o}} = 0.04$, $\lambda_i^{\mathrm{c}} = 0.08$, and $\lambda_i^{\mathrm{nt}} = 0.02$, for all $i \in \mathcal{M}$, unless otherwise stated.

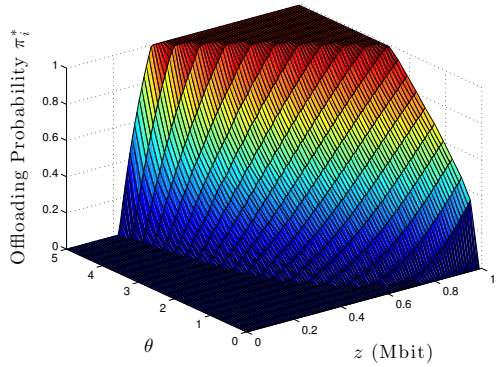


Fig. 3: The optimal offloading probability π_i^* of user i versus task size z and delay parameter θ. Tasks with a larger size z or a higher value of θ will be offloaded to the cloud servers with a higher probability.

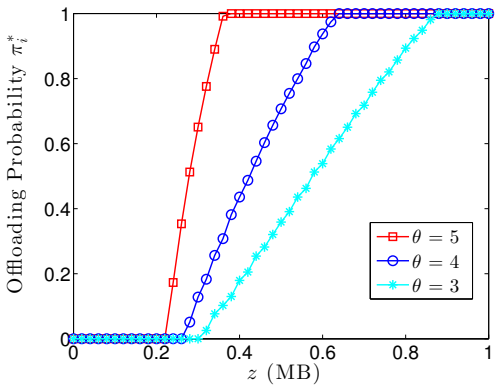


Fig. 4: The optimal offloading probability π_i^* of user i versus task size z. Here, we fixed the value of θ and evaluate the offloading probability for $\theta = 3, 4$, and 5.

A. Dynamic Scheduler

We first study the dynamic scheduler and its offloading probability for a mobile user. Fig. 3 illustrates the offloading probability π_i^* for tasks with different sizes and different values of θ, when the CSP has set a price of $p = 0.03$ \$/Gcycle. For small values of task size z and delay parameter θ, the mobile user prefers to run the application locally. However, for larger tasks, the mobile user is more willing to offload the tasks to the cloud servers to save energy. Moreover, for larger θ, a task is less tolerable to delay. Thus, the mobile user offloads the task to the cloud servers to expedite the execution of the task. The offloading probability eventually approaches 1 in these cases.

To further study how the tasks size affects the offloading decision, we evaluate the offloading probability of a mobile user as a function of z. Fig. 4 shows the offloading probability versus task size z for different values of θ. We assume that $p = 0.03$ \$/Gcycle and fix the value of θ for each experiment. Similar to Fig. 3, the mobile user is not interested in offloading tasks having small sizes. Moreover, the offloading probability increases when the size of the tasks becomes larger. The offloading probability grows faster in z for tasks that are more sensitive to delay (i.e., for larger values of θ). This is because for delay-sensitive tasks, the offloading decision is

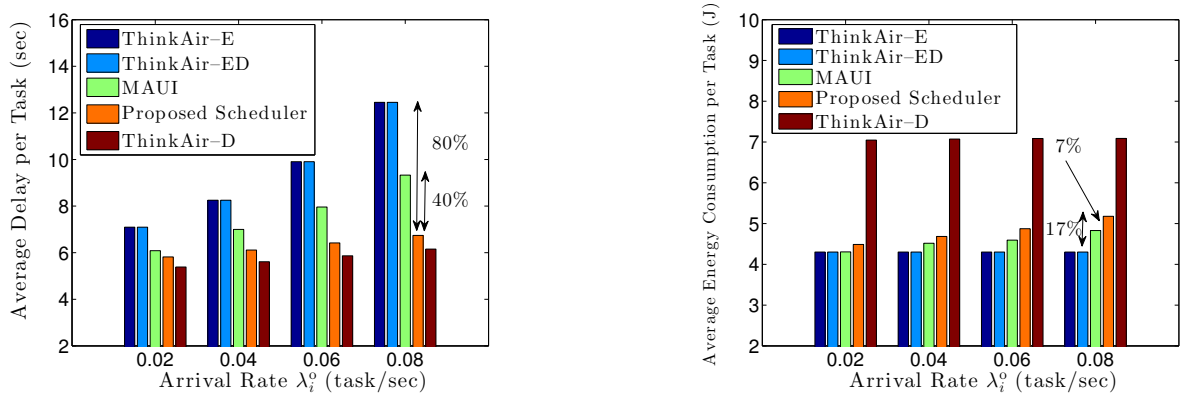


Fig. 5: Comparing the proposed task scheduler in mobile user i with the mechanisms proposed in [7] and [8]. Here, we fixed $\mu = 2$ Mbps and assumed that θ is uniformly distributed in $[0, 5]$. (a) The average delay per task versus different arrival rates of the computing tasks. (b) The corresponding average energy consumption per task.

mainly affected by the delay improvement, whereas the energy consumption saving contributes less to the user's utility.

We now compare the performance between our proposed task scheduler and three task scheduling policies proposed in [7] as well as an energy-delay aware mechanism proposed in [8], which is referred as MAUI. In [7], the first policy prioritizes energy conservation when offloading and offloads the tasks if the energy consumption is expected to improve. We refer this policy as ThinkAir–E. The second policy, referred as ThinkAir–D, optimizes the offloading decision in order to expedite the execution of the tasks. The third policy is based on an energy-delay aware offloading mechanism. The computing tasks will be offloaded only if both the energy consumption and the execution time are expected to improve. We refer this policy as ThinkAir–ED. We further compare our proposed task scheduler with MAUI. To make an offloading decision, the MAUI solver aims to minimize the mobile device's energy consumption subject to a latency constraint. The solver ensures that the total delay experienced by each task does not exceed an application-dependent constant delay, denoted by L. In order to compare our proposed scheduler with MAUI and study the tradeoff between the average delay and average energy consumption, we set $L = 1/\theta$ for each application. We also assume that the cloud services are free (i.e., $p = 0$) in order to compare our proposed scheduler with ThinkAir and MAUI in a fair manner. Figs. 5(a) and 5(b) illustrate the average delay per task for different schemes and the corresponding average energy consumption per task. As can be observed, ThinkAir–E results in a low energy consumption but long delays for the execution of tasks. On the other hand, ThinkAir–D can achieve the fastest task execution, while consuming a large amount of energy. ThinkAir–ED performs similar to ThinkAir–E as the delay improvement is positive in this setting. However, both our proposed task scheduler and MAUI address the tradeoff between delay and energy consumption for different types of tasks. The proposed scheduler consumes slightly more energy to address the delay requirements of delay-sensitive applications. From Fig. 5, we can see that our proposed scheduler reduces the delay by 80% and 40% compared to ThinkAir–E and MAUI, while it only consumes 17% and 7% more energy, respectively.

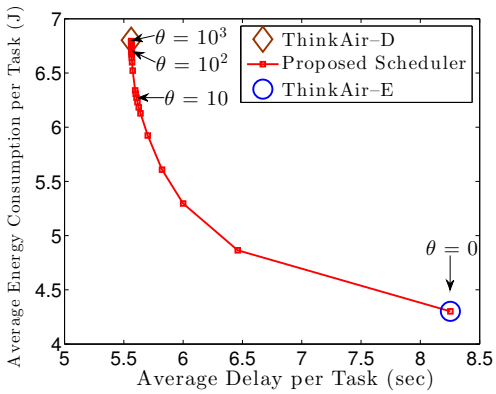


Fig. 6: The tradeoff between the energy consumption and delay in the proposed scheduler in comparison with different scheduling policies proposed in [7]. We vary θ from 0 to 10^3 to study different types of applications.

We further investigate the tradeoff between the energy consumption and delay for different delay requirements of applications. We compare the proposed task scheduler and ThinkAir for price $p = 0$. Fig. 6 shows the average energy consumption per task and the corresponding average delay when we vary the delay parameter θ from 0 to 10^3. A higher delay-sensitivity of tasks (i.e., a larger θ) increases the energy consumption to expedite the execution of the tasks. On the other hand, the proposed task scheduler trades delay for the sake of energy saving for delay-tolerant applications. The proposed scheduler achieves the same performance as ThinkAir–E for delay-tolerant tasks (i.e., $\theta = 0$). Notice that when $\theta = 0$, the proposed task scheduler makes the offloading decision by evaluating only the energy consumption saving. Moreover, the proposed scheduler behaves similar to ThinkAir–D when θ is very large.

Finally, we study the behavior of the dynamic scheduler as a function of the price imposed by the cloud computing services, when θ is uniformly distributed in $[0, 5]$. Fig. 7 illustrates the offloading probability versus price p set by the CSP for different arrival rates of the computing tasks. As

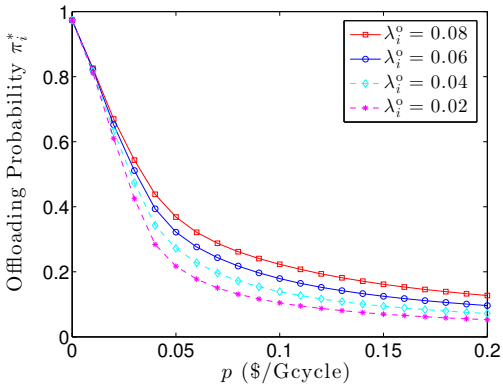


Fig. 7: The optimal offloading probability π_i^* of user i versus price p for different arrival rates of the computing tasks. The mobile user is more interested in offloading the tasks, when its CPU is busy due to a higher arrival rate of the computing tasks.

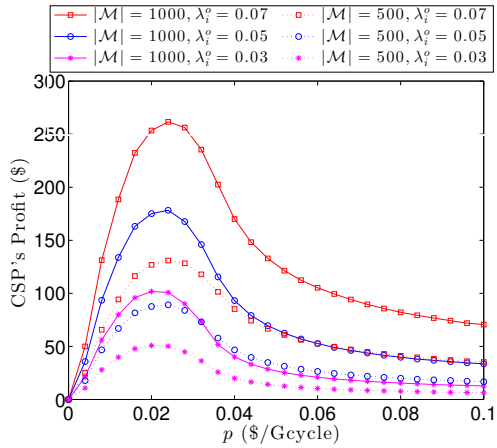


Fig. 8: The profit of the CSP obtained within a 1-minute interval versus price p. The optimal pricing strategy is the price that maximizes the profit shown in this figure.

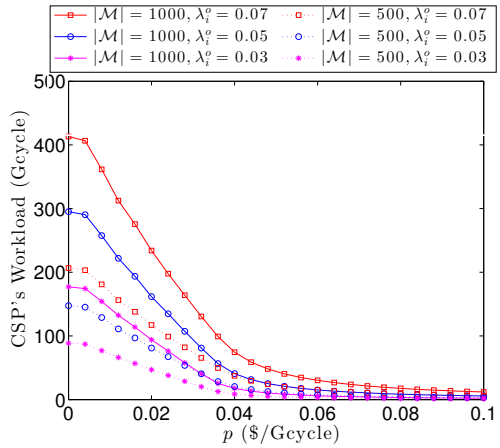


Fig. 9: The workload of the CSP (in Gcycles) versus price p. As expected, fewer computing tasks arrive at the cloud servers, if the CSP sets a high price.

can be observed, the mobile user is more likely to offload its computing tasks to the cloud servers when the CSP sets a low price. Increasing the price reduces the interest of the mobile user in offloading its tasks to the cloud servers. The offloading probability approaches 0, when the CSP sets a high price as stated in Lemma 2. This is because the offloading region, as illustrated in Fig. 2, shrinks fast when p increases. Fig. 7 also shows that for higher arrival rates of the computing tasks, the user is more willing to offload its tasks to the cloud servers to reduce the queuing delay of the CPU and to expedite the execution of the tasks.

B. CSP's Pricing Strategy

In this subsection, we focus on the CSP and study the CSP's profit obtained from providing cloud computing services. We assume that the set $\mathcal{M}$ of mobile users have computing tasks and may offload them to the cloud servers.

Fig. 8 illustrates the profit of the CSP, which is given by $\max(J(p), 0)$ as a function of the price p. Obviously, for very low prices, the payment received from the mobile users is less than the cost incurred to the CSP. Therefore, the CSP's profit is zero as it does not accept any computing tasks for such low prices. When price $p > b\vartheta$, the CSP can make a profit. As the price increases, the total arrival rate of computing tasks at the cloud servers (i.e., the workload of the cloud servers) reduces. However, as can be observed from Fig. 8, the CSP's profit first increases, but eventually decreases again when the CSP further increases the price. This confirms that by employing the optimal pricing strategy, the CSP can achieve the maximum possible profit. For example, when there are $|\mathcal{M}| = 500$ mobile users in the system with $\lambda_i^o = 0.05$, the CSP's profit is maximized if it sets $p = 0.02$ \$/Gcycle. Lower prices than the optimal price encourage the mobile users to offload more tasks to the cloud servers. Nevertheless, the CSP's profit will be decreased due to the lower payment received from the mobile users. Similarly, prices higher than the optimal price reduce the offloading demand and degrade the profit of the CSP.

We now investigate the CSP's workload, which is the amount of total computing tasks in Gcycles offloaded to the cloud servers in a unit of time. Fig. 9 shows the CSP's workload versus price p for different numbers of mobile users and different computing task arrival rates. The cloud servers receive a high workload of computing tasks when the price of cloud services is low. The workload of the cloud servers is reduced when the CSP sets a higher price. This reveals a tradeoff between the price and the interest of the users in offloading their tasks to the cloud servers.

To study the proposed pricing algorithm CoPe, we first evaluate its convergence for $\lambda_i^o = 0.05$, $\epsilon = 10^{-11}$, $\sigma = 5 \times 10^{-11}$, and $\rho = \max_{p \geq 0} \frac{\mathrm{d}^2 J(p)}{\mathrm{d}p^2} + 1$. Fig. 10 shows $q_i^{(k)}$ for 10 mobile users in different iterations. When the algorithm converges, each $q_i, i \in \mathcal{M}$, is equal to p. Thus, the price p can be obtained from $q_i^{(k)}$, when k approaches infinity. As shown in Fig. 10, all $q_i^{(k)}$ quickly converge to the same value, which confirms the fast convergence rate of CoPe.

Next, we evaluate the performance of the proposed optimal pricing strategy and CoPe in comparison with static and dynamic pricing strategies. We first consider an MCC system with $|\mathcal{M}| = 500$ mobile users with arrival rate $\lambda_i^o = 0.01$ and obtain the optimal price that maximizes the CSP's profit. We consider this price as the price in the static approach. In the dynamic pricing strategy, we assume that the price of cloud

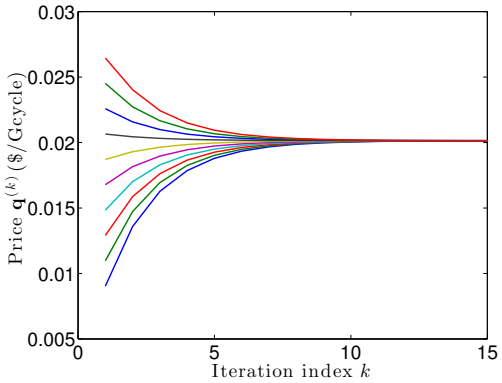


Fig. 10: The price $q_i^{(k)}$ obtained from CoPe in different iterations for 10 users. Results show that CoPe proposed in Algorithm 1 converges quickly.

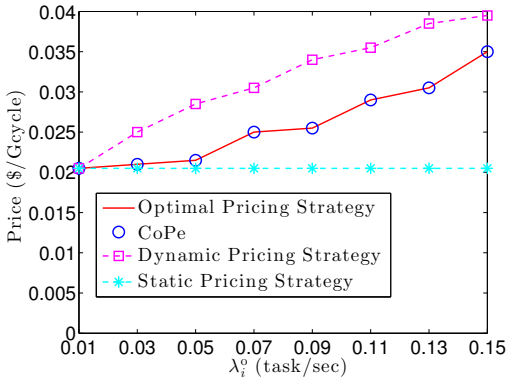


Fig. 11: The price of the cloud services determined by the proposed optimal pricing strategy and CoPe in comparison with static and dynamic pricing strategies. Results show that a larger offloading demand arriving at the cloud servers drives the price of cloud services up. Thus, a static pricing strategy is not suitable for dynamic MCC systems.

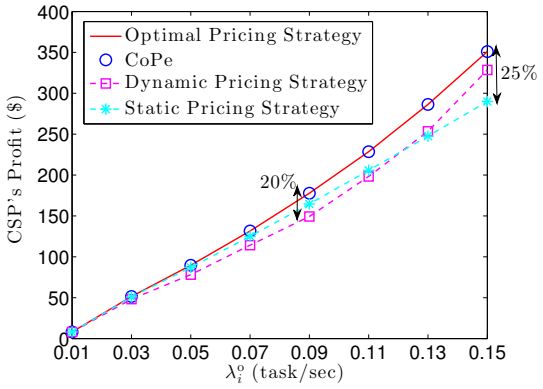


Fig. 12: The profit of the CSP. The proposed pricing strategies outperform the static and dynamic pricing strategies by up to 25% and 20%, respectively.

services changes proportional to the CSP's workload. Fig. 11 illustrates the prices obtained from the optimal pricing strategy and CoPe, respectively, and the prices set by the static and dynamic pricing strategies. The optimal pricing strategy is the price that maximizes the CSP's profit and can be obtained from problem (23). Results show that CoPe yields the same prices as the optimal pricing strategy. This is due to the following reason. The profit function, $J(p)$, is not a concave function in price p as shown in Fig. 8 and discussed in Section III. However, from Fig. 8, we can see that there is an interval in which the profit function is concave and the optimal price lies in this interval. Thus, problem (23) admits a unique optimal solution. Moreover, due to the concavity of $J(p)$ at this point, strong duality holds [44] and the proposed algorithm achieves the global optimal solution of problem (23). Fig. 11 also shows that a higher offloading demand of mobile users caused by more resource hungry applications drives the price set by the CSP up.

Fig. 12 illustrates the CSP's profit obtained via the optimal pricing strategy and CoPe in comparison with the static and dynamic pricing strategies. The results show that the CSP's profit improves when the arrival rate of the computing tasks increases. The proposed pricing strategies both outperform the static and dynamic approaches by up to 25% and 20%, respectively. This reveals that a static pricing strategy cannot capture the dynamics of an MCC system. Furthermore, although the dynamic pricing strategy chooses the prices based on the CSP's workload, it does not consider the effect of the prices on the mobile users' strategies. Notice that our proposed strategies determine the price of cloud services by considering how the prices affect the offloading strategies of mobile users.

V. Conclusion

In this paper, we studied an MCC market to design a dynamic task scheduler for mobile devices and to obtain the pricing strategy of the CSP. We first proposed a task scheduler by taking the energy consumption, delay, and the price of cloud services into account. A queuing delay analysis allowed us to account for both delay-sensitive and delay-tolerant applications. To obtain the CSP's optimal pricing strategy, we then formulated a profit maximization problem, which is non-convex in general. To mitigate the non-convexity, we further proposed the pricing algorithm CoPe. Through numerical experiments, we showed that the proposed task scheduler outperforms task scheduling policies proposed in the literature in terms of energy consumption and delay. We further evaluated the proposed optimal pricing strategy and CoPe. Our results show that the CSP can obtain significantly more profit by employing either the optimal pricing strategy or CoPe in comparison with static and dynamic pricing strategies. In future work, we will consider dynamic voltage scaling to better utilize the computing resources of the mobile users' CPUs. In addition, we will jointly consider the allocation of computing services and wireless channel bandwidth in mobile-edge computing systems where the network operators install the cloud servers within their radio access networks. This is promising for future fifth-generation wireless networks.

Appendix: Proof of Lemma 3

To prove that there exists a finite value of ρ such that problem (27) is a concave maximization problem, we first rewrite the objective function as

$$\sum_{i\in\mathcal{M}} J_i(p_i) - \frac{\rho}{2}(q_i - p_i)^2, \tag{28}$$

where

$$J_i(p_i) \triangleq (p_i - b\vartheta)\pi_i^*(p_i)\lambda_i^{\mathrm{o}}\mathbb{E}[\gamma_i^{\mathrm{C}}(p_i)z_i^{\mathrm{C}}(p_i)] = (p_i - b\vartheta)\lambda_i^{\mathrm{o}}\int_{\mathcal{O}_i(p_i)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu). \quad (29)$$

It suffices to show that the second-order derivative of $J_i(p)$ is bounded above. In this case, for any $\rho > \sum_{i\in\mathcal{M}} \frac{\mathrm{d}^2 J_i}{\mathrm{d}p^2}$, the primal problem (23) is convex. The second-order derivative of $J_i(p)$ is

$$\frac{\mathrm{d}^2 J_i(p)}{\mathrm{d}p^2} = 2\lambda_i^{\mathrm{o}}\frac{\mathrm{d}}{\mathrm{d}p}\int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu) + (p - b\vartheta)\lambda_i^{\mathrm{o}}\frac{\mathrm{d}^2}{\mathrm{d}p^2}\int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu).$$

The first term of $\frac{\mathrm{d}^2 J_i(p)}{\mathrm{d}p^2}$ is always non-positive since $\mathcal{O}_i(p)$ shrinks when p increases. Thus, $\frac{\mathrm{d}^2 J_i(p)}{\mathrm{d}p^2}$ is bounded above if the second term is bounded. Since $p \le p_i^{\mathrm{th}}$ due to Lemma 2, we show that the integral term is bounded above. Given p, we first rewrite the offloading region by using (3) and (6):

$$\mathcal{O}_i(p) = \Big\{(z,\gamma,\theta,\mu) \in \mathbb{R}_+^4 \mid (h_{i,1} - \alpha_i p + \theta h_{i,2})z\gamma - (\beta_i + \theta)\frac{z}{\mu} + \theta w_i(\pi_i) \ge 0\Big\},$$

where $h_{i,1}$ and $h_{i,2}$ are user-dependent constants. Moreover, we define

$$w_i(\pi_i) \triangleq \mathbb{E}[w_{i,\mathrm{L}}(\pi_i)] - \mathbb{E}[w_{i,\mathrm{R}}(\pi_i)] = \frac{((1-\pi_i)\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{c}})\mathbb{E}[s_{i,\mathrm{L}}^2]}{2\left(1 - ((1-\pi_i)\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{c}})\mathbb{E}[s_{i,\mathrm{L}}]\right)} - \frac{(\pi_i\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}})\mathbb{E}[s_{i,\mathrm{R}}^2]}{2\left(1 - (\pi_i\lambda_i^{\mathrm{o}} + \lambda_i^{\mathrm{nt}})\mathbb{E}[s_{i,\mathrm{R}}]\right)}.$$

We derive the second-order derivative of $\int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu)$ utilizing the Reynolds transport theorem [49].

Lemma 4 (*Reynolds Transport Theorem* [49])**.** *Let $g(\mathbf{x},t)$ be a function such that the partial derivative of g with respect to t exists, and is continuous over region $\Omega(t)$. Then,*

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\int_{\Omega(t)} g(\mathbf{x},t)\mathrm{d}\mathbf{V}\right] = \int_{\Omega(t)} \frac{\partial}{\partial t} g(\mathbf{x},t)\mathrm{d}\mathbf{V} + \int_{\partial\Omega(t)} \left(\mathbf{v}^b \cdot \mathbf{n}\right) g(\mathbf{x},t)\mathrm{d}\mathbf{A},$$

where $\partial\Omega(t)$ is the region boundary and $\mathrm{d}\mathbf{V}$ and $\mathrm{d}\mathbf{A}$ are volume and surface elements at $\mathbf{x}$. In addition, $\mathbf{v}^b(\mathbf{x},t)$ is the velocity of the area element and $\mathbf{n}(\mathbf{x},t)$ is the outward-pointing normal, where $\mathbf{v}^b \cdot \mathbf{n}$ represents their inner product.

Utilizing the Reynolds transport theorem, we have

$$\frac{\mathrm{d}}{\mathrm{d}p}\int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu) = \int_{\partial\mathcal{O}_i(p)} z\gamma\left(\mathbf{v}_1^b \cdot \mathbf{n}_1\right) f_\Gamma(\gamma)\mathrm{d}F_{Z,\Theta,M_i}(z,\theta,\mu). \quad (30)$$

The boundary region $\partial\mathcal{O}_i$ is defined by

$$(h_{i,1} - \alpha_i p + \theta h_{i,2})\gamma z - (\beta_i + \theta)\frac{z}{\mu} + \theta w_i(\pi_i) = 0, \quad (31)$$

according to which we obtain

$$\gamma(z,\theta,\mu,p) = \frac{(\beta_i + \theta)\frac{z}{\mu} - \theta w_i(\pi_i)}{(h_{i,1} - \alpha_i p + \theta h_{i,2})z}, \quad (32)$$

$$\partial\mathcal{O}_i(p) = \{(z,\theta,\mu) \in \mathbb{R}_+^3 \mid \gamma(z,\theta,\mu,p) \ge 0\}. \quad (33)$$

Moreover, we have

$$\mathbf{v}_1^b = \frac{\partial}{\partial p}(z,\gamma,\theta,\mu) = \left(0, \frac{\partial\gamma}{\partial p}, 0, 0\right), \quad (34)$$

$$\mathbf{n}_1 = \frac{\partial}{\partial z}(z,\gamma,\theta,\mu) \times \frac{\partial}{\partial\theta}(z,\gamma,\theta,\mu) = \left(1, \frac{\partial\gamma}{\partial z}, 0, 0\right) \times \left(0, \frac{\partial\gamma}{\partial\theta}, 1, 0\right), \quad (35)$$

where $\times$ denotes the cross product. By substituting (33)–(35) into (30), we have

$$\frac{\mathrm{d}}{\mathrm{d}p}\int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu) = \int_{\partial\mathcal{O}_i(p)} -z\gamma\frac{\partial\gamma}{\partial p} f_\Gamma(\gamma)\mathrm{d}F_{Z,\Theta,M_i}(z,\theta,\mu), \quad (36)$$

where $F_{Z,\Theta,M_i}(z,\theta,\mu)$ is the joint cdf of random variables z, θ, and μ. By following the same approach, we obtain the second-order derivative of $\int_{\mathcal{O}_i(p)} z\gamma \mathrm{d}F_{Z,\Gamma,\Theta,M_i}(z,\gamma,\theta,\mu)$ as follows.

$$-\int_{\partial\mathcal{O}_i(p)} z\frac{\partial}{\partial p}\left(\gamma\frac{\partial\gamma}{\partial p} f_\Gamma(\gamma)\right)\mathrm{d}F_{Z,\Theta,M_i}(z,\theta,\mu) - \int_{\partial\partial\mathcal{O}_i(p)} z\gamma\left(\mathbf{v}_2^b \cdot \mathbf{n}_2\right) f_\Gamma(\gamma) f_Z(z)\mathrm{d}F_{\Theta,M_i}(\theta,\mu), \quad (37)$$

where $F_{\Theta,M_i}(\theta,\mu)$ is the joint cdf of random variables θ and μ and $\partial\partial\mathcal{O}_i(p)$ is the region where $\gamma(z,\theta,\mu,p) = 0$, which is equivalent to

$$\partial\partial\mathcal{O}_i(p) = \left\{(\theta,\mu) \in \mathbb{R}_+^2 \mid z(\theta,\mu,p) = \frac{\theta\mu w_i(\pi_i)}{\beta_i + \theta} \ge 0\right\}. \quad (38)$$

In addition,

$$\mathbf{v}_2^b \cdot \mathbf{n}_2 = \left(\frac{\partial z}{\partial p}, 0, 0\right) \cdot \left(\frac{\partial z}{\partial\theta}, 1, 0\right) = \frac{\partial z}{\partial p}\frac{\partial z}{\partial\theta}. \quad (39)$$

According to (38), $\frac{\partial z}{\partial\theta} > 0$. Moreover, $\frac{\partial z}{\partial p} \ge 0$ since $\frac{\partial w_i}{\partial p} = \frac{\partial w_i}{\partial\pi_i}\frac{\partial\pi_i}{\partial p} \ge 0$. Therefore, the second term of (37) is non-positive. In order to show that (37) is bounded above, it is sufficient to prove that its first term is bounded above. We rewrite the first term of (37) as follows.

$$-\int_{\partial\mathcal{O}_i(p)} z\gamma\left(\gamma\frac{\partial^2\gamma}{\partial p^2} f_\Gamma(\gamma) + \gamma\frac{\partial\gamma}{\partial p} f_\Gamma'(\gamma)\right)\mathrm{d}F_{Z,\Theta,M_i}(z,\theta,\mu) - \int_{\partial\mathcal{O}_i(p)} z\left(\frac{\partial\gamma}{\partial p}\right)^2 f_\Gamma(\gamma)\mathrm{d}F_{Z,\Theta,M_i}(z,\theta,\mu). \quad (40)$$

We know that γ, $\frac{\partial\gamma}{\partial p}$, and $\frac{\partial^2\gamma}{\partial p^2}$ approach infinity if the denominator on the right hand side of (32) tends to 0. However, in

this case, $f_\Gamma(\gamma)$ and $f'_\Gamma(\gamma)$ are zero for infinite values of γ. Moreover, since we assume that both the CPU and the wireless interface queues are stable, $\frac{\mathrm{d}w_i}{\mathrm{d}p} = \frac{\mathrm{d}w_i}{\mathrm{d}\pi_i}\frac{\mathrm{d}\pi_i}{\mathrm{d}p}$ is bounded. Thus, (40) is bounded above, which completes the proof.

REFERENCES

[1] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[2] C. Yong, X. Ma, H. Wan, and I. Stojmenovic, "A survey of energy efficient wireless transmission and modeling in mobile cloud computing," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 148–155, Apr. 2013.

[3] Y. Xu and S. Mao, "A survey of mobile cloud computing for rich media applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 46–53, Jun. 2013.

[4] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 369–392, First Quarter 2014.

[5] A. Khan, M. Othman, S. Madani, and S. Khan, "A survey of mobile cloud computing application models," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 393–413, First Quarter 2014.

[6] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.

[7] 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. of IEEE INFOCOM*, Orlando, FL, Mar. 2012.

[8] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. of ACM Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, Jun. 2010.

[9] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. of ACM Conf. on Computer Systems (EuroSys)*, Salzburg, Austria, Apr. 2011.

[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[11] Cisco, "Cisco visual networking index (VNI) mobile forecast projects nearly 10-fold global mobile data traffic growth over next five years," Feb. 2015. [Online]. Available: https://newsroom.cisco.com/press-release-content?articleId=1578507

[12] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. of ACM MobiSys*, Bethesda, ML, Jun. 2011.

[13] 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, Sept. 2013.

[14] S. Chen, Y. Wang, and M. Pedram, "A semi-Markovian decision process based control method for offloading tasks from mobile devices to the cloud," in *Proc. of IEEE GLOBECOM*, Atlanta, GA, Dec. 2013.

[15] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014.

[16] M. Nir, A. Matrawy, and M. St-Hilaire, "An energy optimizing scheduler for mobile cloud computing environments," in *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014.

[17] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Comm.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[18] W. Zhang and Y. Wen, "Energy-efficient task execution for application as a general topology in mobile cloud computing," accepted for publication in *IEEE Trans. Cloud Computing*, 2015.

[19] 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.

[20] Y. Kim, J. Kwak, and S. Chong, "Dual-side dynamic controls for cost minimization in mobile cloud computing systems," in *Proc. of WiOpt*, Mumbai, India, May 2015.

[21] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. of IEEE ICC*, Kuala Lumpur, Malaysia, May 2016.

[22] H. Al-Shatri, S. Muller, and A. Klein, "Distributed algorithm for energy efficient multi-hop computation offloading," in *Proc. of IEEE ICC*, Kuala Lumpur, Malaysia, May 2016.

[23] L. Tong and W. Gao, "Application-aware traffic scheduling for workload offloading in mobile clouds," in *Proc. of IEEE INFOCOM*, San Francisco, CA, Apr. 2016.

[24] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. of IEEE INFOCOM*, San Francisco, CA, Apr. 2016.

[25] F. Liu, P. Shu, and J. C. S. Lui, "AppATP: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Trans. Computers*, vol. 64, no. 11, pp. 3051–3063, Nov. 2015.

[26] T. Zhang, X. Zhang, F. Liu, H. Leng, Q. Yu, and G. Liang, "eTrain: Making wasted energy useful by utilizing heartbeats for mobile data transmissions," in *Proc. of IEEE Int'l Conf. on Distributed Computing Systems (ICDCS)*, Columbus, OH, Jun. 2015.

[27] N. Tran, C. S. Hong, Z. Han, and S. Lee, "Optimal pricing effect on equilibrium behaviors of delay-sensitive users in cognitive radio networks," *IEEE J. Select. Areas Commun.*, vol. 31, no. 11, pp. 2566–2579, Nov. 2013.

[28] H. Shah-Mansouri and V. W.S. Wong, "Profit maximization in mobile crowdsourcing: A truthful auction mechanism," in *Proc. of IEEE ICC*, London, UK, Jun. 2015.

[29] B. Song, H. Shah-Mansouri, and V. W.S. Wong, "Quality of sensing aware budget feasible mechanism for mobile crowdsensing," accepted for publication in *IEEE Trans. Wireless Commun.*, 2016.

[30] K. Wang, F. C. M. Lau, L. Chen, and R. Schober, "Pricing mobile data offloading: A distributed market framework," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 913–927, Feb. 2016.

[31] H. Shah-Mansouri, V. W.S. Wong, and J. Huang, "An incentive framework for mobile data offloading market under price competition," accepted for publication in *IEEE Trans. Mobile Computing*, 2017.

[32] I. Menache, A. Ozdaglar, and N. Shimkin, "Socially optimal pricing of cloud computing resources," in *Proc. of ACM ICST Conf. on Performance Evaluation Methodologies and Tools*, Paris, France, May 2011.

[33] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014.

[34] X. Wang, X. Wang, H. Che, K. Li, M. Huang, and C. Gao, "An intelligent economic approach for dynamic resource allocation in cloud services," *IEEE Trans. Cloud Computing*, vol. 3, no. 3, pp. 275–289, Jul. 2015.

[35] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *IEEE/ACM Trans. Networking*, vol. 24, no. 4, pp. 2060–2073, Aug. 2016.

[36] G. V. Prasad, A. S. Prasad, and S. Rao, "A combinatorial auction mechanism for multiple resource procurement in cloud computing," accepted for publication in *IEEE Trans. Cloud Computing*, 2016.

[37] Z. Guan and T. Melodia, "The value of cooperation: Minimizing user costs in multi-broker mobile cloud computing networks," accepted for publication in *IEEE Trans. Cloud Computing*, 2015.

[38] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Computing*, vol. 1, no. 2, pp. 158–171, Jul. 2013.

[39] W. Zhang, Y. Wen, and X. Zhang, "Towards virus scanning as a service in mobile cloud computing: Energy-efficient dispatching policy under N-version protection," accepted for publication in *IEEE Trans. on Emerging Topics in Computing*, 2016.

[40] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. of ACM MobiSys*, San Francisco, CA, Jun. 2010.

[41] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *Proc. of IEEE INFOCOM*, San Diego, CA, Mar. 2010.

[42] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Dynamic speed scaling for energy minimization in delay-tolerant smartphone applications," in *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014.

[43] D. P. Bertsekas and R. G. Gallager, *Data Networks*, 2nd ed. Prentice-Hall Inc., 1992.

[44] D. P. Bertsekas, "Convexification procedures and decomposition methods for nonconvex optimization problems," *Journal of Optimization Theory and Applications*, vol. 29, no. 2, pp. 169–197, Oct. 1979.

[45] S. Boyd, "Convex Optimization I," *Lecture Notes*. [Online]. Available: http://web.stanford.edu/class/ee364a/index.html

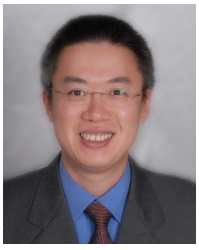
[46] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[47] U.S. Energy Information Administration, "Wholesale electricity and natural gas market data," Aug. 2016. [Online]. Available: http://www.eia.gov/electricity/wholesale/

[48] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," in *Proc. of ACM SIGMETRICS*, Pittsburgh, PA, Jun. 2013.

[49] H. Flanders, "Differentiation under the integral sign," *The American Mathematical Monthly*, vol. 80, no. 6, pp. 615–627, Jun. 1973.

Hamed Shah-Mansouri (S'06, M'14) received the B.Sc., M.Sc., and Ph.D. degrees (Hons.) from Sharif University of Technology, Tehran, Iran, in 2005, 2007, and 2012, respectively all in electrical engineering. From 2012 to 2013, he was with Parman Co., Tehran, Iran. Currently, Dr. Shah-Mansouri is a Post-doctoral Research and Teaching Fellow at the University of British Columbia, Vancouver, Canada. His research interests are in the area of stochastic analysis, optimization and game theory and their applications in economics of cellular networks and mobile cloud computing systems. He has served as the publication co-chair for the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) 2016 and as the technical program committee (TPC) member for several conferences including the IEEE Globecom'15, IEEE VTC–Fall ('16, '17), and IEEE PIMRC'17.



Vincent W.S. Wong (S'94, M'00, SM'07, F'16) received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1994, the M.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the Ph.D. degree from the University of British Columbia (UBC), Vancouver, BC, Canada, in 2000. From 2000 to 2001, he worked as a systems engineer at PMC-Sierra Inc. (now Microsemi). He joined the Department of Electrical and Computer Engineering at UBC in 2002 and is currently a Professor. His research areas include protocol design, optimization, and resource management of communication networks, with applications to wireless networks, smart grid, mobile cloud computing, and Internet of Things. Dr. Wong is an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS. He has served as a Guest Editor of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS and IEEE WIRELESS COMMUNICATIONS. He has also served on the editorial boards of IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and *Journal of Communications and Networks*. He was a Technical Program Co-chair of IEEE SmartGridComm'14, as well as a Symposium Co-chair of IEEE SmartGridComm ('13, '17) and IEEE Globecom'13. He is the Chair of the IEEE Communications Society Emerging Technical Sub-Committee on Smart Grid Communications and the IEEE Vancouver Joint Communications Chapter. He received the 2014 UBC Killam Faculty Research Fellowship.

Robert Schober (S'98, M'01, SM'08, F'10) was born in Neuendettelsau, Germany, in 1971. He received the Diplom (Univ.) and the Ph.D. degrees in electrical engineering from Friedrich-Alexander Universität Erlangen-Nürnberg (FAU), Erlangen, Germany, in 1997 and 2000, respectively. From 2001 to 2002, he was a Post-Doctoral Fellow with the University of Toronto, Canada, sponsored by the German Academic Exchange Service (DAAD). From 2002 to 2011, he was a Professor and Canada Research Chair with The University of British Columbia (UBC), Vancouver, Canada. Since 2012, he has been an Alexander von Humboldt Professor and the Chair for Digital Communication with the FAU. His research interests include the broad areas of communication theory, wireless communications, and statistical signal processing.

Dr. Schober is a fellow of the Canadian Academy of Engineering and a fellow of the Engineering Institute of Canada. From 2012 to 2015, he served as an Editor-in-Chief of the IEEE TRANSACTIONS ON COMMUNICATIONS and since 2014, he is the Chair of the Steering Committee of the IEEE TRANSACTIONS ON MOLECULAR, BIOLOGICAL, AND MULTISCALE COMMUNICATIONS. Furthermore, he is a member-at-large of the Board of Governors of the IEEE Communications Society. He has received several awards for his work, including the 2002 Heinz MaierLeibnitz Award of the German Science Foundation (DFG), the 2004 Innovations Award of the Vodafone Foundation for Research in Mobile Communications, the 2006 UBC Killam Research Prize, the 2007 Wilhelm Friedrich Bessel Research Award of the Alexander von Humboldt Foundation, the 2008 Charles McDowell Award for Excellence in Research from UBC, a 2011 Alexander von Humboldt Professorship, and a 2012 NSERC E.W.R. Steacie Fellowship. In addition, he has received several best paper awards for his research.